Gruppenkommunikation in mobilen Umgebungen

Eine Erweiterung des Gruppenkommunikationsparadigmas um eine Unterstützung für mobile Anwendungsnutzer

Von der Fakultät 1 - MINT – Mathematik, Informatik, Physik, Elektro- und Informationstechnik der Brandenburgischen Technischen Universität Cottbus–Senftenberg genehmigte Dissertation zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

vorgelegt von:

Dipl.-Inf. Jan Gäbler

geboren am 19.07.1979 in Bad Saarow-Pieskow

Vorsitzende:	Prof. Dr.-Ing. Monika Heiner
Gutachter:	Prof. Dr.-Ing. habil. Hartmut König
Gutachter:	Prof. Dr.-Ing. habil. Heiko Krumm
Tag der mündlichen Prüfung:	15.02.2019

Vorwort

Die vorliegende Arbeit entstand während bei meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Rechnernetze und Kommunikationssysteme der Brandenburgischen Technischen Universität. Im Rahmen verschiedener Forschungsprojekte arbeitete ich dort an dem Thema Gruppenkommunikation in mobilen Umgebungen. Die Arbeiten an der *uBeeMe Plattform für mobile kollaborative Anwendungen* haben dabei meine Arbeit wesentlich geprägt.

Bedanken möchte ich mich bei meinen Kolleginnen und Kollegen sowie Studentinnen und Studenten für Ihre gute und konstruktive Zusammenarbeit. Der guten Fee unseres Lehrstuhls, Katrin Willhöft, gilt mein Dank dafür, dass Sie mich immer bei den großen und kleinen Dingen unterstützte, die der Uni-Alltag so mitbringt. Ebenso gilt mein Dank Joachim Paschke, welcher mit technischen Beistand half.

Dank gebührt auch Michael Kirsche und Daniel Rakel für den fruchtbaren fachlichen Austausch rund um die Thematik dieser Arbeit, in deren Ergebnis viele Ideen geboren, diskutiert, ausgefeilt und oft auch wieder verworfen wurden.

Bedanken möchte ich mich auch bei Prof. Dr. Krumm für den Aufwand der Begutachtung der vorliegenden Arbeit.

Ein besonderer Dank geht an Herrn Prof. Dr. König, der diese Arbeit und die in ihrem Zusammenhang entstandenen Veröffentlichungen erst ermöglicht hat. Seine Unterstützung und die hilfreiche kritische Auseinandersetzung mit der Thematik gingen weit über das übliche Maß hinaus. Dafür vielen Dank.

Mein Dank gilt schließlich meinen Lieben, die mir in schwierigen Phasen den Rücken stärkten, mich zum Schreiben animierten, motivierten und dieses umfängliche Werk immer wieder Korrektur lasen. Vor allem mein Vater hat sich hier mit viel Einsatz um die Lesbarkeit dieser Seiten verdient gemacht.

Dipl.-Inf. Jan Gäbler

Cottbus, Juli2019

Preface

The present work arose during my work as a research assistant at the chair of computer networks and communication systems of the Brandenburg Technical University. As part of my research, I worked on the topic of group communication in mobile environments. The work on the *uBeeMe platform for mobile collaborative applications* has greatly influenced my work.

I would like to thank my colleagues and students for their good and constructive cooperation. The good fairy of our chair, Katrin Willhöft, gets my thanks for the fact that she always supported me with the big and small things that the university routine brings with it. Also my thanks to Joachim Paschke, who helped with technical assistance. Thanks also to Michael Kirsche and Daniel Rakel for the fruitful professional exchange around the subject of this work. In the result, many idas were born, discussed, polished and often rejected. I would also like to thank the gentlemen Prof. Heiko Krumm for the effort of reviewing the present work.

Special thanks go to Prof. Hartmut König, who has made this work and the publications created in its context possible. His support and the helpful critical examination of the subject went far beyond the usual measure. Thank you for that.

My thanks finally go to my dear ones, who have strengthened my back in difficult phases, I read for the writing animated, motivated and this extensive work repeatedly proofread. My father, in particular, has made a lot of efforts to make these pages more readable.

Dipl.-Inf. Jan Gäbler

Cottbus, Juli2019

Zusammenfassung

Mit der zunehmenden mobilen Nutzung des Internets steigt auch die Zahl von Anwendungen in mobilen Umgebungen. Diese sind neben ihren spezifischen Anforderungen oft durch eine enge Kooperation ihrer Nutzer gekennzeichnet. Werden mobile verteilte Entscheidungsprozesse modelliert, welche der Bewertung von Sachverhalten oder der Lösung von Aufgaben dienen, so nennt man diese Klasse von Anwendungen *mobile kollaborative Anwendungen*.

Sie werden in dieser Arbeit eingeführt. Ihre Eigenschaften und Besonderheiten werden diskutiert. Die Teilnehmer solcher Anwendungen kollaborieren innerhalb einer geschlossenen Gruppe und bauen ein gemeinsames Wissen auf. Auf dessen Basis kann die Anwendung lokale Entscheidungen treffen, über die in der Gruppe Konsens besteht. Die instabile Kommunikation in mobilen Umgebungen stellt jedoch an die Sicherung der Konsistenz des gemeinsamen Wissens besondere Anforderungen. Da klassische Verfahren zur Konsistenzsicherung in gruppenorientierten Systemen nicht genutzt werden können, ist die Sicherung der Konsistenz des globalen Wissens in mobilen Umgebungen das Leitthema dieser Arbeit.

Es wird ein gruppenorientierter Kommunikationsdienst – *Moversight* genannt – für mobile kollaborative Anwendungen vorgestellt, welcher ihnen ein konsistentes globales Wissen trotz instabilen Kommunikationsumfeldes bereitstellt. Hinzu kommt, dass neben der Konsistenzsicherung die Diensterbringung in einer ressourcenschonenden Art und Weise bei gleichzeitiger kontinuierlicher Adaption der Gruppe an sich ändernde Bedingungen einschließlich der Wartung ihrer Kommunikationsstrukturen gesichert werden müssen. Der Protokollentwurf für *Moversight* wird in dieser Arbeit vorgestellt.

Für *Moversight* wurden neuartige Funktionen zur Behandlung der Nutzermobilität entwickelt, welche mit den notwendigen Mechanismen zur Sicherung der Konsistenz des gemeinsamen Wissens interagieren. Ohne Gegenmaßnahmen würden häufige Verbindungsunterbrechungen das gemeinsame Wissen der Gruppe zerstören, wodurch diese logisch partitioniert und letztlich die Kollaboration der Nutzer verhindert wird.

Grundlage der Sicherung des gemeinsamen Wissens ist die *mobile optimistische virtuelle Synchronität (MOVS)*, welche das übliche gruppenorientierte Modell der Wissenssicherung um eine Behandlung der Nutzermobilität und variierender Verbindungseigenschaften der Gruppenteilnehmer erweitert. Dadurch können Teilnehmer bei Verbindungsproblemen zeitnah in die Gruppe zurückzukehren und ihr Wissen mit dieser abzugleichen.

Das vorgestellte Konzept zur Erkennung und Behandlung von Fehlersituationen mit anschließendem Gruppenwiedereintritt ist deutlich leistungsfähiger als das üblicherweise verwendete Verfahren. Dies wird erreicht durch eine leichtgewichtige Fehlerdetektion unter Nutzung einer neuartigen verteilten Latenzschätzung sowie einer kontinuierlichen Adaption der verwendeten Kommunikationstopologie an die sich permanent ändernden Kommunikationsbedingungen. Die entwickelte Optimierung der clusterbasierten Kommunikationstopologie eignet sich für alle gruppenorientierten clusterbasierten Anwendungsszenarien.

Darüber hinaus verfügt *Moversight* über kollaborative Dienste für das Teilen und Vereinigen von Gruppen, wie sie bisher in keinem gruppenorientierten Kommunikationsdienst zu finden sind. Beide Dienste ermöglichen eine anwendungsgetriebene Neustrukturierung der kollaborativen Gruppe bei gleichzeitiger Sicherung des globalen Wissens.

Abstract

With the increasing mobile use of the Internet, the number of applications in mobile environments is increased as well. In addition to their specific requirements, these applications often have a close group-oriented collaboration among their users. If mobile distributed decision processes are modeled that serve to assess issues or solve tasks this class of applications is called *mobile collaborative applications*.

We introduce this class in this work and discuss their characteristics and special features. The participants of such applications collaborate within a closed group and build a global knowledge. On this basis, the application can make local decisions that enable consensus within the group. However, the unstable communication in mobile environments puts special demands on securing the consistency of the global knowledge. Since it is not possible to use the traditional methods for consistency assurance in group-oriented systems, securing the consistency of global knowledge in mobile environments is the guiding theme of this work.

With *Moversight*, we introduce a group-oriented communication service for mobile collaborative applications, which assures a consistent global knowledge despite an unstable communications environment. In addition to consistency assurance, the resource-efficient provisioning of services is essential for mobile applications. Moreover, the group must continually adapted to the changing conditions including the maintenance of its communication structures.

We developed for *Moversight* novel mobility features, which interact with the mechanisms to ensure the consistency of the global knowledge. Without countermeasures, frequent disconnections would destroy the group's global knowledge, logically partitioning it and ultimately preventing users from collaborating.

The basis for securing the global knowledge is *Mobile Optimistic Virtual Synchrony (MOVS)*, which extends the usual group-centric consistency assurance model to include user mobility and handling of different connection properties of the group participants. Thus, in case of connection problems participants can quickly return to the group and synchronize their knowledge with this.

The presented concept for the detection and treatment of error situations with subsequent group re-entry is much more efficient than commonly used methods. We accomplish this by a lightweight error detection using a novel distributed latency estimation and continuous adaptation of the communication topology to ever-changing communication conditions. The developed optimization of the cluster-based communication topology is suitable for all group-oriented, cluster-based application scenarios.

Furthermore, *Moversight* provides collaborative services for application-driven splitting and merging of groups as they have yet not been found in any group-based communications service. Both services enable application-driven restructuring of the collaborative group while preserving global knowledge.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

Abkürzungsverzeichnis

AS	Autonome Systeme ... 30
B2B	Business-to-Business ... 11
CCN	Content Centric Networking
DCS	Dynamische Clustering-Strategie ... 71
DTN	Delay-Tolerant Networks ... 34
DVMRP	Distance Vector Multicast Routing Protocol ... 25
DHCP	Dynamic Host Configuration Protocol
EVS	Erweiterte Virtuelle Synchronität ... 49
FIFO	First In First Out ... 44
GKS	Gruppenkommunikationssystem ... 23
GD	Group-Data-Nachricht ... 188
GPRS	General Packet Radio Service ... 15
GT	globale Empfangszeit-Nachricht ... 74
ID	Identifikationsnummer ... 16
IP	Internet Protocol ... 13
IPSec	Internet Protocol Security ... 17
JA	Join-Announce-Nachricht ... 70
LA	Leave-Announce-Nachricht ... 71
LAN	Local Area Network ... 15
LEACH	Low-Energy Adaptive Clustering Hierarchy ... 124
LT	lokale Empfangszeit-Nachricht ... 74
LTE	Long Term Evolution ... 25
MA	Merge-Announce-Nachricht ... 153
MAB	Merge-Abort-Nachricht ... 153
MANET	Mobile Ad-hoc Network ... 29
MC	Merge-Confirm-Nachricht ... 150
mDNS	Multicast-DNS ... 16
MF	Merge-Finish-Nachricht ... 153
MFL	Merge-Flush-Nachricht ... 152
MoCA	Mobile Collaboration Architecture ... 15
MOVS	Mobile Optimistic Virtual Synchronity ... 95
MPI	Message Passing Interface ... 9
MR	Merge-Request-Nachricht ... 150
MRJ	Merge-Reject-Nachricht ... 150
MRO	Merge-Roster-Nachricht ... 153
NAT	Network Address Translation ... 17

NDN	Named Data Networking ... 35
NICE	'NICE is the Internet Cooperative Environment' ... 18
NVB	Next-View-Buffer ... 138
OLSR	Optimized Link State Routing ... 29
OMNeT++	OMNeT++ Discrete Event Simulator ... 74
OSI	Open System Interconnection ... 29
OVS	Optimistische Virtuelle Synchronität ... 51
P&S	Publish & Subscribe ... 31
P2P	Peer-to-Peer ... 9
QoS	Quality of Service ... 17
RA	ReJoinAnnounce-Nachricht ... 93
RF	ReJoinFailed-Nachricht
RMS	Rollen-Wartungsdienst ... 132
RPC	Remote Procedure Call ... 33
RRO	Reduced-Roster-Nachricht ... 93
RO	Roster-Nachricht ... 70
RSA	Role-Switch-Announce-Nachricht ... 136
RTT	Round Trip Time ... 109
RU	Resource-Update-Nachricht ... 136
RV	Ressourcenwert ... 134
RVA	Resource-Value-Announce-Nachricht ... 136
SDN	Software-Defined Network ... 32
SPA	Split-Announce-Nachricht ... 149
SpoVNet	Spontaneous Virtual Networks ... 15
TCP	Transmission Control Protocol ... 110
TOM	Total ordered mobile Multicast ... 64
TLS	Transport Layer Security ... 17
uBeeMe	Ubiquitous Mobile Environment ... 15
UDP	User Datagram Protocol ... 34
UMTS	Universal Mobile Telecommunications System ... 15
VS	Virtuelle Synchronität ... 23
WAN	Wide Area Network ... 15
WLAN	Wireless Local Area Network ... 15
WVS	Schwache Virtuelle Synchronität ... 50
XMPP	Extensible Messaging and Presence Protocol ... 18

Teil I

Grundlagen

Kapitel 1

Einleitung

Die Vision des allgegenwärtigen Rechnens und Kommunizierens wird mehr und mehr Wirklichkeit. Die wachsende Mobilität der Internetnutzer bedingt eine Kollaboration auch außerhalb drahtgebundener Netze. Viele Nutzer verfügen heutzutage aufgrund der stark gewachsenen Fähigkeiten und Eigenschaften von Tablets und Smartphones nur noch über mobile Endgeräte.

1.1 Mobile kollaborative Anwendungen

Die Anzahl mobiler Anwendungen und Szenarien nimmt kontinuierlich zu. Durch deren spezifische Anforderungen und die Notwendigkeit einer engen Kooperation zwischen den (mobilen) Nutzern entstand eine neue Art von Anwendungen: die *mobile kollaborative Anwendung*. Typische Vertreter sind mobile Konferenzsysteme, kollaboratives Schreiben oder mobile Spiele, aber auch Umweltmonitoringsysteme. Diese Art von Anwendungen modelliert einen verteilten Entscheidungsprozess, welcher die Bewertung von Sachverhalten bzw. die Lösung von Aufgaben zum Ziel hat. Der wesentliche Unterschied zu anderen verteilten Anwendungen mit mobilen Nutzern ist die enge Kooperation zwischen den Nutzern. Diese ist durch einen häufigen, gruppenorientierten und intensiven Austausch gekennzeichnet, aus dem ein globales gemeinsames Wissen zwischen den Gruppenmitgliedern entsteht.

Die zu lösende Aufgabe beim Entwurf solcher Anwendungen besteht darin, unterschiedliche Szenarien, Geräte, Ressourcen und Kommunikationsschemata zu unterstützen. Dabei sollen gleichzeitig stationäre, nomadische und mobile Nutzer einbezogen werden, wodurch mobile kollaborative Anwendungen in verschiedenen mobilen Umgebungen mit unterschiedlichen Bedingungen Verwendung finden können. Aufgrund des engen Austauschs besteht die kollaborative Gruppe zumeist aus nicht mehr als einhundert Teilnehmern. Die begrenzten Ressourcen der Endgeräte machen es notwendig, dass die entstehenden Kommunikationsaufwände gleichmäßig verteilt werden. Des Weiteren muss ein Kompromiss gefunden werden zwischen der notwendigen Kommunikation zur Erzeugung und Sicherung des gemeinsamen Gruppenwissens und der Lebensdauer der Endgeräte.

Für die Unterstützung mobiler Nutzer wurden bereits verschiedene Kommunikationsschemata vorgeschlagen, z. B. [16, 85, 229]. Kollaborative Anwendungen erfordern jedoch einen zuverlässigen gruppenorientierten Transportdienst, den diese Schemata zumeist nicht unterstützen. Ein Dienst, der die gefragten Anforderungen erfüllt, kann auf zwei Arten angeboten werden:

1. unter Nutzung fester Infrastrukturelemente, beispielsweise durch einen Server, um alle kollaborationsspezifischen Aufgaben zu bearbeiten, oder

2. durch die Integration von gruppenorientierten Kommunikationssystemen in die Endgeräte in einer *Peer-to-Peer* (P2P)-basierten Art und Weise.

Der erste Ansatz kann vor allem in Umgebungen mit einer stabilen Kommunikation eingesetzt werden. Die überwiegende Zahl der mobilen kollaborativen Anwendung

kann jedoch nicht auf eine feste Netzinfrastruktur zurückgreifen. Stattdessen operieren diese Anwendungen mittels des zweiten Ansatzes netzübergreifend auf Basis eines gemeinsamen Transportprotokolls.

Charakteristisch für drahtlose Szenarien sind kurzzeitige Verbindungsverluste bei einzelnen oder mehreren Gruppenmitgliedern, welche deren Kommunikation stören oder völlig unterbinden. Die Unterbrechungsdauer variiert und kann unbegrenzt [6] sein. Die Verbindungverluste sind unter anderem auf die Mobilität der Nutzer und Probleme innerhalb des Kommunikationsnetzes wie Paketverluste, Überlastsituationen oder fehlende Netzabdeckung zurückzuführen. Häufig liegt auch eine Ressourcenerschöpfung in den Endgeräten vor. Als Konsequenz kann der Wissensstand zwischen den einzelnen Gruppenmitgliedern voneinander abweichen.

1.2 Konsistenzsicherung als Leitthema dieser Arbeit

Die Sicherung des globalen Wissens innerhalb einer Gruppe von kooperierenden mobilen Teilnehmern ist das zentrale Thema dieser Arbeit. Dafür stellen die in der vorgelegten Arbeit entwickelten Dienste und Konzepte Lösungsvorschläge dar, welche im Rahmen des so genannten *CAP-Theorems* (C für Konsistenz, A für Verfügbarkeit und P für Ausfalltoleranz) agieren. Das CAP-Theorem von Brewer [38, 39] stellt fest, dass es in einem verteilten System nicht möglich ist, die Eigenschaften Konsistenz und Verfügbarkeit während eines Partitionierungsereignisses gleichzeitig zu garantieren. Die Sicherung eines globalen Wissens innerhalb einer Gruppe von mobilen Teilnehmern ist auf den ersten Blick jedoch genau der Versuch, dieses Theorem zu widerlegen. Auf Details und Hintergründe zum CAP-Theorem wird im Laufe dieser Arbeit ebenso vertiefend eingegangen wie auf grundsätzliche Aussagen zu den Themen Synchronisation, Verfügbarkeit, Konsistenz und Partitionstoleranz.

Ausgehend vom CAP-Theorem stellt sich zunächst die Frage, wie ein gruppenorientiertes Kommunikationssystem nicht erreichbare Teilnehmer zu behandeln hat. In stationären Systemen werden solche Teilnehmer nach einer gewissen Anzahl erfolgloser Versuche von zukünftigen Übertragungen ausgeschlossen, um die Konsistenz des gemeinsamen Wissens und die Verfügbarkeit des Systems zu sichern. In mobilen Umgebungen ist diese Frage komplexer. Die ihnen inhärente Unsicherheit über die Stabilität der Verbindung macht eine sichere Aussage, ob ein Teilnehmer zur kollaborativen Gruppe „*zurückkehrt*“, unmöglich. So kann ein aktuell nicht erreichbarer Teilnehmer beispielsweise seine Verbindungsprobleme überwinden und bezüglich seiner Verbindungseigenschaften in der Folge ein stationäres Verhalten aufweisen, während ein bislang stabil kommunizierender Teilnehmer aufgrund geänderten Mobilitätseigenschaften zukünftig mit Verbindungsproblemen konfrontiert sein kann. Daher stellen mobilitätsinduzierte Verbindungsunterbrechungen die größte Schwierigkeit für die stabile Bereitstellung mobiler kollaborativer Dienste dar.

Die Anzahl der zur Diensterbringung notwendigen Teilnehmer ist abhängig von der kollaborativen Anwendung bzw. dem eingesetzten gruppenorientierten Kommunikationssystem. Wenn aus Sicht der Anwendung alle Teilnehmer gleichrangig bzw. „*gleich wichtig*“ sind, kann die Kooperation bereits beim Ausfall eines Teilnehmers unterbrochen werden. Hier wird deutlich, dass trotz der strengen Eingrenzung des Anwendungsgebiets auf mobile kollaborative Anwendungen bestimmte Reaktionen auf Fehler nicht durch das verteilte System getroffen werden können, sondern dem Anwendungsentwickler überlassen bleiben müssen.

Verbindungsunterbrechungen sind in mobilen kollaborativen Anwendungen häufig zu erwarten. Die in dieser Arbeit vorgestellten Dienste sollen u. a. den Wiedereintritt eines von der Gruppe getrennten Teilnehmers zeitnah und mit geringem Aufwand ermöglichen. Dazu muss er die Änderungen am gemeinsamen Wissen nachvollziehen können, welche in seiner Abwesenheit innerhalb der Gruppe entstanden sind, um sich mit der Gruppe wieder zu synchronisieren.

Eine Lösung zur Sicherung der Konsistenz des gemeinsamen Wissens in gruppenorientierten Kommunikationssystemen auch in Gegenwart von nicht reagierenden Teilnehmern

existiert bisher nicht. Beim Entwurf solch einer Lösung entsteht eine Reihe von Fragen wie:

- Wie ist ein nicht reagierender Teilnehmer bei der Übertragung einer Gruppennachricht zu behandeln?
- Wie gestaltet sich der Wiedereintritt des ehemals nicht reagierenden Teilnehmers?
- Wie wird die Konsistenz des gemeinsamen Wissens wieder hergestellt?
- Ab wann ist dieser Teilnehmer wieder ein normales Gruppenmitglied?

Bisher sind diese und weitere Fragen unbeantwortet. Ohne eine geeignete Mobilitätsunterstützung muss jede Anwendung für sich die zuverlässige Nachrichtenübertragung sicherstellen, einschließlich der Einhaltung der Nachrichtenordnung und der Wissenssynchronisation. Damit entsteht für den Anwendungsentwickler ein erheblicher Mehraufwand, verbunden mit zusätzlichen Fehlerquellen. Im Rahmen dieser Arbeit wird ein neues semantisches Modell zur Wahrung der virtuellen Synchronität in mobilen Gruppen vorgestellt, welches die bestehenden Ansätze für den Einsatz in mobilen Umgebungen weiterführt. Die Definition dieses Modells erfolgt unter Berücksichtigung der Funktionen kollaborativer Anwendungen.

1.3 Funktionen kollaborativer Anwendungen

Kollaborative Anwendungen bestehen aus einer Reihe von Basisfunktionen, welche je nach konkretem Anwendungsszenario kombiniert werden. In dieser Arbeit werden die Funktionen eines gruppenorientierten Kommunikationssystems herausgearbeitet, welche für die Umsetzung mobiler kollaborativer Anwendungen notwendig sind. Eine analytische Untersuchung verschiedener Ansätze zeigt, dass Gruppenkommunikationssysteme am besten für diese Umsetzung geeignet sind. Insbesondere die Sicherung des gemeinsamen Wissens der Gruppe in mobilen Umgebungen ist durch bisherige Ansätze nicht gewährleistet, da diese nicht für Anwendungsfälle mit häufigen Verbindungsabbrüchen konzipiert sind.

Bisher existierende Gruppenkommunikationsansätze schließen zumeist die Teilnehmer aus der Gruppe aus, welche nicht oder fehlerhaft reagieren. Dadurch gehen in der Regel sämtliche anwendungs- und gruppenbezogenen Informationen verloren. Das stört die Kooperation insbesondere bei häufigen und unerwarteten Verbindungsunterbrechungen. Ohne eine Wiedereintrittsfunktion müssen diese Teilnehmer erneut in die Gruppe eingeladen werden, was den Ressourcenverbrauch erhöht. Das verlorengegangene gemeinsame Wissen muss beim wiedereingeladenen Teilnehmer erneut hergestellt werden. Um eine gruppenorientierte Kommunikation in mobilen Szenarien stabil zu ermöglichen, ist folglich eine Wiedereintrittsfunktion wünschenswert, die eine beständige Kollaboration unter Sicherung des gemeinsamen Wissens ermöglicht.

Neben der Sicherung der Konsistenz bei kurzzeitigen Verbindungsunterbrechungen beschäftigt sich diese Arbeit mit weiterführenden Operationen für die Unterstützung von Prozessen in mobilen Gruppen. Die dazu notwendigen Funktionen werden als eigenständige Dienste innerhalb des Gruppenkommunikationssystems definiert. Ein Dienst zur Mobilitätsunterstützung umfasst alle Funktionen für den nahtlosen Wiedereintritt von Gruppenteilnehmern, welche aufgrund einer Verbindungsunterbrechung von der Gruppe getrennt wurden. Der Dienst gewährleistet die Wiederaufnahme der Verbindung zur Gruppe, die Reintegration in diese und den Abgleich bzw. die Wiederherstellung des gemeinsamen Wissens. Ebenso werden die theoretischen Grundlagen für die Nutzung dieser Funktionen hergeleitet.

Wegen der ihnen eigenen Dynamik unterliegt die Kommunikationsstruktur einer kollaborativen Gruppe fortlaufend Änderungen, beispielsweise durch Gruppeneintritte und -austritte oder Änderungen der Teilnehmereigenschaften. Das kann die Leistungsfähigkeit der Gruppe negativ beeinflussen. Eine kontinuierliche Anpassung der Gruppe an die aktuellen Gegebenheiten, d. h. eine Wartung, ist daher vorteilhaft.

In verteilten Entscheidungsprozessen ist es unter Umständen notwendig, eine weitere Gruppe in die kollaborative Gruppe zu integrieren bzw. eine Teilgruppe aus dieser zu entfernen. Solche eine Teilgruppe kann beispielsweise einen konkreten Sachverhalt getrennt von der Restgruppe bearbeiten und ihre Ergebnisse später mit der Gruppe austauschen. Dabei ist es wichtig, dass bei der Teilung bzw. Vereinigung der Gruppen das Wissen der/in den Teilgruppen erhalten bleibt. Vergleichbar sind diese Operationen mit den „*branch*“ und „*merge*“ Operationen von Versionskontrollsystemen. Waren Mitglieder während dieser Operationen von der Gruppe getrennt, so müssen auch diese Änderungen beim Wiedereintritt abgeglichen werden. Um dies zu ermöglichen, werden kollaborative Dienste für die Aufteilung der Gruppe sowie die Vereinigung mit anderen Gruppen vorgestellt. Für den Entwurf beider Dienste nimmt wiederum die Sicherung der Konsistenz des gemeinsamen Wissens trotz Verbindungsausfällen zwischen den Teilnehmern eine Schlüsselstellung ein.

Verschiedene Funktionen in einem kollaborativen System müssen durch die Gruppe überwacht werden, um beispielsweise ein fehlerhaftes Verhalten von Teilnehmern zu erkennen. Hierzu kommen verschiedene Timer zum Einsatz. Ihre Laufzeit wird bisher entweder statisch definiert oder mittels der Übertragungsverzögerung lokal abgeschätzt, welche in mobilen Umgebungen allerdings deutlich schwankt. Eine statische Definition reduziert demgegenüber die Leistungsfähigkeit der Systeme, weshalb eine dynamische Einstellung zu bevorzugen ist. In kollaborativen Anwendungen muss sichergestellt werden, dass die gesamte Gruppe für bestimmte Operationen gleiche Laufzeiten für die Timer verwendet. Deshalb wird ein generisches Verfahren eingeführt, das die zu erwartenden Verzögerungen für Gruppennachrichten abschätzt.

1.4 Gliederung der Arbeit

Nach Einführung des Begriffs der mobilen kollaborativen Anwendung in Kapitel 2 werden anhand eines Beispielszenarios die Anforderungen an solche Anwendungen sowie deren notwendigen Eigenschaften und Funktionen abgeleitet. Daran schließt sich in Kapitel 3 eine analytische Untersuchung verschiedener existierender Ansätze gruppenorientierter Kommunikationssysteme an, die ihre Nutzung für mobile kollaborative Anwendungen evaluiert. Die Ergebnisse zeigen, dass Gruppenkommunikationssysteme am geeignetsten für solch eine Umsetzung sind, obwohl sie nicht alle Anforderungen erfüllen.

Insbesondere die Sicherung der Konsistenz des gemeinsamen Wissens der Gruppe in mobilen Umgebungen ist durch bestehende Gruppenkommunikationssysteme nicht gewährleistet, da sie die Mobilität und die damit zusammenhängenden häufigen Verbindungsabbrüche unzureichend berücksichtigen. Nach einer Einführung in die theoretischen Grundlagen zum Thema Gruppenkommunikation und Sicherung der Konsistenz des gemeinsamen Wissens in Kapitel 4 werden die Anforderungen der Konsistenzsicherung in mobilen Umgebungen herausgearbeitet und dafür bestehende Konzepte vorgestellt. Anhand eines Experiments werden abschließend die Auswirkungen einer fehlenden Mobilitätsunterstützung auf die Konsistenz des Wissens in mobilen Umgebungen demonstriert.

Basierend auf diesen Vorbetrachtungen wird in den folgenden Abschnitten ein allgemeiner Ansatz zur Behandlung der Nutzermobilität präsentiert, welcher sich insbesondere auf die Sicherung der Konsistenz des gemeinsamen Wissens innerhalb einer mobilen Gruppe konzentriert. Dieser Ansatz basiert auf einem neuen semantischen Modell zur Wahrung der virtuellen Synchronität, welches Anforderungen, Eigenschaften und Funktionen an Gruppenkommunikationssysteme definiert, die sicherstellen sollen, dass auch in mobilen Umgebungen das virtuelle Wissen zwischen allen Teilnehmern konsistent ist. Dieses Modell erweitert die bestehenden theoretischen Grundlagen zur virtuellen Synchronität konsequent und legt dabei den Schwerpunkt auf die Identifizierung und Behandlung von Teilnehmern, welche aufgrund von temporären Verbindungsunterbrechungen nicht protokollkonform agieren. In diesem Zusammenhang wird mit dem Gruppenkommunikationsprotokoll *Moversight* ein praktikabler Lösungsansatz vorgestellt, welcher dieses Modell umsetzt. Zunächst werden in Kapitel 5 die Funktionen und die Basisdienste von *Moversight* eingeführt. Darauf aufbauend wird die Behandlung

der Nutzermobilität im Kapitel 6 diskutiert. Der Schwerpunkt liegt dabei in der Erkennung und Behandlung von Verbindungsabbrüchen einzelner Gruppenteilnehmer und ihrem Wiedereintritt in die Gruppe unter bestimmten Bedingungen bei Wahrung der Konsistenz des gemeinsamen Wissens.

Im Kapitel 7 wird der verwendete Ansatz für eine gruppenweite Abschätzung der Verzögerungen bei der Übertragung von Gruppennachrichten erläutert. Er ist verteilt und kann lokal so ausgeführt werden, dass die Schätzergebnisse gruppenweit identisch sind. Für die dynamische Adaption der Gruppentransferdienste an die sich ändernden Kommunikationsbedingungen sind die Funktionen des vorgestellten Ansatzes notwendig. Im Ergebnis einer anschließenden Variantendiskussion wird dessen fachliche Richtigkeit geprüft.

Die ständig sich ändernden Bedingungen und Zusammensetzungen einer Gruppe erfordern eine kontinuierliche Anpassung bzw. Optimierung der Gruppentopologie. Dafür wird im Kapitel 8 ein verteilter Wartungsdienst vorgestellt. Er definiert ein allgemeines Konzept zur Nachrichtenbehandlung während eines Sichtwechsels, welches die Konsistenz des Gruppenwissens sichert. Im Verlauf des Sichtwechsels wird die Gruppe an die geänderten Bedingungen anpasst. Die Leistungsfähigkeit des Ansatzes wird in verschiedenen Szenarien nachgewiesen.

In kollaborativen Anwendungen kann es notwendig sein, eine Gruppe anwendungsgetrieben in zwei oder mehrere eigenständige Gruppen aufzuspalten bzw. diese umgekehrt wieder zu vereinen. Die in Kapitel 9 vorgestellten Dienste ermöglichen dies. Dabei steht wiederrum die Sicherung der Konsistenz des gemeinsamen Wissens im Mittelpunkt. Auf Basis des CAP-Theorems werden die Herausforderungen bei der Sicherung des gemeinsamen Wissens dargestellt und mögliche Ansätze skizziert. Die gefundene Lösung für das Aufteilen und Vereinigen von Gruppen mit Schwerpunkt Konsistenzsicherung wird mit einer Leistungsanalyse beendet.

Die Ausführungen in dieser Arbeit schließen in Kapitel 10 mit einer Zusammenfassung der Anforderungen, Konzepte und Realisierungsmöglichkeiten eines gruppenorientierten Kommunikationssystems für mobile kollaborative Anwendungen. Ein Ausblick auf mögliche Erweiterungsarbeiten rundet die Darstellungen ab.

Kapitel 2

Mobile kollaborative Anwendungen

Verteilte Anwendungen, die die Kooperation von Anwendern an verschiedenen Standorten in einer vernetzten Infrastruktur unterstützen, bezeichnet man als kollaborative Anwendungen. Sie sind in der Forschung intensiv untersucht worden. Auf Basis dieser Arbeiten haben sich verschiedene Formen herausgebildet, welche eine breite Nutzung erfahren. Als Beispiel wären Kollaborationsplattformen zu nennen wie Sharepoint Workspaces von Microsoft, soziale Plattformen wie Facebook oder Twitter sowie Werkzeuge, z. B. Whiteboard, Application Sharing und Filetransfer, welche in andere Anwendungen integriert werden.

Kollaborative Anwendungen, die im Festnetzbereich genutzt werden, haben charakteristische Eigenschaften. So ist ihre Teilnehmerstruktur relativ stabil und ohne häufige Ein- und Austritte. Ihre Kommunikation unterliegt nur wenigen Störungen, ihre Ressourcen sind potentiell unbeschränkt. Dies ermöglicht klare und stabile Systemlösungen. Mit der steigenden Mobilität von Nutzern im Internet müssen zunehmend Teilnehmer eingebunden werden, die unterwegs sind. Sie sind nur über eine drahtlose Kommunikationsschnittstelle zu erreichen. Die Mobilität der Teilnehmer und die drahtlose Kommunikation stellen eine Reihe zusätzlicher Anforderungen an die Gestaltung dieser Anwendungen, die man deshalb auch als *mobile kollaborative Anwendungen* bezeichnet.

Der Begriff der mobilen kollaborativen Anwendung wird in der Literatur nicht konsistent verwendet. So nutzen beispielsweise Barolli und Xhafa [22] in ihrer Plattform für das verteilte kollaborative Arbeiten ein klassisches *Peer-to-Peer* (P2P) Overlay unter Nutzung der JXTA-P2P-Protokollfamilie [103]. Auf dieser Basis realisieren sie weit verteilte Berechnungen durch eine große Zahl von Systemen, vergleichbar mit dem *Message Passing Interface* (MPI) Standard [157].

In der Arbeit von Cheverst *et al.* [53], welche sich mit dem Thema Konsistenz in mobilen kollaborativen Anwendungen befasst, steht das häufige Teilen von Informationen in einem Gruppenkommunikationssystem im Mittelpunkt.

Cacho *et al.* [41] untersuchen in mobilen kollaborativen Anwendungen die Auswirkungen von Fehlern auf die Konsistenz von Informationen innerhalb einer Gruppe. Im Gegensatz zu Cheverst *et al.* basieren die Anwendungen bei Cacho *et al.* jedoch auf einer Middleware [186].

Mobilität und Kooperation werden zusammen mit der zunehmenden Vernetzung und Computerisierung des Alltags im Sinne des Ubiquitous Computing [215] immer mehr zu den prägenden Elementen des Internets. Ein wichtiges Teilgebiet des Ubiquitous Computing sind die mobilen kollaborativen Anwendungen. Dieses Kapitel gibt einen Überblick über die Eigenschaften mobiler kollaborativer Anwendungen und die benötigten Funktionen. Dabei werden speziell die Anforderungen diskutiert, die an die Gestaltung der Gruppenkommunikation innerhalb dieser Anwendungen gestellt werden.

2.1 Eigenschaften mobiler kollaborativer Anwendungen

Die in der Literatur dokumentierten mobilen kollaborative Anwendungen sowie die dazu verwendeten Ansätze variieren in ihren Eigenschaften und Merkmalen. Allen gemein ist die Repräsentation der kooperierenden Endsysteme in einer geschlossenen virtuellen

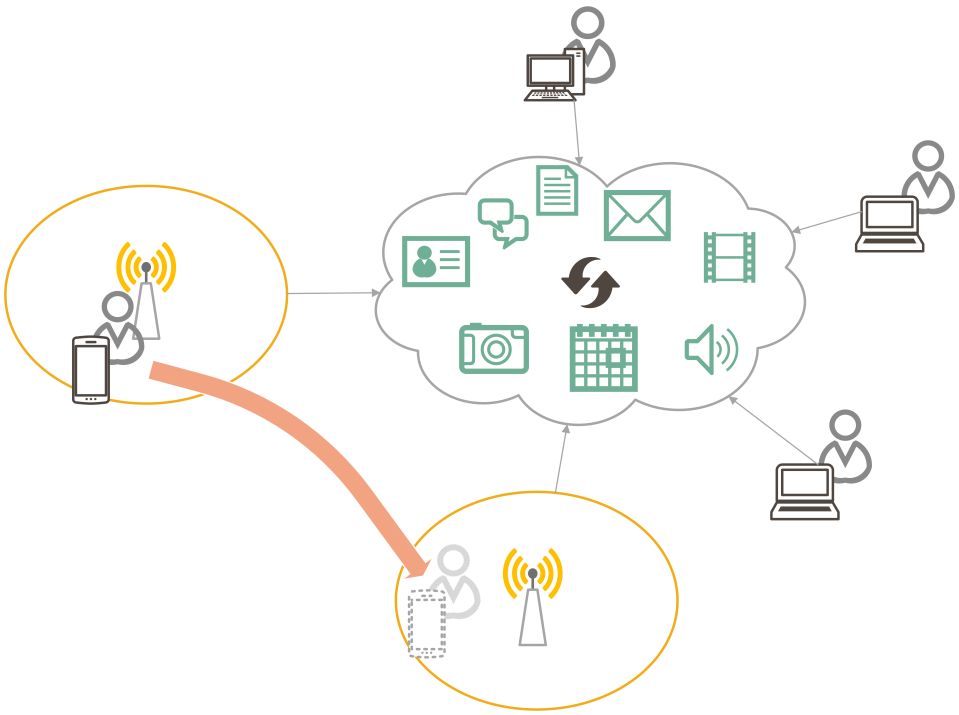


Abbildung 2.1: Prinzipieller Aufbau einer mobilen kollaborativen Anwendung

Gruppe. Innerhalb dieser Gruppe besteht eine enge Kollaboration bzw. Kooperation im Sinne eines regelmäßigen Informationsaustauschs.

Abbildung 2.1 zeigt den prinzipiellen Aufbau einer solchen Anwendung. Im Gegensatz zu Anwendungen wie Facebook oder Twitter zeichnen sich die mobilen kollaborativen Anwendungen im Allgemeinen durch eine geringe Anzahl von Teilnehmern aus (maximal einige Hundert), welche sich häufig spontan verbinden, um zusammenzuarbeiten und dabei eng zu interagieren. Des Weiteren weisen die nomadischen oder mobilen Teilnehmer variierende Latenzen und häufige Unterbrechungen in ihren Kommunikationsverbindungen auf.

Gruppen in mobilen kollaborativen Anwendungen sind sehr dynamisch. So ändert sich beispielsweise die Anzahl ihre Teilnehmer häufig. Auch die Dauer der Zusammenarbeit ist im Vergleich zu anderen Kollaborationsformen deutlich reduziert. Ein typischer Anwendungsfall wäre z. B. ein Reisender am Flughafen, der seine Wartezeit mit einem Mehrteilnehmerspiel überbrücken möchte. Aufgrund der Mobilität und der Spontanität einzelner Teilnehmer treten bei diesem Spiel immer wieder Verbindungsverluste im Sinne von Churns [169] zur restlichen Gruppe auf, welche die Konsistenz der Gruppe gefährden können.

Durch ihre enge Kooperation und das häufige Teilen von Informationen ist die Zusammenarbeit in mobilen kollaborativen Anwendungen sehr individuell [134, 208]. Viele Anwendungsgebiete sind jedoch von einer spontanen und dezentralen Kollaboration geprägt. So ist es beispielsweise in einem Flughafen problematisch, eine zusätzliche zentralisierte Infrastruktur für ein Mehrteilnehmerspiel zu etablieren. Die Installation wäre ökonomisch aufwendig – sofern überhaupt möglich – und daher zumeist nicht zu rechtfertigen. Aus diesem Grunde werden dezentrale Konzepte genutzt, welche die bereits vorhandene Netzinfrastruktur wiederverwenden.

Eine weitere charakteristische Eigenschaft mobiler kollaborativer Anwendungen ist das *globale Wissen*. Darunter versteht man die Informationen, welche allen Teilnehmern in der Gruppe bekannt und zugänglich sind. Beispiele für globales Wissen sind die aktuelle Gruppenzusammensetzung, Zustände und Eigenschaften der einzelnen Gruppenmitglieder, die Belegung von globalen Variablen oder die Kenntnis über die in der Gruppe ausgetauschten Nachrichten.

Um die Konsistenz des globalen Wissens zu sichern, wird die Gruppenzusammenset-

zung oft in *Sichten* (engl. Views) [55] strukturiert. Das Konzept der Sicht stammt aus dem Bereich der Gruppenkommunikation. Es beschreibt eine nummerierte Folge von Gruppenzusammensetzungen in Abhängigkeit von der Zeit. Eine solche Zusammensetzung besteht aus einer Menge von Teilnehmern, welche zu diesem Zeitpunkt Mitglied der Gruppe sind, wobei auch die leere Menge zulässig ist. Möchte ein Gruppenmitglied das globale Wissen ändern, muss diese Änderung allen Mitgliedern der aktuellen Sicht angezeigt werden, damit die Konsistenz des globalen Wissens gewahrt bleibt.

Nicht zuletzt soll auf die notwendige Toleranz gegenüber Partitionierungen verwiesen werden. Eine Gruppe kann aufgrund verschiedener Effekte in Teilgruppen zerfallen, welche daraufhin zumeist nicht mehr miteinander kommunizieren können. Änderungen des globalen Wissens wirken sich somit nur innerhalb einer Teilgruppe aus. Die Konsistenz des globalen Wissens geht verloren; das Wissen zwischen den Teilgruppen „*driftet auseinander*“. Partitionierungstolerante Anwendungen besitzen Funktionen, welche das Auseinanderdriften im Falle einer Partitionierung verhindern und nach Behebung der Partitionierungsursache die Teilgruppen wieder konsistent zusammenführen. Dabei ist zwischen einer kommunikativen und einer semantischen Partitionierung zu unterscheiden:

Die *kommunikative Partitionierung* entsteht auf der Netzebene aufgrund von Kommunikationsunterbrechungen, ausgelöst durch Churn-Effekte oder Störungen in der Netzinfrastruktur, sodass einzelne oder mehrere Mitglieder nicht mehr durch die Gruppe erreichbar sind. Nach Behebung der Verbindungsunterbrechung müssen die betroffenen Mitglieder erneut in die Gruppe integriert und die Konsistenz des globalen Wissens wiederhergestellt werden.

Die *semantische Partitionierung* wird hingegen durch die kollaborativen Anwendungen induziert. Hier möchte die Gruppe sich bewusst in Teilgruppen aufgliedern (im Sinne eines *Split*) bzw. zuvor getrennte (Teil-) Gruppen (wieder-) vereinen (im Sinne eines *Merge*). Im Gegensatz zur kommunikativen Partitionierung handelt es sich also um eine beabsichtigte Strukturänderung innerhalb der betroffenen Gruppe bzw. Teilgruppe, welche sich ebenfalls auf die Konsistenz des globalen Wissens auswirkt. Die Sicherung des globalen Wissens ist bei Strukturänderungen eine anspruchsvolle Aufgabe, da das Risiko besteht, dass Nachrichten nicht an alle Mitglieder einer Sicht ausgeliefert werden können und dadurch das globale Wissen nicht mehr konsistent zu den Sichten der übrigen Mitglieder ist.

2.2 Beispielszenario

Die Zahl der Anwendungsfälle und -szenarien für mobile kollaborative Anwendungen ist sehr groß und wächst kontinuierlich. Typische Beispiele sind mobile Audio-/Videokonferenzen, Telediagnosen, kollaboratives Schreiben, Überwachung und Tracking im Rahmen von *Business-to-Business* (B2B)-Anwendungen, mobile Spiele und E-Learning-Anwendungen. Im Folgenden werden anhand eines detaillierten Anwendungsszenarios die Herausforderungen und notwendigen Komponenten für die Realisierung von mobilen kollaborativen Anwendungen identifiziert. Auf dieses Beispiel wird in der Arbeit bei der Darstellung von Diensten und deren Umsetzung immer wieder zu Illustrationszwecken zurückgegriffen.

Die Schäden, welche durch Naturkatastrophen oder Menschenhand entstehen, werden von Jahr zu Jahr größer. Eine mögliche Begleiterscheinung von Katastrophen kann die Freisetzung von giftigen oder gefährlichen Stoffen sein, verursacht durch chemische Unfälle, technische Störfälle, terroristische Angriffe oder Umweltereignisse, wie das Beispiel Fukushima zeigt. Dadurch wird oft ein großer Teil der lokalen Infrastruktur zerstört, Straßen werden unpassierbar und eine große Anzahl von Personen wird verschüttet oder vermisst. Ohne zusätzliche und weitgehend autonom operierende Hilfsmittel ist die Arbeit der Rettungs- und Hilfskräfte sehr gefährlich. Roboter beziehungsweise autonome Systeme können in diesen Situationen eine effektive Unterstützung für die Rettungskräfte darstellen.

Gegenstand der Betrachtung ist eine Gruppe von autonomen Robotern, die kooperativ ein Katastrophengebiet erkunden. Die Roboter kommunizieren untereinander und

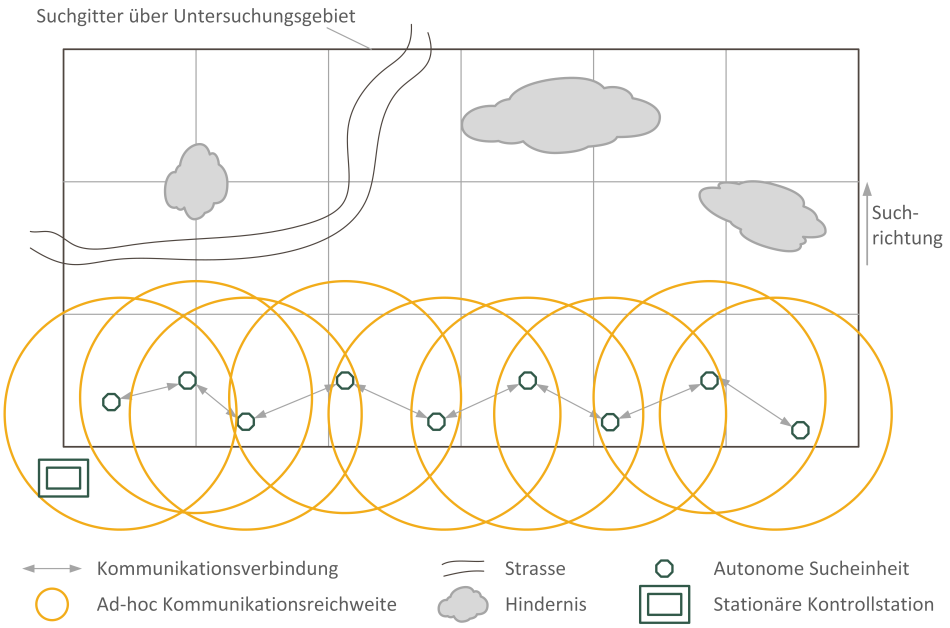


Abbildung 2.2: Schematische Darstellung des kollaborativen Roboter-Anwendungsszenarios

zusätzlich mit einer stationären Kontrollstation, welche sich außerhalb des betroffenen Gebietes befindet und die Roboter bei Bedarf operativ koordinieren kann. Die Roboter sollen eigenständig das Katastrophengebiet entsprechend vorgegebener Kriterien, beispielsweise nach Opfern oder zur Bestimmung der Konzentration vorhandener chemischer Stoffe, durchsuchen. Ihren Weg bestimmen sie neben den Vorgaben der stationären Kontrollstation in enger Zusammenarbeit mit den anderen an der Suche beteiligten Robotern.

Abbildung 2.2 stellt das Anwendungsszenario schematisch dar. Die Roboter tauschen untereinander und mit der Kontrollstation permanent relevante Daten aus. Dadurch entsteht innerhalb der Gruppe ein globales Wissen über das Katastrophengebiet. Durch das Wissen über die Position der anderen Roboter kann jeder Roboter selbstständig seine Suchstrategie an die aktuelle Situation anpassen. Die Sicherung des globalen Wissens und dessen Konsistenz ist eine Schlüsselfunktion für eine erfolgreiche Kollaboration der Roboter. Im Falle eines kompletten technischen Ausfalls eines Roboters bleiben dadurch das globale Wissen und seine bisher ermittelten Daten erhalten. Auf den Ausfall können die verbleibenden Gruppenmitglieder mit einer Anpassung ihrer Suchstrategie reagieren.

Die Aufgabe der Sucheinheiten ist es, das Gebiet vollständig und flächendeckend zu erkunden. Üblicherweise wird für die Suche die so genannte W-Formation genutzt (siehe Abbildung 2.2). Dabei befindet sich jeder Roboter (außer denen im Randbereich) mit mindestens zwei Robotern in Kommunikationsreichweite. Das Suchgebiet ist in ein Suchraster unterteilt. Im Verlauf der Durchsuchung des initialen Gitters kann die eingenommene Suchformation häufig nicht aufrechterhalten werden. Durch örtliche Gegebenheiten wie Hindernisse, Bewuchs, Geröll, Schutt usw., muss die Suchstrategie kontinuierlich angepasst werden. Dadurch besteht die Gefahr, dass die Kommunikationsverbindungen zwischen benachbarten Robotern abreißen. Fällt die Kommunikation zu einem Roboter aus, wird zunächst eine allgemeine Fehlerbehandlung gestartet und im Anschluss, wenn nötig, die Suchstrategie der Gruppe entsprechend adaptiert. Unabhängig von der konkreten Ursache sind Funkausfälle und Ressourcenerschöpfung die häufigsten Fehlerquellen in mobilen kollaborativen Anwendungen. Durch diese sind die Konsistenz des globalen Wissens und der Gruppenzusammenhang gefährdet. Maßnahmen zum Erkennen von Fehlern, welche die Konsistenz des globalen Wissens beeinträchtigen, zu deren Behandlung bzw. Vorbeugung sind wiederum aufwändig und ressourcenintensiv und können somit selber zu einem Ausfall der Roboter beitragen.

Im Laufe der Suche kann es notwendig werden, dass die Gruppe sich in Teilgruppen aufteilt, welche unabhängig voneinander agieren und z. B. in unterschiedlichen Suchgebieten oder mit unterschiedlichen Suchaufträgen im selben Gebiet agieren. Dies ist von Vorteil, wenn im Laufe der Suche sich Teile des Suchgebietes temporär als nicht durchsuchbar (z. B. aufgrund von lokalen Überschwemmungen, Bränden oder anderweitigen Hindernissen) erweisen. Dann ist es notwendig, dass mobile kollaborative Anwendungen die Fähigkeit haben, die Gruppe entsprechend der Anwendungsanforderungen semantisch zu teilen. Die semantische Teilung unterscheidet sich von einem Gruppenaustritt und einer anschließenden Gruppenneugründung dadurch, dass bereits erworbenes Wissen der betroffenen Teilnehmer erhalten bleibt und nur die mitgliederspezifischen Informationen angepasst werden.

Analog zur semantischen Teilung ist eine semantische Vereinigung zweier unabhängiger Gruppen wünschenswert. So können unabhängige Suchgruppen in einer gemeinsamen Gruppe vereinigt und dabei das jeweils erworbene Wissen in der neuen Gruppe aggregiert werden. Als ein analoges Beispiel für diese beiden Operationen können die Operation *branch* und *merge* des Git-Versionskontrollsystems[1] verstanden werden. So teilt die Git-Operation *branch* das gemeinsame Wissen einer Entwicklergruppe in zwei unabhängige Entwicklungspfade (analog zur semantischen Teilung der Gruppe) und die Git-Operation *merge* vereinigt zwei bis zu diesem Zeitpunkt unabhängige Entwicklungspfade zu einem gemeinsamen, wobei das Wissen zwischen beiden Teilpfaden im neuen Pfad vereinigt wird (analog zur semantischen Vereinigung).

In Katastrophengebieten kann die Kommunikation oft nur eingeschränkt über infrastrukturbasierte Medien realisiert werden. Aus diesem Grunde sind die Roboter zusätzlich mit Ad-hoc-Kommunikationsfähigkeiten ausgestattet. Sofern jedoch Netze mit einer höheren Bandbreite und/oder Übertragungskapazität als im Ad-hoc-Netz verfügbar sind, werden diese bevorzugt genutzt. Durch verschiedene Gründe, wie Mobilität der Roboter, Routenanpassungen, Kommunikationsausfälle, sich ändernde Gruppenzusammensetzungen oder Störeinflüsse aus der Umwelt ändert sich die Güte der Kommunikationsverbindungen zwischen den Robotern kontinuierlich. Um trotzdem die Kollaboration zu sichern, ist es notwendig, dass sich die Gruppe permanent an die sich wechselnden Eigenschaften anpasst und diese in ihrer Kommunikation berücksichtigt.

Um die Arbeit der Roboter zu gewährleisten, ist aus technischer Sicht eine Reihe von Randbedingungen zu beachten:

1. *Netzabstraktion*: Im Katastrophengebiet steht nicht immer eine einheitliche Netzversorgung zur Verfügung. Die Roboter kommunizieren untereinander über ein Ad-hoc-Netz. Sofern ein stationäres Netz vorhanden ist, welches eine bessere Dienstgüte aufweist, soll dieses verwendet werden. Für die einheitliche, transparente Kommunikation über Ad-hoc-basierte und stationäre Netze wird eine Netzabstraktion benötigt, welche das Routing und die Nachrichtenweiterleitung zwischen den verfügbaren Netzen regelt. Dadurch wird eine Kommunikation zwischen allen Gruppenmitgliedern möglich. Die Kommunikation innerhalb der Gruppe ist dadurch sicherzustellen, dass die Roboter innerhalb der Suchformation gegebenenfalls die Weiterleitung der Nachrichten übernehmen (im Sinne von Relaypeers).

2. *Lokalisierung*: Voraussetzung für den Gruppenaufbau ist ein Lokalisierungsmodul, das die kooperierenden Einheiten und die *Internet Protocol* (IP)-Adresse der Roboter ermittelt. Die Funktion der Lokalisation muss sowohl in stationären als auch in Ad-hoc-Netzen gesichert sein. Zu den Aufgaben gehört auch das Wiederauffinden von verlorenen Robotern, um diese wieder in die Gruppe zu integrieren.

3. *Gruppenorientiertes Kommunikationssystem*: Für die mobile Kollaboration, wird ein gruppenorientiertes Kommunikationssystem benötigt. Dieses stellt das globale Wissen der Gruppe bereit und sichert dessen Konsistenz. Dieses muss sowohl in stationären als auch in Ad-hoc-Netzen eingesetzt werden können. Des Weiteren

[1] Siehe https://git-scm.com/doc

muss es sowohl stationäre als auch nomadische und mobile Gruppenmitglieder unterstützen. Die meisten gruppenorientierten Kommunikationssysteme entfernen Mitglieder aus der Gruppe, die Nachrichten nicht bestätigen bzw. nicht wie erwartet reagieren. Dadurch wird das globale Wissen der Restgruppe gesichert [19, 26]. Dabei gehen jedoch sämtliche anwendungs-, zustands- und ressourcenbezogenen Informationen für das entfernte Gruppenmitglied verloren. Es muss erneut explizit in die Gruppe eingeladen werden, nachdem die Kommunikationsverbindung wiederhergestellt wurde. Damit ist ein erheblicher Re-Synchronisationsaufwand verbunden. Bisher nicht bekannt sind Wiedereintrittsoperationen, welche es einem Gruppenmitglied erlauben, sich nach einer kurzeitigen Kommunikationsunterbrechung ohne die genannten Informationsverluste erneut in die Gruppe zu integrieren.

4. *Gruppentopologie und Wartung*: Das gruppenorientierte Kommunikationssystem organisiert die Mitglieder innerhalb einer Gruppentopologie. Diese definiert die Kommunikationsbeziehungen zwischen den einzelnen Mitgliedern. Somit beeinflusst die Gruppentopologie den Aufwand jedes Mitglieds bei der Nachrichtenverteilung sowie die Rollen der einzelnen Teilnehmer dabei. Häufige Gruppenein- und -austritte aufgrund von Churns, fehlender Netzabdeckung oder Ressourcenerschöpfung destabilisieren die Gruppentopologie und erhöhen die Wahrscheinlichkeit für Kommunikationsfehler, weil die Mitglieder innerhalb der Topologie zunehmend ungleichmäßig verteilt sind. Dadurch kann es zu einer Überlastung einzelner Kommunikationsverbindungen oder deren Zusammenbruch kommen. Des Weiteren ändern sich durch die Mobilität der Roboter und die damit einhergehenden Positionsänderungen die jeweiligen Verbindungseigenschaften (z. B. Bandbreite, Latenz, Paketverlustrate, Position innerhalb des Ad-hoc-Netzes). Das kann zu einer nicht mehr optimalen Gruppentopologie führen, weshalb der Aufwand für die Sicherung der Konsistenz des globalen Wissens deutlich steigt. Daher benötigt das gruppenorientierte Kommunikationssystem einen *Wartungsdienst*, welcher periodisch aktiviert wird, um die Zuverlässigkeit der Topologie zu erhöhen und diese entsprechend der aktuellen Gruppen- und Netzsituation zu optimieren.

5. *Automatischer Netzwechsel*: Die automatische Wahl des optimalen Netzes und der anschließende Wechsel dorthin kann durch ein lokales Handover-Modul erfolgen. Es ermöglicht die Übertragung der Kommunikationsbeziehungen aus dem aktuell genutzten Netz in das Zielnetz, idealerweise ohne die Kommunikation zu unterbrechen oder zu stören. Das Handover-Modul wird durch die Überwachung spezifizierter Parameter aktiviert, z. B. Signalstärke, Datenübertragungskapazität sowie des bei der Nutzung entstehenden Energieverbrauches.

6. *Adaption an die Netzbedingungen*: Klassische gruppenorientierte Kommunikationssysteme verwenden zumeist fest definierte Zeitschranken für das Erkennen von Kommunikationsfehlern. Durch die sich stetig ändernden Eigenschaften der genutzten Netze, den Wechsel zwischen verschiedenen Netzen sowie die Asynchronität der gruppenorientierten Kommunikation ist das Erkennen von Kommunikationsfehlern im skizierten Anwendungsszenario jedoch eine komplexe Aufgabe. Das Verwenden fest definierter Antwortzeiten zur Fehlererkennung birgt das Risiko falscher Annahmen in sich. Verfrühte und damit unnötige Sendewiederholungen führen zu einem erhöhten Energieverbrauch; längere Warteperioden aufgrund falscher Schwellwerte erhöhen die Latenz. Dies kann im äußersten Fall fälschlicherweise zu Gruppenausschlüssen von korrekt agierenden Robotern führen. Generell stellt die zeitliche Koordination einer Gruppe in dezentralen asynchronen Systemen hohe Anforderungen. Basis für eine Koordination – beispielsweise für die genannte Erkennung von Kommunikationsstörungen – sind gruppenweit adaptierte Schwellwerte. Im Gegensatz zu einer bidirektionalen Kommunikation kann im Anwendungsszenario keine direkte Latenzmessung zur Schwellwertadaptierung genutzt werden, da keine zentrale Instanz existiert. Daher ist ein verteilter Adaptionsalgorithmus notwendig, welcher unter Nutzung des globalen Wissens die notwendigen Schwellwerte adaptiert.

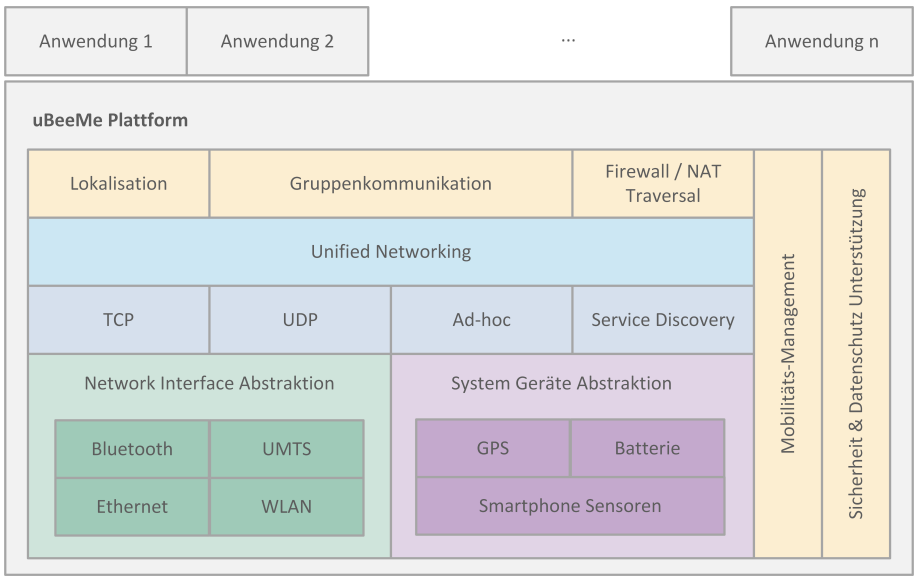


Abbildung 2.3: Aufbau der uBeeMe-Plattform

Zusammenfassend ist festzustellen, dass bereits das dargestellte kleine Anwendungsszenario einer mobilen kollaborativen Anwendung eine Vielzahl von Anforderungen an deren technische Umsetzung stellt. Die dabei zu berücksichtigenden technischen Rahmenbedingungen betreffen unterschiedliche Fachgebiete. Deshalb ist es zielführender, mobile kollaborative Anwendungen auf Basis einer Plattform zu realisieren, welche Lösungen für die einzelnen Anforderungen unter Berücksichtigung der oben kurz umrissenen Rahmenbedingungen bietet. Im folgenden Abschnitt sollen Funktionen einer solchen Plattform vorgestellt sowie fehlende Funktionen identifiziert werden.

2.3 Notwendige Basisfunktionen einer Plattform für die Unterstützung mobiler kollaborativer Anwendungen

Plattformen für die Umsetzung von mobilen kollaborativen Anwendungen stellen die notwendigen Basisfunktionen bereit. Beispiele für diese Art von Plattformen sind:

- *Mobile Collaboration Architecture* (MoCA) [186],
- *Spontaneous Virtual Networks* (SpoVNet) [34] und
- *Ubiquitous Mobile Environment* (uBeeMe) [96].

Abbildung 2.3 zeigt exemplarisch den Aufbau der uBeeMe-Plattform. Im Folgenden sollen die sechs wesentlichen Basisfunktionen dieser Plattform vorgestellt werden.

1. *Netzarten.* Das Konzept der mobile kollaborative Anwendungen ist nicht auf eine bestimmte Netzart beschränkt. Vielmehr sind Anwendungsbeispiele aus allen Netzarten bekannt, von zellbasierten und Weitverkehrsnetzen (engl. *Wide Area Networks* (WANs)), über lokale Netze (engl. *Local Area Networks* (LANs)) bis hin zu Nahfunknetzen wie Bluetooth [36] oder ZigBee [230].

    - Die MoCA-Plattform ist nur für die Verwendung in infrastrukturbasierten drahtlosen Netzen entworfen worden, konkret für IEEE 802.11-*Wireless Local Area Networks* (WLANs) [117]. Eine Portierung für zelluläre Netzprotokolle wie *General Packet Radio Service* (GPRS) [76] oder *Universal Mobile Telecommunications System* (UMTS) [1] wurde angedacht, aber nicht realisiert [186].

- Die SpoVNet-Plattform unterstützt neben einer IEEE 802.11-WLAN unter anderem auch Ethernet oder zellbasierte Netze wie UMTS [34] sowie Nahfunktechniken wie Bluetooth.
- Die uBeeMe-Plattform wurde nicht für eine konkrete Netztechnologie entworfen und unterstützt sowohl infrastrukturbasierte drahtlose Netze, drahtgebundene Netze als auch zellulare und Ad-hoc-Netze [96].

Bei Verwendung eines zellbasierten Kommunikationsnetzes ist eine Verwendung der MoCA-Plattform in zum geschilderten Anwendungsbeispiel vergleichbaren Anwendungsszenarien ohne umfangreiche Anpassungen und Erweiterungen der Plattform nicht möglich. Dagegen wären die Plattformen uBeeMe und SpoVNet direkt einsetzbar. Für die Nutzung von Ad-hoc-Netzen wären für alle Plattformen Anpassungen bzw. Erweiterungen notwendig.

2. *Netzabstraktion.* In allen Ansätzen werden die Netzschnittstellen gekapselt und die Daten unabhängig von ihrer Herkunft in der Gruppe verteilt.

   - In MoCA erfolgt dies auf Anwendungsebene [118], wobei ein Proxy-Server die Kapselung der Schnittstellen zur kollaborierenden Gruppe übernimmt und eingehende Daten an die Gruppe weiterleitet [186].
   - In der SpoVNet-Plattform wird oberhalb der Transportschicht [118] ein sogenanntes Kommunikations-Underlay zwischen allen Knoten etabliert. Darüber werden Daten aus den unterschiedlichen Quellnetzen, unabhängig von der konkreten Netztechnologie, an die Empfänger weitergeleitet [34].
   - In der uBeeMe-Plattform wird dieses umfassende virtuelle Netz – *Unified Networking* genannt – im Gegensatz zu SpoVNet nicht zwischen allen Plattformnutzern aufgespannt, sondern nur zwischen den Mitgliedern der kollaborativen Anwendung [97].

   Alle Ansätze realisieren so einen weitgehend transparenten und einheitlichen Zugang zu verschiedenen Teilnetzen. Konzeptionell können auch Ad-hoc-Netze mit infrastrukturbasierten Netzen gekoppelt und eine sichere Paketweiterleitung zwischen beiden realisiert werden. Aus Energie- und Sicherheitsperspektive ist beim SpoVNet-Ansatz die große Anzahl der in die Paketweiterleitung involvierten Zwischensysteme als nachteilig zu betrachten. Auch die Art und Weise der für die Underlay-Konstruktion notwendigen Informationsgewinnung und -verteilung wirken sich negativ auf die Energiebilanz von SpoVNet aus. Demgegenüber sind die lokalen Ansätze von uBeeMe und MoCA eher mit klassischen Routingverfahren zu vergleichen, was positiv für deren energetische Beurteilung ist.

3. *Mobilitätsmanagement.* Die Abdeckung des Einsatzgebietes mit unterschiedlichen Netzen und die Mobilität der Nutzer zwingen häufig zu Netzwechseln. Idealerweise wird dem Nutzer einer solchen Plattform jedoch ein virtueller Kommunikationskanal zur Verfügung gestellt, welcher Netzwechsel und IP-Adressänderungen verbirgt und diese möglichst nahtlos, d. h. ohne Kommunikationsunterbrechung, realisiert.

   - In MoCA muss sich der mobile Teilnehmer für einen Netzwechsel bei den „*nächstgelegenen*“ Proxys an- und abmelden. Ermittelt wird dieser „nächstgelegene“ Proxy über eine Reihe von MoCA-Diensten, welche als Infrastrukturdienste über das Netz verteilt sind und durch den mobilen Teilnehmer erreichbar sein müssen [186].
   - SpoVNet realisiert den Zugang zum Kommunikations-Underlay durch Nutzung von *Multicast-DNS* (mDNS) [52], IP-Multicast [63, 66] und dem Bluetooth Service Discovery Profile[35].[2] Die Adressierung und das Routing auf Basis von *Identifikationsnummern* (IDs) unabhängig von der konkret genutzten Transportadresse werden im Kommunikations-Underlay ausgeführt. Dieses Adressierungsmuster wird auch *Address-Identifier-Split* [34] genannt.

[2] vgl. [115]

- In der uBeeMe-Plattform ist in erster Linie ein Mobilitätsmanagementdienst [178] für den nahtlosen Netzwechsel (engl. Handover) verantwortlich. Dieser unterstützt auch vertikale Handover zwischen unterschiedlichen Netztechnologien. Darüber hinaus werden auch Firewall- und *Network Address Translation* (NAT)-Traversalfunktionen[3] eingesetzt, welche bei der uBeeMe-Plattform als eigenständige Dienste ausgebildet sind. Das uBeeMe-Mobilitätsmanagement geht über Konzepte wie Mobile IP [200] hinaus. Es scannt die Umgebung kontinuierlich nach nutzbaren Netzen und bewertet diese. Entsprechend der Nutzerdienstgüteanforderungen wie benötigte Datenübertragungskapazität, Erreichbarkeit, Nutzungsentgelte und maximale Verzögerung wird aus den zur Verfügung stehenden Netzen das optimale ausgewählt. Die aktuellen Verbindungen werden auf dieses übertragen, ohne die aktuelle Kommunikation zu unterbrechen. Eine weitere Funktion des uBeeMe-Mobilitätsmanagements ist die Prädiktion der nutzbaren Übertragungskapazitäten bekannter Kanäle [167]. Durch diese kann die mobile kollaborative Anwendung den Down- oder Upload größerer Datenmengen zeitlich variieren, wenn beispielsweise durch die Prädiktion im virtuellen Kanal Kapazitätsschwankungen vorhergesagt werden.

Insgesamt ist festzustellen:

- Das Mobilitätsmamangement von MoCA ist stark abhängig von Proxy-Diensten, welche ausgebracht, betrieben und erreichbar sein müssen. Daher ist eine Verwendung von MoCA in Anwendungsfällen wie dem Beispielszenario nur sehr eingeschränkt möglich.
- SpoVNet ermöglicht eine mobilitätsunabhängige Kommunikation auf Basis von SpoVNet-spezifischen IDs nur zwischen Endgeräten, welche Teil des Underlays sind. Damit sind SpoVNet-basierte Anwendungen nicht kompatibel zu bestehenden IP-basierten Anwendungen und begrenzen den potentiellen Teilnehmerbereich erheblich.
- Das uBeeMe-Mobilitätsmanagement ist konzeptionell mit Verfahren aus den zellbasierten Netzen vergleichbar. Adressänderungen aufgrund eines Netzwechsels werden der Anwendung ebenso signalisiert wie erwartete oder erkannte Änderungen an den *Quality of Service* (QoS)-Eigenschaften des Kommunikationskanals. Ziel des uBeeMe-Mobilitätsmanagements ist die vollständige, unterbrechungsfreie Nutzung eines virtuellen Kommunikationskanals.

4. *Datenschutz und Datensicherheit.* Im mobilen Umfeld sind Themen wie Datensicherheit, Authentizität, Autorisation oder Datenschutz von besonderer Bedeutung.

   - In MoCA werden sicherheitsrelevante Operationen in erster Linie zwischen dem mobilen Teilnehmer und dem MoCA-Proxy ausgeführt. Dieser ist verantwortlich für die Nutzerauthentifizierung sowie die Verschlüsselung [186] der zu übertragenden Daten.
   - Die SpoVNet-Plattform gewährleistet primär die Transportsicherheit. Die Übertragung innerhalb des Kommunikations-Underlay kann mittels *Transport Layer Security* (TLS) [68] oder *Internet Protocol Security* (IPSec) [127, 129, 153] sowohl asymmetrisch als auch symmetrisch [34] verschlüsselt werden.
   - Ein vergleichbares Schutzniveau wird auch durch die uBeeMe-Plattform realisiert. Über die genannten Schutzziele hinaus wird jedoch auch eine zuverlässige Verschlüsselung der Kommunikation in einem gruppenorientierten Kommunikationssystem realisiert, was beispielhaft von Liu und König in [145] gezeigt wird.

[3] z. T. auf Deutsch als NAT-Durchdringungsmethoden übersetzt

Insgesamt gilt: Durch die verfügbare Transportverschlüsselung ist die Verwendung der Plattformen SpoVNet und uBeeMe vorteilhafter gegenüber der MoCA-Plattform. Besteht der Bedarf für eine gruppenweite Authentifizierung und Ende-zu-Ende-Verschlüsselung, so bietet nur die uBeeMe-Plattform eine verfügbare Lösung.

5. *Nutzerlokalisierung.* Die Nutzerlokalisierung ist eine wesentliche Komponente einer mobilen kollaborative Anwendung, da sie notwendige Funktionen für die Initialisierung der Kommunikation zwischen den Beteiligten bereitstellt [96, 133]. Sie nutzt zumeist intensiv die Firewall- und NAT-Traversalfunktionen der Plattformen, um die Transportadresse der gesuchten Kommunikationspartner zu bestimmen.

   - In der MoCA-Plattform werden für diesen Zweck spezielle Discovery-Server etabliert, welche Informationen wie Namen, Eigenschaften oder Transportadressen von Nutzern und Diensten anbieten [186].
   - In SpoVNet treten mobile Teilnehmer dem Kommunikations-Underlay explizit über so genannte Bootstrap-Server [34] bei. Diese verteilen eine Liste mit Einstiegspunkten in das Kommunikations-Underlay. Somit kann man hier nur dem Kommunikations-Underlay beitreten. Eine Einladung externer Kommunikationspartner in eine SpoVNet-basierte kollaborative Anwendung ist nicht vorgesehen. Das Grundprinzip der Lokalisation ähnelt damit dem des *'NICE is the Internet Cooperative Environment'* (NICE)-Gruppenkommunikationsprotokolls [19].
   - Die uBeeMe-Plattform lokalisiert Nutzer mittels des XMPP-Protokolls [188]. Den lokalisierten Kommunikationspartnern wird eine unveränderliche ID zugeordnet, welche auf die jeweils ermittelte aktuelle Transportadresse abgebildet wird. Schwierig ist hier vor allem in Ad-hoc-Umgebungen die Bereitstellung der Lokalisation als verteilten Dienst, welcher standardisierte Signalisierungsprotokolle unterstützt [96]. Dafür nutzt uBeeMe einen Ansatz in Kombination mit der *Extensible Messaging and Presence Protocol* (XMPP)-Protokollerweiterung Serverless Messaging [187], welcher auf Zero-Conf, DNS-based Service Discovery und mDNS [12, 51, 205] beruht.

   Semantisch und funktional unterscheiden sich die verschiedenen Konzepte deutlich. Sowohl MoCA als auch SpoVNet nutzen vergleichbare Zugangskonzepte, welche einen freien Zugang zur kollaborativen Gruppe bzw. dem Underlay ermöglichen. Die uBeeMe-Plattform ist flexibler und unterstützt verschiedene Konzepte. So können sowohl offene als auch geschlossene Gruppen realisiert werden, welche auf Basis von Bootstrap- und Service-Discovery-Diensten oder einladungsbasiert umgesetzt werden.

6. *Gruppenorientierter Kommunikationsdienst.* Die letzte notwendige Komponente zur Realisierung mobiler kollaborativer Anwendungen ist ein gruppenorientierter Kommunikationsdienst, welcher das globale Wissen innerhalb einer Gruppe bereitstellt und dessen Konsistenz sichert.

   - MoCA stellt solch einen Dienst nicht bereit [186].
   - Der gruppenorientierte Kommunikationsdienst der SpoVNet-Plattform basiert auf dem NICE-Protokoll [19], welches nicht für mobile Anwendungsszenarien konzipiert und somit nicht für den Einsatz im Zusammenhang mit mobilen kollaborativen Anwendungen geeignet ist.
   - Diese Arbeit konzentriert sich auf Fragen der Konzeption und Umsetzung eines gruppenorientierten Kommunikationsdienstes für mobile kollaborative Anwendungen, welcher beispielsweise in der uBeeMe-Plattform verwendet werden könnte. Dabei wird unterstellt, dass die übrigen Funktionen und Dienste, die für die Umsetzung einer mobilen kollaborativen Anwendung notwendig sind, durch die vorgestellten Plattformen bereitgestellt werden.

2.4 Anforderungen an die Gestaltung gruppenorientierter Kommunikationsdienste in mobilen kollaborativen Anwendungen

Die Anforderungen an die Gestaltung eines gruppenorientierten Kommunikationssystems leiten sich aus dem vorgestellten Anwendungsszenario und den notwendigen Basisfunktionen einer mobilen kollaborativen Anwendung ab. Im Mittelpunkt steht dabei die Frage, wodurch eine mobile kollaborative Gruppe charakterisiert ist und welcher Eigenschaften sie bedarf.

1. *Kleinteilige geschlossene Gruppe für spontane Kollaboration*

   Die wesentlichste Eigenschaft der in Abschnitt 2.1 eingeführten Charakterisierung mobiler kollaborativer Anwendungen ist die Geschlossenheit der Gruppe. Alle Mitglieder kennen sich untereinander einschließlich ihrer relevanten Zustands- und Statusinformationen. Nur auf dieser Basis lässt sich die enge Kollaboration realisieren. Ohne das Wissen, wer zu der kollaborativen Gruppe gehört, können sich beispielsweise die Roboter nicht auf eine gemeinsame Suchstrategie einigen.

   Ausgehend von dem Anwendungsbeispiel soll der Schwerpunkt des gruppenorientierten Kommunikationssystems bei der Unterstützung kleiner Gruppen bis zu einhundert Mitgliedern liegen, deren Kollaboration spontan und kleinteilig ist (im Sinne von: häufiger Austausch von geringen Datenmengen bzw. inkrementelle Aktualisierungen). Dies entspricht auch den Schwerpunkten anderer bekannter Anwendungsszenarien wie mobile Spiele, Videokonferenzsysteme oder Car-to-X-Kommunikation.

2. *Gemeinsames Wissen gleichberechtigter Gruppenmitglieder*

   Eine direkte Konsequenz aus der geschlossenen Gruppe als Basis für die Kollaboration ist das gemeinsame Wissen innerhalb der Gruppe. Dieses muss zu jeder Zeit, auch im Falle einer Kommunikationsunterbrechung zwischen dem Teilnehmer und der Restgruppe, konsistent und lokal abrufbar sein. Das gemeinsame Wissen entsteht aus einer engen Kollaboration im Sinne einer starken Synchronisation und häufigen Kommunikation, d. h. die Nutzer treten der Gruppe bei, um sich mit dieser aktiv auszutauschen und Wissen mit dieser zu teilen. Dabei kann ein Gruppenmitglied sowohl Sender als auch Empfänger von Wissen sein, wodurch die Kommunikation unabhängig von einem konkreten Schema oder der zugrundeliegenden Netzsituation bzw. Netzstruktur ist. Verkürzt lässt sich der Wissensaustausch innerhalb der Gruppe mit einer *one-to-all*-Semantik (im Sinne von „einer an alle“) beschreiben. Daraus folgt, dass innerhalb der geschlossenen Gruppe alle Mitglieder gleichberechtigt sind.

3. *Robust gegenüber Verbindungsabbrüchen und deren zeitnahe Behandlung*

   In mobilen Umgebungen kann es jederzeit zu Verbindungsabbrüchen kommen, von denen entweder einzelne Nutzer oder eine Reihe von Nutzern betroffen sind. Diese sind zumeist jedoch nur von kurzer Dauer. Gründe für diese Kommunikationsunterbrechungen sind unter anderem:

   - die Nutzermobilität,
   - eine fehlende Netzabdeckung,
   - physikalische Effekte bei der Funkübertragung wie Streuung und Reflexion, Abschattung, Signal-Fading,
   - Überlastung oder Ausfall von Zwischensystemen,
   - Paketkollisionen und Überlastungen im Übertragungskanal sowie
   - Ressourcenerschöpfung oder sonstige Fehler in den Endgeräten.

   Eine Unterbrechung der Kommunikation gefährdet generell die Konsistenz des globalen Wissens. Daher müssen durch die mobile kollaborative Anwendung

Verbindungsabbrüche zeitnahe erkannt und behandelt werden. Da vor allem mit Verbindungsabbrüchen aufgrund der Nutzermobilität häufiger zu rechnen ist, muss deren Behandlung möglichst nahtlos erfolgen. Kurzzeitige Unterbrechungen dürfen keinen Einfluss auf die Gruppenzusammensetzung oder die Konsistenz des globalen Wissens haben. Bei einer signifikanten Dauer der Kommunikationsunterbrechung können betroffene Teilnehmer aus der Gruppe entfernt werden. Ihre spätere Reintegration muss jedoch jederzeit gesichert sein.

Wird eine Unterbrechung der Kommunikationsverbindung erkannt, muss das gruppenorientierte Kommunikationssystem zwischen zwei Fällen unterschieden können:

- Verbindungsabbruch zu einem einzelnen Teilnehmer und
- Partitionierung der Gruppe in zwei oder mehr Teilgruppen auf Netzebene.

Da diese beiden Fälle einen signifikanten Einfluss auf das gemeinsame Wissen der Gruppe haben, muss das gruppenorientierte Kommunikationssystem für sie Fehlerbehandlungsmethoden bereitstellen und eine Reintegration der betroffenen Mitglieder ermöglichen.

4. *Optimierung für Endgeräte mit begrenzten Ressourcen*

   Da in mobilen kollaborativen Anwendungen sowohl mobile als auch stationäre Endgeräte zum Einsatz kommen, muss das gesuchte gruppenorientierte Kommunikationssystem in beiden Geräteklassen einsatzfähig sein. Weil mobile Endgeräte im Allgemeinen in ihrer Ressourcenausstattung im Vergleich zu ihren stationären Pendants deutlich begrenzter sind, muss sich die Auslegung der einzelnen Funktionen jeweils an den Anforderungen der mobilen Endgeräte bzgl. Speicher- und Energieverbrauch orientieren. Insbesondere ist dabei die Anzahl der zu übertragenden Nachrichten je Teilnehmer einhergehend mit einer gleichmäßigen und fairen Lastverteilung zwischen allen Gruppenteilnehmern zu optimieren. Im Falle einer notwendigen Abwägung zwischen der Sicherung der Konsistenz des globalen Wissens und der Reduktion des Ressourcenverbrauches ist jedoch immer der Konsistenzsicherung Vorrang zu geben, da dies die originäre primäre Zielstellung des gruppenorientierten Kommunikationsdienstes ist, die ressourcenorientierte Optimierung dagegen nur eine sekundäre Zielstellung.

5. *Integration in existierende Systeme mittels eines dezentralen Ansatzes*

   Ziel der Entwicklung soll die Unterstützung beliebiger, existierender stationärer, mobiler und zellulärer Netze sein, sofern diese eine IP-basierte Kommunikation ermöglichen. Eine Integration in bestehende Netzstrukturen, sowohl in infrastrukturbasierte als auch Ad-hoc-basierte Netze, ist somit gewünscht. Prinzipiell soll die Lösung unabhängig von einer speziellen Netzart sein. Um eine schnelle Verbreitung der zu erreichenden Lösung sicherzustellen, wird gefordert, dass das gesuchte gruppenorientierte Kommunikationssystem keine zusätzlichen Abhängigkeiten von einer für den Betrieb notwendigen Infrastruktur erzeugt. Neben der angestrebten schnellen Verbreitung sprechen auch ökonomische Aspekte gegen eine dedizierte Infrastruktur. Zusätzlich stellen zentrale Elemente potentielle Fehlerpunkte dar, welche durch zusätzlichen Aufwand gesichert werden müssten. Unter Berücksichtigung dieser verschiedenen Szenarien ist ein dezentraler, vollständig verteilter und infrastrukturunabhängiger Architekturansatz vorteilhaft.

6. *Kontinuierliche Anpassung der Gruppe an die sich ändernden Netz- und Gruppenbedingungen*

   Entsprechend der Forderung nach einem dezentralen Ansatz zur Realisierung der kollaborativen Gruppe werden die Kommunikationsverbindungen zwischen den einzelnen Gruppenmitgliedern über eine Gruppentopologie definiert. Diese ist so zu gestalten, dass sowohl der Gruppendynamik, d. h. den potentiell häufigen Gruppenein- und -austritten, als auch mobilitätsinduzierten Verbindungsabbrüchen Rechnung getragen wird und zudem eine faire Rollen- und Lastverteilung

zwischen den einzelnen Gruppenmitgliedern unter Berücksichtigung der individuellen Ressourcensituation erreicht wird. Um die Gruppentopologie an die sich ständig ändernden Kommunikations-, Gruppen- und Ressourcenbedingungen anzupassen, ist es erforderlich, die Topologie der Gruppe entsprechend der aktuellen Situation regelmäßig zu optimieren und gegebenenfalls umzustrukturieren. Dabei muss die Konsistenz des globalen Wissens zwischen den Gruppenmitgliedern erhalten bleiben und ein Auseinanderdriften der Gruppe verhindert werden.

7. *Adaptierende, verteilte gruppenweite Koordination*

   Grundlage für Konsistenzsicherung und Wartung ist eine gruppenweite Koordination. Durch den verteilten Ansatz der mobilen kollaborativen Anwendungen kann dies im Allgemeinen nicht über eine zentrale ausfallgefährdete Instanz erfolgen. Basis einer dezentralen Koordination ist die Verknüpfung globaler Ereignisse mit einheitlichen lokalen Handlungsweisen. Dadurch werden gruppenweit Operationen koordiniert, wie beispielsweise der Ausschluss eines Mitgliedes.

   Aufgrund der einer mobilen kollaborativen Gruppe inhärenten Asynchronität können zur Initialisierung der für diese Koordination notwendigen lokalen Timer keine fest definierten Zeitintervalle verwendet werden. Daher ist es notwendig, die Zeitintervalle auf Basis einer feingranular verteilten Schätzung in Abhängigkeit von der aktuellen Gruppenzusammensetzung und Lastsituation im Netz zu bestimmen. Auch hier besteht wiederum ein Zusammenhang zwischen dem globalen Wissen und dem lokalen Handeln, welches das prägende Gestaltungsprinzip des gruppenorientierten Kommunikationsdienstes sein muss. Dieses Prinzip kann kurz als globales Wissen, lokales Handeln (engl. *global knowledge, local decisions*) umschrieben werden.

8. *Anwendungsgetriebene Gruppenteilung bzw. Gruppenvereinigung*

   Wie im Anwendungsbeispiel gezeigt, kann es für eine kollaborative Gruppe wünschenswert sein, diese semantisch in Teilgruppen aufzuteilen oder sich mit einer bis dahin unabhängigen Gruppe zu vereinigen. Diese anwendungsgetriebene Gruppenteilung bzw. Gruppenvereinigung ist von der durch Kommunikationsstörungen ausgelösten Gruppenpartitionierung und anschließender Wiedervereinigung zu unterscheiden. Solche Partitionierungen sollen automatisch durch das gruppenorientierte Kommunikationssystem erkannt und behoben werden, wohingegen die semantische Aufteilung bzw. semantische Vereinigung zweier Gruppen unter Berücksichtigung der Sicherung der Konsistenz des jeweils gemeinsamen Wissens explizit durch die Anwendung ausgelöst werden soll. Daher muss das gruppenorientierte Kommunikationssystem Operationen unterstützen, welche das semantische Aufteilen einer Gruppe in Teilgruppen und das Vereinigen unabhängiger Gruppen ermöglichen. Hierbei ist jeweils das gemeinsame Wissen abzugleichen und dessen Konsistenz zwischen allen Teilnehmern der neuen Gruppe(n) zu sichern.

Auf Basis dieser Anforderungen werden anschließend Ansätze für die Realisierung eines entsprechenden gruppenorientierten Kommunikationssystems diskutiert. Dazu wird ein Überblick über existierende Ansätze aus den unterschiedlichen Bereichen gegeben und deren Eignung bzgl. der Erfüllung der dargestellten Anforderungen geprüft.

Kapitel 3

Kollaborative Ansätze

Gruppenorientierte Kommunikationssysteme sind Kernkomponenten in kollaborativen Anwendungen. Sie werden häufig auch in anderen Anwendungsgebieten genutzt, z. B. in middleware-, cluster- oder datencenterbasierten Anwendungsszenarien. Ein breites Spektrum verschiedener Lösungsansätze mit unterschiedlichen Kommunikationsschemata, z. B. ISIS [33], JGroups [161], Pilot [149] und JEDI [64] ermöglicht die Kooperation von Teilnehmern in mobilen Netzen [90]. In diesem Kapitel wird ein Überblick über die wichtigsten Ansätze gegeben, die für die Realisierung von mobilen kollaborativen Anwendungen genutzt werden können. Bei ihrer Bewertung wird auf die in Abschnitt 2.4 definierten Anforderungen an mobile kollaborative Anwendungen Bezug genommen. Dabei ist zu beachten, dass keiner der in der Folge vorgestellten Ansätze alle Anforderungen vollständig erfüllt. Teilaspekte, wie die Unterstützung der Nutzermobilität, Konsistenzsicherung, Bereitstellung von kollaborativen Funktionen sowie Mechanismen zur Behandlung von dynamischen Änderungen der Gruppentopologie, sind jedoch in allen Ansätzen vorhanden. Sie unterscheiden sich aber in funktionaler und semantischer Sicht teilweise deutlich. Zur Vereinfachung der Diskussion wird für jeden Ansatz ein typischer Vertreter diskutiert, der die notwendigen Eigenschaften und Funktionen zur Unterstützung mobiler kollaborativer Anwendungen am deutlichsten charakterisiert. Dabei wird das im Abschnitt 2.2 vorgestellte Beispielszenario genutzt.

3.1 Herkömmliche Gruppenkommunikationssysteme und Derivate

Eine naheliegende Möglichkeit, Gruppenarbeit in mobilen Anwendungen zu unterstützen, besteht in der Nutzung von *Gruppenkommunikationssystemen*, die für kabelgebundene Netze entworfen wurden. Konzeptionell eng verwandt sind auch das *Application-Level Multicast* [57] sowie *Middlewaresysteme* und *Tupelräume*. Sie werden hier mit betrachtet.

3.1.1 Gruppenkommunikationssysteme

Gruppenkommunikationssystem (GKS) (engl. *Group Communication Systems*) werden für die Sicherung des gemeinsamen Wissens von geschlossenen Gruppen genutzt. Typische Beispiele sind ISIS [33], Transis [69], JGroups [161] und GCP [231]. Sie nutzen zumeist Paradigmen wie die *Virtuelle Synchronität* (VS) [33] oder den *Paxos-Algorithmus* [137], um eine gemeinsame Sicht auf den aktuellen Wissenszustand und die Gruppenzusammensetzung zu erzielen.

In offenen Gruppen können eine solche gemeinsame Sicht bzw. ein solches Wissen nicht garantiert werden. Für das Roboterbeispiel gehören dazu Informationen wie die aktuelle Position und der zurückgelegte Weg jedes Roboters, die aktuelle Suchstrategie der Gruppe, der Zustand jedes Roboters (Energievorrat, Bandbreite, Paketverlustrate, Signalqualität oder Beschädigungsgrad) oder die Beschreibungen der gesuchten Personen und Objekte. Tritt ein Roboter aus der kollaborativen Gruppe aus bzw. verliert er die Verbindung zu ihr, ändert dies die Gruppenzusammensetzung, über welche die übrigen Roboter informiert werden müssen. Generell ist ein globales Wissen immer

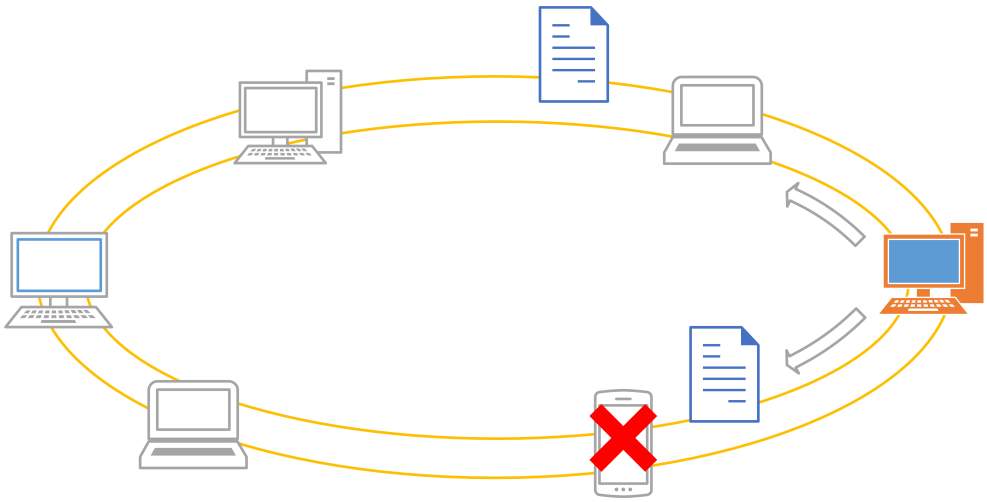


Abbildung 3.1: Kommunikation in einer GCP-Gruppe mit einem fehlerhaften Teilnehmer (nach [231])

dann notwendig, wenn (mobile) Gruppenteilnehmer gemeinsam Entscheidungen treffen oder Aktionen verteilt synchronisiert ausführen müssen, wie es mobile kollaborative Anwendungen erfordern.

Mobilität gehörte nicht zu den Entwurfskriterien herkömmlicher Gruppenkommunikationssysteme. Dadurch sind deren Einsatzmöglichkeiten in mobilen Umgebungen begrenzt. Reagieren die Roboter beispielsweise aufgrund von Verbindungsproblemen nicht protokollkonform, so wird bei herkömmlichen Gruppenkommunikationssystemen fälschlicherweise davon ausgegangen, dass dieser Roboter dauerhaft ausgefallen ist. Sie würden den betroffenen Roboter unverzüglich aus der Gruppe ausschließen. Diese Strategie wird u. a. in ISIS [31], JGroups [161], Spread [10], Apia [158] und GCP [231] angewendet. Diese Situation ist beispielhaft für das Gruppenkommunikationsprotokoll GCP in Abbildung 3.1 dargestellt. GCP ist ein Token-Protokoll mit einer logischen Doppelringstruktur zur Unterstützung von *Peer-to-Peer* (P2P)-Videokonferenzen. Der orange dargestellte Rechner ist der aktuelle Tokenhalter. Nur er darf Nachrichten an die Gruppe versenden. Er tut dies, indem er die Nachrichten in beide Richtungen auf dem Ring versendet. Der Rechner, der zuerst beide Nachrichten erhält, stoppt die Weiterleitung. Wenn, wie dargestellt, ein mobiler Teilnehmer aufgrund eines Verbindungsfehlers nicht erreichbar sein sollte, wird der Ring als unterbrochen angenommen, was die zuverlässige Datenübertragung innerhalb der Gruppe verhindert. Um den Ring wieder zu schließen, wird daher bei GCP der nicht erreichbare Teilnehmer unverzüglich aus der Gruppe ausgeschlossen.

Bei einem solchen Gruppenausschluss gehen i. d. R. alle kontextbezogenen Daten bezüglich des betroffenen Teilnehmers verloren. In dem Roboterszenario wäre also der Gruppe nicht mehr bekannt, welche Wege der ausgeschlossene Roboter bereits abgesucht hat. Wäre der Roboter zeitnah in der Lage, seine Kommunikationsprobleme zu beheben, so ist es ihm trotzdem nicht möglich, der Restgruppe wieder beizutreten. Stattdessen ist es notwendig, ihn explizit wieder in die Gruppe einzuladen und den Anwendungszustand vor dessen Ausscheiden wieder herzustellen (vergleichbar mit einem Rollback bei Datenbanken). Außerdem muss auch das Wissen des wieder eingetretenen Roboters aktualisiert werden, um ihn über Änderungen des Gruppenwissens bzw. des Suchstatus der Gruppe bis zum seinem Wiedereintritt zu informieren. Unabhängig von der konkreten Ausprägung und dem Umfang dieser Operationen impliziert dies einen nicht unerheblichen Aufwand für alle Gruppenmitglieder, da (1) ein umfangreicher Nachrichtenaustausch für die Synchronisation der einzelnen Operationen zwischen den beteiligten Robotern erforderlich und (2) die Kooperation innerhalb der Gruppe während des Teilnehmerausschlusses und der Wiedereinladung unterbrochen ist.

Diese Konsequenzen wirken sich umso gravierender aus, je höher die Wahrscheinlichkeit von mobilitätsbedingten Kommunikationsunterbrechungen ist. Da mobile Gruppen durch instabile Netzverbindungen mit temporären Kommunikationsausfällen charakterisiert sind [6], wird in mobilen Szenarien eine fehlerhafte oder ausbleibende Reaktion einzelner Gruppenteilnehmer in erster Linie durch Störungen der Netzverbindungen verursacht und nicht durch Störungen der Endsysteme. Spiker *et al.* zeigen in [201], dass dies vor allem für spontane Kommunikationstechniken gilt. Reagiert ein Roboter nicht innerhalb einer definierten Zeitspanne protokollkonform, ist es wahrscheinlicher, dass er aktuell keine Funkverbindung zur Gruppe hat, als dass er ausgefallen ist. Kim *et al.* [131] untersuchen die Verwendung von *Long Term Evolution* (LTE)-basierten Kommunikationsnetzen im Kontext der Gruppenkommunikation. Dabei zeigt sich, dass die Skalierbarkeit und die Latenzeigenschaften von Gruppenkommunikationsdiensten verbessert werden können, was jedoch wegen der im Allgemeinen nicht gegebenen Verfügbarkeit von LTE-Systemen keine Lösung für das Problem der temporären Kommunikationsausfälle darstellt.

Das dezentrale Gruppenkommunikationssystem *Atum* [105] strukturiert die Gruppe mittels Cluster, welche durch ein Overlay verbunden sind. Innerhalb eines Clusters werden Nachrichten über ein Replikationsprotokoll zuverlässig ausgetauscht, zwischen den Clustern über ein gerüchtebasiertes Protokoll. Im Gegensatz zu den oben genannten Protokollen werden fehlerhafte Knoten nicht ausgeschlossen, sondern gleichmäßig auf die Cluster verteilt. Wie die Gruppe reagiert (im Sinne von Abgleich globales Wissen, Overlayposition, etc.), wenn die fehlerhaften Knoten ihre Probleme beheben, ist nicht definiert. Damit zielt Atum eher auf die Verfügbarkeit der Gruppe als auf das konsistente Wissensmanagement zwischen den Gruppenmitgliedern.

Die Nutzung herkömmlicher Gruppenkommunikationssysteme für mobile kollaborative Anwendungen würde zu einer dauerhaft reduzierten Kollaborationsqualität zwischen den Teilnehmern führen, da Operationen wie Gruppenausschluss, Gruppeneintritt, Zustandstransfers usw. oft blockierend ausgeführt werden. Blockierend heißt in diesem Kontext, dass die Anwendung keine weiteren Operationen ausführen kann, bis die angefangene Operation beendet wurde. Zusätzlich ist ein erhöhter Ressourcenverbrauch durch den Nachrichtenaustausch für die Gruppenänderungen und den Zustandstransfer die Folge. Je höher die Churn-Rate bei der Kommunikation zwischen den Teilnehmern ist, desto stärker sinkt die Kollaborationsqualität bei gleichzeitig steigendem Ressourcenverbrauch. Das widerspricht den Anforderungen mobiler kollaborativer Anwendungen bezüglich eines optimalen Ressourcenverbrauchs der mobilen Endgeräte. Auf das Roboterszenarium bezogen hieße dies, dass die Roboter mehr Energie in die Kommunikation innerhalb der Gruppe investieren als in ihre eigentliche Aufgabe, die Suche nach vermissten Personen oder Objekten.

3.1.2 Application-Level Multicast

Wie von Chu *et al.* mit Narada [57] erstmals gezeigt, können Gruppenkommunikationssysteme auch unter Nutzung des *Application-Level Multicast* umgesetzt werden. Weitere Ansätze sind u. a. NICE [19], Gossamer [49], Yoid [85] und MTor [143]. All diese Ansätze zielen in erster Linie auf stationäre Teilnehmer ab und spannen ein Overlay zwischen ihnen in der Anwendungsebene auf. Die zugrundeliegenden Semantiken bzgl. Gruppeneigenschaften und Zuverlässigkeit unterscheiden sich jedoch. So besitzen beispielsweise die Gruppenmitglieder in Nice im Gegensatz zu den Mitgliedern einer Narada-Gruppe nur ein beschränktes Wissen über andere Mitglieder.

Die Overlays werden mittels unterschiedlicher Metriken erzeugt und gewartet, um das Overlay an sich dynamisch ändernde Netzbedingungen anzupassen. Verschlechtern sich beispielsweise einzelne Verbindungen zwischen den Robotern, so können sie durch robustere Verbindungen ersetzt werden, ohne die Suchanwendung in ihrer Funktion zu beeinträchtigen. Hauptziel aller Overlaymetriken ist die Reduzierung der Ende-zu-Ende-Verzögerung. Für das Routing innerhalb der Overlays wird beispielsweise das *Distance Vector Multicast Routing Protocol* (DVMRP) [211] genutzt [85]. Die dadurch entstehenden Topologien sind sehr robust gegenüber Knoten- und Verbindungsausfällen.

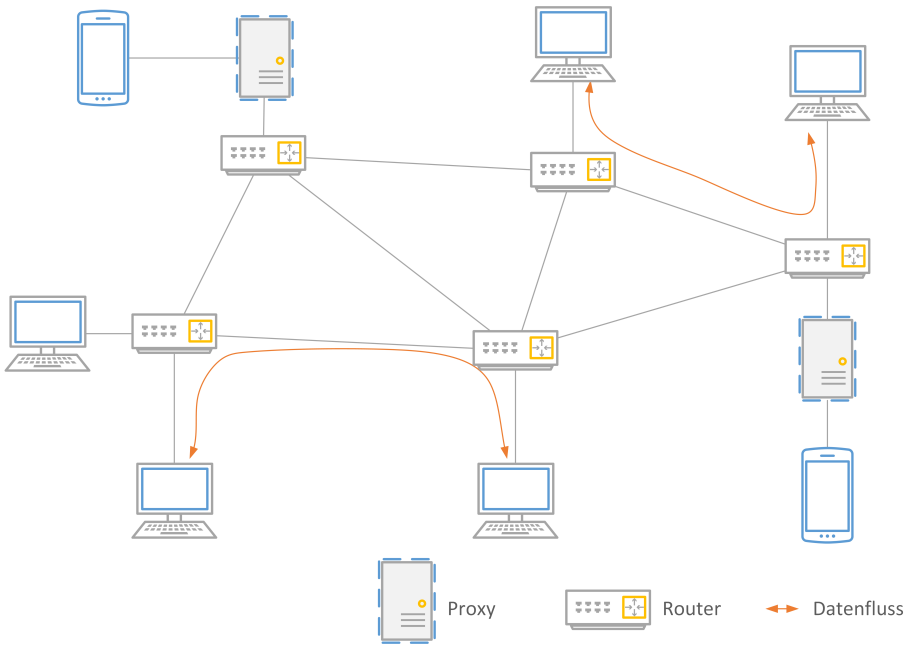


Abbildung 3.2: Systemarchitektur für Konferenzanwendungen im Internet (nach [58])

Häufige Gruppen Ein- und Austritte mobiler Teilnehmer wirken sich jedoch negativ auf deren Stabilität aus.

Um die Stabilität der Overlays zu verbessern, wurden einzelne Application-Level Multicastansätze (z. B. die Multicast Architektur von Chu *et al.* [58], Gossamer [49] und Yoid [85]) zu hybriden Proxyansätzen weiterentwickelt, die Stellvertreter für mobile Teilnehmer in die Gruppe einführen. Der Lösungsansatz Chu *et al.* [58] ist beispielhaft dafür und in Abbildung 3.2 dargestellt. Die mobilen Teilnehmer werden in die Gruppenkommunikation über je einen Proxy eingebunden, welcher als Stellvertreter agiert und von Bandbreiten- und Latenzschwankungen abstrahiert. Dabei handelt er für ein oder mehrere mobile Teilnehmer jeweils als vollständiges, virtuelles Gruppenmitglied. Der Proxy ist persistent über die gesamte Laufzeit in die Gruppe eingebunden und stellt somit auch einen Single-Point-of-Failure dar. Im Roboterszenario wäre mit diesem Lösungsansatz nur die stationäre Kontrollstation direkt mit der Gruppe verbunden. Die Roboter wären beispielsweise jeweils durch einen Proxy repräsentiert. Dies führt zu einer zweistufigen Kommunikation zwischen den stationären Teilnehmern und den Proxies sowie zwischen diesen und den Robotern, wie in Abbildung 3.3 für acht Roboter dargestellt (vereinfachte Darstellung gegenüber Abbildung 3.2).

Zusammenfassend ist festzustellen, dass sich Application-Level Multicastsysteme nicht für die Realisierung mobiler kollaborativer Systeme eignen. Die Bereitstellung der Proxies kann nur durch einen Infrastrukturprovider realisiert werden. Die Abhängigkeit von ihm widerspricht jedoch den Anforderungen kollaborativer Anwendungen. Darüber hinaus fehlt den Application-Level Multicastansätzen oft die Semantik einer geschlossenen Gruppe oder eines gemeinsamen Wissens. Die Anpassung an sich ändernde Netz- und Gruppenbedingungen ist nur eingeschränkt möglich. Weiter können häufige Kommunikationsunterbrechungen nur bedingt behandelt werden. Zwar kann die Gruppe einen mobilen Teilnehmer über den Proxyansatz erreichen, die Sicherung der Konsistenz des gemeinsamen Wissens jedoch wird blockiert, sobald die Verbindung zwischen dem mobilen Teilnehmer und dem Proxy abreißt.

3.1.3 Middlewareplattformen

Kollaborative Anwendungen könnten auch auf Basis einer der vielen Middlewareplattformen umgesetzt werden. Sie bieten viele interessante Vorteile wie Abstraktion von der gegebenen Netzinfrastruktur und Unterstützung von Selbstorganisation und Skalierbar-

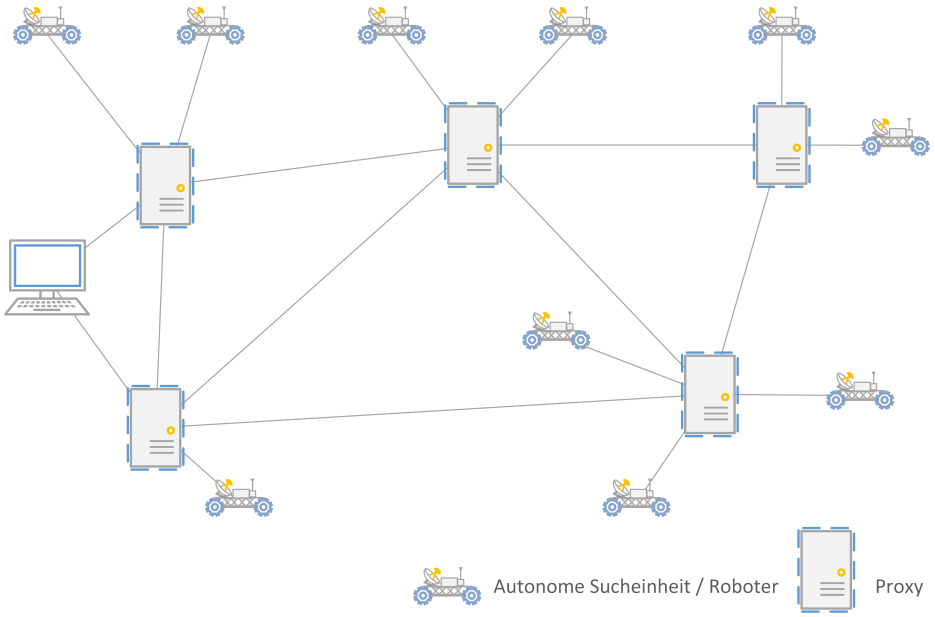


Abbildung 3.3: Roboterszenario unter Verwendung des Proxyansatzes

keit. Sie bestehen aber üblicherweise aus einer Vielzahl von Komponenten und Diensten, welche nicht alle im Kontext mobiler Kooperation benötigt werden. Aus den verfügbaren Lösungen eigenen sich vor allem die P2P-basierten Ansätze für mobile kollaborative Anwendungen, weshalb im Weiteren nur solche Ansätze betrachtet werden.

Die Mehrzahl der P2P-Ansätze basiert auf einer Kombination aus einem spezifischem Service-Overlay mit einem globalen Kommunikations-Underlay, z. B. JXTA [22], SpoVNet [34] oder ROAM [172]. Diese Plattformen bieten u. a. auch gruppenorientierte Dienste an, welche eine Multicastkommunikation ermöglichen. Sie sind zumeist von weiteren Komponenten für die Nutzerverwaltung, Synchronisation, Sicherung von Ordnungs- und Konsistenzeigenschaften abhängig. Für die Wartung der Topologie werden durch die Plattformen häufig sehr viele XML-basierte oder binäre Nachrichten ausgetauscht, was sich nachteilig auf die Ressourcen der mobilen Endgeräte auswirkt und die zu erwartende Anwendungslebenszeit reduziert [34, 212]. ROAM greift als Real-Time Middleware bei der Nachrichtenübertragung in Funktionen des Betriebssystems ein und wäre ohne umfangreiche Anpassungen nicht für mobile kollaborative Anwendungen einsetzbar. Die definierten Optimierungsfunktionen könnten für den Netzzugriff in mobilen Systemen interessant sein, betreffen aber nicht mehr den Fokus dieser Arbeit.

Die Kommunikations-Underlays umfassen viele Peers und haben eine umfangreiche Funktionalität. Nicht alle Peers des Underlays stehen jedoch in Verbindung mit den Peers der Anwendung, die das Service-Overlay bildet. Dieser Zusammenhang ist in Abbildung 3.4 für SpoVNet dargestellt. Für das Roboterszenario aus Abschnitt 2.2 würde dies bedeuten: Alle Rettungskräfte und Roboter sind Teilnehmer des Underlays. Auf dessen Basis kommunizieren die Roboter über ein Service-Overlay miteinander. Für den Nachrichtenaustausch muss jeder Peer innerhalb des Underlays mehrere Kommunikationsverbindungen zu anderen Peers aufrechterhalten, ohne dass dieser Mehraufwand immer in einem direkten Zusammenhang zu einem Service-Overlay steht, dessen Teilnehmer der jeweilige Peer ist. Das würde die Ressourcen der mobilen Endgeräte stark belasten und damit die zu erwartende Servicequalität bzw. die Anwendungslebenszeit deutlich reduzieren.

In SpoVNet wird die Kommunikationsadresse eines Teilnehmers durch ein Tupel modelliert, das aus einer eindeutigen Teilnehmeridentifikation und der aktuellen Transportadresse (IP-Adresse, Bluetooth-Adresse o. ä.) besteht. Die Teilnehmeridentifikation wird beim Gruppeneintritt vergeben und bleibt bis zum Austritt des Teilnehmers an diesen gebunden. Ändert sich die Transportadresse eines Teilnehmers, beispielsweise durch einen mobilitätsinduzierten Netzwechsel, so wird diese im Adresstupel des Teilnehmers

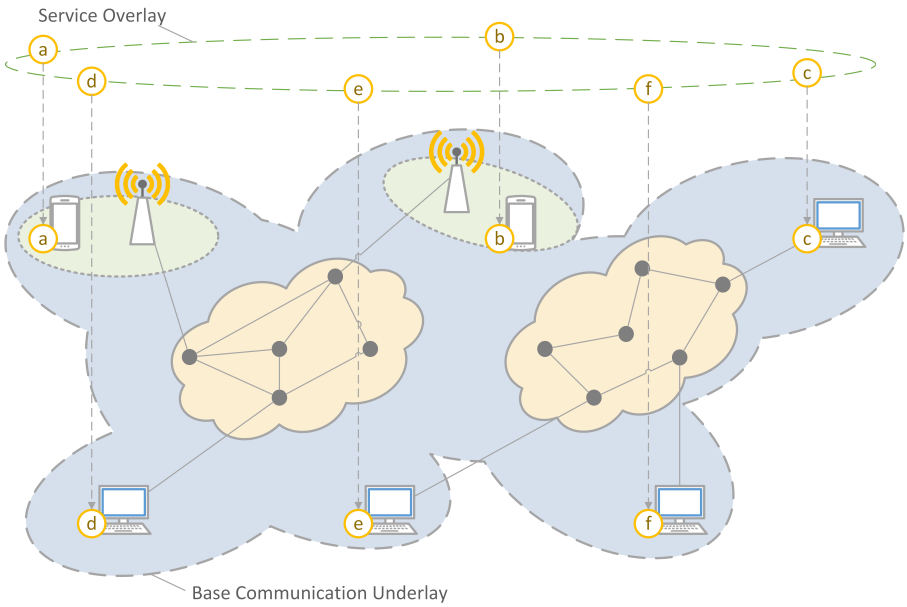


Abbildung 3.4: SpoVNet Kommunikations-Underlay mit einem darauf aufbauenden dedizierten Service-Overlay (nach [34])

aktualisiert. Dadurch ist es den Teilnehmern möglich, mit festen logischen Adressen – den Teilnehmeridentifikationen – zu kommunizieren. Änderungen an der Netztopologie bleiben vor den Anwendungen des Service-Overlays verborgen. So könnte beispielsweise ein Roboter durch den Netzwechsel zwischen WLANs verschiedener Anbieter seine IP-Adresse wechseln, ohne dass sich seine Adresse aus Sicht der Anwendung ändert. Damit bieten Middlewareplattformen eine Mobilitätsunterstützung, welche mit dem Grundgedanken von Mobile IP (siehe folgenden Abschnitt) vergleichbar ist. Jedoch fehlen den Middlewareansätzen wesentliche Funktionen, um die Synchronisation zwischen den Gruppenmitgliedern nach einer Verbindungsunterbrechung wieder herzustellen. Zahlreiche Kommunikationsausfälle könnten somit den Zusammenhalt der Gruppe zerstören.

3.1.4 Tupelräume

Ein weiteres mögliches Konzept sind *Tupelräume*. Entwickelt wurde dieses Konzept 1985 von David Gelernter im Rahmen seiner Arbeiten an der Programmiersprache *Linda* [99]. Konkret stellen die Tupelräume eine Implementierung des *inhaltsadressierten Speicherparadigmas* [181] dar. Innerhalb einer Gruppe wird ein virtueller verteilter Speicher bereitgestellt, welcher einen konkurrierenden Zugriff erlaubt – der *Tupelraum*. Innerhalb des Tupelraums können unterschiedlich strukturierte Daten als Tupel zwischen den Teilnehmern geteilt werden, z. B. die aktuelle Position des einzelnen Roboters. Somit repräsentieren diese Daten den geteilten Anwendungszustand.

Mit LIME [176] wurde eine mobile Variante des Linda-Konzeptes sowohl für kabelgebundene als auch für Ad-hoc-Netze vorgeschlagen. Der Tupelraum zwischen den Peers ist transient, d. h. die Menge von Tupeln, auf welche durch die einzelnen Peers zugegriffen werden kann, ist abhängig von der zugrundeliegenden Konnektivität zwischen den Peers. Es wird angenommen, dass die Peers in einem Tupelraum jederzeit vollständig miteinander verbunden sind und Verbindungen nur explizit nach Ankündigung abgebaut werden. Damit ist der Tupelraum in LIME nur vorübergehend geteilt. Für das Roboterbeispiel wäre der erreichbare Anwendungskontext instabil, da durch spontane Verbindungsabbrüche auf Teile des Tupelraums nicht mehr zugegriffen werden kann. Die Konsistenz zwischen den Robotern wäre dann nicht mehr gesichert, da die (Teil-)Tupelräume sich unabhängig von einem gemeinsamen Ausgangszustand weiterentwickeln können. Eine

Synchronisation dieser Teilräume wird durch LIME nicht unterstützt. Ohne zusätzliche Maßnahmen wären daher die Anforderungen mobiler kollaborativer Anwendungen nicht erfüllt.

Pinciroli *et al.* [177] präsentieren einen mobilen Tupelraumansatz, welcher prinzipiell genau dem in Abschnitt 2.2 skizzierten Anwendungsfall entspricht, jedoch nur für sehr kleine Gruppen mit wenigen Teilnehmern realisiert werden kann. Weiterführende Ansätze, zumeist im Zusammenhang mit anderen Techniken, werden in [100, 165, 175, 182, 218] diskutiert. Diese schränken jedoch die Konsistenzeigenschaften der mobilen Tupelräume stark ein bzw. reduzieren den Umfang der kollaborativen Operationen.

Live Objects [168] stellen eine höhere objektorientierte Abstraktion des Tupelraums (*Live Object Space*) dar. Eine geschlossene Gruppe kooperiert hier auf der Basis der gemeinsamen Nutzung geteilter Objekte. Ein Beispiel für solch ein Objekt ist das Membership-Datenobjekt. Änderungen an diesem Objekt werden in der Gruppe mit Hilfe des Paradigmas der Virtuellen Synchronität [31] synchronisiert und vom tatsächlichen Datenaustausch abstrahiert. Wird die Gruppe partitioniert, dürfen nur solche Teilnehmer die Objekte ändern, welche der Mehrheit der Gruppe angehören. Dadurch ist eine Wiederherstellung der Konsistenz begünstigt, da die kleinere Gruppe die geänderten Daten einfach übernehmen kann. Der Abgleich dieser Änderungen in der Gruppe bewirkt eine nicht zu vernachlässigende Verzögerung, da die virtuelle Synchronität sehr kommunikationsaufwendig ist. Bisher existiert keine Umsetzung dieses Paradigmas, die in mobilen Umgebungen nutzbar wäre.

3.1.5 Mehrschichtansätze

Bei neueren Ansätzen werden wesentliche Funktionen der Nutzermobilität in die Netzschicht verlagert [138, 139, 142, 199] und die Systeme um Funktionen für die spontane Multi-Hop-Kommunikation [89, 147] oder spezifische Kommunikationsprotokolle [124, 149, 161] erweitert.

Ein Beispiel für eine solche Lösung ist die Erweiterung für das MChannel [147] Gruppenkommunikationssystem von Costagliola *et al.* [62], in dem ein energieeffizientes und verzögerungsoptimiertes Application-Level Multicastrouting auf Basis des *Optimized Link State Routing* (OLSR) Protokolls [60] in mobilen Ad-Hoc-Netzen vorgeschlagen wird. Die Routenwahl basiert auf einer Hop-Metrik. Diese Konzepte sind jedoch durch ihre Routenauswahl nur in *Mobile Ad-hoc Networks* (MANETs) anwendbar und aufgrund der ihnen inhärenten Trägheit nicht in der Lage, auf häufige temporäre Kommunikationsausfälle und Synchronisationsfehler zu reagieren.

3.2 Netzbasierte Mobilitätsunterstützung

Eine weitere Möglichkeit der Mobilitätsunterstützung in kollaborativen Anwendungen ist die Integration mobiler Teilnehmer in eine Gruppenkommunikationssitzung unter Nutzung eines generischen Dienstes der Netzschicht (*Open System Interconnection* (OSI) Schicht 3).

3.2.1 Mobile IP

Der am häufigsten genutzte Ansatz für solch einen Dienst ist *Mobile IP* [173, 174]. Ein Gruppenkommunikationssystem mit Mobile IP[1] würde die Probleme der Nutzermobilität durch die Aufgabenteilung zwischen beiden Komponenten theoretisch zu großen Teilen lösen. Im Roboterbeispiel wäre das Gruppenkommunikationssystem für die konsistente Datenverteilung verantwortlich, während Mobile IP die übertragungstechnische Behandlung der Positionsveränderungen der Roboter gewährleistet, und so eine stabile Ende-zu-Ende-Verbindung realisiert.

Die Kommunikation mit mobilen Teilnehmern wird hierbei über „*Mobilitätsanker*“ – *Visitor Location Registers* oder *Home Agents* – realisiert. Sie verbinden das Heim-

[1] Ein Überblick über die Vielzahl von Mobile IP-Ansätzen gibt der RFC 6301 [228]. Deren spezifische Anforderungen sind in RFC 4886 [74] beschrieben.

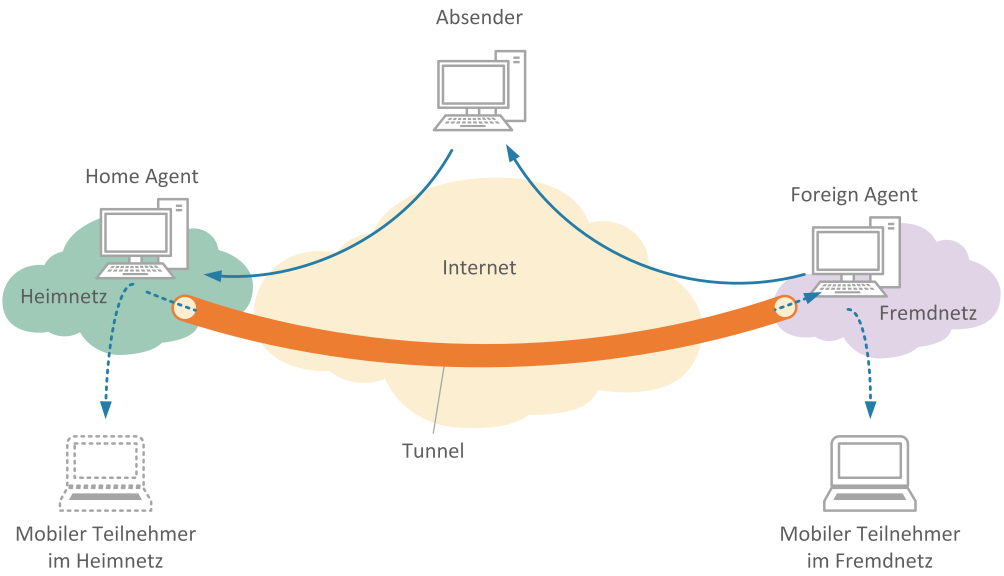


Abbildung 3.5: Mobile IP Funktionsprinzip

und das Fremdnetz über einen transparenten Tunnel, über den Nachrichten an den mobilen Teilnehmer weitergeleitet werden. Zusätzlich muss das Netz mit so genannten *Tracking-Entities* – den *Foreign Agents* – ausgestattet sein, die die Ortsveränderungen der mobilen Teilnehmer überwachen. Wird eine solche erkannt, passt das Netz die Datenverteilung entsprechend an. Aus Sicht eines Mobile IP nutzenden Gruppenkommunikationssystems wird die Mobilität der Teilnehmer durch Beibehaltung der sogenannten *Schicht 3 Zuordnung* verdeckt. Dadurch bleiben die Verbindungen, abgesehen von den sich ändernden Latenzen, stabil.

Abbildung 3.5 zeigt das Funktionsprinzip von Mobile IP. Die Nachrichtenübertragung an den mobilen Teilnehmer wird über den Home Agent gesteuert. Befindet sich der mobile Teilnehmer in seinem Heimnetz, wird ihm die Nachricht zugestellt. Wenn der mobile Teilnehmer in ein Fremdnetz wechselt, muss er sich beim Foreign Agent registrieren, welcher den Tunnel zum Home Agenten aufbaut, über den ihm dann die Nachrichten zugestellt werden.

Die Nutzung von Mobile IP ist aber nur bedingt zielführend. Es sprechen drei wesentliche Gründe gegen die Kombination von Gruppenkommunikationssystemen mit Mobile IP:

1. Die Mehrheit der öffentlichen Kommunikationsnetze unterstützt aktuell kein Mobile IP. Das ist vor allem durch finanzielle Überlegungen begründet. Im Jahre 2014 unterstützen rund 11700 von insgesamt 43128[2,3] *Autonome Systeme* (AS) das Protokoll, was einem Anteil von 27,15 % entspricht. Mobile IP hat sich technologisch also nicht durchgesetzt.

2. Mobile Endgeräte verfügen noch immer lediglich über limitierte Ressourcen. Herkömmliche Gruppenkommunikationsprotokolle würden diese rasch erschöpfen, da die Protokolle nicht bezüglich des Ressourcenverbrauchs optimiert sind.

3. Für mobile Anwendungen wirkt sich auch die Erhöhung der Ende-zu-Ende-Verzögerung in mobilen Kommunikationsnetzen nachteilig aus. Bei der Verwendung von Mobile IP kann diese weiter zunehmen. Herkömmliche Gruppenkommunikationssysteme nutzen jedoch oft statisch definierte Laufzeiten für diverse Timer (z. B. SpovNet/MCPO [212], Zookepper [116], JGroups [161]), welche für die variierenden Latenzen in mobilen Netzen nur bedingt geeignet sind. Steigt bei

[2]Siehe dazu: `http://www.cidr-report.org/as2.0/`
[3]Siehe dazu: `http://bgp.potaroo.net/v6/as2.0/index.html`

einer relativ stabilen Verbindung die durchschnittliche Latenz, so erhöht sich mit dieser ebenso das Risiko für Fehlannahmen bezüglich des vermissten Teilnehmers, was wiederum zu unberechtigten Gruppenausschlüssen führen würde.

3.2.2 Weiterentwicklungen auf Basis von Mobile IP

Die Grundidee von Mobile IP wurde mehrfach aufgenommen und weiterentwickelt. Stellvertretend soll hier der Ansatz von Babu *et al.* [16] genannt werden, in dem Mobile IP auf eine Gruppe von mobilen Peers erweitert wird, welche ein mobiles Netz formen. Dieses ist über einen mobilen Router mit der Restgruppe verbunden und bildet innerhalb der Gruppe eine Untergruppe. Die Teilnehmer dieser Untergruppe sollten gleiche Eigenschaften in Bezug auf Mobilität, Ende-zu-Ende-Verzögerung und Datenpfade aufweisen. Ist dies der Fall, kann eine Leistungsfähigkeit wie bei Mobile IP erreicht werden. In den meisten Anwendungsfällen fehlen jedoch solche gemeinsamen Eigenschaften. So ist im Roboterbeispiel die Bewegung der einzelnen Roboter sehr individuell. Ihre unterschiedlichen Positionen im Suchgebiet führen zu unterschiedlichen Netzeigenschaften. Dadurch müsste für jeden Roboter eine eigene, durch einen mobilen Router verwaltete Untergruppe erzeugt werden.

Auch die Weiterentwicklungen von Mobile IP weisen dessen Nachteile auf und erhöhen lediglich die Komplexität des Gesamtsystems, ohne signifikante Vorteile für mobile kollaborative Systeme zu bieten.

3.3 Publish & Subscribe-Systeme

Ein weiterer häufig genutzter Ansatz für die mobile Kollaboration ist *Publish & Subscribe* (P&S). Dabei werden Nachrichten oder Ereignisse von so genannten Erzeugern – den *Publishern* – produziert und von einer beliebigen Anzahl von Empfängern – den *Subscribern* – konsumiert. Die Nachrichten werden nicht direkt an die Empfänger versandt, sondern erst durch Merkmale wie Klasse, Typ, diverse Schlagworte oder ähnliches beschrieben. Die Subscriber geben ihr Interesse an einem bestimmten Nachrichtentyp bekannt. Das P&S-System sorgt daraufhin dafür, dass die angebotenen Nachrichten allen interessierten Empfängern zugestellt werden.

Systeme dieser Art bestehen aus einer Anzahl von Instanzen, die lose in Zeit, Raum und bzgl. ihrer Synchronisation gekoppelt sind. Die ersten P&S-Systeme basierten auf einem Gruppenkommunikationssystem unter Nutzung des Paradigmas der virtuellen Synchronität. Weiterentwicklungen beschäftigen sich auch mit der konsistenten Ereignisverarbeitung bei instabilen Verbindungen [17, 64, 82, 188, 194]. Diese definieren zumeist Proxy-Systeme, welche Daten als Stellvertreter für nicht erreichbare Teilnehmer speichern. Die Datenspeicherung erfolgt dabei proaktiv oder reaktiv.

Eines der am weitesten verbreiteten P&S-Systeme, welches in erster Linie auf zentralisierte Elemente zurückgreift, ist das *Extensible Messaging and Presence Protocol* (XMPP) [188]. Es vermittelt zwischen den Publishern und Subscribern unter Nutzung einer dezentralen Brokerinfrastruktur, die aus miteinander verbundenen Domänen besteht. Jede Domäne wird durch einen Server verwaltet, mit dem sich die Rechner verbinden. Der prinzipielle Aufbau der XMPP-Architektur für zwei XMPP-Domänen *A* und *B* sowie fünf Nutzer ist in Abbildung 3.6 dargestellt. Der Austausch zwischen den beiden XMPP-Domänen wird über so genannte *Federations* realisiert, die es ermöglichen, dass (in Abbildung 3.6) Nutzer (1) und (4) miteinander kommunizieren können.

Unter Nutzung der Multi-User-Chat-Protokollerweiterung [189] ist es möglich, eine offene Gruppe (Raum genannt) für den Nachrichtenaustausch aufzubauen. Ist ein Teilnehmer aktuell nicht erreichbar, so werden die Gruppennachrichten gespeichert und können nach dem Wiedereintritt abgefragt werden. Damit ermöglicht XMPP jedoch weder ein globales Wissen, noch wird eine enge Kollaboration sichergestellt. Die Abhängigkeit von einer Infrastruktur kann durch die Protokollerweiterung *Serverless Messaging* [187] (basierend auf *DNS-based Service Discovery* und *Multicast DNS* [12, 51, 205]), umgangen werden, sollte aber aus Sicherheitsgründen nur in lokalen Netzen eingesetzt werden. Neuere P&S-Ansätze wie [11, 30, 108], welche sich mit der Nutzermo-

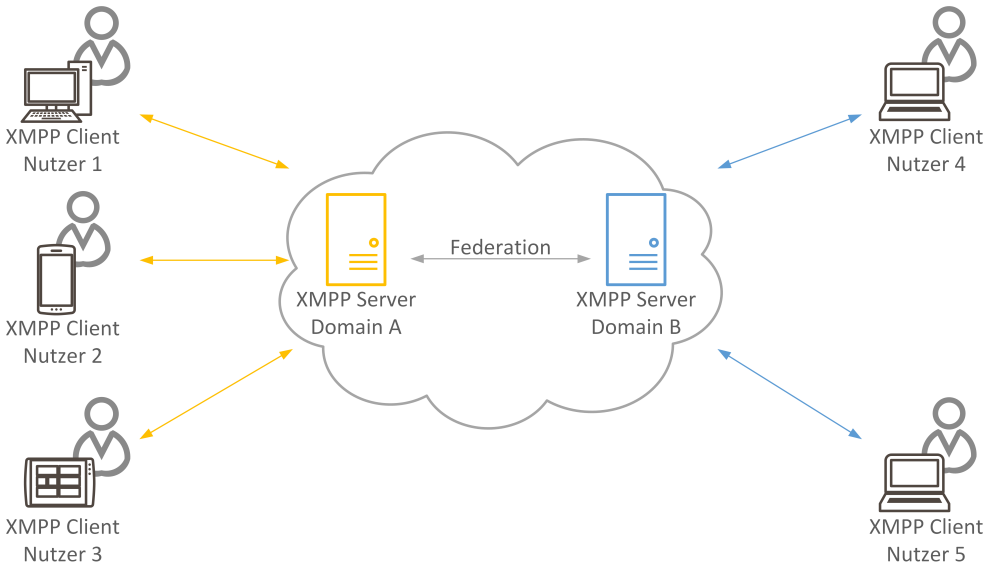


Abbildung 3.6: Prinzipielle Aufbau einer XMPP-Architektur mit zwei Domänen und fünf Nutzern

bilität befassen, benutzen zumeist Cloud- oder *Software-Defined Network* (SDN)-basierte Techniken.

Die Beispiele zeigen, dass P&S-Systeme nicht für die Umsetzung mobiler kollaborativer Anwendungen geeignet sind, da solche Anwendungen

1. eine stärkere Konsistenz des Anwendungskontextes erfordern, als P&S-Systeme sie üblicherweise bieten,
2. eine gruppenweite Koordination auf Basis eines konsistenten globalen Wissens benötigen, und
3. ihre Dienste ohne eine definierte feste Infrastruktur erbringen müssen.

Zwar bieten die oben erwähnten Ansätze Re-Synchronisationsfähigkeiten bzgl. entgangener Daten, mittels derer beispielsweise die Roboter den gleichen Wissensstand (wieder-)herstellen könnten. Ein gemeinsamer geteilter Zustand innerhalb der Gruppe wird dadurch aber nicht erreicht. Die wesentliche Eigenschaft von P&S-Systemen – die lose Kopplung in Raum, Zeit und bzgl. Synchronisation – steht somit im Widerspruch zu den Anforderungen an mobile kollaborative Anwendungen.

3.4 Cloudbasierte Kollaboration

Gegenwärtig entwickeln sich Middlewaresysteme mehr und mehr hin zu cloudbasierten Plattformen, wobei unter einer *Cloud* [155] ein Pool von virtualisierten Rechnerkomponenten verstanden wird. Solche Plattformen können für die Kooperation mit mobilen Nutzern genutzt werden, indem sie in einen vereinheitlichten Arbeitsablauf zusammen mit stationären Nutzern integriert werden.

Little *et al.* [144] identifizierten den Koordinationsaspekt als das entscheidende Problem beim Entwurf einer cloudbasierten Middleware. Bestehende Systeme modellieren die Kommunikation aus einer Client-Perspektive unter Nutzung eines zentralen Koordinators und sind somit nach Little *et al.* in einer Multipeerumgebung nicht anwendbar. Um das Koordinationsproblem zu lösen, verwenden cloudbasierte Middlewaresysteme zumeist Gruppenkommunikationsdienste, z. B. für die Zustandsverwaltung, die Lastbalancierung sowie die Datenreplikation. Der Nachrichtentransfer dieser Dienste basiert wiederum überwiegend auf dem Paradigma der *Virtuelle Synchronität* (VS) oder einem Konsensprotokoll wie Paxos. Little *et al.* [144] verwenden für diese Zwecke das JGroups

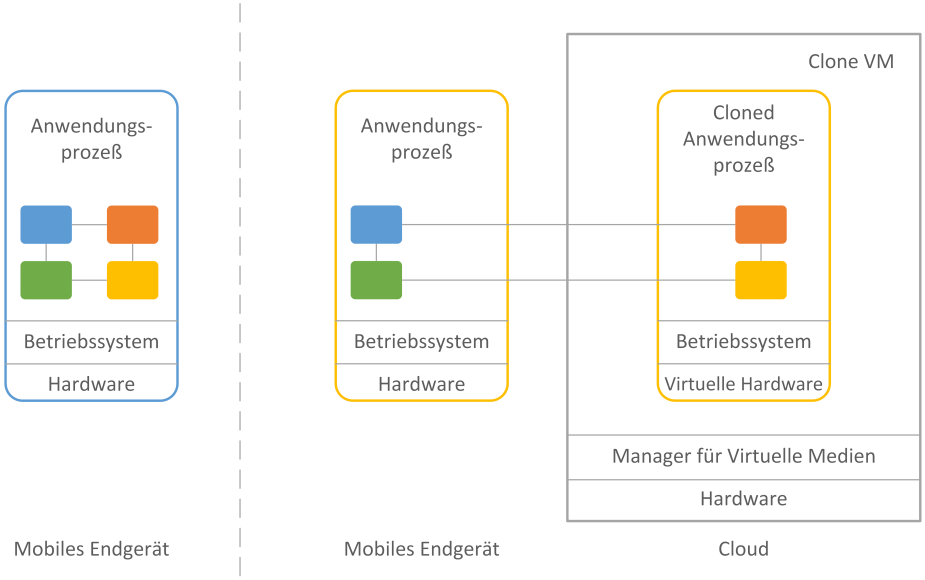


Abbildung 3.7: Gegenüberstellung einer herkömmlichen Einzelmaschinen Prozessausführung und einer verteilten Prozessausführung mittels des Clone Cloud-Ansatzes (nach [59])

Toolkit [161]. Die gruppenorientierte Kommunikation der cloudbasierten Middlewaresysteme ist somit vergleichbar mit einem klassischen Gruppenkommunikationsansatz mit ressourcenstarken Zentralsystemen.

Das Clone Cloud Framework von Chun *et al.* [59] ist ein hybrider Ansatz von Middleware- und Cloudsystemen. Dieser verlagert (Teil-)Prozesse in die Cloud, um komplexe Strukturen zu vermeiden und von der höheren Ressourcenausstattung zu profitieren. Die automatische Verlagerung von Prozessen zwischen Mobilgerät und Cloud erfolgt auf Basis statistischer Analysen und dynamischen Profilings unterbrechungsfrei mittels *Remote Procedure Calls* (RPCs) [216] und hypervisorbasierten Techniken [179]. Durch die so erreichte Ressourceneinsparung auf den mobilen Endgeräten kann die Gesamtausführungszeit einer Anwendung um den Faktor 20 verbessert werden.

Der prinzipielle Aufbau einer kollaborativen Anwendung auf einem mobilen Endgerät ist für eine herkömmliche Anwendung und eine Clone Cloud-basierte Anwendung in Abbildung 3.7 vergleichend dargestellt. Auf der linken Seite ist der Anwendungsaufbau bei einer herkömmlichen Prozessausführung zu sehen. Demgegenüber ist rechts die Architektur einer Anwendung für eine Clone Cloud-basierte Ausführung gezeigt. Für die Realisation der Prozessmigration sind weitgehende Anpassungen der Anwendung erforderlich (z. B. Anpassung der Zieladressen und Timeouts).

Eine Kombination aus dem Cloud- und dem Proxyansatz mit der Bezeichnung *Cloudlets* wurde von Satyanarayanan *et al.* [190] entwickelt. Ansätze dieser Art werden auch *Follow Me Cloud* genannt. Potentielle Ressourcenengpässe mobiler Endgeräte sollen durch die Nutzung der Cloud überwunden und typische Netzcharakteristiken[4] durch die örtliche Verlagerung der Cloudinstanzen (der Cloudlets) hin zum Nutzer verbessert werden. Durch die Positionierung der Cloudlets auf Proxyservern im Netz des Nutzers „*folgt*“ ihm somit die Cloud.

Zusammenfassend ist festzustellen, dass sowohl die cloudbasierte Umsetzung klassischer Gruppenkommunikationsprotokolle, wie beispielsweise [25, 151, 170], als auch die spezifische Anpassung bestehender Lösungen (im Sinne von cloudbasierter Middleware, Cloudlets oder Clone-Cloud-Ansatz) keinen Vorteil gegenüber den klassischen Gruppenkommunikationsprotokollen bieten. Die verwendeten Techniken steigern zudem die Abhängigkeit von einem Infrastrukturanbieter und erzeugen komplexe Lösungen.

[4] wie Verzögerung, Jitter oder Bandbreite

3.5 Kombinierte Ansätze

Die bisher dargestellten Ansätze bieten in unterschiedlichem Umfang die Funktionen an, die von mobilen kollaborativen Anwendungen benötigt werden. Keiner dieser Ansätze gewährleistet jedoch eine vollständige Gruppenabstraktion für mobile Umgebungen, welche Gruppenaufbau, zuverlässige Übertragung von Nachrichten an alle Gruppenmitglieder und das globale Wissen auch bei häufigen Verbindungsunterbrechungen sichert. Eine Möglichkeit, sie für mobile kollaborative Anwendungen bereitzustellen, wäre die Kombination eines leichtgewichtigen Gruppenkommunikationssystems mit innovativen Netztechniken, welche eine zuverlässige Übertragung auch über unzuverlässige und instabile Kommunikationskanäle in mobilen Umgebungen gewährleisten. Das Gruppenkommunikationssystem wäre dabei für die Gruppenverwaltung und die Konsistenzsicherung zwischen den Gruppenmitgliedern verantwortlich, während die Netztechniken einen zuverlässigen transparenten Kommunikationskanal zu den Gruppenmitgliedern bereitstellen, der aber im Vergleich zu drahtgebundenen Medien durch eine größere, variablere Latenz gekennzeichnet ist. Die in Frage kommenden innovativen Netztechniken sind Mobility Management Frameworks, verzögerungstolerante Netze und datenorientierte Netze. Allen drei Ansätzen ist gemein, dass ihre Verbreitung noch sehr begrenzt ist, sodass die folgende Diskussion nur perspektivischen Charakter hat.

3.5.1 Mobility Management Frameworks

Mobility Management Frameworks, auch als Handover-Frameworks bezeichnet, dienen der Bereitstellung eines unterbrechungsfreien Kommunikationskanals auch für mobile Nutzer. Das wird durch einen proaktiven nahtlosen Wechsel zwischen verschiedenen Netzen erreicht, ohne die laufende Kommunikation zu unterbrechen. Mit den bisher vorgestellten Ansätzen, z. B. von Lee *et al.* [138], Skorepa und Klugl [199], Karetsos *et al.* [126], He und Perkins [111], Fotouhi *et al.* [84] und Yen *et al.* [224] könnten die Roboter mit Mechanismen ausgestattet werden, die in Kombination mit einem klassischen Gruppenkommunikationssystem einen nahtlosen Wechsel zwischen verschiedenen heterogenen Zugangsnetzen erlauben.

Solch eine Lösung greift jedoch in die Kommunikationsabläufe der Endgeräte ein und weist ähnliche Nachteile wie Mobile IP auf (z. B. Skalierungs- und Effizienzprobleme, Ressourcenengpässe). Bislang sind die meisten Ansätze nur simulativ untersucht worden, wodurch die Eigenschaften realer Netze nur unzureichend repräsentiert werden. Die Sicherung der Konsistenz des globalen Wissens bei einem temporär fehlenden Netzzugang eines oder mehrerer Teilnehmer müsste durch das Gruppenkommunikationssystem erfolgen.

3.5.2 Verzögerungstolerante Netze

Verzögerungstolerante Netze (*Delay-Tolerant Networks* (DTN)) [45, 78] zielen auf spärlich verbundene, heterogene und verteilte Netze, welche aufgrund der geringen Ende-zu-Ende-Stabilität oft partitionieren. Ursprünglich wurden sie für die Satellitenkommunikation [42] entworfen und werden zunehmend für die *Car-to-Car*-Kommunikation [140, 207] verwendet. Dieses Anwendungsgebiet ist mit dem Roboterszenario vergleichbar: Eine Anzahl von mobilen Einheiten möchte ein konsistentes Wissen über ein definiertes Gebiet teilen (z. B. die Innenstadt oder das Suchgebiet). Dazu wird der zuverlässige Kommunikationskanal auf Basis des „*store and forward*“-Prinzips [45] realisiert. Eine erste Architektur für diese Netze stammt von Kevin Fall [77] und definiert ein Overlay [193] zwischen den Teilnehmern – den *DTN Nodes* – dessen Eigenschaften P2P-Netzen ähnlich sind. Weitere Lösungen wurden unter anderem von Wood *et al.* [219] sowie von Farell und Chahill [79] vorgeschlagen.

Die Nachrichtenübertragung innerhalb des Overlays erfolgt Hop-by-Hop über *User Datagram Protocol* (UDP)/IP mit optionaler Ende-zu-Ende-Bestätigung. Im Overlay sind Proxys für die Nachrichtenspeicherung und -weiterleitung platziert [45]. Knoten und Anwendungen im Overlay werden über eine ID adressiert. Für die Kollaboration in Gruppen teilen sich mehrere Teilnehmer eine ID. Der Nachrichtenaustausch in der

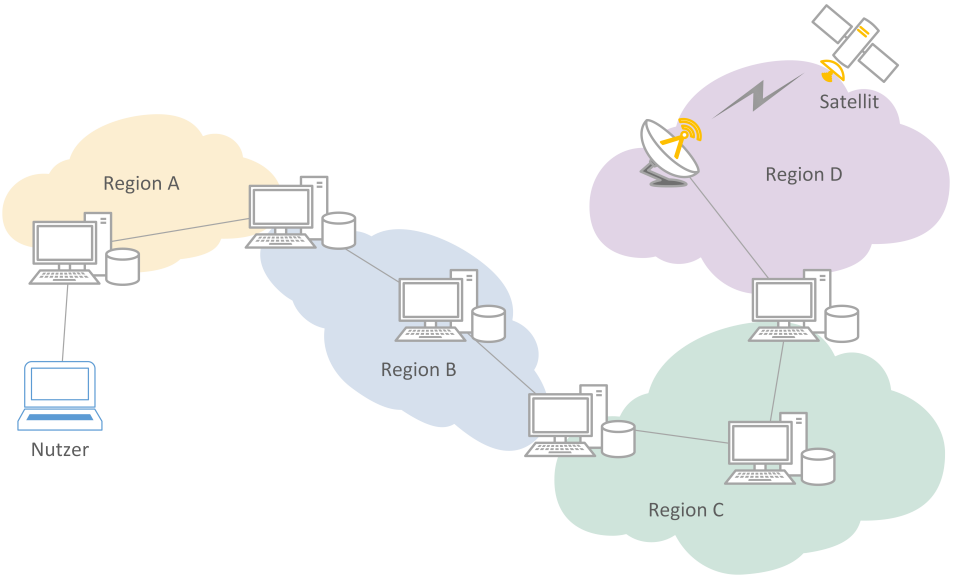


Abbildung 3.8: Exemplarische DTN-Architektur mit vier Regionen (nach [77])

Gruppe erfolgt unter Verwendung spezifischer Routingprotokolle [4]. Die Umsetzung eines Multicast ist in verzögerungstoleranten Netzen jedoch ein komplexes Problem, für das nur eingeschränkte Lösungen existieren [214, 220, 221].

Abbildung 3.8 stellt ein vereinfachtes Szenario nach [77] dar, in dem ein Anwender mit einem Satelliten Daten austauscht. Die Kommunikation erfolgt über vier Regionen (Region A bis D), in welchen mehrere Proxys die ausgetauschten Daten zwischenspeichern. Grundsätzlich können in den Regionen unterschiedliche Netztechniken eingesetzt werden, welche durch das Overlay überbrückt werden. Im Roboterszenario wäre die Region D die stationäre Kontrollstation, alle anderen Rechner mobile Einheiten.

Für die Verwendung von verzögerungstoleranten Netzen in mobilen kollaborativen Systemen müsste eine Multicastlösung definiert werden, welche ein konsistentes gemeinsames Wissen zwischen den Mitgliedern der verzögerungstoleranten, kollaborativen Gruppe realisiert. Die größte Herausforderung für solch einen Ansatz wäre der Umgang mit der potentiell ungebundenen Round-Trip-Time bei der Definition der Gruppenkonsistenz.

3.6 Datenorientierte Netzarchitekturen

Datenorientierte Netzarchitekturen, thematisch eng verwandt mit den CCN [202], stellen die Strukturen und Technologien bestehender Netze infrage. Sie gehören mit zu den *Future Internet Projekten* [75]. Im Mittelpunkt dieses Konzepts stehen anstelle ihres Speicherorts die Daten selbst. Van Jacobson beschreibt dies als Paradigmenwechsel von einer hostzentrierten hin zu einer datenorientierten Netzarchitektur [123]. Damit einher geht auch eine Änderung der Semantik des Netzes. Aus dem Zustellen von Daten an definierte Adressen wird das Abrufen von Daten, welche über ihren Namen identifiziert werden. Im Kontext kollaborativer Anwendungen würde die Gruppe über den Austausch benannter Daten kooperieren, was vergleichbar mit den Tupelraumansätzen ist.

Named Data Networking (NDN)[5] [226, 227] ist ein Ansatz für solch eine Netzarchitektur und basiert auf einem Vorschlag von van Jacobson [122]. Eingesetzt wird diese Netzarchitektur beispielsweise im Internet der Dinge und der Car-to-Car-Kommunikation [196]. Sie modelliert das Kommunikationsnetz als Verteilnetz für Inhalte, welche dedizierte Datenmengen (engl. *Content Chunks*) verteilen. Das Adressierungsschema der NDN leitet die Benennung der Chunks direkt aus den ausgetauschten Daten ab. Ein Name könnte die Adresse eines Endpunkts, ein Teil einer größeren Datenmenge (z. B.

[5] siehe auch: `www.named-data.net`

alle Messdaten der letzten Stunden von Roboter X), ein Kommando (z. B. Robotersuche starten) oder ähnliches sein. Die Datenverteilung basiert auf einem Konzept vergleichbar dem P&S, indem Gruppenmitglieder ihr Interesse an bestimmten Daten bekanntgeben.

Datenorientierte Netze eignen sich prinzipiell für kollaborative Gruppen, da sie die Konsistenzprobleme von Tupelraumansätzen überwinden. Ihr Kommunikations- und Adressierungsschema ist jedoch grundlegend anders als in existierenden Netzen. Sie bieten keinerlei Gruppenabstraktion, keine Ende-zu-Ende-Relation und keine Funktionen zur Bereitstellung eines gemeinsamen Wissens. Named Data Netze erfordern zur Diensterbringung eine spezifische Netzinfrastruktur sowie ein getrenntes Routing und Forwarding. Ihr Adressierungsschema ist nicht kompatibel mit bestehenden Netzen. Ein leichtgewichtiges Gruppenkommunikationssystem, welches oberhalb der datenorientierten Netze angesiedelt ist, müsste neben der fehlenden Gruppenabstraktion auch alle Funktionen bzgl. des gemeinsamen Wissens bereitstellen. Es könnte beispielsweise über die konsistente Datenanforderung durch alle Gruppennutzer erzeugt werden. Damit würde das Konsistenz- in ein Koordinationsproblem transformiert und eine andere Struktur der Gruppenkommunikationsprotokolle erforderlich werden: Die Teilnehmer müssten aktiv Wissen anfordern und nicht wie bisher über geändertes Wissen informiert werden.

3.7 Fazit

Keiner der vorgestellten Ansätze erfüllt alle Anforderungen an die Gestaltung mobiler kollaborativer Anwendungen. Zumeist werden nur einzelne Aspekte wie Datentransfer, Adressierung oder Nutzermobilität unterstützt. Deshalb müssen Vor- und Nachteile, Einsatzbereiche und Anforderungen an die im Kapitel 2.4 Ansätze hinsichtlich der Erfüllung der formulierten Ansprüche verglichen und bewertet werden. Im Ergebnis dieses Vergleichs ist abzuleiten, mit welchem Ansatz ein gruppenorientiertes Kommunikationssystem realisiert werden kann.

Geschlossene Gruppe Die für die Kollaboration geforderten geschlossenen Gruppen werden durch die meisten Gruppenkommunikations-, Middleware- und Tupelraumansätze unterstützt. Auch einzelne Ansätze aus den Bereichen Application-Level Multicast, P&S und Cloud modellieren geschlossene Gruppen.

Gemeinsames Wissen Innerhalb einer Gruppe hängt die Verfügbarkeit des gemeinsamen Wissens oft von der konkreten Umsetzung der Wissensspeicherung ab. Punktuelle Speicher, wie [34, 168, 176], sind nur eingeschränkt für mobile kollaborative Anwendungsszenarien verwendbar. Verliert beispielsweise eine Reihe von Robotern ihre Verbindung zur Gruppe, geht bei diesen Ansätzen ein substantieller Teil des globalen Wissens verloren, der somit für die Restgruppe nicht mehr verfügbar ist. Verstärkt wird dieses Problem dadurch, dass viele Ansätze mit dem unverzüglichen Ausschluss einzelner Gruppenmitglieder auf deren Nichterreichbarkeit reagieren. Die Sicherung des Anwendungszustandes wird dabei im Fehlerfall dem Anwendungsentwickler überlassen. Die kombinierten Ansätze, welche zukünftige Techniken wie Mobility Management Frameworks, verzögerungstolerante Netze und datenorientierte Netzarchitekturen nutzen, konzentrieren sich ausschließlich auf den Datenaustausch. Die Sicherung eines gemeinsamen Wissens ist nicht Gegenstand dieser Konzepte. Ein mit diesen Ansätzen zu kombinierendes leichtgewichtiges Gruppenkommunikationssystem hätte das gemeinsame Wissen auch während Verbindungsunterbrechungen zu sichern. Aktuell existiert jedoch kein Gruppenkommunikationssystem, welches diese Fähigkeiten besitzt.

Umgang mit Verbindungsstörungen Kollaborative Anwendungen müssen robust gegenüber Verbindungsabbrüchen sein und diese zeitnah transparent behandeln. Nur ausgewählte Ansätze berücksichtigen den Einfluss der Nutzermobilität auf die Kommunikation zwischen den Gruppenmitgliedern, so die netzbasierten Ansätze, die verzögerungstoleranten Netze und vereinzelt Middlewareplattformen. Sie geben der Anwendung

die Möglichkeit zum Datenaustausch. Der Umgang mit Datenverlust oder Kommunikationsunterbrechungen ist jedoch nicht Gegenstand dieser Konzepte.

Optimierung für mobile Endgeräte Mobile Endgeräte benötigen einen allgemein sparsamen Umgang mit den verfügbaren Ressourcen. Viele Mechanismen, welche zur Sicherung bzw. Adaption der Dienstgüte in den komplexen Ansätzen eingesetzt werden (z. B. Probing, Sendewiederholung, zuverlässige Übertragung, Partitionierungserkennung) sind ressourcenintensiv. Der Aspekt der Verringerung des Ressourcenverbrauchs wird nur von einigen Middlewareansätzen, den Mobility Management Frameworks und den verzögerungstoleranten Netzen als primäres oder sekundäres Ziel verfolgt. Die übrigen diskutierten Ansätze vernachlässigen diesen Aspekt zumeist.

Integration in existierende Systeme mittels eines dezentralen Ansatzes Mobile kollaborative Anwendungen zielen auf eine universelle Einsetzbarkeit. Eine Vielzahl der vorgestellten Lösungen ist von einer bestimmten Infrastruktur abhängig bzw. setzt auf zentralisierte Komponenten, z. B. [107, 146]. Sind diese nicht verfügbar bzw. erreichbar, so ist die Funktionsfähigkeit der kollaborativen Anwendung nicht mehr gegeben. Zusätzlich sprechen folgende Umstände gegen solche Elemente:

1. Sie sind potentielle Engpässe, welche sich negativ auf die Leistung des Systems auswirken können.

2. Sie stellen einen Single Point of Failure dar.

3. Sie erhöhen die Systemkomplexität, da sie zusätzliche Anforderungen wie Skalierbarkeit und Zuverlässigkeit bei minimalen Kosten erfüllen müssen.

Einige P2P, einzelne Tupelraum- und Application-Level Multicastsysteme sind unabhängig von zentralen Komponenten und Diensten. Andere Ansätze nutzen Proxy- oder Brokerkomponenten.

In Anbetracht der häufig relativ kleinen Gruppen mobiler kollaborativer Anwendungen steht der Aufwand für eine spezifische Infrastruktur oft nicht in einem vernünftigen Verhältnis zum erzielbaren Nutzen. Mit der Verbreitung der mobilen Ad-hoc-Netze und des Internets der Dinge steigt die Anzahl von Einsatzszenarien, in welchen zentrale Komponenten oder eine speziellen Infrastruktur verfügbar sind. Viele dieser Szenarien sind jedoch ebenso wie die kollaborativen Anwendungen kleinteilig strukturiert. Um heterogene, inkompatible Insellösungen zu vermeiden, ist es vorteilhaft, bei Anwendungsfällen mit einer begrenzten Teilnehmermenge Ansätze zu präferieren, die mit unterschiedlichen Netztechnologien umgehen können und keine spezifischen Infrastrukturabhängigkeiten erzeugen. Für einige der dargestellten Ansätze existieren Teillösungen, um die Infrastrukturabhängigkeit zu reduzieren, wie beispielsweise die Serverless-Messaging Erweiterung XEP-0174 [187] für das Nachrichtenprotokoll XMPP.

Anpassung an sich ändernden Bedingungen Aufgrund der ihnen inhärenten Dynamik und der Asynchronität mobiler Gruppe ist es notwendig, deren interne Struktur kontinuierlich an sich ändernde Bedingungen anzupassen. Einzelne GKS-, Application-Level Multicast- und Middlewareplattformen besitzen solche Funktionen. Keiner der vorgestellten Ansätze bietet jedoch Mechanismen, welche die Beziehungen innerhalb der Gruppe entsprechend den aktuellen Gruppen- und Netzbedingungen warten.

Verteilte gruppenweite Koordination Eine verteilte gruppenweite Koordination ist mit einer Vielzahl der vorgestellten Ansätze möglich, sofern sie die Forderung nach einer geschlossenen Gruppe und nach einem gemeinsamen Wissen erfüllen. Das trifft vor allem auf die Gruppenkommunikations-, Middleware- und Tupelraum- und Application-Level Multicastansätze zu. Auf Basis der Verknüpfung des gemeinsamen Wissens mit gruppenweiten Ereignissen kann eine gruppenweite Koordination entsprechend des „*global knowledge, local decisions*“ Prinzips realisiert werden.

Tabelle 3.1: Vergleich der vorgestellten Ansätze bzgl. der Anforderungen mobiler kollaborativer Anwendungen

	Herkömmliche GKS und Derivate	**netzbasierte Mobilitätsunterstützung**	**Publish & Subscribe**	**cloudbasierte Kollaboration**	**Kombinierte Ansätze**
Geschlossene Gruppe	+	–	o	o	–
Gemeinsames Wissen	o	–	–	–	o
Umgang mit Verbindungsstörungen	o	o	–	–	o
Optimierung für mobile Endgeräte	o	–	–	–	o
Integration in existierende Systeme mittels eines dezentralen Ansatzes	o	–	o	–	–
Kontinuierliche Anpassung an sich ändernde Bedingungen	o	–	–	–	o
Verteilte gruppenweite Koordination	+	–	o	o	–
Anwendungsgetriebene Gruppenteilung bzw. Gruppenvereinigung	–	–	–	–	–

Anwendungsgetriebene Gruppenteilung bzw. Gruppenvereinigung Höhere Operationen, wie das semantische Teilen der Gruppe oder die semantische Vereinigung mit anderen Gruppen werden durch keinen der vorgestellten Ansätze unterstützt. Die Gruppenkommunikationssysteme unterstützen vereinzelt mit der Untergruppenbildung eine konzeptionell ähnliche Operation. P&S-Systeme können vereinzelt offene Gruppen auf Basis der Teilnehmerinteressen bilden, die durch die Variation der Interessen verwaltet werden. So könnten beispielsweise zunächst alle Roboter in einer Gruppe „*Temperatur*“ zusammengefasst sein. Um diese Gruppe semantisch zu teilen, wird das Interesse umformuliert bzw. spezialisiert. Aus dem Interesse „*Temperatur*“ wird „*Temperatur in der Nähe von [...]*“. Eine Vereinigung wäre umgekehrt durch eine Vereinfachung/Generalisierung der Interessen denkbar. Beide Konzepte garantieren jedoch nicht ein gemeinsames Wissen aller Teilnehmer ab bzw. bis zu einem definierten Zeitpunkt. Diese Forderung mobiler kollaborativer Systeme bleibt bisher unerfüllt.

In Tabelle 3.1 wird die Erfüllung der Anforderungen an mobile kollaborative Anwendungen durch die verschiedenen Ansätze zusammengefasst. Bewertet ist die Erfüllung der Anforderungen aus Sicht der kollaborativen Gruppe, deren Ziel die aktive Kollaboration auf Basis eines gemeinsamen Wissens in Anbetracht von Kommunikationsunterbrechungen ist. Dabei entspricht „+“ einer weitgehenden Erfüllung der Anforderungen durch Ansätze der jeweiligen Kategorie. Eine teilweise Erfüllung der jeweiligen Anforderung wird durch „*o*“, die Nicht- oder unzureichende Erfüllung durch „–“ symbolisiert.

Es zeigt sich, dass keiner der vorgestellten Ansätze alle Anforderungen und Aspekte mobiler Kollaboration abdeckt. Häufige Probleme wie Verbindungsabbrüche variabler Dauer, Sicherung bzw. Wiederherstellung des globalen Wissens zwischen allen Gruppenteilnehmern und Behandlung von Netzpartitionierungen werden durch keinen der Ansätze zufriedenstellend gelöst. Auch fehlt es an allgemeingültigen Lösungen für den Wiedereintritt in eine Gruppe nach einer kurzfristigen Verbindungsunterbrechung, das semantische Vereinigen oder Aufteilen kollaborativer Gruppen, die ressourcenschonende und gleichberechtigte Selbstorganisation und den Umgang mit dem Ausfall von Kommunikationspartnern bei einer Gruppenoperation.

Im Vergleich der einzelnen Ansätze ist erkennbar, dass herkömmliche Gruppenkommunikationssysteme (und deren Derivate) die meisten Anforderungen erfüllen. Aus Sicht des Autors bilden sie daher den zu bevorzugenden Ansatz für einen gruppenorientierten Kommunikationsdienst in mobile kollaborativen Anwendungen, da sie deren strengen Konsistenzanforderungen gerecht werden und den Zugriff auf das gemeinsame Wissen sowohl in Infrastruktur- als auch in Ad-hoc-Netzen ermöglichen.

Ziel dieser Arbeit ist es, für die in diesem Kapitel umrissenen Probleme Lösungskonzepte zu erarbeiten.

Kapitel 4

Sicherung der Konsistenz des globalen Wissens

Wie im Abschnitt 3 dargelegt, ist zur Umsetzung einer Mobilitätsunterstützung für mobile kollaborative Anwendungen am ehesten ein GKS geeignet, welches das globale Wissen auf Basis einer geschlossenen Gruppe verwaltet. Diese Kommunikationssysteme bilden eine Teilmenge der so genannten „*sichtbewussten*“ Gruppenkommunikationssysteme. Sie verfügen über einen Gruppenverwaltungs- (engl. *Membership Service*) und einen zuverlässigen Übertragungsdienst (engl. *Transfer Service*). In diesem Kapitel werden Eigenschaften von Gruppenkommunikationssystemen eingeführt, welche eine Voraussetzung für die Wahrung der Konsistenz in mobilen Umgebungen sind.[1]

Für die Bereitstellung eines globalen konsistenten Wissens in einer geschlossenen Gruppe wird zumeist das Paradigma der virtuellen Synchronität [33] verwendet. Klassische Gruppenkommunikationssysteme wie ISIS [33], Transis [69] oder GCP [231] gehen dabei von einer permanent verfügbaren Kommunikationsverbindung zwischen den Gruppenteilnehmern aus. Kommt es aufgrund der Nutzermobilität oder wegen Kommunikationsstörungen zu einer, wenn auch nur kurzzeitigen Kommunikationsunterbrechung (beispielsweise zwischen dem mobilen Roboter und der restlichen Suchgruppe), würden diese Systeme den nicht erreichbaren Roboter umgehend aus der Gruppe ausschließen, um die Konsistenz der Gruppe und damit des globalen Wissens zu wahren. Eine mobile Datenübertragung ist unsicher und gekennzeichnet von häufigen Verbindungsabbrüchen. Das Wiederherstellen der Verbindungen ist jedoch durch den beschriebenen Gruppenausschluss und den expliziten Wiedereintritt ressourcenintensiv und aufwendig. Es ist folglich für mobile Endgeräte mit ihren begrenzten Ressourcen ungeeignet. Daher muss auf anderem Wege gewährleistet werden, dass in einer kollaborativen Gruppe das gemeinsame Wissen auch bei kurzzeitigen Kommunikationsunterbrechungen und Netzpartitionierungen erhalten bleibt.

4.1 Schnittstellen und Interaktionsschema

Abbildung 4.1 stellt den Zusammenhang zwischen der kollaborativen Gruppe, dem Gruppenkommunikationssystem und der Anwendung dar. Die Nutzer der Anwendung bilden eine geschlossene Gruppe. Jedes Gruppenmitglied besteht aus der eigentlichen kollaborativen Anwendung und dem untergeordneten Gruppenkommunikationssystem. Alle Gruppenmitglieder haben in der Gruppe die gleichen Rechte und Fähigkeiten und können beispielsweise Gruppennachrichten versenden oder neue Mitglieder einladen.

Die Kollaboration zwischen den Gruppenmitgliedern wird über den Austausch von Nachrichten realisiert. Dafür wird im Rahmen dieser Arbeit immer ein Broadcast an alle Gruppenmitglieder angenommen. Die Anwendung manipuliert das gemeinsame Wissen der Gruppe, indem es die Verteilung einer Gruppennachricht initiiert, für

[1] Wir lehnen uns dabei an die Darstellungen in Chockler *et al.* [55], Defago *et al.* [67], Babaoğlu *et al.* [14] und [15] an. Eine formale Definition der hier diskutierten Begriffe, Eigenschaften und Funktionen ist im Anhang A enthalten.

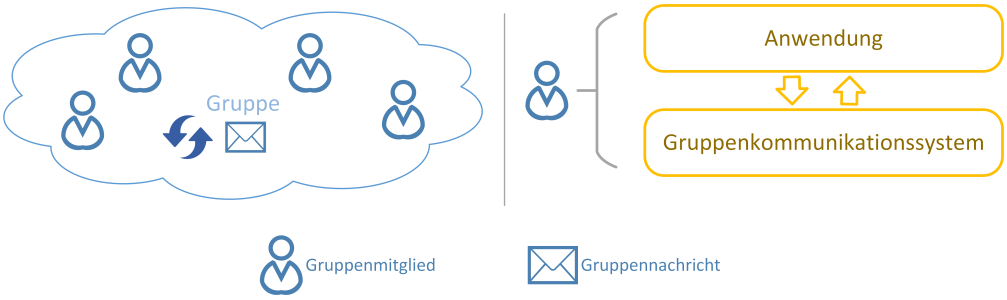


Abbildung 4.1: Darstellung Zusammenhang zwischen kollaborativer Gruppe, Gruppenkommunikationssystem und Anwendung

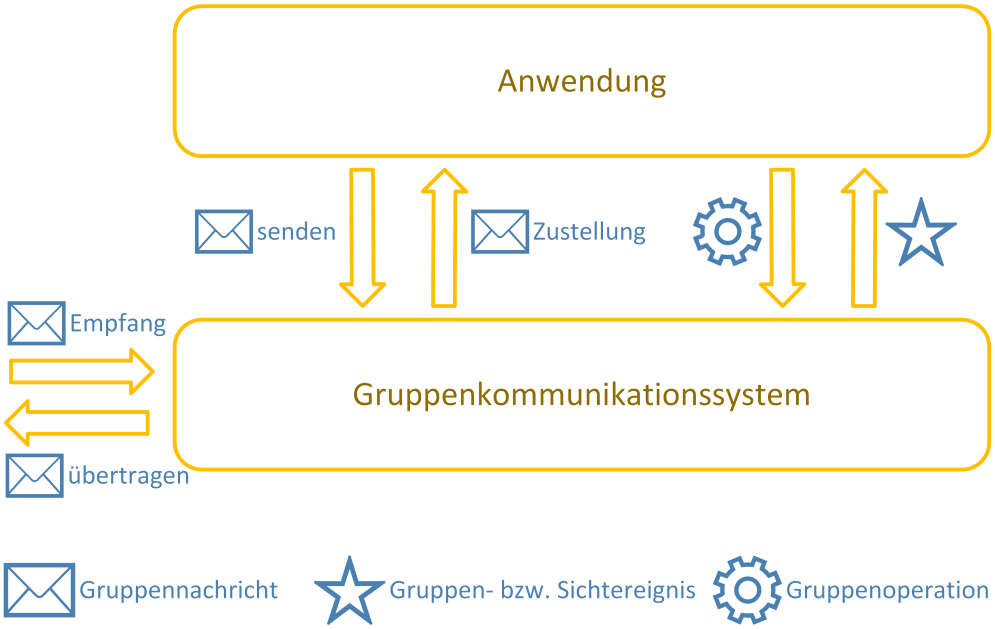


Abbildung 4.2: Darstellung der Interaktion zwischen der Anwendung, dem Gruppenkommunikationssystem und der kollaborativen Gruppe

deren sichere, konsistente und geordnete Zustellung das Gruppenkommunikationssystem verantwortlich ist.

Abbildung 4.2 zeigt die Interaktion zwischen der Anwendung, der kollaborativen Gruppe und dem Gruppenkommunikationssystem im Detail. Eine von der Anwendung versandte Gruppennachricht wird vom Gruppenkommunikationssystems unter Wahrung einer bestimmten Nachrichtenordnung, d. h. von spezifischen Zuverlässigkeits- und Zustellungseigenschaften, übertragen. Wenn ein Gruppenmitglied eine Nachricht von der Gruppe empfängt, wird zunächst geprüft, ob die Nachricht gültig ist oder verworfen werden muss. Eine Nachricht ist beispielsweise ungültig, wenn der Absender kein Gruppenmitglied ist. Der Empfang einer gültigen Nachricht wird dem Absender bestätigt und das Gruppenkommunikationssystem reiht die empfangene Nachricht in die gruppenweite Nachrichtenordung ein. Ist die Nachricht entsprechend dieser Ordnung zustellbar, so kann sie an die Anwendung ausgeliefert werden. Mögliche Varianten der Nachrichtenordnung werden in den folgenden Abschnitten noch ausführlicher diskutiert.

Ändert sich die Zusammensetzung der Gruppe, beispielsweise durch den Beitritt eines neuen Mitglieds, so wird dies der Anwendung vom Gruppenkommunikationssystem durch ein Sichtereignis angezeigt. Beispiele für solche Sichtereignisse sind:

- Ein- und Austritt von Gruppenmitgliedern,
- Status- bzw. Eigenschaftsänderungen von Gruppenmitgliedern oder

- die Unterbrechung der Kommunikation zwischen ihnen.

Für die folgende Diskussion der Eigenschaften von Gruppenkommunikationssystemen sind bezüglich der Übertragung der Nachrichten zwei Begriffe zu unterscheiden:

- Als *Empfang* wird das Eintreffen einer Nachricht bei einem Gruppenmitglied beschrieben, welches sie i.d.R. in die Eingangswarteschlange einordnet.
- Als *Zustellung* wird die Auslieferung der Nachricht durch den Gruppenkommunikationsdienst an die Anwendung bezeichnet. Sofern für die Zustellung eine bestimmte Nachrichtenordnung einzuhalten ist, kommt es zwischen dem Empfang der Nachricht und der Auslieferung an die Anwendung zu einer nicht bekannten Verzögerung.

4.2 Sichten und Zustellungsprinzipien in Gruppenkommunikationssystemen

Ein von einem GKS verwaltetes Gruppenmitglied wird in der Regel als *Peer* p, die Menge der möglichen Nutzer einer mobilen kollaborativen Anwendung wiederum wird als Peers $\mathbb{P}$ bezeichnet.

Der *Membership Service* verwaltet eine Liste der gegenwärtig aktiven und in der Gruppe verbundenen Peers und stellt diese Liste anderen Diensten als *Sicht* (engl. *View*) zur Verfügung. Die Sicht V ist definiert als $V = (id, members)$, wobei id ($id \in VID$) die Sichtkennung und $members$ ($members \in 2^P$) die Menge der Mitglieder der aktuellen Sicht bezeichnet. Die Menge VID der Sichtenkennungen (engl. *View IDs*) ist partiell über den Operator $<$ geordnet. Ändert sich die Sicht, so wird ein Sichtänderungsereignis (engl. *view change event*) mit der neuen Sicht angezeigt. Allgemein wird dieser Vorgang Installation oder Einrichtung einer Sicht in der Gruppe genannt. Dies erfolgt durch den Gruppenverwaltungsdienst (engl. *Membership Service*). Eine neue Sicht wird beispielsweise installiert, wenn Peers der Gruppe beitreten, aus dieser austreten oder deren Eigenschaften sich ändern. Die Eigenschaften der angezeigten Sicht können über $V.id$ bzw. $V.members$ ausgelesen werden. Im Laufe einer Gruppenkommunikationssitzung werden in der Regel mehrere Sichten installiert.

4.2.1 Basistransfereigenschaften eines Gruppenkommunikationssystems

Gruppenkommunikationssysteme können verschieden realisiert werden. Allen gemein ist eine Menge von Basistransfereigenschaften, welche sie den Nutzern zur Verfügung stellen. Diese beziehen sich auf den Versand einer Nachricht innerhalb der Gruppe. Die vier wesentlichsten Eigenschaften, die im Folgenden zur Charakterisierung der Protokollstrukturen benötigt werden, sollen hier kurz eingeführt werden:

- Die *Selbstinklusion* besagt, dass eine von Peer p an die Gruppe versendete Nachricht auch ihm selbst zugestellt wird. Der Vorteil der Selbstinklusion besteht darin, dass durch den Empfang der Nachricht das Senden gegenüber dem Peer implizit bestätigt wird.
- Die *Duplikationsfreiheit* garantiert, dass alle versandten Nachrichten maximal einmal an die Empfänger ausgeliefert werden. Dies entspricht der at-most-once Semantik.
- Die *Zustellintegrität* drückt aus, dass nur Nachrichten von fehlerfrei arbeitenden Peers in der Gruppe zugestellt werden. Fehlerhafte Peers können also weder Nachrichten an die Gruppe senden noch werden durch das Gruppenkommunikationssystem selbständig Nachrichten erzeugt.
- Die *Zustellungslebendigkeit* besagt, dass jede Nachricht, welche durch einen Peer p in Sicht V versandt wurde, von allen fehlerfreien Peers in V empfangen wird. Damit wird sichergestellt, dass es immer einen Fortschritt in der Protokollabarbeitung gibt.

Das den Gruppenkommunikationssystemen zugrundeliegende theoretische Modell bezieht seine Aussagen und Garantien nur auf fehlerfreie Peers, nicht jedoch auf fehlerbehaftete Peers. Ein fehlerhafter Peer könnte sich beispielsweise als Gruppenmitglied ausgeben, die Gruppe mit Nachrichten fluten oder deren Konsistenz beeinträchtigen. Es ist daher die Aufgabe des Gruppenkommunikationssystems, fehlerhafte Peers zu erkennen und von der weiteren Kommunikation auszuschließen. Dazu werden Fehlererkennungsmechanismen genutzt, welche auf den hier ausgeführten Eigenschaften aufbauen. Detailliertere Ausführungen zu den Fehlererkennungsmechanismen folgen im Kapitel 6. Weitere Eigenschaften von Gruppenkommunikationssystemen sind im Anhang A vorgestellt.

4.2.2 Ordnungen und Zuverlässigkeitseigenschaften

Gruppenkommunikationssysteme nutzen für die Nachrichtenübertragung in der Regel Multicasttransferdienste (engl. *Multicast Transfer Services*), welche durch unterschiedliche Ordnungs- und Zuverlässigkeitseigenschaften insbesondere hinsichtlich der Einhaltung der Nachrichtenordnung und der Zuverlässigkeit der Übertragung gekennzeichnet sind [164].

Sie unterschieden sich durch die Art und Weise der Zustellung der Nachrichten an die einzelnen Gruppenmitglieder, welche durch mehrere Gruppenmitglieder zu verschiedenen Zeitpunkten an die Gruppe versandt wurden. Die Reihenfolge, in der die Nachrichten an die Peers zugestellt werden, nennt man *Nachrichtenordnung*. Diese ist eine globale Gruppeneigenschaft, welche die erlaubten Nachrichtenreihenfolgen über alle Sichten hinweg definiert.

Die *Zuverlässigkeitsgarantie* eines Multicastdienstes definiert, ob die lokale Nachrichtenordnung eines Peers Ausfallstellen (fehlende, nicht ausgelieferte Nachrichten) beinhalten darf. Dadurch werden die erlaubten Nachrichtenmengen über alle Sichten hinweg definiert. Eine zuverlässige geordnete Zustellung über Sichtgrenzen hinweg ist komplexer als die Zustellung innerhalb einer Sicht und wird beispielsweise in Chockler *et al.* [55] diskutiert.

Um das Verwerfen von Nachrichten sowie Blockaden zu vermeiden, muss für jede versandte Nachricht entschieden werden, welche Zuverlässigkeitsgarantien zweckmäßig sind. Der Entwickler muss entscheiden, welche Zustellungseigenschaften und welche Nachrichtenordnung die Anwendung erfordert. Aus der Vielzahl der bekannten Ansätze (z. B. [10, 32, 48, 161, 164, 231]) für eine geordnet gestaltete Nachrichtenübertragung in einem Gruppenkommunikationssystem werden hier nur die betrachtet, welche die erfolgreiche Übertragung an alle fehlerfreien Peers einer Gruppe garantieren[2]. Die am häufigsten genutzten Ansätze sind [67]:

1. FIFO Multicast
2. Kausaler Multicast
3. Total geordneter Multicast[3]

***First In First Out* (FIFO) Multicast** Ein FIFO-Multicastdienst [113, 191] garantiert, dass Nachrichten eines Senders bei allen Gruppenmitgliedern in der exakt gleichen Reihenfolge empfangen werden, wie sie gesendet wurden, d. h. wenn ein Peer p zwei Nachrichten m und m' sendet, empfängt sie ein fehlerfreier Empfänger auch in diese Reihenfolge. Für einen fehlerbehafteten Peer ist die Empfangsreihenfolge der Nachrichten jedoch nicht definiert. Eine *zuverlässige FIFO-Übertragung* bezeichnet einen Multicasttransfer, bei dem garantiert ist, dass jeder fehlerfreie Peer sowohl m als auch m' empfangen hat.

[2] für eine Formaldefinition siehe Anhang A
[3] Auch atomarer Multicast oder übereinstimmender Multicast genannt.

Kausaler Multicast Die kausale Ordnung [210] wurde von Lamport auf Basis der FIFO-Ordnung entwickelt [136]. Diese besagt, dass eine Nachricht m' als Antwort auf eine Nachricht m bei jedem Peer, der m empfangen hat, immer nach m zugestellt wird. Auf Basis der kausalen Ordnung lässt sich ein zuverlässiger kausaler Multicastdienst definieren. Der CBCAST-Dienst in ISIS [33] ordnete als einer der ersten Dienste die Nachrichten kausal.

Wird durch einen Peer s eine Nachricht m'' versendet, welche nicht kausal abhängig zu m und m' ist, so kann m'' unabhängig von m und m' zugestellt werden. Eine kausale Ordnung der Nachrichten ist beispielsweise in Chats nützlich, welche garantieren müssen, dass eine Antwort nicht vor der eigentlichen auslösenden Nachricht zugestellt wird.

Total geordneter Multicast Der total geordnete Multicast ist eine Weiterentwicklung des kausalen Multicasts. Er wird am häufigsten in Gruppenkommunikationssystemen verwendet[67]. Die wichtigsten Varianten[4] sind:

- die strenge totale Ordnung und
- die schwache totale Ordnung.

Die strenge totale Ordnung (engl. *strong total order*) stellt sicher, dass die Zustellreihenfolge für alle Nachrichten bei allen fehlerfreien Peers identisch ist. Sie fordert die gleiche Zustellreihenfolge auch für Peers, welche von der Gruppe getrennt werden. Diese Eigenschaft ist im Fehlerfall (bei Verlust der Verbindung zwischen den Gruppenteilnehmern) jedoch nur schwer sicherzustellen.

Die schwache totale Ordnung (engl. *weak total order*) fordert dieselbe Nachrichtenordnung, jedoch nur für miteinander verbundene Peers. Es wird daher verlangt, dass Peers, welche gemeinsam von Sicht V in Sicht V' wechseln, die Nachrichten in V in derselben Reihenfolge empfangen haben. Für Peers, welche beispielsweise aufgrund von Kommunikationsunterbrechungen den Wechsel nach V' nicht vollziehen und für immer in V verweilen (im Sinne ihrer letzten Sicht), muss gemäß der Definition der schwachen totalen Ordnung sichergestellt werden, dass für diese in V ebenso die gleiche Nachrichtenreihenfolge gilt.

Im Fehlerfall kann diese Forderung nicht immer gewährleistet werden, da es möglich ist, dass Änderungen am gemeinsamen Wissen bei nicht verbunden Peers in einer anderen Reihenfolge ausgeführt werden. Die nicht verbundenen Peers bilden von den verbundenen Peers unabhängige Partitionen, wodurch das gemeinsame Wissen in unabhängige Instanzen getrennt wird. Die Konsistenz des Wissens innerhalb der Gruppe kann nach Beendigung der Partitionierung in diesem Fall nicht mehr garantiert werden.

4.2.3 Zustellungseigenschaften

Zustellungseigenschaften werden in Gruppenkommunikationssystemen verwendet, um die Konsistenz der Gruppe zu sichern. Unabhängig von der verwendeten Nachrichtenordnung können für einzelne Nachrichten oder für alle ausgetauschten Nachrichten Zustellungseigenschaften definiert werden. Diese Zustellungseigenschaften erweitern somit die zuvor eingeführten Basistransfereigenschaften und stellen strengere Anforderungen an die Zustellung einer Nachricht. Friedman und van Renesse [87] zeigen, dass es für ein konsistentes Gruppenwissen notwendig ist, solche Zustellungseigenschaften mit einer geeigneten Nachrichtenordnung zu kombinieren, um die Konsistenz global sicherzustellen[5]. Zur Sicherung eines globalen Wissens sollten u. a. folgende Zustellungseigenschaften erfüllte sein:

- Sendesichtzustellung,
- Gruppensichtzustellung.

[4] Ein Überblick über die Vielzahl von existierenden Varianten geben Defago *et al.* in [67] oder Wilhelm und Schiper [217].

[5] Für eine Definition der Eigenschaften siehe Anhang A

Sendesichtzustellung In sichtbewussten (engl. *view aware*) Gruppenkommunikationssystemen werden Ereignisse wie das Senden und Empfangen einer Nachricht jeweils einer Sicht zugeordnet. Die so genannte Sendesichtzustellung (*sending view delivery*) besagt, dass eine Nachricht in derselben Sicht an die Empfänger-Peers zugestellt wird, in der sie gesendet wurde. Wenn also ein Peer p eine Nachricht in Sicht V empfängt, welche durch einen Peer q in Sicht V' versandt wurde, so muss gelten:

$$V = V'.$$

Wird eine Nachricht außerhalb der Sendesicht empfangen, wird sie verworfen, um die Konsistenz zu sichern, da jede sichtbewusste Nachricht immer in einem spezifischen Anwendungskontext gesendet wird. Wenn beispielsweise ein Roboter eine neue Suchstrategie vorschlagen möchte und parallel dazu ein anderer Roboter die Gruppe verlässt, ist die Gültigkeit des Vorschlags abhängig von der Nachrichtenordnung. Wird der Gruppenaustritt vor dem Vorschlag einer neuen Suchstrategie ausgeliefert, ändert sich die Gruppenzusammensetzung und damit die Sicht. Als Konsequenz ist der Suchvorschlag wegen der geänderten Sicht nicht mehr gültig. Ordnet der Transferdienst die Nachrichten invers, so können beide Nachrichten zugestellt werden (im Sinne von: erst der Suchvorschlag, dann der Gruppenaustritt).

Durch das Prinzip der Sendesichtzustellung wird der Aufwand für die Sicherung des Anwendungszustands reduziert. Jeder Nachricht wird durch die Sendesicht ein konkreter Anwendungszustand zugeordnet, was weitere Kontextinformationen bei der Verarbeitung der empfangenen Nachricht erübrigt. Als Beispiel ist hier der Versand von Anweisungen an die Roboter zu nennen. Da jeder Peer die Reihenfolge der Gruppenmitglieder in der Sicht kennt, können die Anweisungen direkt als Vektor versandt werden, dessen i-tes Vektorelement dem i-ten Roboter in der aktuellen Sicht zugeordnet ist [87].

In Abhängigkeit von der Gruppengröße und sonstigen Ereignissen kann jedoch die strenge Anwendung der Sendesichtzustellung einen erheblichen Aufwand erzeugen, ohne dass ein signifikanter Anwendungsfortschritt erreicht wird. Löst beispielsweise in einer Gruppe von zehn Robotern der Empfang einer bestimmten Nachricht bei jedem Roboter die Einladung eines weiteren Roboters aus, kann jeweils nur eine Einladung „*gewinnen*“. Die übrigen neun Einladungen werden verworfen und müssen in der neuen Sicht erneut an die Gruppe versandt werden. Dadurch können über $n!$ –Sendeversuche ($n = zehn$) entstehen, was zu einem erheblichen Aufwand bei den Gruppenmitgliedern führt.

Blockierende Zustellung Alternativ zum Verwerfen diskutieren Friedman und van Renesse [87] das Blockieren des Sendevorgangs eines jeden Peers, bevor eine neue Sicht installiert wird. Die Anwendung wird über den Beginn einer Blockierung informiert. Bis zu deren Ende werden durch das Gruppenkommunikationssystem von der Anwendung sowie von anderen Peers keine neuen Nachrichten mehr angenommen und eine „*Flush*“-Operation ausgeführt. Diese Operation leert die Sende- und Empfangspuffer des Multicasttransferdienstes, indem begonnene Nachrichtenübertragungen abgearbeitet werden (im Sinne von übertragen, bestätigen, ausliefern, etc.). Dadurch wird das Verwerfen von Nachrichten aufgrund eines Sichtwechsels vermieden. Sind alle Nachrichten verarbeitet, wird die neue Sicht eingerichtet und die Blockierung aufgehoben. Nachteilig dabei ist, dass die Blockierungsdauer nur bedingt abgeschätzt werden kann, wodurch die mobile kollaborative Anwendung potentiell träge wird. Im Allgemeinen liefert der Blockierungsansatz gute Ergebnisse für kleinere Gruppen, wobei jedoch mit zunehmender Gruppengröße auch die Blockierungszeit zunimmt [87].

Gruppensichtzustellung Die schwächste Zustellungsvariante ist die Gruppensichtzustellung (engl. *same view delivery*). Dabei wird eine durch einen Peer empfangene Nachricht in der gleichen Sicht zugestellt, in der alle anderen Peers diese Nachricht zustellen. Die Zustellung *kann* also unabhängig von der Absendersicht sein, *muss* jedoch in einer gemeinsamen Sicht erfolgen.

4.3 Herausforderungen der Konsistenzsicherung

Die eingeführten Prinzipien und Eigenschaften von Gruppenkommunikationssystemen beinhalten eine Reihe von Schwierigkeiten für die Sicherung der Konsistenz des Gruppenwissens. Die wichtigsten sollen hier herausgehoben werden:

1. Die Instanzen des Gruppenkommunikationssystems sind über die Gruppe asynchron gekoppelt. Die Verbindungen zwischen den einzelnen Peers können unterschiedliche Verzögerungen aufweisen, da die zu passierenden Zwischensysteme verschieden belastet sind. Dadurch werden Gruppennachrichten durch die einzelnen Peers zu unterschiedlichen Zeitpunkten empfangen. Nachrichtenverluste und daraus resultierende wiederholte Übertragungen können die zeitliche Spreizung innerhalb der Gruppe weiter vergrößern.

2. Die Zustellung der Gruppennachricht an die Anwendung wird durch weitere Faktoren verzögert, wie beispielsweise:

   - den Pufferfüllstand des Gruppenkommunikationssystems,
   - die verwendete Nachrichtenordnung, sowie
   - die genutzten Paradigmen zur Konsistenzsicherung (siehe unten).

3. Durch ausbleibende bzw. verzögerte Nachrichten (im Sinne von Bestätigungen, fehlenden Nachrichten, etc.) können einzelne Peers veranlasst werden, Nachrichten lokal zu verwerfen. Dies beinhaltet grundsätzlich das Risiko, dass das gemeinsame Wissen der Gruppe „*auseinander driftet*“. Diese Situationen werden auch als *Split Brain*[6] bezeichnet. Das nachfolgend eingeführte Paradigma der virtuellen Synchronität und deren Variationen können dieses Problem nicht vollständig lösen, vor allem nicht in mobilen Umgebungen. Zur Minimierung dieses Risikos ist eine geeignete Abwägung zwischen:

   - der Anzahl von wiederholten Nachrichtenübertragungen und
   - der Dauer der Warteperioden für die wiederholte Anforderung von Bestätigungen bzw. vor dem Verwerfen von Nachrichten

   zu finden. Des Weiteren müssen Operationen definiert werden, welche *Split Brain* Situationen erkennen und geeignet behandeln können. Grundlage für diese Operationen ist die Frage nach den zu erwartenden Folgen, wenn eine Nachricht endgültig nicht an eines oder mehrere Gruppenmitglieder zugestellt werden kann.

Diese Besonderheiten beeinflussen die Verwaltung des globalen Wissens. Im mobilen Umfeld nimmt die Wahrscheinlichkeit des Eintretens von Nachrichtenverlusten zu. Die Übertragungsverzögerungen der einzelnen Nachrichten steigen, was die Situation weiter verschärft. Die im Folgenden vorgestellten Paradigmen zur Sicherung der Konsistenz globalen Wissens berücksichtigen diese Herausforderungen, um Inkonsistenzen innerhalb der Gruppe zu verhindern. Sie stellen damit immer eine Abwägung zwischen den übergreifenden Themen Konsistenz, Reaktionsfähigkeit und Fehlerdetektion dar.

4.4 Paradigma der Virtuellen Synchronität

Viele dezentrale GKS nutzen für die Sicherung der Konsistenz des globalen Wissens in einer geschlossenen Gruppe das Paradigma der *Virtuellen Synchronität* (engl. *virtual synchronity*) [31]. Innerhalb einer Gruppe, welche Daten auf Basis der virtuelle Synchronität verarbeitet, haben alle Gruppenmitglieder die gleiche Sicht auf die Gruppe und können damit verteilte Entscheidungen treffen. Typische Anwendungsfälle für dieses Paradigma sind verteilte Datenbanken und Konferenzsysteme. Verteilte Datenbanken halten mit Hilfe des VS-Paradigmas die Daten zwischen den einzelnen Instanzen synchron. Sollen in einer verteilten Datenbank Daten in eine Tabelle eingefügt, geändert

[6] Siehe dazu: `http://www.jgroups.org/manual/html/user-advanced.html#d0e3041`

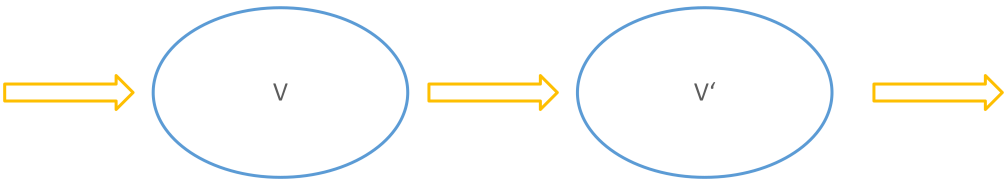


Abbildung 4.3: Sichtübergang der Verwendung des VS-Paradigma

oder gelöscht werden, so erfolgt dies auf Basis der virtuellen Synchronität. Die einzelnen Datenbankinstanzen sind dabei die Mitglieder der geschlossenen Gruppe. Analog wird das Paradigma bei den verteilten Objekten genutzt. Jeder Gruppenteilnehmer bietet einen lokalen Speicher, in denen eine Kopie der ausfallsicheren Objekte abgelegt wird. Leseoperationen können damit direkt auf diesen Objekten erfolgen. Alle übrigen Operationen werden zwischen den Nutzern der verteilten Objekte mit Hilfe des Paradigmas koordiniert.

4.4.1 Funktionsprinzip

Das Paradigma der virtuellen Synchronität stellt sicher, dass zwei Peers die Sicht V' in V installieren, nachdem sie die gleiche Teilmenge $m \in M$ in V empfangen haben. Haben zwei Peers, ausgehend vom gleichen Anwendungszustand, genau die gleiche Teilmenge an Nachrichten erhalten, so gewährleistet die virtuelle Synchronität, dass die Nachrichtenreihenfolge bei beiden Peers die gleiche ist und somit beide Peers denselben Anwendungszustand erreicht haben – die gemeinsame Sicht V'. Mit anderen Worten: Über die empfangenen Gruppennachrichten wurde der gemeinsame Anwendungszustand beider Peers so modifiziert, dass der resultierende Zustand identisch ist. Ein expliziter Zustandstransfer, wie er bei anderen Ansätzen notwendig ist, entfällt dadurch. Für die Umsetzung der virtuellen Synchronität ist die Sendesichtzustellungseigenschaft oder die blockierende Zustellung notwendig [87].

Abbildung 4.3 zeigt das Einrichten einer neuen Sicht bei Verwendung der virtuellen Synchronität. Es handelt sich dabei um einen regulären Übergang von Sicht V zu V', ausgelöst etwa durch einen Gruppeneintritt. Kann beispielsweise ein Roboter aufgrund einer mangelhaften Netzabdeckung die notwendigen Nachrichten für den Sichtübergang nicht empfangen, so wird Sicht V' bei ihm nicht installiert. Er ist somit kein Mitglied der Sicht V' und der Gruppe (Der jeweils gemeinsame Anwendungszustand wird in der Abbildung durch die blaue Ellipse illustriert).

Deshalb setzt die Umsetzung der virtuellen Synchronität eine zuverlässige unterbrechungsfreie Verbindung zwischen den einzelnen Peers voraus. Der Nachrichtenaustausch muss durch ein total geordnetes Multicast gesichert werden, was einen sehr hohen Kommunikationsaufwand induziert.

Üblicherweise werden Gruppenkommunikationssysteme mit virtueller Synchronität in Rechenzentren oder zuverlässigen Netzumgebungen eingesetzt. Mobile Kommunikationsumgebungen erfüllen diese Voraussetzungen nicht. Aus diesem Grund werden in traditionellen Gruppenkommunikationssystemen, wie ISIS [33] oder GCP [231], Peers, die sich nicht protokollkonform verhalten beziehungsweise Nachrichten nicht bestätigen, unverzüglich aus der Gruppe ausgeschlossen, um deren Konsistenz zu sichern. Das ist für eine Reihe von Anwendungen durchaus sinnvoll. So erkennen beispielsweise die Teilnehmer einer Audio- oder Videokonferenz, dass ein oder mehrere Teilnehmer ausgefallen sind. Sie können nun entscheiden, ob sie die Diskussion fortsetzen können oder warten müssen, bis diese(r) Teilnehmer der Konferenz wieder beigetreten sind (ist). Ohne diesen Mechanismus würde ein Teil der Teilnehmer die Diskussion nicht bemerken.

Das Paradigma der virtuellen Synchronität ermöglicht verteilte Entscheidungen und stellt eine Implementierung des Zustandstransferansatzes[7] von Lamport [137] dar. Dieser wird auch als ein „*gemeinsames Fortschreiten*“ von Sicht zu Sicht interpretiert. Hiltunen

[7] engl. *state transfer approach*

und Schlichting [113] führen aus, wie der dazu notwendige gemeinsame konsistente Anwendungszustand der Gruppe durch die Verteilung aller Gruppennachrichten mit Hilfe eines total geordneten Multicasts erreicht wird. Anwendungen, welche auf dem Zustandstransferansatz basieren, ändern ihren Zustand auf der Basis von Nachrichten anderer Gruppenmitglieder. Der Zustandstransfer ist über die Gruppenkommunikation sichergestellt.

4.4.2 Varianten des VS-Paradigmas

Der Nachteil der virtuellen Synchronität – der hohe Kommunikationsaufwand für die total geordnete Nachrichtenübertragung sowie die Forderung nach einer stabilen Kommunikationsverbindung zur Vermeidung von Gruppenausschlüssen und Synchronisationsfehlern – hat zu mehreren Erweiterungen des Paradigmas geführt:

***Erweiterte Virtuelle Synchronität* (EVS)** Der oben diskutierte Gruppenausschluss zur Sicherung der virtuellen Synchronität hat zur Folge, dass fehlerhafte Peers nach Beseitigung ihrer Kommunikationsprobleme nur explizit als neue Mitglieder wieder in die Gruppe eingeladen werden können. Im Falle einer Gruppenpartitionierung können ferner nur die Mitglieder der so genannten primären Partition (zumeist mehr als 50 % der Gruppenmitglieder) weiterarbeiten, alle weiteren blockieren.

Moser *et al.* schlugen dafür bei der Entwicklung des Totem-Protokolls [162] die EVS (engl. *extended virtual synchrony*) vor, welche auch im Falle eines möglichen Kommunikationsausfalls die Verarbeitung einzelner Nachrichten zulässt, indem die „*all or nothing*“ Semantik der virtuellen Synchronität durch eine „*almost all or nothing*“ Semantik[8] ersetzt wird.

Eingesetzt wird die erweiterte virtuelle Synchronität z. B. in Flugbuchungssystemen [162]. Diese Systeme sollen auch im Falle einer Netzpartitionierung so viele Tickets wie möglich verkaufen – auch über Systeme, welche nicht der primären Partition angehören. Der Verkauf erfolgt auf Basis des lokal verfügbaren globalen Wissens. Mögliche Inkonsistenzen (im Sinne von überbuchten Flügen) werden zu Gunsten des Ticketverkaufs akzeptiert. Das gleiche Prinzip findet ebenfalls bei Bankautomaten Anwendung. Hier wird die kollaborative Gruppe von den einzelnen Geldautomaten und den Kontoservern der jeweiligen Banken gebildet. Arbeitet die Gruppe fehlerfrei, wird vor jeder Auszahlung der Verfügungsrahmen des Kontos geprüft. Im Falle einer Netzpartitionierung wird nach einer lokalen Autorisierung die Auszahlung direkt vorgenommen. Bei der späteren Zusammenführung der Partitionen wird die Auszahlung auf dem entsprechenden Konto wirksam. Mögliche Überschreitungen des verfügbaren Guthabens werden durch die Banken akzeptiert.

Dazu wird die virtuelle Synchronität um das Konzept der „*safe messages*“[9] bzw. des „*safe delivery*“[10] erweitert. Es sieht vor, dass ein Peer p eine Nachricht m erst dann an die Anwendung weiterreicht, wenn sie *stabil* ist. Eine Nachricht ist genau dann stabil, wenn alle Peers der aktuellen Sicht diese Nachricht empfangen haben. Dies wird über die eingehenden Bestätigungen von den Empfängern festgestellt. Danach bestätigt der Sender die *safe*-Eigenschaft der Nachricht mittels einer *safe*-Nachricht gegenüber der Gruppe. Trifft dieser Hinweis nicht innerhalb einer bestimmten Zeitspanne ein, so wird die Nachricht durch den Empfänger verworfen. Bleibt die *safe*-Nachricht auch nur bei einzelnen Gruppenmitgliedern aus, kann es durch das Verwerfen zu inkonsistenten Gruppenzuständen kommen. Die erweiterte virtuelle Synchronität definiert keine Operationen, um solch eine Situation zu behandeln.

Abbildung 4.4 veranschaulicht das Prinzip. In der Sicht V empfangen die Peers p und q die Nachricht m jeweils zu unterschiedlichen Zeitpunkten. Mit dem Erhalt der *safe*-Eigenschaft für m in V wird diese Nachricht bei Peer p zustellbar. Bei Peer q ist dies nicht möglich, da dieser noch keine *safe*-Nachricht empfangen hat.

[8] deutsch: *fast* bzw. *annähernd alle oder keiner*

[9] deutsch: *sichere Nachricht* im Sinne von: Zustellbarkeitsnachricht für eine vorausgegangene Nachricht

[10] deutsch im Sinne von: *sichere Zustellung*

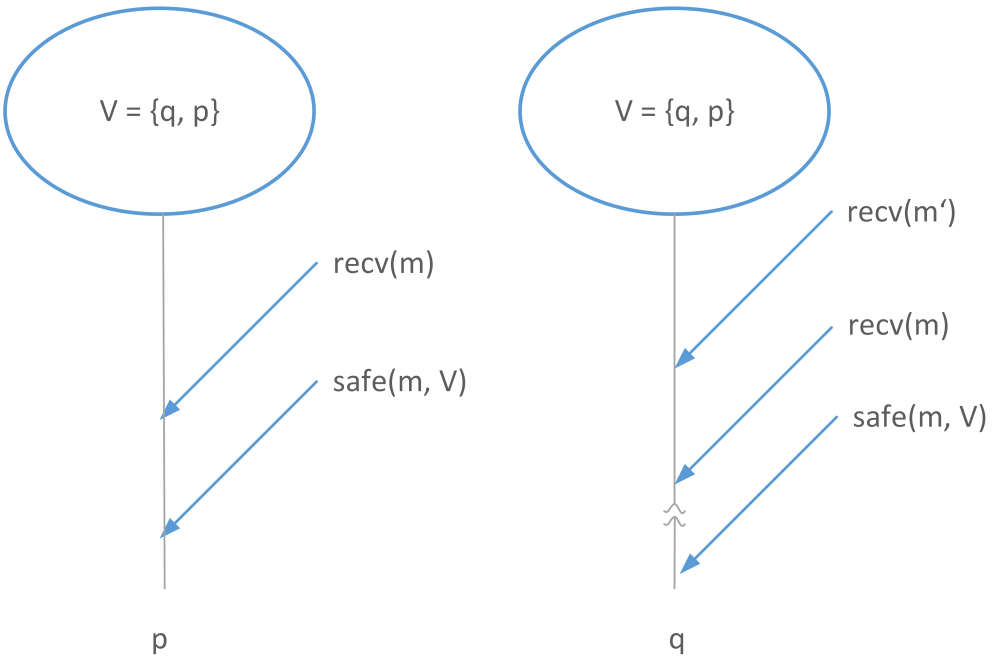


Abbildung 4.4: Konzept der *safe-messages* am Beispiel des Austausches der Nachrichten m und m'

Durch dieses Konzept verzögert sich die Auslieferung der Nachrichten an die Anwendung. In [80] wird eine vielgenutzte Variante diskutiert, welche diesen Nachteil nicht hat. Die Nachrichtenauslieferung ist hier mit einem sogenannten *safe indication*-Primitive verbunden, das der Anwendung anzeigt wird, wenn die ausgelieferte Nachricht stabil ist. Dadurch wird die endgültige Entscheidung, ob die Nachricht verarbeitet oder verworfen wird, vom Gruppenkommunikationsprotokoll auf die Anwendungsebene verlagert. Diese Variante wird bei dem geschilderten Anwendungsbeispiel des Flugbuchungssystems und der Geldautomaten eingesetzt. Die Gesamtverzögerung der einzelnen Nachrichten bleibt gegenüber dem ursprünglichen Ansatz der erweiterten virtuellen Synchronität unverändert. Zerfällt die Gruppe jedoch in mehrere Partitionen, so kann die Synchronität der Gruppe nicht mehr gesichert werden. Das Problem der potentiell inkonsistenten Gruppenzustände durch ausbleibende *safe*-Nachrichten besteht auch bei dieser Variante.

***Schwache Virtuelle Synchronität* (WVS)** Die WVS (engl. *weak virtual synchrony*) unterstützt im Gegensatz zu den beiden vorangegangenen Varianten die Sendesichtzustellungseigenschaft nicht. Sie wurde von Friedman und van Renesse im Horus-Gruppenkommunikationssystem als Alternative zur virtuellen Synchronität entwickelt [87]. Die schwache virtuelle Synchronität eignet sich für Anwendungen, welche auch während eines Sichtwechsels Nachrichten an die Gruppe senden wollen, ohne dass alle Mitglieder der letzten Sicht diese unbedingt empfangen. Ein Beispiel dafür wäre die Verteilung von Veranstaltungsinformationen innerhalb eines Hotels. Alle angemeldeten Empfänger (z. B. Fernseher oder Tablets in den Gästezimmern) sollen diese Informationen anzeigen. Wird ein Empfänger während der Übertragung beispielsweise durch einen Gast deaktiviert, können die Informationen dennoch an die übrigen ausgeliefert werden. Eine wiederholte Übertragung würde für die Anwendung keine Verbesserung bedeuten.

Bei der schwachen virtuellen Synchronität wechselt das Gruppenkommunikationssystem von der regulären Sicht V über eine vorgeschlagene (engl. *suggested*) Sicht V^* wieder in eine reguläre Sicht V' (siehe Abbildung 4.5)[11]. Eine vorgeschlagene Sicht wird durch das Gruppenkommunikationssystem installiert, wenn der Fehlerdetektor den Ausfall eines Peers bzw. die Unterbrechung der Kommunikation zu diesem erkennt. Dabei

[11] Bedingung: $V.members \in V^*.members$

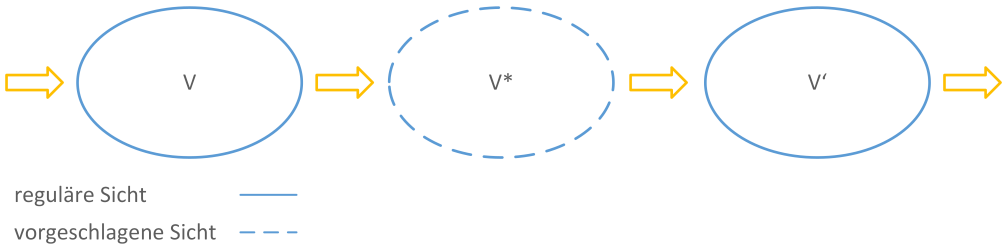


Abbildung 4.5: Sichtübergänge in der schwachen virtuellen Synchronität

können mehrere vorgeschlagene Sichten aufeinanderfolgen. Alle Mitglieder der regulären Sicht V' müssen auch Mitglied der vorgeschlagenen Sicht V^* sein. Eine Identität der Mengen wird nicht gefordert, da beispielsweise weitere Peers aus V^* zwischenzeitlich ausgefallen sein können oder abgeschaltet worden sind. Statt der Sendesichtzustellung wird erwartet, dass jede Nachricht, welche in einer vorgeschlagenen Sicht versendet wird, in der nächsten regulären Sicht zugestellt wird. Alle in V^* versendeten Nachrichten müssen also in V' zugestellt werden, während dies für Nachrichten der Sicht V nicht gilt. Dadurch kann das Gruppenkommunikationssystem auch bei Änderungen an der Gruppenzusammensetzung blockierungsfrei weiterarbeiten.

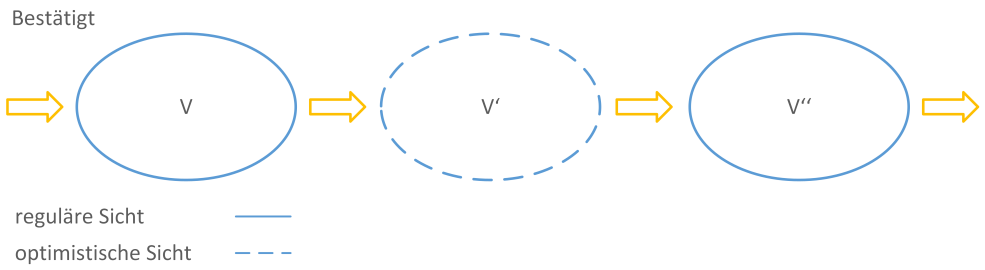
Wurde eine vorgeschlagene Sicht installiert, ist es aus Gründen der Wahrung der Konsistenz nicht erlaubt, dass neue Peers der nächsten regulären Sicht V' beitreten. Soll ein Peer der Gruppe hinzugefügt werden, muss zunächst die Installation der regulären Sicht V' abgewartet und anschließend eine weitere reguläre Sicht V'' mit dem beitretenden Peer eingerichtet werden. Dies erhöht die Anzahl der installierten Sichten und damit den Kommunikationsaufwand.

***Optimistische Virtuelle Synchronität* (OVS)** Die optimistische virtuelle Synchronität (engl. *optimistic virtual synchrony*) von Sussman *et al.* [204] versucht die Nachteile der zuvor beschriebenen Ansätze zu beheben. Sie wurde erstmals für das Gruppenkommunikationssystem Transis umgesetzt. Kann ein Teilnehmer der Gruppe durch einen Kommunikationsausfall kurzzeitig nicht erreicht werden, installiert das Gruppenkommunikationssystem eine optimistische Sicht, vergleichbar mit der vorgeschlagenen Sicht der schwachen virtuellen Synchronität. Im Gegensatz zur vorgeschlagenen Sicht wird bei der optimistischen Sicht angenommen, dass das Gruppenmitglied seine Netzprobleme zeitnah lösen kann und zur Gruppe zurückkehrt. In der optimistischen Sicht werden Nachrichten unter der Annahme ausgetauscht, dass die nächste reguläre Sicht der optimistischen Sicht entspricht. Zusätzlich können Nachrichten durch die Anwendung mit Bedingungen verknüpft werden. Beispiele für solche Bedingungen sind:

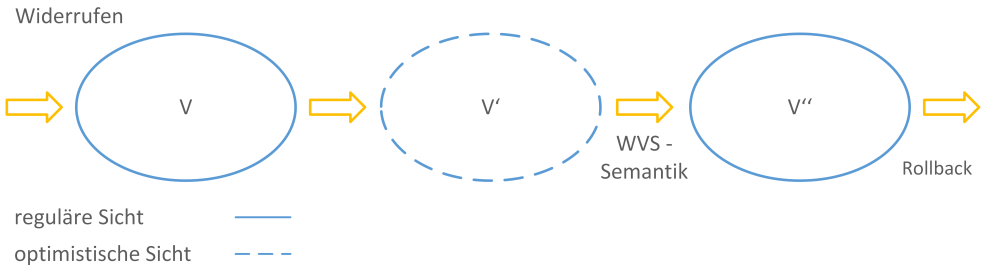
- ausgesuchte Gruppenmitglieder (z. B. der Moderator einer Konferenz) müssen in der Sicht enthalten sein oder
- eine bestimmte Anzahl von Gruppenmitgliedern muss die Nachricht erhalten.

Empfängt ein Gruppenmitglied eine Nachricht während der optimistischen Sicht, so überprüft er zunächst, ob die mit der Nachricht verknüpften Bedingungen erfüllt sind. Ist dies der Fall, wird die Nachricht lokal ausgeliefert, anderenfalls verworfen.

Ein typischer Anwendungsfall für die optimistische virtuelle Synchronität wäre wiederum das oben erwähnte verteilte Datenbanksystem. Wäre für die verschiedenen Datenbanktabellen jeweils ein Mitglied der Gruppe zuständig, würde bei einer Partitionierung im Fall der virtuellen Synchronität die Anwendung blockieren und keine Anfragen mehr verarbeiten. Bei der Verwendung der optimistischen Variante könnte die Anwendung jedoch Anfragen senden, solange das für die jeweilige Tabelle verantwortliche Gruppenmitglied Teil der optimistischen Sicht ist. Ebenso könnte bei quorumbasierten Anwendungen jeweils bestimmt werden, ob noch ausreichend Kopien (das Quorum) von zu bearbeitenden Objekten in der aktuellen Sicht enthalten sind. Auch in diesen Fall könnte die Anwendung ohne Blockierung weiterarbeiten. Auch mobile kollaborative



(a) Erfolgreicher OVS-Sichtübergang



(b) OVS-Sichtübergang im Fehlerfall

Abbildung 4.6: Sichtübergänge bei der Verwendung der optimistischen virtuellen Synchronität

Anwendungen könnten die optimistische virtuelle Synchronität nutzen, da über die optimistische Sicht potentiell kurzzeitige Verbindungsabbrüche mobiler Gruppenmitglieder modelliert werden könnten.

Aufgrund von kurzzeitigen Verbindungsabbrüchen, Überlastungen in Zwischensystemen und anderen Gründen ist in mobilen Umgebungen häufig ein Verlust von Gruppennachrichten oder deren Verwerfen (z. B. aufgrund von nichterfüllten Zustellungseigenschaften) zu erwarten. Als Konsequenz ist die Konsistenz des globalen Wissens innerhalb der Gruppe nicht mehr gesichert. Einen Ansatz für die dadurch notwendig Resynchronisation gibt Sussman *et al.* jedoch nicht. Weitere Anwendungsbeispiele sind in [204] gegeben.

Kann das Mitglied, das den Kontakt zur Gruppe verloren hat, seine Netzprobleme beheben und zur Gruppe zurückkehren, wird die optimistische Sicht bestätigt, anderenfalls widerrufen. Für den Fall eines Widerrufs muss der Anwendungsentwickler die Semantik für den Nachrichtenaustausch in der optimistischen Phase definieren. Zur Wahl stehen die virtuelle und die schwache virtuelle Synchronität. Gilt letztere bei Widerruf der optimistischen Sicht, können Nachrichten, die in der optimistischen Sicht zugestellt wurden, durch die Anwendung weiterhin verwendet werden, sofern die neue Sicht eine Teilmenge der optimistischen Sicht ist. Bei der virtuellen Synchronität werden alle Nachrichten der optimistischen Sicht durch die Anwendung zurückgenommen, d. h. es findet ein *Rollback* statt. Ein Widerrufen der optimistischen Sicht ist im Allgemeinen nur notwendig, wenn während der Installation der regulären Sicht weitere Verbindungsprobleme bei den Teilnehmern der Restgruppe auftreten, d. h. die Kommunikation zu einem weiteren Teilnehmer abbricht [55].

Verbindet sich ein von der Gruppe getrenntes Mitglied wieder mit der Gruppe, so unterscheidet sich sein Anwendungszustand bzw. sein globales Wissen von dem der Gruppe. Damit wäre eine Resynchronisation erforderlich, die jedoch durch die optimistische virtuelle Synchronität nicht unterstützt wird. Dafür ist die Anwendung verantwortlich, wodurch zwar die Blockierungsdauer der kollaborativen Anwendung, nicht jedoch der erforderliche Resynchronisationsaufwand reduziert wird. In Kapitel 6 wird eine Weiterentwicklung der optimistischen virtuellen Synchronität vorgestellt, welche einen neuen Ansatz für die notwendige Resynchronisation und die Nachrichtenauslieferung in der optimistischen Sicht/Phase nutzt.

Den Sichtwechsel der optimistischen virtuellen Synchronität zeigt Abbildung 4.6. Aus der regulären Sicht V wird in die optimistische Sicht V' gewechselt. Kehrt das vermisste Mitglied rechtzeitig in die Gruppe zurück, wird aus der optimistischen Sicht eine reguläre, der gemeinsame Anwendungszustand ist durch die Anwendung wiederherzustellen (Abbildung 4.6a)[12]. Kehrt das Mitglied nicht innerhalb eines bestimmten Zeitraums zurück, wird die optimistische Sicht widerrufen und eine neue reguläre Sicht ohne das vermisste Mitglied installiert (Abbildung 4.6b)[13]. Anschließend ist zu prüfen, ob ein Rollback der empfangenen Nachrichten notwendig ist. Dieser ist gegebenenfalls auszuführen. In diesem Fall sind die durch die Gruppe in der optimistischen Phase erbrachten Synchronisationsaufwände (im Sinne von: Austausch und Speicherung von Nachrichten sowie der damit zusammenhängende Energieverbrauch) verloren.

Zusammenfassend kann festgestellt werden, dass der virtuellen Synchronität und ihren Varianten im Wesentlichen folgende Annahmen zugrunde liegen[55]:

1. Die Konnektivität zwischen den Gruppenteilnehmern ist weitgehend stabil,
2. Unterbrechungen der Kommunikationsverbindungen sind zumeist durch Fehler im Endsystem begründet und
3. Partitionierungen der Gruppe ändern nicht die Kommunikationsadressen der Gruppenteilnehmer.

Diese Annahmen gelten jedoch in mobilen Umgebungen nur eingeschränkt. Um die Auswirkungen auf die Konsistenz des Gruppenwissen zu demonstrieren, wird im folgenden Abschnitt ein Experiment vorgestellt, das zeigt, dass die bisher verwendeten Maßnahmen zur Konsistenzsicherung in mobilen Umgebungen nicht ausreichen. Eine Lösung dieses Problems wird in den Kapiteln 5 bzw. 6 vorgestellt.

4.5 Auswirkungen einer fehlenden Mobilitätsunterstützung auf die Wissenskonsistenz

Die Kommunikation in mobilen Umgebungen ist durch folgende Eigenschaften charakterisiert:

1. Drahtlose Kommunikationsverbindungen haben höhere Fehlerraten, wodurch sie weniger stabil im Vergleich zu drahtgebundenen Medien sind.
2. Aufgrund der Nutzermobilität kann es zu Kommunikationsunterbrechungen unterschiedlicher Dauer kommen, welche nicht ausschließlich durch Endgerätefehler begründet sind. Zusätzlich müssen Fehlerquellen wie fehlende Netzabdeckungen, räumliche Begrenzung der drahtlosen Netze, limitierende Effekte bei der drahtlosen Ausbreitung (z. B. Fading, Streuung, Reflexion, Dämpfung) und mobilitätsinduzierte Netzwechsel berücksichtigt werden.
3. Diese Kommunikationsfehler haben einen direkten Einfluss auf die Gruppe im Falle einer Partitionierung. Der Partitionierungsbegriff muss hier weiter gefasst werden. Bisher galt: Sind Teilnehmer in stationären Netzen nicht erreichbar, so wurde deren Ausfall oder eine Netzpartitionierung angenommen. Konnte die Partitionierung behoben werden, waren die Teilnehmer wieder unter ihrer bereits bekannten Adresse erreichbar. In mobilen Umgebungen kann die Gruppe jedoch auch aufgrund der genannten Kommunikationsfehler oder eines Netzwechsels partitionieren. Die Partitionen existieren getrennt weiter. Bei einem Netzwechsel sind die Teilnehmer potentiell erreichbar, aber zumeist unter einer anderen Kommunikationsadresse, wie beispielsweise bei einem Wechsel aus einem WLAN zu UMTS.

Das Experiment soll am Beispiel des Gruppenkommunikationssystems JGroups [161] die Auswirkungen solcher Netzwechsel auf die Konsistenz des globalen Wissens

[12] Bedingung bestätigt: $V''.members \equiv V.members$
[13] Bedingung widerrufen: $V''.members \equiv V.members \backslash p$

illustrieren und zeigen, dass die in Festnetzen verwendeten Maßnahmen zur Konsistenzsicherung in mobilen Umgebungen nicht ausreichen. In Extremfällen kann es zu einer logischen Partitionierung der Gruppe kommen. Logische Partitionierungen sind dadurch gekennzeichnet, dass Verbindungen auf der Netzebene bestehen bleiben, aber die Instanzen auf Gruppenkommunikationsebene logisch getrennt werden. Die Konsistenz der Gruppe ist dann zumeist nicht mehr sichergestellt. Gründe dafür können beispielsweise verlorene oder verworfene Gruppennachrichten sein. Logische Partitionierungen werden durch das Gruppenkommunikationssystem nicht immer korrekt erkannt bzw. führen zum Blockieren einzelner Instanzen.

Vorbetrachtung Gruppenkommunikationssysteme verfügen oft über einen Fehlerdetektor, welcher die Nichterreichbarkeit einzelner Peers erkennt. Wird eine Unterbrechung der Kommunikation festgestellt, so versucht das Gruppenkommunikationssystem den Peer wieder zu erreichen und – wenn möglich – die Gruppe über spezielle Protokolle[14] wieder zusammenzuführen. Wechselt ein Peer jedoch das Netz, so müssen alle aktuellen Kommunikationssockets geschlossen und daraufhin neu initialisiert werden. Da aktuelle Gruppenkommunikationsprotokolle keine Mobilitätsunterstützung bieten, entspricht dieses Vorgehen einer vollständigen Terminierung der lokalen Instanz des Gruppenkommunikationssystems und einem anschließenden Wiederaufsetzen. Die kollaborative Anwendung bleibt davon unberührt.

Aufbau des Experiments Für das fehlerfreie Zusammenwirken der Roboter im Anwendungsszenario aus Abschnitt 2.2 ist es notwendig, dass die Gruppe immer (auch während Kommunikationsunterbrechungen zu einzelnen Robotern) die korrekte Anzahl von Teilnehmern kennt. Die erfassten Daten sind folglich so in der Gruppe zu verteilen, dass das globale Wissen zu jeder Zeit gesichert ist.

Das Experiment soll eine kollaborative Gruppe von zehn Teilnehmern nachstellen, für die angenommen wird, dass ein Teil der Gruppe eine stabile stationäre Verbindung besitzt, während die Verbindung der mobilen Peers periodisch unterbrochen wird. Nur wenn die Gruppe vollständig ist, d. h. aus *zehn* aktiven Mitgliedern besteht, sendet ein ausgewähltes Mitglied periodisch Nachrichten an die Gruppe[15].

Wenn die Kommunikationsverbindung zu einem mobilen Peer unterbrochen wird, kann dieser eine bestimmte Zeit nicht mehr mit der Gruppe kommunizieren. Es wird angenommen, dass dieser Peer einen Netzwechsel vollzieht, was mit einer erneuten Initialisierung des Sockets verbunden ist. Hat der Peer die Konnektivität wiederhergestellt, tritt er der Gruppe erneut bei. Dieser Zyklus wird insgesamt zehnmal für jeden mobilen Teilnehmer wiederholt. Ein Teilnehmer verarbeitet eine empfangene Nachricht nur dann, wenn die aktuelle Gruppengröße zehn Teilnehmer umfasst.

Ablauf des Experiments Es wurden insgesamt 18 Testläufe durchgeführt, wobei ein Testlauf charakterisiert ist durch:

- die Anzahl der mobilen Peers,
- die Dauer der Kommunikationsunterbrechung,
- das Sendeintervall.

Die Gesamtzahl der Testläufe ergibt sich aus der Variation der charakteristischen Eigenschaften, d. h. der Anzahl der mobilen Peers (eins oder vier), der Dauer der Kommunikationsunterbrechungen (10, 30 oder 60 s) und der Sendeintervalle (1000, 500 und 100 ms). Für einen Zeitraum von einheitlich 30 s haben die mobilen Teilnehmer jeweils eine stabile Verbindung zur Gruppe. Die resultierenden Konfigurationen der Testläufe ist in Tabelle 4.1 zusammengestellt.

Während eines Testlaufs wird durch jeden Gruppenteilnehmer die Anzahl der gesendeten und empfangenen Nachrichten protokolliert. Beim Empfang der Nachrichten

[14] Im Falle von JGroups ist dies das Merge3-Protokoll, siehe dazu: `http://www.jgroups.org/manual-3.x/html/protlist.html#d0e5133`

[15] Die Auswahl des Senders erfolgt über die JGroups-Leader-Algorithmus, vergleiche dazu [161]

Tabelle 4.1: Überblick über die Testläufe für den Konsenstest

Testfall	**Anzahl stationärer Teilnehmer**	**Anzahl mobiler Teilnehmer**	**Dauer der Kommunikationsunterbrechung** [s]	Sendeintervall [ms]
TC_1 1000 ms	9	1	10	1000
TC_1 500 ms	9	1	10	500
TC_1 100 ms	9	1	10	100
TC_2 1000 ms	6	4	10	1000
TC_2 500 ms	6	4	10	500
TC_2 100 ms	6	4	10	100
TC_3 1000 ms	9	1	30	1000
TC_3 500 ms	9	1	30	500
TC_3 100 ms	9	1	30	100
TC_4 1000 ms	6	4	30	1000
TC_4 500 ms	6	4	30	500
TC_4 100 ms	6	4	30	100
TC_5 1000 ms	9	1	60	1000
TC_5 500 ms	9	1	60	500
TC_5 100 ms	9	1	60	100
TC_6 1000 ms	6	4	60	1000
TC_6 500 ms	6	4	60	500
TC_6 100 ms	6	4	60	100

ermittelt jeder Teilnehmer die aktuelle Gruppengröße über eine lokale Abfrage beim Gruppenverwaltungsdienst. Zusätzlich werden für jeden Teilnehmer die installierten Sichten (d. h. die aktuelle Gruppenzusammensetzung und die Anzahl der Teilnehmer) protokolliert. Als Zustelleigenschaft wird die Gruppensichtzustellung verwendet.

Im Verlauf eines Testlaufs werden zunächst jeweils zehn JGroups-Instanzen gestartet und die Gruppe beginnt zu kooperieren. Nach 30 s wird die Kommunikationsverbindung zu dem/den mobilen Teilnehmer(n) für die Dauer der jeweiligen Kommunikationsunterbrechung getrennt. Nach der Unterbrechung wird die Kommunikationsverbindung neu initialisiert und die Verbindung zur Gruppe wiederaufgenommen.

Ergebnisse Ausgehend von der Prämisse, dass

1. der Sender nur sendet, wenn es zehn aktive Gruppenmitglieder gibt,
2. der Multicastdienst von JGroups zuverlässig ist,
3. der Empfänger die eingehende Nachricht nur dann verarbeitet, wenn die aktuelle Gruppengröße zum Empfangszeitpunkt zehn Gruppenmitglieder umfasst,

wird erwartet, dass innerhalb der Gruppe über alle ausgetauschten und verarbeiteten Nachrichten Konsens besteht. Eine Nachricht wird also durch einen Teilnehmer verarbeitet. Wenn dies für alle Gruppenteilnehmer der Fall ist, beträgt der Anteil an Übereinstimmung innerhalb der Gruppe dann 100 %.

Tabelle 4.2 zeigt die wichtigsten Ergebnisse der einzelnen Testläufe. Für jeden Teilnehmer wurde die Anzahl der verarbeiteten Nachrichten gemessen. Daraus wurde für die Gruppe die minimale und maximale Anzahl der durch die Anwendung empfangenen

Tabelle 4.2: Anteil an Übereinstimmung der Testläufe

Testfall	**Anteil an Übereinstimmung** [%]	**Maximal empfangene Nachrichten**	**Minimal empfangene Nachrichten**	**Maximal registrierte Gruppengröße**
TC_1 1000 ms	91,75	340	312	10
TC_1 500 ms	94,32	669	631	10
TC_1 100 ms	94,30	3300	3112	10
TC_2 1000 ms	94,51	328	310	10
TC_2 500 ms	93,95	661	621	10
TC_2 100 ms	43,42	2699	1172	14
TC_3 1000 ms	94,13	341	321	10
TC_3 500 ms	94,03	670	639	10
TC_3 100 ms	93,73	1197	1122	10
TC_4 1000 ms	94,51	328	310	10
TC_4 500 ms	93,94	660	620	10
TC_4 100 ms	94,22	3269	3080	10
TC_5 1000 ms	98,33	660	649	10
TC_5 500 ms	97,58	1322	1290	10
TC_5 100 ms	97,11	6581	6391	10
TC_6 1000 ms	96,97	660	640	10
TC_6 500 ms	97,55	1306	1274	10
TC_6 100 ms	97,38	648	6317	10

Nachrichten bestimmt. Die maximale Anzahl von Nachrichten wurde dabei ausschließlich von stationären Teilnehmern empfangen, die minimale Anzahl dagegen von einem mobilen Teilnehmer. Im Gegensatz zur Erwartung, dass die Anzahl der empfangenen Nachrichten bei allen Teilnehmern identisch ist, treten bei den Testergebnissen deutliche Unterschiede auf. Damit besteht ein Dissens über das gemeinsame Wissen in der Gruppe zwischen den mobilen und stationären Teilnehmern.

Abbildung 4.7 zeigt dieses Problem, wobei für alle Teilnehmer die Anzahl der minimal und maximal empfangenen Nachrichten angegeben ist. Der daraus resultierende Anteil an Übereinstimmung wird in Tabelle 4.2 sowie in Abbildung 4.8 dargestellt. Im Allgemeinen schwankt die Übereinstimmung zwischen 94 % und 98 %. Die Testläufe TC_1 100 ms, TC_2 500 ms und TC_2 100 ms schneiden mit 91,75 %, 93,95 % bzw. 43,42 % teilweise deutlich schlechter ab, was in den zeitlichen Differenzen zwischen wiederholtem Ein- und Austritt des mobilen Teilnehmers und dem Sendeintervall begründet ist.

Eng mit dem Anteil an Übereinstimmung hängt auch die Entwicklung der Gruppenzusammensetzung in den einzelnen Testläufen zusammen. Abbildung 4.9 zeigt die Entwicklung der Gruppenzusammensetzung für den Testlauf TC_1 500 ms mit einem mobilen und neun stationären Teilnehmern. In der Darstellung entspricht die x-Achse nur der logischen Reihenfolge der Änderungen der Gruppenzusammensetzung. Die Änderung der Gruppenzusammensetzung wird der Testanwendung über Sichtänderungsereignisse angezeigt. Durch den Testverlauf und die zeitliche Entkoppelung der einzelnen Teilnehmer existiert keine Korrelation zwischen den unterschiedlichen Kurven. Sie wurden lediglich aus Darstellungsgründen zusammengefasst und sind jeweils einzeln zu interpretieren. Das Ergebnis entspricht dabei der Erwartung vor dem Test: Für den mobilen Teilnehmer ist die Gruppengröße immer zehn, wenn er mit der Gruppe

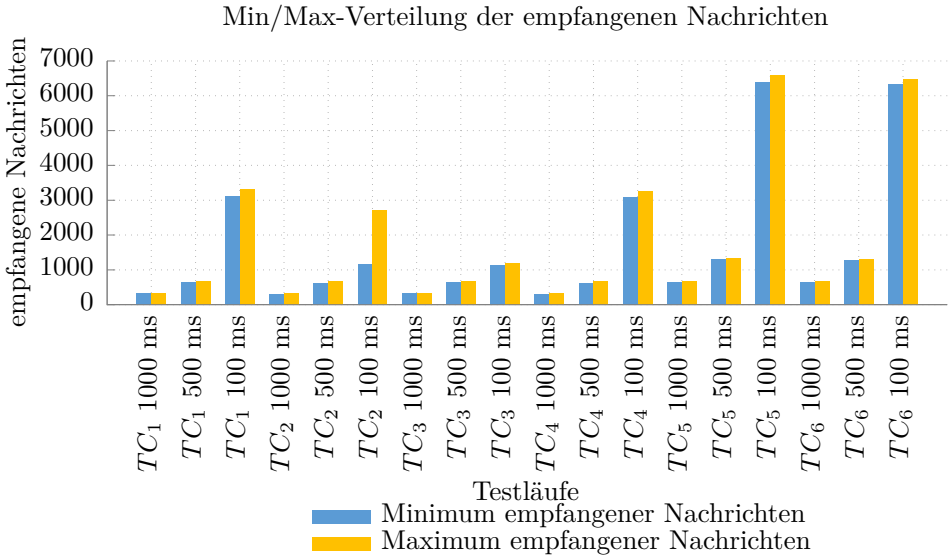


Abbildung 4.7: Min/Max-Verteilung der durch die Testanwendung empfangenen Nachrichten

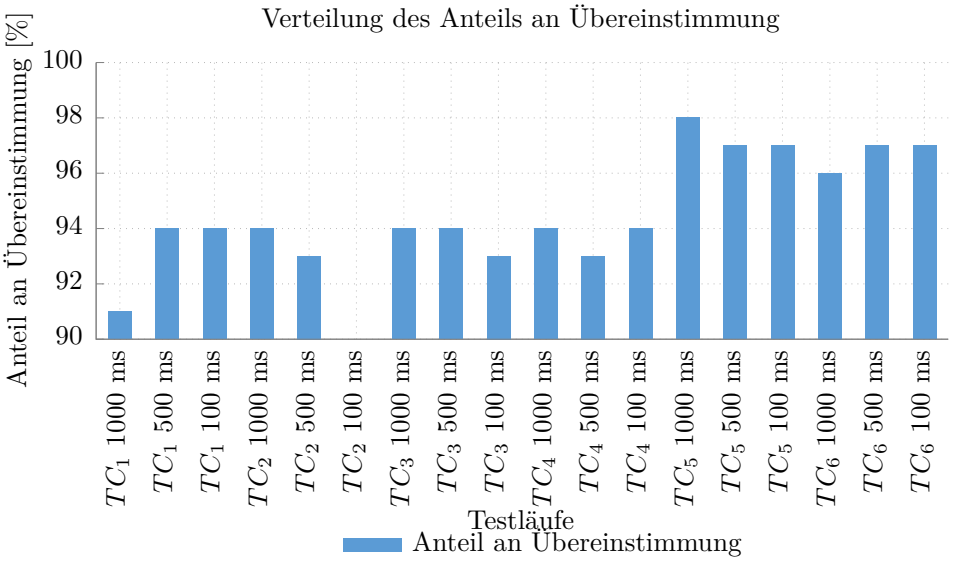


Abbildung 4.8: Verteilung des Anteils an Übereinstimmung auf die unterschiedlichen Testläufe (der Anteil an Übereinstimmung für den Testlauf TC_2 100 ms liegt unter 90 %)

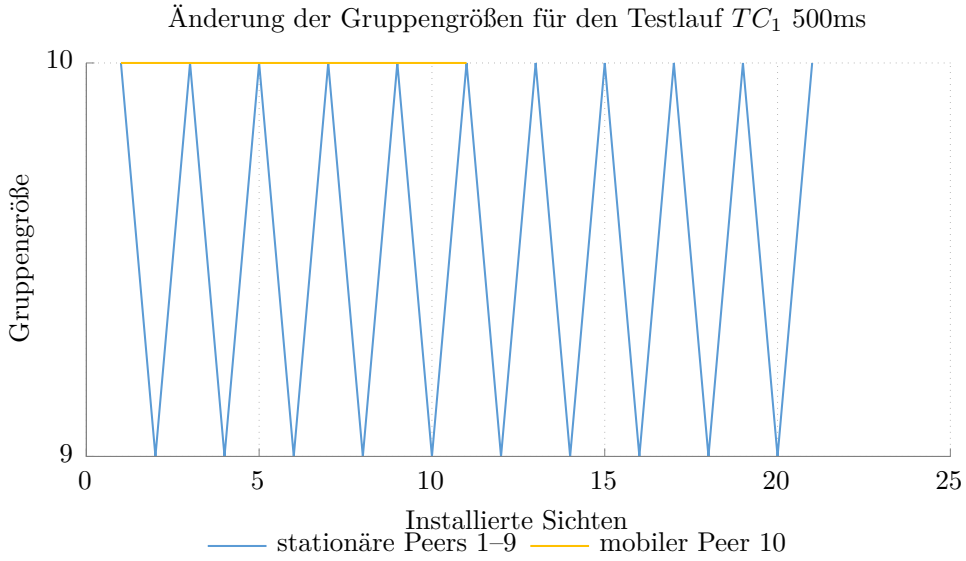


Abbildung 4.9: Exemplarische Darstellung der Änderung der Gruppengröße für den Testlauf TC_1 500 ms

verbunden ist. Für die stationären Teilnehmer schwankt diese zwischen neun und zehn. Dieses Ergebnis stellt sich jedoch nicht bei allen Testläufen ein.

In Testlauf TC_2 100 ms kam es durch die lokalen JGroups-Instanzen zu fehlerhaften Ergebnissen bei der Gruppenzusammensetzung, welche die Probleme der virtuellen Synchronität (vgl. Abschnitt 4.3) besonders gut illustrieren. In Abbildung 4.10 ist die Entwicklung der lokal angenommenen Gruppenzusammensetzung für den Testlauf graphisch dargestellt. Der Lauf wird mit einer Gruppe, bestehend aus sechs stationären und vier mobilen Teilnehmern, durchgeführt. Die stationären Teilnehmer 1, 2 sowie $4-6$ haben im Testverlauf die gleiche (identische) Gruppenzusammensetzung und werden deshalb zusammengefasst (rote Linie). Auffällig ist, dass sich die Gruppenzusammensetzung bei Teilnehmer 3 – dem primären Sender in der Gruppe – deutlich von der der anderen stationären Teilnehmer unterscheidet. Bei diesem und den mobilen Teilnehmern werden bis zu 14 Gruppenmitglieder in der Gruppe angenommen. Die Gruppe bestünde also aus vier Teilnehmern mehr, als überhaupt existieren (da max. 10 JGroups-Instanzen gleichzeitig gestartet werden). Diese Fehlannahme liegt in der Detektionsträgheit des Fehlerdetektors begründet. Alte, nicht mehr existierende Verbindungen zum mobilen Teilnehmer werden als aktiv geführt. Dadurch täuscht dessen „*Wiedereintritt*“ in die Gruppe dieser temporäre fiktive Teilnehmer vor.

Durch die falschen Gruppenannahmen werden viele der versendeten Nachrichten durch einige Gruppenteilnehmer verworfen, da sie die Verarbeitungsanforderung von zehn aktiven Gruppenmitgliedern für nicht erfüllt halten. In diesem Testfall kann somit nicht mehr von einem gemeinsamen Wissen gesprochen werden, da nur 43 % aller Nachrichten tatsächlich alle Gruppenmitglieder erreichten. Hier liegt eine „*Split-Brain*“-Situation vor. Spiker *et al.* zeigen in [201] über verschiedene Experimente, dass beim Einsatz von WLAN MANETs und Wireless Sensor Networks das Gruppenkommunikationsprotokoll ebenfalls keinen Konsens über das globale Wissen oder die Gruppenzusammensetzung zusichern kann.

Zusammenfassend ist festzustellen, dass die gegenwärtig existierenden Gruppenkommunikationssysteme in Szenarien, in denen mobile Teilnehmer das Netz wechseln, ohne zusätzliche Maßnahmen das globale Wissen nicht garantieren können. Ein Konsens in der Gruppe kann trotz gegenteiliger Aussagen durch die lokale Gruppenkommunikationsinstanz nicht sichergestellt werden. Dieser einfache Test zeigt, dass bereits bei einem oder mehreren Teilnehmern, welche aufgrund ihrer Mobilität häufig die Kommunikationsnetze

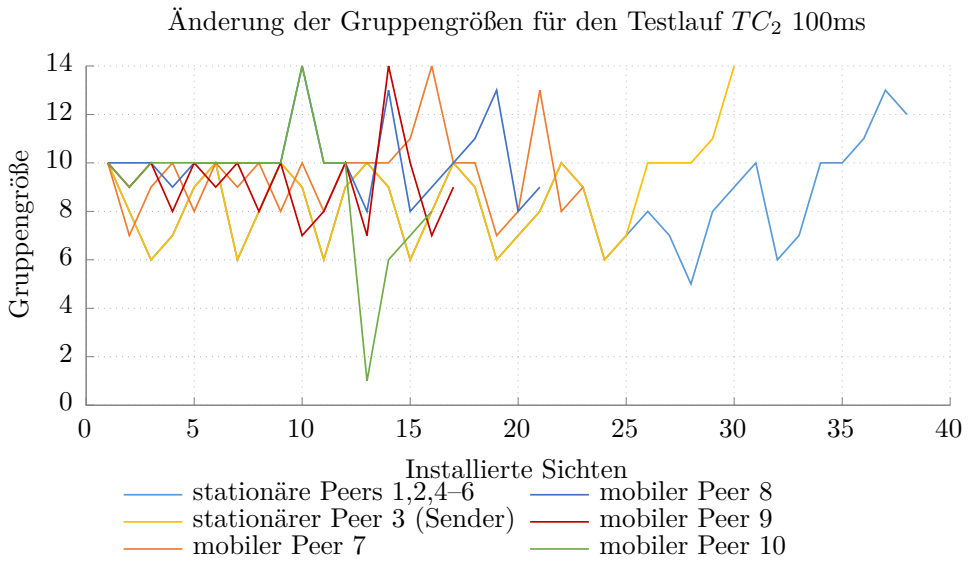


Abbildung 4.10: Exemplarische Darstellung der Änderung der Gruppengröße für den Testlauf TC_2 100 ms

wechseln, das gemeinsame Wissen bezüglich der aktuellen Gruppenzusammensetzung sowie der erfolgreich ausgetauschten Nachrichten fehlerhaft ist.

Teil II

Moversight

Kapitel 5

Das Gruppenkommunikationsprotokoll *Moversight*

Weiter oben wurde gezeigt, dass bisherige Gruppenkommunikationssysteme nur einen Teil der Anforderungen an gruppenorientierte Kommunikationssysteme erfüllen. Viele der untersuchten Ansätze eignen sich kaum für den Einsatz in mobilen kollaborativen Anwendungen. Aus diesem Grunde wurde mit *Moversight* ein neues flexibles P2P-basiertes Gruppenkommunikationsprotokoll für mobile kollaborative Anwendungen entwickelt, das sowohl mobile, nomadische als auch stationäre Teilnehmer [92] unterstützt und auf UDP/IP-aufbaut. Die Teilnehmer sind in einer geschlossenen Gruppe organisiert, der nur auf explizite Einladung beigetreten werden kann. Das Protokoll ist für Gruppengrößen bis zu 100 Mitglieder ausgelegt, um die Kosten für die Aufrechterhaltung der Konsistenz zwischen den Teilnehmern zu beschränken. Aus Sicht des Autors erscheint dies eine angemessene obere Schranke, da mobile Kollaboration mit strengen Konsistenzanforderungen üblicherweise nur eine geringe Anzahl von Teilnehmern involviert.

Moversight stellt verschiedene Gruppentransferdienste bereit, die eine geordnete, zuverlässige und sichtorientierte Datenübertragung gewährleisten. Im Gegensatz zu anderen vergleichbaren Protokollen erlaubt *Moversight* kurzzeitige mobilitätsbedingte Verbindungsunterbrechungen zu einzelnen Teilnehmern. Das Protokoll sichert auch in diesem Fall die Konsistenz des globalen Gruppenwissens und gestattet über eine spezielle Funktion *Rejoin* den Wiedereintritt des Teilnehmers. Zudem unterstützt *Moversight* das Aufteilen der Gruppe in zwei unabhängige Teilgruppen sowie deren (Wieder-)Vereinigung jeweils unter Wahrung der Konsistenz des kollaborativen Gruppenwissens. Es stellt weiterhin verschiedene gruppenweite Verfahren zur Anpassung der Kommunikationstopologie und der Transferdienste an sich ändernde Kommunikationsbedingungen und Gruppenzusammensetzungen sowie einen Dienst zur Fehlererkennung bereit.

Moversight nutzt eine clusterbasierte Topologie für die Kommunikation, welche robust gegenüber Verbindungsabbrüchen mobiler Teilnehmer ist. Die Topologie ist Grundlage aller Dienste von *Moversight* und hat Einfluss auf deren Funktionsweise.

Aufgrund der Komplexität des Protokolls werden seine Funktionen in mehreren Kapiteln beschrieben. Das folgende Kapitel behandelt einführend den grundlegenden Aufbau und die Funktionsweise von *Moversight* sowie die wichtigsten Dienste. Aufbauend darauf wird im Kapitel 6 vertiefend auf die Behandlung der Nutzermobilität eingegangen. Dabei werden insbesondere Fragen des Wiedereintritts von Teilnehmern und der Wiederherstellung der Gruppenkonsistenz diskutiert. Dem schließen sich in den Kapiteln 7, 8 und 9 Darstellungen zu Details der Adaptions- und Wartungsverfahren sowie der Dienste für das Aufteilen und Wiedervereinigen von Gruppen an.

5.1 Architektur

Mobile kollaborative Anwendungen unterscheiden sich in den beabsichtigten Anwendungsszenarien, den Kommunikationsschemata und den Anforderungen an die Dienst-

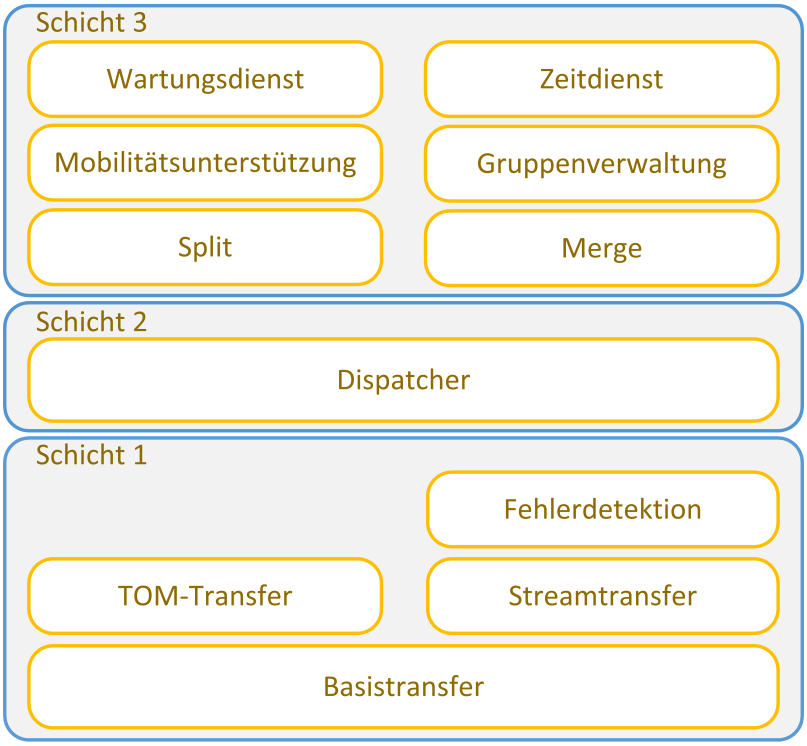


Abbildung 5.1: Die Architektur von *Moversight*

qualität. Um unterschiedlichen Anforderungen gerecht zu werden, nutzt *Moversight* eine erweiterbare modulare Architektur, die in Abbildung 5.1 dargestellt ist. Sie besteht aus einer Reihe von Diensten, welche lose über eine ereignisbasierte Schnittstelle gekoppelt sind. Im Anhang B sind weitere Details gegeben.

Eine Instanz von *Moversight* ist funktional – wie in der Abbildung dargestellt – in drei Schichten gegliedert.

Schicht 1 Kollaborative Anwendungen haben unterschiedlich strenge Anforderungen an die Nachrichtenübertragung. Die unterste Schicht bietet daher drei Transferdienste mit verschiedenen Eigenschaften: den Basis-Transfer, den Stream-Transfer und den *Total ordered mobile Multicast* (TOM). Sie werden im Abschnitt 5.4 ausführlich vorgestellt.

Ebenfalls dieser Schicht zugeordnet ist der Fehlerdetektor (engl. *Failure Detector*), der für die Überwachung der Ein-Hop-Kommunikationspartner eines Peers verantwortlich ist. Seine Aufgabe ist es, Ausfälle von Peers und Netzpartitionierungen zu erkennen. Er wird im Kapitel 6 beschrieben.

Schicht 2 Die zweite Schicht enthält einen Dispatcher, der die Interaktion zwischen den verschiedenen Diensten steuert. Er bietet der kollaborativen Anwendung unter anderem über Schnittstellen Zugang zu den Operationen des Gruppenkommunikationssystems und dessen Ereignisse. Jeder Dienst von *Moversight* kann Ereignisse aussenden, welche über empfangene Nachrichten oder interne Zustandswechsel informieren. Die Anwendung oder auch andere Dienste können Nachrichten empfangen, sofern sie sich beim Dispatcher für den konkreten Ereignistyp registriert haben.

Schicht 3 Die dritte Schicht enthält den Gruppenverwaltungsdienst (engl. *Membership Service*), der den Anwendungen typische Operationen wie den Gruppenein- und -austritt anbietet. Außerdem können Informationen über Gruppen und Peers abgefragt oder geändert werden. Die Gruppenverwaltung wird im Abschnitt 5.3 erläutert.

Die *Mobilitätsunterstützung* ist für den nahtlosen Gruppenwiedereintritt von Peers nach Verbindungsabbrüchen verantwortlich. Nicht erreichbare Peers werden nicht sofort

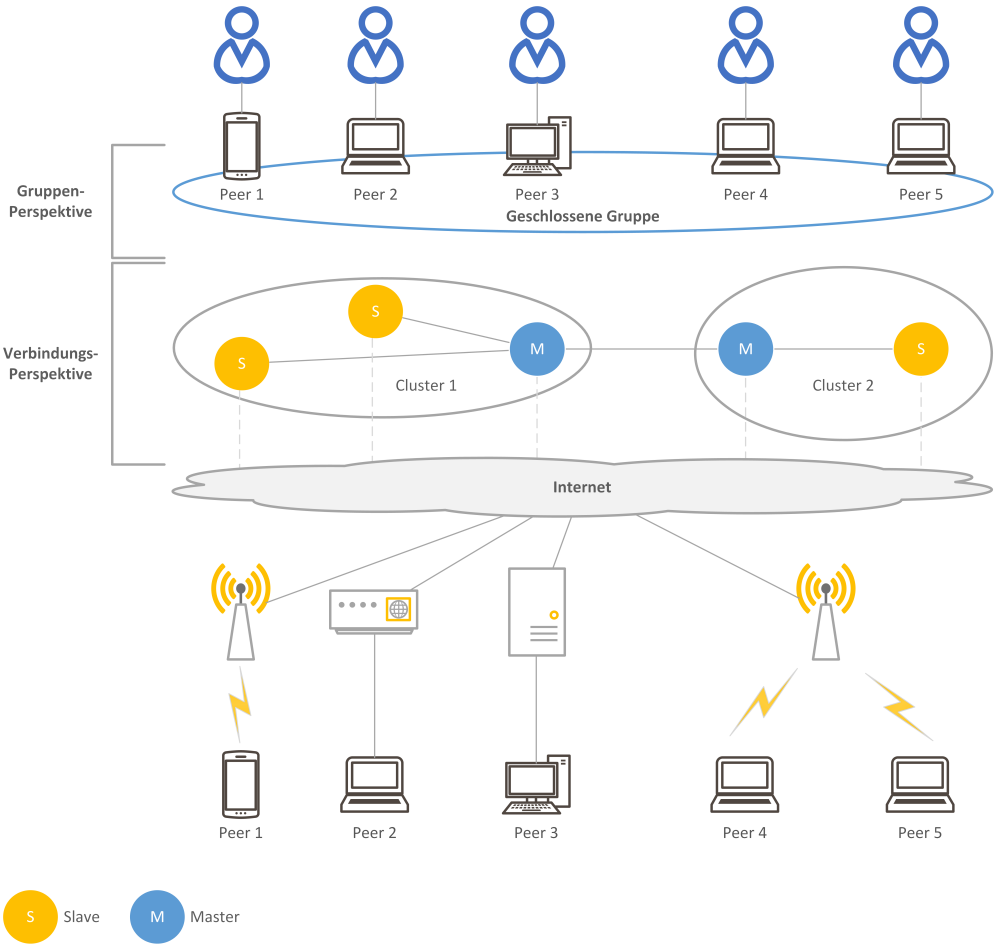


Abbildung 5.2: Das Kommunikationsmodell von *Moversight*

aus der Gruppe entfernt, sondern bleiben eine Zeitlang passive Mitglieder. Die Peers setzen ihre Arbeit fort, bis das Verbindungsproblem gelöst ist. Die Arbeitsweise dieses Dienstes ist Gegenstand des 6 Kapitels.

Der *Zeitdienst* stellt den Peers verschiedene Funktionen für die lokale Abschätzung der Verzögerung von Nachrichtenübertragungen (Latenz) zwischen (1) zwei Peers der Gruppe, (2) einem Peer und seinem Cluster oder (3) einem Peer und der gesamten Gruppe zur Verfügung. Die Abschätzung erfolgt dynamisch nach dem im Kapitel 7 beschriebenen Ansatz.

Der *Wartungsdienst* passt kontinuierlich die Kommunikationstopologie der Gruppe an Veränderungen im Netz und bei den Peers an. Er wird ausführlich im Kapitel 8 erläutert.

Der *Split*dienst ermöglicht eine aktive Aufteilung der Gruppe in zwei Gruppen. Eine Teilmenge der aktuellen Gruppenmitglieder kann sich von der Gruppe abspalten und fortan unabhängig von der Restgruppe agieren. Der *Merge*dienst stellt die semantische Umkehrfunktion zum Splitdienst dar und vereinigt zwei unabhängige Gruppen. Die Funktionsweise beider Dienste ist Gegenstand des Kapitels 9.

5.2 Kommunikationsmodell

Das in *Moversight* verwendete Kommunikationsmodell unterscheidet zwei Perspektiven: die Gruppen- und die Verbindungsperspektive (siehe Abbildung 5.2).

Die *Gruppenperspektive* modelliert eine geschlossene Gruppe, in der alle Peers gleiche Rechte und die gleiche Sicht auf die Gruppe besitzen. Jeder Peer ist in der Lage, auf alle Gruppendaten (z. B. Transportadresse, Rolle, Overlayposition) lokal

zuzugreifen und Gruppenoperationen auszuführen. Die Synchronisation bzw. Modifikation des gemeinsamen Wissens wird durch den TOM-Transfer gesichert. Änderungen der Gruppenzusammensetzung werden allen Gruppenmitgliedern über so genannte Sichtänderungsereignisse angezeigt.

Die *Verbindungsperspektive* modelliert das P2P-Overlay zwischen den Gruppenteilnehmern, das die Pfade für die Datenübermittlung und die Kontrollnachrichten definiert. Das Overlay besteht aus mehreren Clustern. Jeder Peer ist genau einem Cluster zugeordnet und agiert in diesem als Master oder Slave. Ein Master ist für die Verteilung der Nachrichten an die Slaves im Cluster verantwortlich. Ebenso tauscht er Nachrichten mit den Mastern der anderen Cluster aus. Sofern dies notwendig ist, kann ein Peer seine Rolle jederzeit ändern.

5.2.1 Wahl der Clustertopologie

Der Entscheidung für eine Clustertopologie lagen Abwägungen der Aspekte Ressourceneffizienz, robuste Verbindungen versus Kommunikationsausfälle und der durch die Topologie induzierten Übertragungsverzögerung zugrunde. Eine ausführliche Diskussion dieser Aspekte ist in Anhang B enthalten. Sie wird hier nur verkürzt dargestellt.

Mobile kollaborativen Anwendungen agieren auf Anwendungsebene dezentral und unabhängig von einer spezifischen Anwendungsinfrastruktur (vgl. Kapitel 2). So soll die im Anwendungsszenario beschriebene Robotergruppe in verschiedenen Umgebungen eingesetzt werden können, ohne von einer bestimmten Infrastruktur abhängig zu sein. Daher wurden unter Beachtung der oben genannten Kriterien folgende Topologien für das Overlay untersucht:

- Ringtopologie,
- voll vermaschte Topologie,
- virtuelle Pfadtopologie,
- (einstufige) Clustertopologie,
- Binärbaumtopologie und die
- k-Baumtopologie.

Die Anzahl von Verbindungen pro Knoten in einer Topologie kann als Kenngröße für deren Robustheit gegenüber Kommunikationsausfällen angesehen werden. Eine kollaborative Anwendung auf Basis einer voll vermaschten Topologie wäre folglich in mobilen Umgebungen robuster gegenüber Verbindungsabbrüchen, da alternative Routen bestehen. Jedoch steigen mit der Anzahl der zu verwaltenden Verbindungen die Ressourcenanforderungen an die Topologie. Daher muss bei der Entscheidung für eine Topologie zwischen Robustheit und Ressourceneffizienz abgewogen werden.

In Abbildung 5.3 ist die Anzahl der zu verwaltenden Verbindungen der einzelnen Topologien pro Peer für verschiedene Gruppengrößen dargestellt. Bei der Clustertopologie wurde eine Anzahl von $\sqrt{n}$-Mastern angenommen, wobei n die Gruppengröße ist. Die robuste voll vermaschte Topologie weist, wie erwartet, die meisten Verbindungen pro Peer auf. Demgegenüber haben die Ring-, Pfad- und Baumtopologien die geringste Anzahl von Verbindungen zu verwalten, was deren Ressourcenverbrauch minimiert.

Die Clustertopologie stellt einen Kompromiss zwischen stabilen Verbindungen und Verwaltungsaufwand dar. Die Anzahl der Verbindungen eines Slaves sind vergleichbar zu denen der Ring-, Pfad- oder Baumtopologien, während die Anzahl der Verbindungen eines Masters von der Gruppengröße abhängig und deutlich höher als die eines Slaves ist. Idealerweise wechseln Peers innerhalb der Topologie regelmäßig ihre Rollen, um eine faire Lastverteilung zu sichern. Daher kann bei der Clustertopologie für die Master und Slaves die durchschnittliche Verbindungsanzahl pro Peer, welche deutlich geringer ist als die Verbindungsanzahl eines Masters, für die Betrachtung genutzt werden.

Im Anhang B wird eine idealisierte Abschätzung für die durch die Topologie hervorgerufene Übertragungsverzögerung vorgenommen. Dazu wird ermittelt, wie lange

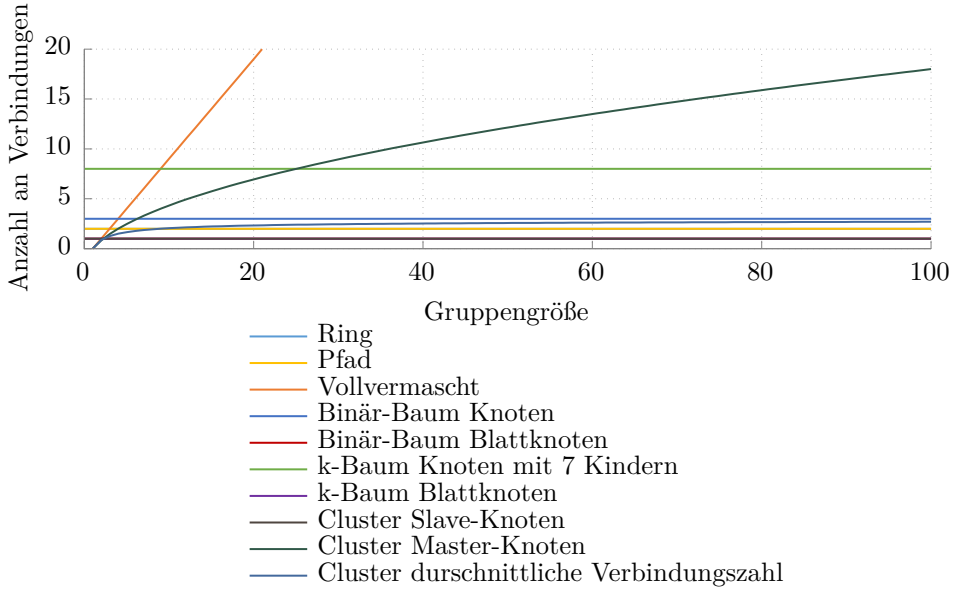


Abbildung 5.3: Anzahl von Verbindungen je Peer

die Verteilung einer Nachricht innerhalb der Gruppe dauert, wenn jede Übertragung eine Verzögerung von 1 s erzeugt. Die Clustertopologie hat im besten Fall eine geringere Übertragungsverzögerung als die Ring-, Pfad- oder die baumbasierten Topologien und wird nur von der voll vermaschten Topologie unterboten, die jedoch den höchsten Verwaltungsaufwand aufweist. Für die Details der Abschätzung sowie der zugrundeliegenden Annahmen sei auf den Anhang verwiesen.

Zusammenfassend lässt sich feststellen, dass eine Clustertopologien unter Beachtung der genannten Aspekte aus drei Gründen das beste Abwägungsergebnis bietet:

1. Im Gegensatz zu den in [19, 34, 89, 130] verwendeten Topologien ist die Anzahl der durch die Peers verwalteten Verbindungen begrenzt, was den Ressourcenverbrauch für den Nachrichtentransfer reduziert und damit die Funktionszeit der (mobilen) Endgeräte verbessert.

2. Die induzierten Verzögerungen sind bei gleichzeitig günstigem Ressourceneinsatz nahe dem Optimum [203].

3. Mit der Anzahl von Verbindungen zwischen den Peers werden gleichzeitig der Einfluss von schwankenden Netzverbindungen reduziert und durch die Clusterstruktur deren Auswirkungen begrenzt. Es handelt sich also um eine relativ robuste Topologie.

Die Clustertopologie skaliert gut. Neu eintretenden Peers wird über eine Peer-Platzierungsstrategie (siehe Abschnitt 5.3) ein Cluster zugewiesen und ihre Rolle bestimmt. Dadurch kann auch bei Churns die Clusterstruktur stabil gehalten werden.

5.2.2 Kommunikationsschema

Für die Gruppenkommunikation ergibt sich durch die Wahl der Clustertopologie das folgende Kommunikationsschema (siehe Abbildung 5.4): Wenn ein Slave eine Nachricht an die Gruppe senden möchte, sendet er diese zunächst an seinen Master. Dieser wird als *primärer Master* bezeichnet. Er ist verantwortlich für die Weiterleitung der Nachricht an die Master der anderen Cluster – die *sekundären Master* (aus Sicht des Absenders), die die Nachricht dann an ihre Slaves verteilen. Somit kommuniziert ein Master nur mit den anderen Mastern und seinen Slaves.

Durch die Overlaystruktur wird die Leistungsfähigkeit des P2P-Gruppenkommunikationsprotokolls maßgeblich beeinflusst. Um einen ausreichenden Durchsatz bei der

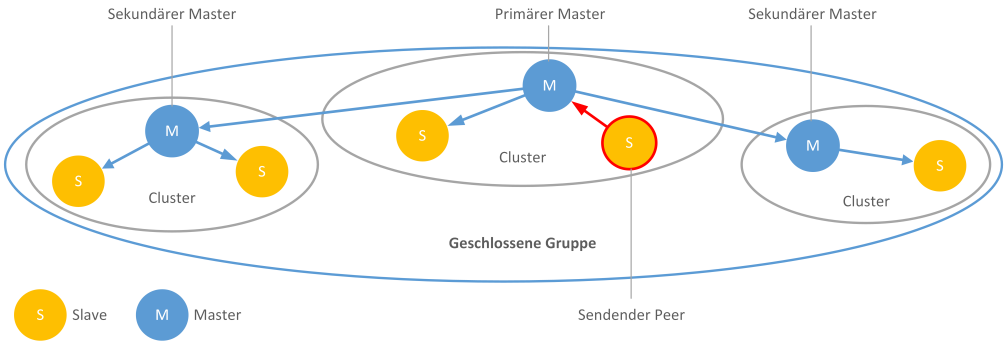


Abbildung 5.4: Kommunikationsschema von Moversight

Datenübertragung zu erzielen, müssen deshalb die definierten Datenpfade effizient und skalierbar sein. Der durch dieses Schema induzierte Nachrichtenverteilungsbaum ist aus einigen Application-Level Multicastprotokollen [19, 49, 57] bekannt und minimiert die zu übertragenden Nachrichten pro Peer bei potentiell höheren Latenzen.

5.2.3 Bewertung der resultierenden Datenpfade

Die Metriken Link Stress und Router Stress sind nach Banerjee *et al.* [19] geeignet, die Güte von Kommunikationsoverlays zu beurteilen. Allgemein ist der *Link Stress* definiert als die durchschnittliche Anzahl von Nachrichten, welche bei einer Multicastübertragung über eine Verbindung versendet werden. Der *Router Stress* dagegen gibt die durchschnittliche Anzahl von Nachrichten an, die der Router innerhalb des zugrundeliegenden Netzes während einer Multicastübertragung verarbeitet. In dem zuvor definierten Kommunikationsschema agieren die Master-Peers gleichzeitig als Endsystem und Router, Slaves hingegen nur als Endsystem. Somit gibt der Router Stress an, wie viele Nachrichten im Durchschnitt durch einen Master pro Multicastübertragung zu verarbeiten sind. Der Link Stress beschreibt, wie viele Nachrichten durchschnittlich pro Peer versendet werden. Je geringer die Anzahl der verbrauchten Ressourcen in den jeweiligen Metriken, desto ressourcenschonender ist das gewählte Kommunikationsschema.

Die im Kommunikationsschema von *Moversight* entstehenden Datenpfade sind vergleichbar mit denen des NICE-Protokolls [19] – einem leichtgewichtigen clusterbasierten Application-Level Multicastprotokoll. Abbildung 5.5 zeigt zwei Übertragungen von Gruppennachrichten in einer Gruppe von 16 Peers für das *Moversight* und das NICE Protokolle. Die Gruppe ist jeweils in vier Cluster unterteilt, als B_0 bis B_3 bezeichnet. Die Cluster sind in der Abbildung durch einen Kreis markiert. Pfeile stellen die Sendebeziehungen zwischen den Peers des Overlays dar. Die Peers a_0 und a_3 sind die Sender. Die Abbildungen a und c zeigen die Nachrichtenverteilung in NICE, entsprechend verdeutlichen b und d den Sendevorgang in *Moversight*. Die aus- und eingehenden Kanten zu/von den anderen Clustern identifizieren deren jeweilige Master. Die Abbildungen zeigen exemplarisch die Datenpfade für den Versand einer Gruppennachricht durch die Peers a_0 bzw. a_3.

Der Aufwand für die Datenübertragung in NICE wurde in [19] bezüglich der beiden Metriken Link- und Routerstress ermittelt. Die vergleichbare Struktur der Datenpfade in *Moversight* lässt den Schluss zu, dass der Stress für den (unzuverlässigen) Transfer einer Multicastnachricht in beiden Protokollen vergleichbar ist. In NICE liegt der Routerstress bei einer unzuverlässigen Multicastübertragung für Gruppen von 8 bis 128 Mitgliedern zwischen $2,3$ und $3,5$ Nachrichten, der Linkstress zwischen $1,63$ und $3,24$ Nachrichten. Die im folgenden Abschnitt 5.4 eingeführte zuverlässige Multicastübertragung basiert auf einem 3-Wege-Handshake Verfahren. Dadurch wird der Routerstress auf bis zu $10,5$ Nachrichten für große Gruppen gesteigert. Nach [19] ist dies eine gute Datenpfadqualität bei einem vergleichsweise geringen durchschnittlichen Aufwand.

Der Vergleich mit den Messergebnissen von NICE zeigt, dass für das definierte Kommunikationsschema der Kommunikationsaufwand auch für große Gruppen gering

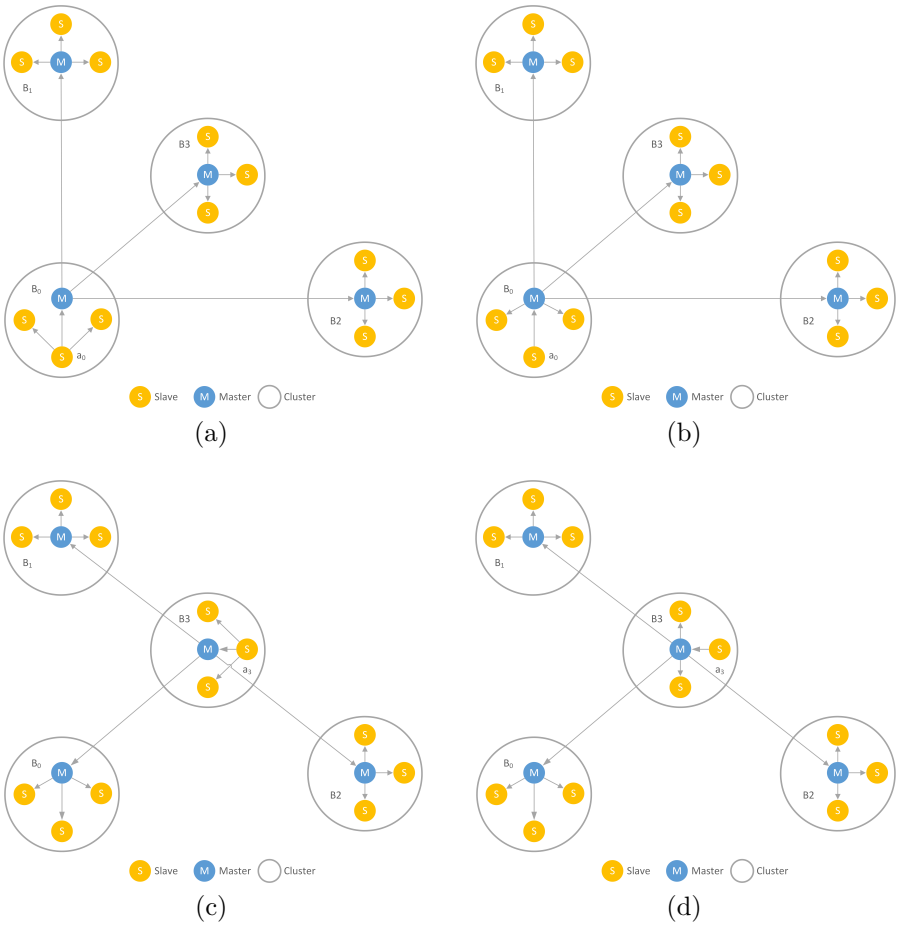


Abbildung 5.5: Gegenüberstellung der Datenverteilungspfade in NICE (*a*, *c*) und *Moversight* (*b*, *d*). Peers a_0 und a_3 senden jeweils eine Nachricht an die Gruppe.

ist. Damit wird das Ziel einer ressourcensparenden Kommunikation erreicht. Er zeigt aber auch, dass der Master im Vergleich zum Slave einen höheren Ressourcenbedarf hat. Daher ist ein regelmäßiger Rollenwechsel zwischen den Peers vorteilhaft.

5.3 Gruppenverwaltung

Die Gruppenverwaltung überwacht und steuert die Zusammensetzung der geschlossenen mobilen Gruppe. Sie wird dezentral auf jedem Peer ausgeführt. Infolge ihrer Geschlossenheit besitzen alle Peers ein konsistentes Wissen über die Gruppe und die Rolle der anderen Peers innerhalb der Clustertopologie. Dies wird im so genannten *Gruppenregister* verwaltet. Da alle Peers gleichberechtigt sind, kann jeder Peer innerhalb der Gruppe die gleichen Aufgaben übernehmen. Somit ist jeder Peer austauschbar, ohne dass die Gruppe funktional eingeschränkt wird. Um das zu erreichen, propagiert jeder Peer seine Eigenschaften wie Transportadresse, Ressourcensituation oder Mobilitätstyp in der Gruppe. Dadurch können die Peers lokal Operationen ausführen und Entscheidungen treffen, die global konsistent sind.

Die propagierten Eigenschaften der Peers – die Gruppenverwaltungsdaten (im Sinne der Sicht auf die aktuelle Clustertopologie) – und die durch den Anwender ausgewählten Anwendungsdaten bilden das *gemeinsame Wissen der Gruppe*. Es wird innerhalb der Gruppe mit Hilfe des total geordneten Multicastdienstes TOM konsistent gehalten (siehe Abschnitt 5.4).

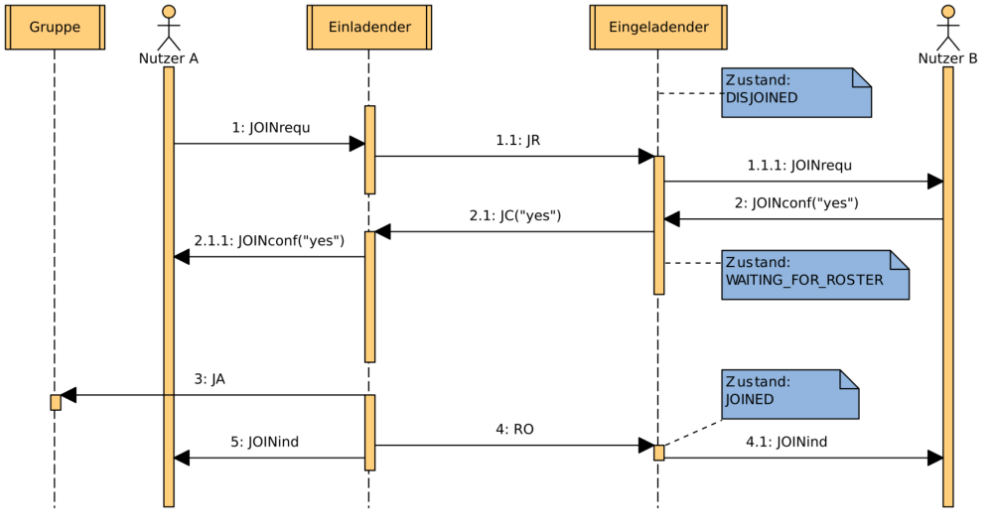


Abbildung 5.6: Ablauf Gruppenbeitritt

5.3.1 Gruppeneintritt – die Operation *JOIN*

Der Beitritt zu einer Gruppe wird über eine explizite Einladung mittels der *JOIN*-Operation eingeleitet, deren Ablauf in Abbildung 5.6 dargestellt ist.

Das einladende Gruppenmitglied sendet eine Einladung an den einzuladenden Peer. Wird die Einladung akzeptiert, so verteilt der einladende Peer die Beitrittsinformationen des neuen Peers mittels einer *Join-Announce-Nachricht* (JA) an die Gruppe. Sie enthält folgende Angaben:

- seine Transportadresse (IP Adresse und UDP-Port),
- eine kurze anwendungsspezifische Beschreibung (beispielsweise seine Lokalisierungsadresse, z. B. <nutzer_abc@beispiel.de>), sowie
- seine Betriebsmittel (Abstraktion der verfügbaren Teilnehmerressourcen und deren Mobilitätstyp).

Voraussetzungen für den Beitritt Jeder Peer prüft zunächst, ob die Sendesichtzustellungseigenschaft für diese Nachricht eingehalten wurde (vgl. Kapitel 4). Das ist notwendig, da durch die asynchrone ereignisbasierte Kommunikation unter den *Moversight*-Diensten andere Ereignisse einen Sichtwechsel zwischen der Auslieferung und dem Empfang von Gruppennachrichten (durch die Gruppenverwaltung) auslösen können. Ist die Sendesichtzustellung erfüllt, kann die Nachricht lokal verarbeitet werden. Anderenfalls wird die Nachricht verworfen und der Beitritt abgebrochen. Danach registriert jeder Empfänger den neuen Peer und platziert ihn in der Clustertopologie. Der einladende Peer informiert den neuen Peer mittels einer *Roster-Nachricht* (RO) (im Sinne von Mitgliederverzeichnisnachricht) über seine Position in der Topologie[1]. Abschließend werden die Anwendung und die anderen Dienste über den Beitritt des Peers informiert.

Peer-Platzierung Für die Platzierung des neuen Peers in die Clusterstruktur wird eine spezielle Peer-Platzierungsstrategie verwendet. Sie wird von jedem Peer unabhängig ausgeführt und kommt zu dem jeweils gleichen Ergebnis. In *Moversight* können prinzipiell verschiedene *Peer-Platzierungs-* oder *Clustering-Strategien* eingesetzt werden. Eine Platzierungsstrategie ist ein Balancierungsalgorithmus, der auf alle lokal verfügbaren

[1] Der Aufbau der Gruppenbeschreibung entspricht den Informationen, welche in Tabelle B.3 im Anhang B erläutert werden.

Gruppeninformationen zugreifen kann, vergleichbar mit der Balancierung von Baumstrukturen. Der Algorithmus ist verantwortlich für das Erstellen, Zusammenfassen bzw. Auflösen von Clustern sowie die Zuordnung der Peers zu ihnen. Die verwendete Strategie wird dem neuen Teilnehmer bei der Einladung mitgeteilt, um das neue Gruppenmitglied korrekt zu initialisieren. *Moversight* unterstützt folgende Strategien:

- die sequentielle Clustering-Strategie,
- die parallele Clustering-Strategie,
- die Distanz Clustering-Strategie und
- die *Dynamische Clustering-Strategie* (DCS).

Die *sequentielle Clustering-Strategie* ordnet dem gleichen Cluster solange neue Peers zu, bis eine obere Schranke erreicht ist. Erst dann wird ein weiterer Cluster erzeugt. Demgegenüber zielt die *parallele Clustering-Strategie* auf eine gleichmäßige Anzahl von Peers pro Cluster, jeweils für eine fest definierte Anzahl von Clustern. Die angestrebte Anzahl von Clustern bzw. deren Master wird so früh wie möglich erzeugt. Die *Distanz Clustering-Strategie* ordnet Peers dem Cluster zu, dessen Master logisch „*nahe*“ liegt. Die Entfernung zu einem Master wird dabei über eine Metrik bestimmt, z. B. über die Anzahl der Hops, die Latenz zu gegebenen Ankerpunkten oder über die Verwendung von logischen virtuellen Koordinaten wie in [40].

Mit der DCSs kann eine Clustering-Strategie vergleichbar mit den hierarchischen Beitrittsalgorithmen in NICE [19] und Narada [57] dezentral realisiert werden. Diese Strategie wird aufgrund ihrer Komplexität ausführlich erst in Kapitel 8 vorgestellt.

Abbildung 5.7 illustriert das Prinzip der parallelen Clustering-Strategie. Der erste Peer der Gruppe agiert als Master des ersten Clusters (Schritt 1). Jeder weitere beitretende Peer wird solange Master eines neuen Clusters (Schritt 2 bis 4), bis die maximale Anzahl von Clustern erreicht ist. Im Beispiel ist die maximale Clusteranzahl drei. Nachfolgende Peers werden ausgehend vom ersten Cluster in einem Round-Robin Verfahren als Slaves auf die bestehenden Cluster verteilt (Schritt 5).

5.3.2 Gruppenaustritt – die Operation *LEAVE*

Ein Peer kann auf zweierlei Weise die Gruppe verlassen: durch expliziten Austritt oder durch Ausschluss. Letzterer wird ausgeführt, wenn ein Peer dauerhaft nicht mehr erreichbar ist. Der Ausschluss ist die letzte Konsequenz des von *Moversight* unterstützten *REJOIN*-Prinzips, das im folgenden Kapitel behandelt wird.

Über die *LEAVE*-Operation kann ein Peer selbstbestimmt aus der Gruppe austreten. Dazu versendet er eine *Leave-Announce-Nachricht* (LA) an die Gruppe, die seine ID enthält. Sie wird mit der Sendesichtzustellungseigenschaft versandt. Danach wechselt der Peer in den Status *LEAVING*. Zunächst wartet er, ob/bis diese Nachricht an die Gruppe zugestellt wurde. Während dieser Phase ist der Peer weiterhin Mitglied der Gruppe, akzeptiert aber keine neuen Gruppennachrichten mehr. Im Fehlerfall wird der Versand der LA-Nachricht mehrfach wiederholt.

Wenn alle Peers die LA-Nachricht erhalten haben, entfernen sie den austretenden Peer aus der Gruppe. Der Peer verbleibt jedoch noch ein Vielfaches der Umlaufzeit einer Gruppennachricht in der Gruppe, bevor er diese lokal verlässt. Diese Warteperiode ist notwendig, da ein unverzüglicher Austritt aktive Nachrichtentransfers abbrechen würde, wodurch das globale Wissen der Gruppe zerstört und die Gruppe geteilt werden kann. Über die Warteperiode wird sichergestellt, dass der austretende Peer alle Transfers korrekt abschließt. Tritt der vorletzte Peer aus der Gruppe aus und existieren keine weiteren Eintrittsoperationen, wird die Gruppe global geschlossen.

Wiederholte *JOIN*- und *LEAVE*-Operationen können die Güte der Clustertopologie reduzieren. Deshalb muss die Topologie periodisch angepasst werden. Diese Optimierung übernimmt der Wartungsdienst (siehe Kapitel 8).

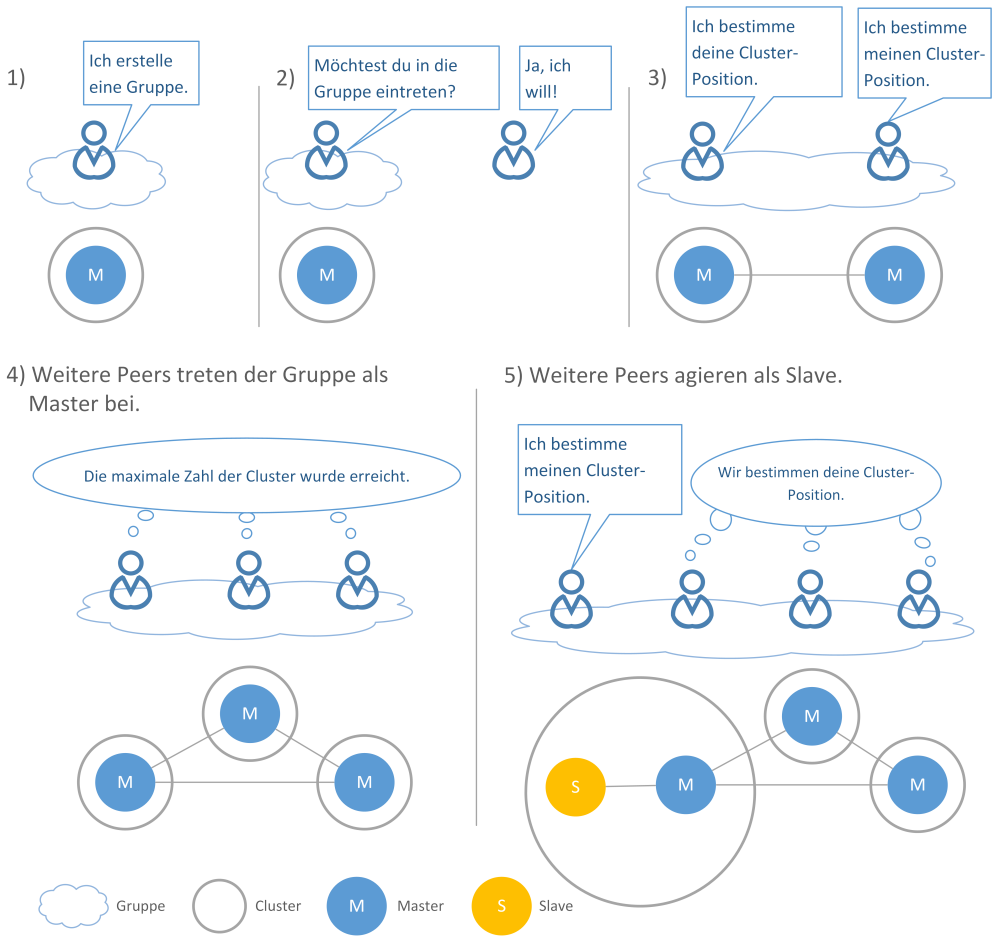


Abbildung 5.7: Eintritt von vier Teilnehmern in eine Gruppe unter Nutzung der parallelen Clustering-Strategie

5.4 Die Transferdienste von *Moversight*

Die *Moversight*-Transferdienste bieten dem Nutzer bzw. den anderen Diensten von *Moversight* unterschiedliche Zuverlässigkeits- und Ordnungseigenschaften, welche diese entsprechend ihrem Anwendungskontext auswählen können. Alle Transferdienste identifizieren die Nachrichten über eine Nachrichtenreferenz. Sie ist innerhalb der kollaborativen Gruppe für den jeweiligen Transferdienst eindeutig und über die Operation $<$ geordnet. Die Nachrichtenreferenz ist ein 2*er* Tupel der Form (Nachrichtennummer, Peer-ID), wobei die Peer-ID[2] und die Nachrichtennummer fortlaufende Zähler sind. Um Überläufe im Wertebereich zu vermeiden, wird bei jedem Sichtwechsel die Nachrichtennummer zurückgesetzt. Die Transferdienste sind:

- der Basis-Transfer,
- der Stream-Transfer und
- der Multicast-Transfer TOM.

Dabei ist der Basistransfer Peer-orientiert, d. h. eine Nachricht wird von einem Sender immer genau an einen Empfänger geschickt. Der Stream-Transfer und der TOM-Transfer arbeiten hingegen gruppenorientiert, d. h. sie übertragen eine zu versendende Nachricht immer an alle aktuell erreichbaren Peers der Gruppe.

[2] die Identifikation des sendenden Peers

5.4.1 Basis-Transfer

Der *Basis-Transfer* ist der einfachste Transferdienst. Er bietet eine Unicast-Übertragung auf Basis des UDP-Transportprotokolls an, welche wahlweise zuverlässig und unzuverlässig ausgeführt werden kann. Der Basistransfer wird von den anderen Diensten direkt oder indirekt für den Versand oder den Empfang von Nachrichten genutzt. Die Kommunikationspartner werden über die Transportadresse identifiziert, also üblicherweise über die IP-Adresse und den UDP-Port. Sendevorgänge werden innerhalb der Dienste über die Referenz der versandten Nachricht identifiziert. Für jede zuverlässig zu übertragende Nachricht wird der sendende Dienst mittels eines Ereignisses über Erfolg oder Misserfolg der Übertragung informiert. Die Art des Ereignisses und dessen Nachrichtenreferenz identifizieren dabei die Übertragung des betreffenden Transferdienstes. Bei einer zuverlässigen Übertragung wird für jede Nachricht eine konfigurierbare Anzahl von Sendeversuchen ausgeführt, bevor die Übertragung als fehlgeschlagen angezeigt wird.

Für jede erfolgreiche zuverlässige Übertragung misst der Dienst die Round-Trip-Time und übermittelt sie dem Zeitdienst, der sie als Basis für die Timereinstellungen nutzt (siehe Kapitel 7).

5.4.2 Stream-Transfer

Der *Stream-Transfer* ist ein leichtgewichtiger und unzuverlässiger Multicastdienst, vergleichbar dem Übertragungsdienst des NICE-Protokolls [19]. Er kann für die Übertragung von gruppenorientierten Daten genutzt werden, die nicht Teil des globalen Wissens sind, da es zu Übertragungsverlusten von beispielsweise Audio- oder Videodaten kommen kann. Die zu sendenden Nachrichten werden dabei durch den Dienst an alle erreichbaren Gruppenteilnehmer unter Nutzung des zuvor eingeführten Basis-Transfers übertragen.

Für die Übertragung spannt der Stream-Transfer einen Verteilungsbaum auf, wie er in Abbildung 5.2.3 dargestellt ist und auch durch den TOM-Transfer genutzt wird. Ist der Sender ein Slave, so übermittelt er die Gruppennachricht an den primären Master. Ist der Sender selbst Master, so wird die Nachricht direkt in der Gruppe verteilt. Der primäre Master verteilt die Nachricht mit der Identifikation des sendenden Peers an die sekundären Master sowie innerhalb seines Clusters. Diese Master übermitteln die Nachricht an die Slaves ihres jeweiligen Clusters.

5.4.3 *Total ordered mobile Multicast* (TOM)-Transfer

Der *Total ordered mobile Multicast* (TOM)-Transfer realisiert eine spezielle Art des Paradigmas der virtuellen Synchronität. Er sichert das gemeinsame Wissen der mobilen kollaborativen Anwendung. Damit nimmt der TOM-Transfer eine Schlüsselstellung in *Moversight* ein.

Das Übertragungsprinzip des TOM-Transfers ist eine Weiterentwicklung des *abcast*-Ansatzes aus [33], der ein total geordnetes Application-Level Multicast in einer Clustertopologie realisiert. Es arbeitet nach dem 3-Wege-Handshake-Prinzip und nutzt eine virtuelle Zeit zur totalen globalen Nachrichtenordnung.

Die zu sendende Gruppennachricht wird unter Nutzung des Basis-Transfers an alle Gruppenmitglieder übermittelt. Dieser Sendevorgang wird ebenfalls über eine Nachrichtenreferenz identifiziert. Bei der Übertragung versucht der TOM immer, die Nachricht an alle Peers (auch an nicht erreichbare) zu senden. Eine Gruppennachricht gilt als erfolgreich zugestellt, wenn sie mindestens einem Mitglied der Gruppe zugestellt werden konnte. Wurde die Nachricht von einigen Peers der Gruppe nicht empfangen, so wird ein Ereignis erzeugt, welches die vermissten Peers anzeigt. In diesem Fall wird mit der Installation einer optimistischen Sicht der *REJOIN*-Mechanismus ausgelöst (ausführlich mit den semantischen Konsequenzen für die virtuelle Synchronität im Kapitel 6 erläutert).

Empfängt ein Peer solch eine Gruppennachricht, so bestätigt er sie. Mit Hilfe dieser Bestätigungen wird dann die globale Empfangszeit für die Nachrichtenordnung ermittelt.

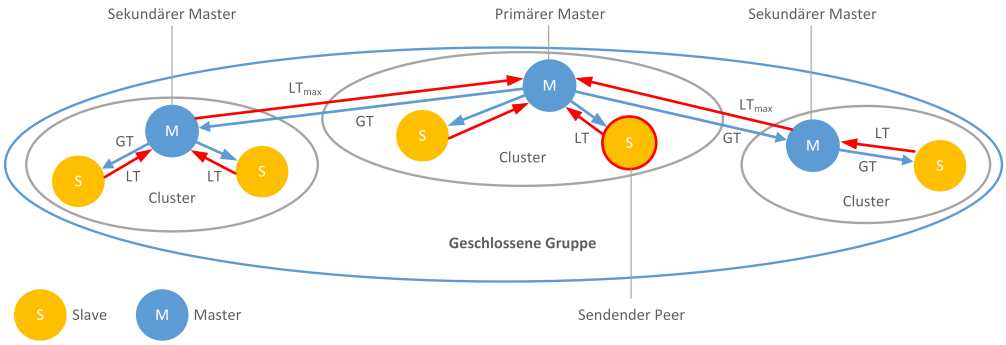


Abbildung 5.8: Bestimmung der globalen Zustellzeit für eine TOM-Übertragung

Anschließend wird eine Zustellungsbestätigung an alle Gruppenmitglieder versandt. Diese enthält die ermittelte Empfangszeit der Nachricht sowie eine Liste der Peers der Gruppe, welche die Nachricht nicht empfangen haben. Diese Informationen werden dem Empfänger bei der Nachrichtenauslieferung angezeigt.

Der Dienst unterstützt dabei die Zustelleigenschaften (1) Gruppensichtzustellung und (2) Sendesichtzustellung (vgl. Kapitel 4). Dazu enthalten alle Multicastnachrichten die ID des sendenden Peers und die ID der Sendesicht (engl. *View ID*). Als implizite Bestätigung für den Versand wird die Nachricht auch dem Absender zugestellt. Anderenfalls erhält der Sender eine Fehlermitteilung[3].

Die zu sendende Nachricht wird zunächst in der Gruppe entsprechend dem im Abschnitt 5.2 erläuterten Kommunikationsprinzip verteilt und zur Wahrung der virtuellen Synchronität in eine gruppenweit einheitliche Nachrichtenreihenfolge gebracht[4], welche durch die globale Empfangszeit bestimmt wird. Die Empfangszeit basiert auf der Lamport-Zeit [136] und stellt einen gruppenweiten Ereigniszähler für die Nachrichten dar. Um deren Reihenfolge zu bestimmen, wurde das Kommunikationsschema entsprechend der Abbildung 5.8 wie folgt erweitert:

Eingehende Nachrichten werden in jedem Peer in einer Warteschlange gespeichert und zunächst als *nichtzustellbar* markiert. In der Warteschlange werden die Nachrichten entsprechend der logischen Empfangszeit geordnet. Die Slaves bestätigen eine Nachricht gegenüber ihrem Master mit der logischen *lokale Empfangszeit-Nachricht* (LT). Der sekundäre Master sammelt alle Bestätigungen der Slaves seines Clusters. Aus den lokalen LT-Werten für die Nachricht wird der maximale LT-Wert des Clusters bestimmt. Dieser wird an den primären Master übermittelt, der seinerseits alle LT-Werte seines Clusters und aller sekundären Master erfasst. Das Maximum beider Werte ist die *globale Empfangszeit-Nachricht* (GT) für die Nachricht. Der ermittelte GT-Wert wird durch den primären Master nach dem Schema der Gruppennachrichten in der Gruppe verteilt. Jedes Gruppenmitglied aktualisiert beim Empfang des GT-Werts die Empfangszeit der entsprechenden Nachricht in seiner Warteschlange. Anschließend wird die Nachricht als zustellbar markiert und die Warteschlange nach den globalen Empfangszeiten sortiert.

Jede als zustellbar markierte Nachricht am Anfang der Warteschlange wird an die Anwendung ausgeliefert. Die Auslieferung wird beendet, sobald sie auf die erste nicht auslieferbare Nachricht trifft. Ausgeliefert werden die eigentliche Gruppennachricht, die Peer-ID des Senders, die ID der Sendesicht und optional eine Liste mit den ID der Peers, welche den Empfang der Nachricht nicht bestätigt haben.[5]

5.5 Evaluation

Die Eignung von *Moversight* für mobile kollaborative Anwendungen wurde simulativ unter Verwendung des *OMNeT++ Discrete Event Simulator* (OMNeT++)-Frameworks[6]

[3] Ein Überblick über alle ausgesendeten Ereignisse des TOM-Transfers ist im Anhang B gegeben.
[4] vergleiche Kapitel 4
[5] siehe vertiefend Kapitel 6
[6] siehe `https://omnetpp.org/`

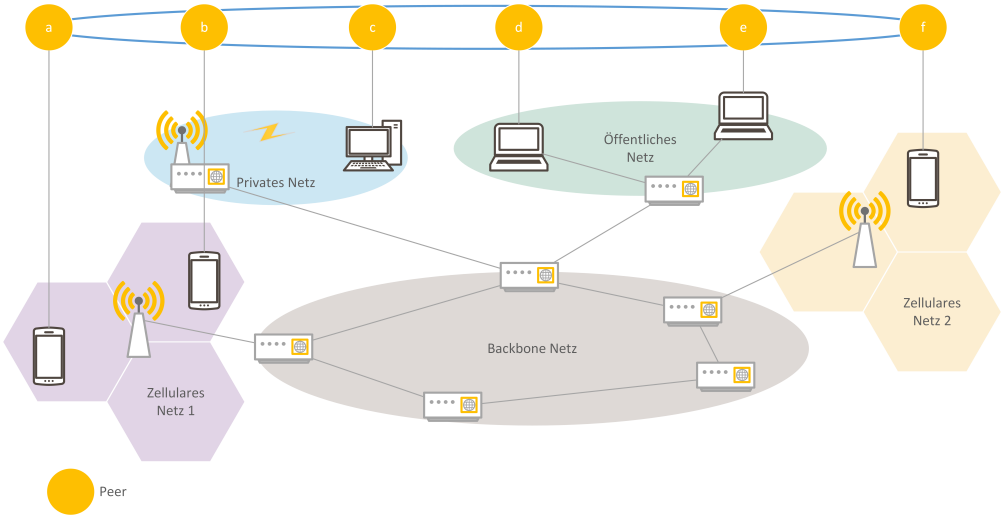


Abbildung 5.9: Aufbau des Simulationsszeanrios

evaluiert. Es sollte herausgearbeitet werden, ob bzw. inwieweit *Moversight* die am Ende von Kapitel 2 formulierten Anforderungen an die Gestaltung mobiler kollaborativer Anwendungen (insbesondere die Sicherung des Gruppenwissens, die Optimierung der Ressourcenauslastung und die Stabilität des Protokolls) erfüllt.

Zunächst wurden die Basisfunktionen für den Gruppenaufbau und die Nachrichtendistribution untersucht. In der einschlägigen Literatur werden die Latenzen für diese Protokollfunktionen zumeist nur theoretisch betrachtet. Klassische zentralisierte Gruppenkommunikationssysteme realisieren den Gruppenbeitritt im Allgemeinen über die Verteilung einer neuen Sicht nach der Anmeldung des eintretenden Peers beim zentralen Server. Daher entsprechen die angegebenen Latenzen für die Nachrichtenverteilung der Dauer des Gruppenbeitritts. Die wenigen Angaben aus der Literatur geben als maximale Latenz bei der Nachrichtenverteilung Werte von 0,02 s [164] bzw. 0,03 s [209] bis 0,06 s [163] an. Diese wurden für Gruppengrößen von acht bis 1000 Peers bestimmt.

Trotz des dezentralen Ansatzes sollte die maximale Dauer eines Gruppenbeitritts und die Verteilung einer Gruppennachricht bei dem hier vorgestellten *Moversight*-Ansatz idealerweise vergleichbar kurz sein. Für die Simulation eines vollständigen und realistischen Internet-Protokollstapels wird *Moversight* darin unter Nutzung des *INET*-UDP/IP-Stapels[7] implementiert.

Simulationsszenario Für die Leistungsanalyse wurde das in Abbildung 5.9 dargestellte Szenario verwendet, welches sich an das Roboterszenario aus 2.2 anlehnt. Die mobilen Teilnehmer stellen die Suchroboter dar. Die initiale kollaborative Gruppe besteht aus sechs Teilnehmern, die in unabhängigen Netzen platziert sind. Die Teilnehmer a, b und f sind mobil, c, d und e dagegen stationär. Letztere können eine Kontrollstation bzw. sonstige stationäre Einheiten im Suchgebiet sein.

Die einzelnen Netze sind über ein Backbonenetz verbunden, das aus 10, 100 bzw. 1000 *INET*-Routern besteht. Durch die unterschiedlich großen Backbonenetze werden verschiedene geographische Ausdehnungen der kollaborativen Gruppe simuliert. Ein Beispiel für eine kollaborative Anwendung, die über 10 Backbone-Router kommuniziert, wäre die Suchgruppe aus dem Roboterszenario in Abschnitt 2.2. Eine mobile Videokonferenz über große Entfernungen könnte ein Anwendungsbeispiel für ein Backbonenetz mit 1000 Routern sein.

Für die einzelnen Varianten von Backbonenetzen wurden jeweils variierende Gruppengrößen untersucht. Die Gruppengröße wurde schrittweise von 12 auf 24, 48, 72 bis 96 Mitglieder mit jeweils etwa zehn Prozent stationären Mitgliedern erhöht. Die maximal erlaubte Clustergröße wurde mit sechs Peers festgelegt. Um die Eigenschaften des

[7] siehe `http://inet.omnetpp.org/`

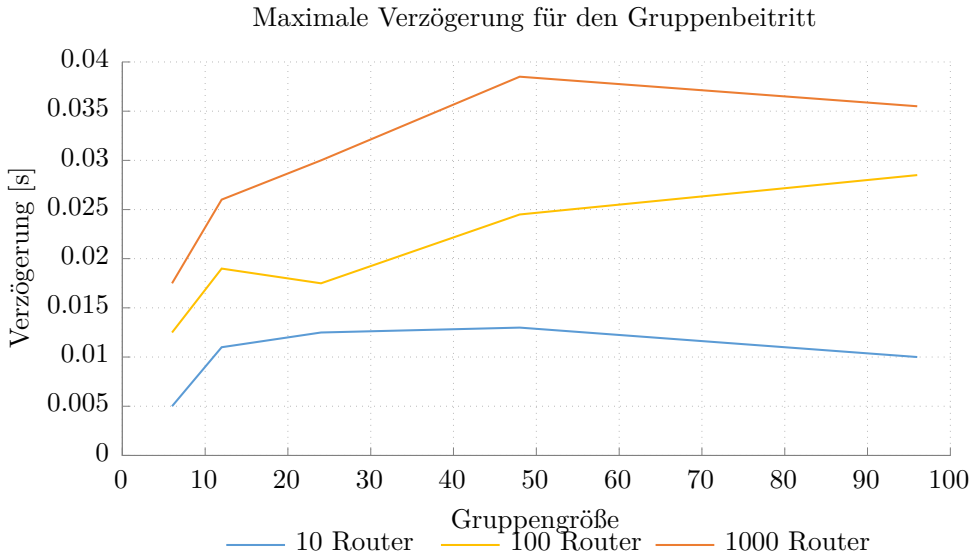


Abbildung 5.10: Maximale Verzögerung eines Gruppenbeitritts für unterschiedliche Gruppen und Backbonenetze

Backbonenetzes realistisch abzubilden, wurden mit Hilfe des Brite Topologiegenerators[8] auf Basis des Barabasi-Albert-Modells [8] die verschiedene Netze generiert und in die OMNeT++-Simulationsumgebung portiert.

Das Barabasi-Albert-Modell ist ein Algorithmus zur Erzeugung großer skalenfreier Netze [21]. Skalenfreie Netze sind dadurch gekennzeichnet, dass deren Verbindungseigenschaften trotz Veränderung der Betrachtungsgrößen (Skalierung) weitgehend konstant bleiben, also „*selbstähnelnd*“ sind. Dazu modelliert Barabasi das Netz als ungerichteten Graphen mit den Routern als Knoten und den Verbindungen zwischen diesen als Kanten. Die Knotenverteilung sowie die Anzahl von Kanten pro Knoten folgt dabei dem mathematischen Potenzgesetz[9]. Beispiele für skalenfreie Netze sind das Internet oder das World-Wide-Web.

Die Verwendung des Barabasi-Albert-Modells in der vorliegenden Arbeit basiert auf der Annahme, dass mobile kollaborative Anwendungen ebenfalls über Netze kooperieren, welche Teil des Internets sind oder deren Aufbau mit diesem vergleichbar ist. Über das Barabasi-Albert-Modell wurden in dieser Arbeit sowohl die Kanten zwischen den Routern im Backbonenetz als auch deren Verbindungseigenschaften (Bandbreite und Paketverlustrate) definiert.

Gruppenbeitritt Zunächst wurde der Gruppenbeitritt analysiert. Dazu wurde die Verzögerung ermittelt, die in typischen Gruppeneintrittssituationen entsteht. Abbildung 5.10 stellt die maximale Verzögerung für den Beitritt eines Peers zur Gruppe für die Gruppengrößen von 6 bis 96 Mitglieder und Backbonenetzgrößen von 10, 100 und 1000 Routern dar. Es ist zu erkennen, dass die Ergebnisse für die unterschiedlichen Netzgrößen –abgesehen von der Erhöhung der allgemeinen Verzögerung – vergleichbar sind. An den ähnlichen Verzögerungswerten für die verschiedenen Gruppengrößen innerhalb einer Netzgröße ist ebenfalls erkennbar, dass die Clustertopologie sehr gut skaliert.

Abbildung 5.11 zeigt die kumulierte Verzögerung für den Gruppenbeitritt unter Nutzung der sequentielle Clustering-Strategie bei unterschiedlichen Backbonenetzgrößen. Die kumulierte Verzögerung entsteht, wenn ein einzelner Peer die Gruppe nacheinander ohne parallele Einladungen aufbaut. Die Ergebnisse zeigen, dass die maximale Verzögerung für Gruppengrößen von 96 Peers 3,4 s beträgt. Ein Vergleich der verschiedenen

[8] siehe `http://www.cs.bu.edu/brite/`
[9] siehe `https://de.wikipedia.org/wiki/Potenzgesetz_(Statistik)`

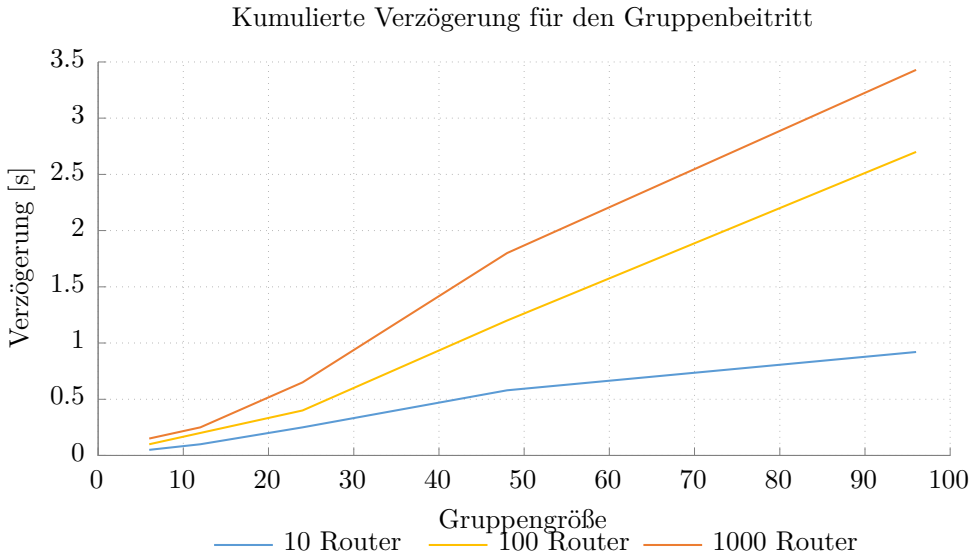


Abbildung 5.11: Kumulierte Verzögerung aller Gruppenbeitritte für unterschiedliche Gruppen und Backbonenetze

Backbonenetzgrößen zeigt beim großen Netz für eine Gruppe von 96 Mitgliedern eine Erhöhung der Verzögerung etwa um den Faktor drei gegenüber dem kleinen Backbonenetz. Damit einhergehend korreliert die Anzahl der versendeten Nachrichten mit der Größe der Gruppe und nicht mit der des Backbonenetzes. Der Verlauf der Verzögerungen ist unabhängig von der Größe des Backbonenetzes, welches der Verzögerung nur einen konstanten Anteil hinzufügt.

Nachrichtenverteilung Nach dem Gruppenbeitritt wurde die Verzögerung für den Nachrichtenaustausch mit dem TOM- und dem Stream-Transfers untersucht. Die Anzahl der in der Gruppe über den TOM-Transfer parallel versendeten Nachrichten hat, wie Abbildung 5.12 zeigt, einen deutlichen Einfluss auf die Verzögerung einer TOM-Übertragung, da der Aufwand für die Nachrichtenordnung mit der Anzahl paralleler Nachrichten steigt. In Abhängigkeit von der Länge und dem Füllstand der Warteschlange eines Peers und dessen Position innerhalb der Clustertopologie variiert die Verzögerung einer Nachricht für jeden Peer. Um diesen Einfluss zu reduzieren, wird die Datenverteilung über die durchschnittliche Verzögerung bewertet. In Abbildung 5.12 ist die durchschnittliche Verzögerung für eine TOM-Nachricht für unterschiedliche Gruppen und Backbonenetze dargestellt. Sie variiert zwischen 0,007 s und 0,021 s für das kleine, zwischen 0,011 s und 0,034 s für das mittlere und zwischen 0,014 s und 0,04 s für das große Backbonenetz. Da über den TOM-Transfer nur Daten ausgetauscht werden, die für den Zustand der kollaborativen Anwendung relevant sind (z. B. Statusinformationen, Nutzerdaten oder Live Objekte[10], Teilnehmereigenschaften oder Gruppenschlüssel), ist die Verzögerung angemessen. Für Anwendungen mit schwächeren Konsistenzanforderungen ist es vorteilhafter, andere Transferdienste zu wählen. Schwächere Konsistenzmodelle erzeugen weniger Nachrichten und geringere Latenzen, wodurch die Ressourcen des Endgerätes weniger beansprucht werden[11].

Die durchschnittliche Verzögerung des Stream-Transfers für die unterschiedlichen Backbone-Netze ist in Abbildung 5.13 dargestellt. Den Vergleich mit den Verzögerungen des TOM-Transfer für ein großes Backbonenetz zeigt Abbildung 5.14. Die Nachrichtenverteilung in der Gruppe ist beim Stream-Transfer signifikant schneller als beim zuverlässigen TOM. Die um den Faktor $2,5$ reduzierte Nachrichtenverzögerung wird durch den Verzicht auf die zuverlässige Übertragung und fehlende Konsistenz bzgl. des

[10] siehe Kapitel 3 oder [168]
[11] siehe Kapitel 4

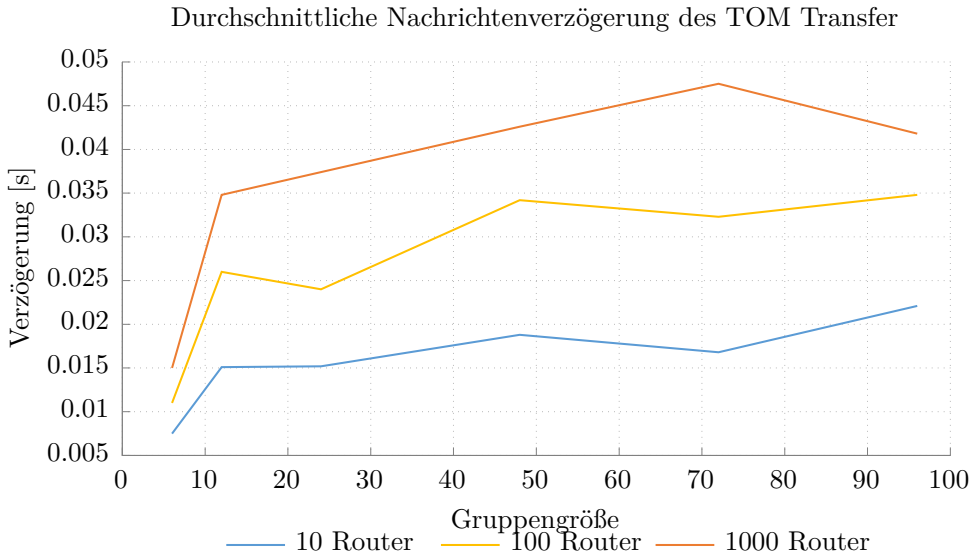


Abbildung 5.12: Durchschnittliche Verzögerung einer Nachrichtenübertragung mit dem TOM-Transfer für unterschiedliche Gruppen und Backbonenetze

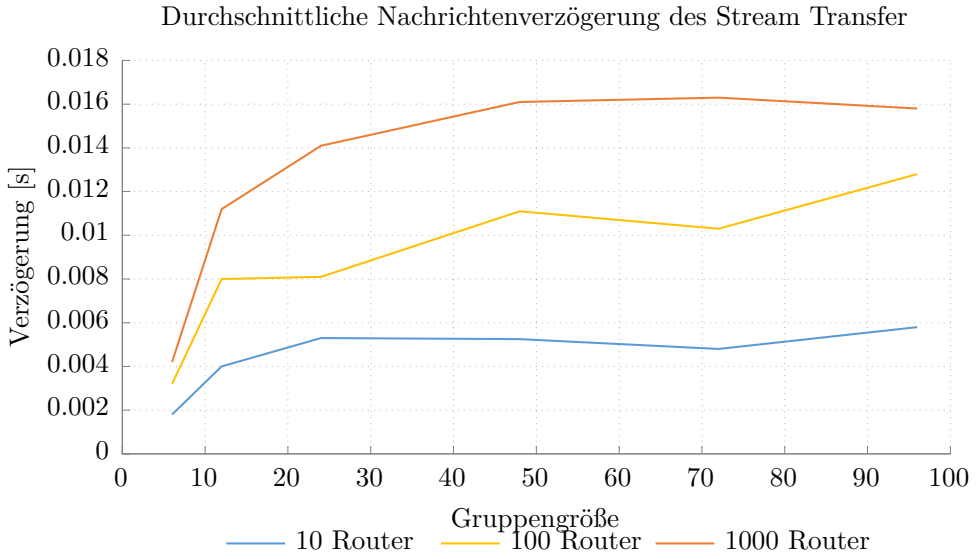


Abbildung 5.13: Durchschnittliche Verzögerung einer Nachrichtenübertragung mit dem Stream-Transfer für unterschiedliche Gruppen und Backbonenetze

ausgetauschten Wissens zwischen den Gruppenmitgliedern erreicht. Mit zunehmender Größe des Backbonenetzes wird die Gesamtverzögerung jedoch in erster Linie durch die Übertragungsverzögerung und nicht durch die Nachrichtenordnung bestimmt. Daher kann bei räumlich weit verteilten Gruppen der TOM-Transfer auch für nicht notwendigerweise total geordnete Anwendungsdaten genutzt und somit die Anwendungsentwicklung vereinfacht werden.

Die durchgeführten Experimente zeigen, dass das *Moversight* die gewünschten Leistungsanforderungen erfüllt und für typische Anwendungsfälle mobiler kollaborativer Anwendungen gut einsetzbar ist.

Durch den vollständig P2P-basierten Ansatz von *Moversight* können Single-Point-of-Failure-Situationen vermieden werden. Die Teilnehmer der mobilen kollaborativen Anwendung sind durch die netzunabhängige Architektur des Protokolls über verschiede-

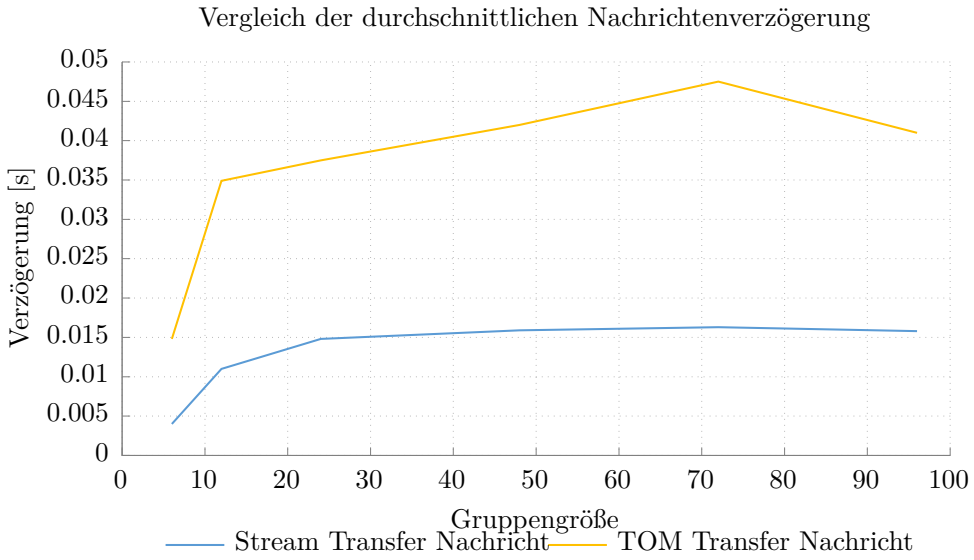


Abbildung 5.14: Vergleich der durchschnittlichen Nachrichtenverzögerung beim Stream-Transfer und dem TOM-Transfer für unterschiedliche Gruppen mit dem 1000-Router-Backbonenetz

ne Netze mit unterschiedlichen Netzarten erreichbar. Für die Kopplung dieser Netze ist ausschließlich ein UDP/IP-Stack notwendig. Die Analyse der durchschnittlichen Verzögerungen der verschiedenen Transferdienste sowie des Gruppenbeitritts zeigt, dass die gewählte Clustertopologie für die anvisierte Anzahl von maximal einhundert Teilnehmern gut geeignet ist. Die beim Gruppeneintritt und -austritt sowie der Datenverteilung mit den unterschiedlichen Transferdiensten entstehenden Verzögerungen werden in erster Linie durch das jeweils genutzte Netz und nicht durch das Protokoll verursacht. Somit ist der *Moversight*-Ansatz geeignet, gemeinsames Wissen in mobilen Umgebungen zu sichern. Des Weiteren lassen die ermittelten durchschnittlichen Nachrichtenverzögerungen die Schlussfolgerung zu, dass *Moversight* auch für Gruppengrößen über 100 Mitglieder gut nutzbar sein könnte. Dies müsste jedoch in eigenständigen Untersuchungen überprüft werden.

Kapitel 6

Nutzermobilität

Die Behandlung der Peer-Mobilität ist eines der zentralen Elemente beim Entwurf von *Moversight*. Sie nimmt deshalb eine Schlüsselstellung ein, weil hohe Churn-Raten die verfügbaren Ressourcen der Peers stark beeinflussen, was häufig zu Verbindungsaufällen führt und die Qualität der Zusammenarbeit nachteilig beeinflusst. Bisherige Ansätze zur Sicherung des gemeinsamen Wissens in Gruppenkommunikationssystemen berücksichtigen mobile Teilnehmer kaum, da der Fokus bislang zumeist auf stationären Teilnehmern in kabelgebunden Netzen mit wenigen Verbindungsunterbrechungen lag (vgl. Abschnitt 4). Die verschiedenen Varianten der virtuellen Synchronität beinhalten dennoch teilweise Strategien für den Umgang mit temporär nicht erreichbaren Teilnehmern. Diese ziehen jedoch häufig einen hohen Ressourcenverbrauch bei den einzelnen Teilnehmern nach sich, z. B. infolge von Rollbackmechanismen. Des Weiteren blockieren sie die Anwendungsausführung oder liefern Nachrichten nur unter einschränkenden Zustellgarantien aus. Die verschiedenen Varianten erkennen nur Fehlverhalten von Peers, nicht jedoch Netzpartitionierungen.

Bei der Netzpartitionierung wird die Gruppe aufgrund bestehender Verbindungsprobleme einzelner Mitglieder oder wegen fehlender Netzabdeckung in mehrere Teile aufgespalten. Die Ursachen dafür liegen also auf Netzebene. Es sind dabei verschiedene Szenarien möglich: (1) die Abspaltung eines isolierten Peers, (2) die Abspaltung von mehreren isolierten Peers oder (3) die Entstehung mehrerer Teilgruppen mit einer variablen Anzahl von Teilnehmern.

In *Moversight* ist es aus Sicht der Gruppe unerheblich, ob sich die nicht erreichbaren Peers in einer gemeinsamen Partition befinden. Für die Anwendung stellt sich im Fehlerfall die Gruppe immer zweigeteilt in erreichbare und nichterreichbare Peers dar. Wird durch die Anwendung die Kommunikation beim Vorliegen einer Partitionierung nicht beendet, läuft das Wissen der Gruppenmitglieder auseinander. Wird die Konnektivität mit der Gruppe bei ein oder mehreren Peers der nicht erreichbaren Partition wiederhergestellt, so versuchen die Peers sich mit der Gruppe wieder zu verbinden und in diese zurückzukehren. Dazu muss das globale Wissen der wieder eintretenden Peers mit dem der Restgruppe abgeglichen werden. Dieser Vorgang wird *Resynchronisation* genannt.

Die Aufteilung der Gruppe aufgrund von Verbindungsunterbrechungen ist semantisch von der anwendungsgetriebenen Teilung bzw. Wiedervereinigung der Gruppe durch die im Kapitel 9 betrachten Operationen *SPLIT*und *MERGE* abzugrenzen. Letztere sind aktive Entscheidungen der Gruppenmitglieder. Bei der Behandlung der durch die Nutzermobiltät ausgelösten Separierungen muss das System hingegen auf eine Netzsituation reagieren.

Bisher existierende Varianten der virtuellen Synchronität können nur bedingt mit nicht erreichbaren Teilnehmern umgehen. Die initiale Variante von Birman *et al.* [31] betrachtet Verbindungsprobleme nicht als eine Netzpartitionierung, sondern als ein Ausfall des Teilnehmers. Diese werden daher aus der Gruppe ausgeschlossen mit den bereits in Kapitel 4 beschriebenen Konsequenzen. Implementierungen dieses Paradigmas sind also nicht verwendbar in Szenarien, die zeitweilige Ausfälle tolerieren.

Bei der erweiterten virtuellen Synchronität werden Nachrichten im Falle einer Netzpartitionierung erst dann zugestellt, wenn sie in der Gruppe als „*safe*“ gekennzeichnet wurden. Reagieren Peers nicht auf Übertragungen, so können die jeweiligen Nachrichten nicht zugestellt werden. Die betreffenden Peers werden daraufhin aus der Gruppe entfernt. Die Partitionstoleranz wird somit auf Kosten einer reduzierten Systemverfügbarkeit gesichert.

Auch bei der schwachen virtuellen Synchronität werden nichterreichbare Teilnehmer aus der Gruppe entfernt. Über einen zusätzlichen Sichttyp, die so genannte *vorgeschlagene Sicht*, wird sichergestellt, dass das Gruppenkommunikationssystem während des Sichtwechsels weiter arbeiten kann. Diese Varianten modellieren also eine kurzzeitige Unterbrechung der Verbindung als Fehlersituation.

Demgegenüber sind kurzzeitige Verbindungsunterbrechungen bei der optimistischen virtuellen Synchronität ein regulärer Gruppenzustand, in dem angenommen wird, dass die betreffenden Peers zeitnah zur Gruppe zurückkehren. Dazu wird eine *optimistische Sicht* eingeführt, in der empfangene Nachrichten nur dann zugestellt werden, wenn bestimmte Bedingungen erfüllt sind. Sind sie nicht gegeben, werden die Nachrichten verworfen. Kehrt der getrennte Teilnehmer nicht in die Gruppe zurück, so ist der Nachrichtenaustausch in der optimistischen Sicht eventuell rückgängig[1] zu machen. Kehrt der Peer rechtzeitig zurück, so ist es Aufgabe der Anwendung, sein Wissen mit dem der Gruppe abzugleichen. Damit benachteiligen diese Ansätze Gruppenmitglieder, die keine Verbindungsprobleme haben. Werden Nachrichten nachträglich verworfen, weil Teilnehmer nicht zur Gruppe zurückkehren, sind die damit verbunden Ressourcen ergebnislos verbraucht. Somit wird auch bei der optimistischen virtuellen Synchronität die Ausfalltoleranz durch eine Reduzierung der Verfügbarkeit der Anwendung erreicht.

Es ist zu erkennen, dass bisherige Konzepte zur Sicherung der Wissenskonsistenz keine zufriedenstellende Lösung für den Umgang mit Partitionierungssituationen bei gleichzeitiger Sicherstellung der Anwendungsverfügbarkeit bieten. Der in diesem Kapitel beschriebene Ansatz zur Behandlung der Nutzermobilität in *Moversight* schlägt hierfür eine Lösung vor. Zunächst ist es jedoch erforderlich, die Zusammenhänge zwischen Konsistenz, Verfügbarkeit und Partitionstoleranz zu betrachten und das damit zusammenhängende *CAP-Theorem* einzuführen. Das CAP-Theorem zeigt, dass verteilte Systeme immer einen Kompromiss zwischen Konsistenz, Verfügbarkeit und Partitionstoleranz finden müssen. Die Umsetzung dieses Kompromisses kann für die Modellierung der Mobilitätsunterstützung in mobilen kollaborativen Anwendungen genutzt werden.

6.1 CAP-Theorem im Kontext mobiler kollaborativer Anwendungen

Mobile Systeme zeichnen sich durch ein inhärentes asynchrones Verhalten aus. Im Jahre 2000 äußerte Eric Brewer im Rahmen des *Symposiums Principles of Distributed Computing* [39] erstmals die Vermutung, dass die Asynchronität letztlich eine strikte Konsistenz innerhalb einer geschlossen Gruppe nicht zulässt. Weiter postulierte er, dass von den Entwurfszielen verteilter Systeme:

- Konsistenz (C, consistency),
- Verfügbarkeit (A, availability) und
- Partitionstoleranz (P, partition tolerance)

im Fehlerfall jeweils nur zwei gleichzeitig erfüllt werden können. Diese Vermutung wurde von Gilbert und Lynch im Jahre 2002 [101] bewiesen und wird seitdem als CAP-Theorem [38] bezeichnet (siehe Abbildung 6.1). Es verallgemeinert die Arbeiten von Fischer *et al.* [83] über die Unmöglichkeit einer gleichzeitigen Erfüllung dieser Parameter bei einem fehlerhaften Prozess in verteilten Systemen. Die Arbeiten von Friedman und Birman [86] bestätigen das Theorem ebenfalls.

[1] siehe Kapitel 4

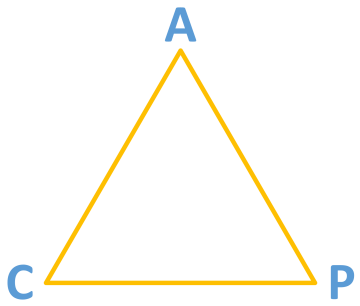


Abbildung 6.1: Illustration des CAP-Theorems (nach [38])

Nach diesem Theorem kann der Entwickler beim Entwurf eines verteilten Systems nur zwei von drei Eigenschaften des Dreiecks erfüllen. Besteht beispielsweise die Möglichkeit einer Netzpartitionierung – wie in mobilen kollaborativen Anwendungen – werden als Konsequenz zumeist Konsistenz (C) und Verfügbarkeit (A) beim Entwurf als Auslegungsschwerpunkte gewählt. Brewer [38] weist darauf hin, dass das Theorem nur während einer Netzpartitionierung gilt.

Die Kommunikationseigenschaften verteilter Systeme ändern sich kontinuierlich. Als Beispiel sei hier das Gruppenkommunikationssystem JGroups [161] genannt. Es sichert in Situationen ohne Netzpartitionierung eine strikte Konsistenz und hohe Verfügbarkeit. Im Falle einer Netzpartition wechselt es in einen Partitionierungsmodus, um die verfügbaren Operationen einzuschränken. In diesen Modus versucht das Protokoll, die Gruppe wieder zu vereinen und deren Konsistenz herzustellen. Dominiert wird die Anwendung jedoch von Phasen, in denen keine Partitionierung vorliegt und Verfügbarkeit und Konsistenz in der Gruppe hoch sind.

Beim Entwurf von *Moversight* sollten nach Möglichkeit Situationen vermieden werden, die ein Rollback des gemeinsamen Wissens nach der Wiedervereinigung der Gruppe erfordern. Ein Beispiel dafür wäre die Ausführung unterschiedlicher sichtändernder Operationen in den einzelnen Netzpartitionen. Die Entscheidung, ob eine Gruppenpartitionierung vorliegt oder Nachrichten nur verzögert sind, wird unter anderem von den Verbindungslatenzen in der Gruppe beeinflusst und bestimmt die Verfügbarkeit und die Konsistenz der kollaborativen Anwendung. Um einen guten Kompromiss entsprechend des CAP-Theorems zu erreichen, fordert Brewer [38], dass das Kommunikationssystem:

- eine Partitionierungssituation erkennt,
- in einen Partitionsmodus mit eventuell eingeschränkten Operationen wechselt, und
- die Partitionierung überwindet.

Abbildung 6.2 zeigt diesen Ablauf bei der Trennung einer kollaborativen Gruppe in zwei Partitionen. Die gemeinsame Sicht V der Gruppe trennt sich durch die Partition in die Sichten V^* und V°. Mit dem Verlassen des Partitionierungsmodus muss die Gruppe die Konsistenz in Sicht V' wiederherstellen.

In mobilen kollaborativen Anwendungen ist davon auszugehen, dass sich die Anwendung aufgrund von Verbindungsunterbrechungen häufiger im Partitionsmodus befindet. Folglich muss im Falle einer Partitionierung zwischen der Konsistenz (C) und der Verfügbarkeit (A) einer Operation abgewogen werden. Aus Anwendersicht sollten jedoch idealerweise stets alle drei Attribute erfüllt sein. Daher kann die in dieser Arbeit vorgestellte Behandlung der Nutzermobilität auch als ein Kompromiss auf der Grundlage des „*2-aus-3*“-Prinzips verstanden werden.

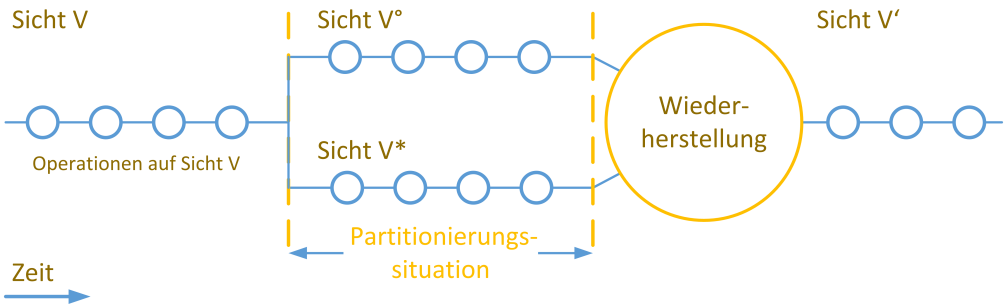


Abbildung 6.2: Trennung einer Gruppe in zwei Partitionen und Wiedervereinigung (nach [38])

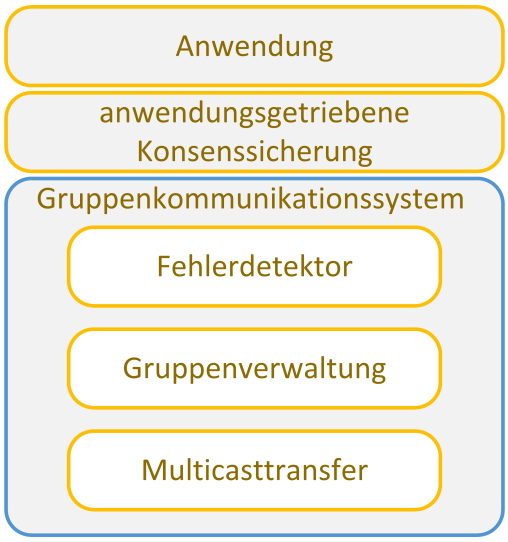


Abbildung 6.3: Mobilitätsunterstützung durch die Nutzung bestehender Funktionen und deren Ergänzung

6.2 Notwendigkeit einer Mobilitätsunterstützung in mobilen Gruppenkommunikationssystemen

Wie in Abschnitt 4.5 gezeigt, kann ohne weiterführende Maßnahmen in einer Gruppenkommunikationssitzung mit mobilen Teilnehmern das globale Wissen nicht gesichert werden. Die gewünschte Mobilitätsunterstützung kann auf zweierlei Wegen erreicht werden: (1) durch die Nutzung bestehender Funktionen und deren Ergänzung durch schnelle Fehlerdetektionsmechanismen und Konsistenzsicherungsmaßnahmen auf Anwendungsebene und (2) durch eine dedizierte Mobilitätsunterstützung als Teil des Protokollablaufs, die die Konsistenz des gemeinsamen Wissens sichert. In Abbildung 6.3 und 6.4 sind beide Ansätze schematisch dargestellt.

Der Vorteil einer dedizierten Mobilitätsunterstützung im Vergleich zu einer nachträglichen Konsistenzsicherung innerhalb der Anwendung soll durch die Abschätzung des Mehraufwands bei hohen Churn-Raten in einem infrastrukturlosen P2P-basierten Gruppenkommunikationssystem veranschaulicht werden. Dafür wird wieder das Roboterbeispiel aus Abschnitt 2.2 verwendet. Ausgehend von einem angenommenen Verbindungsverlust eines Roboters zur Gruppe werden zwei mögliche Situationen des Roboterbeispiels betrachtet:

1. Der Roboter kann erfolgreich zur Gruppe zurückkehren.

2. Der Roboter bleibt dauerhaft von der Gruppe getrennt.

Abbildung 6.4: Dedizierte Mobilitätsunterstützung

Für beide Situationen wird der Aufwand über die Anzahl der zwischen den Robotern ausgetauschten Nachrichten ermittelt. Dabei wird unterstellt, dass die Gruppe aus m Mitgliedern besteht, von denen m_{mob} mobile Roboter sind. Die übrigen Gruppenmitglieder agieren stationär. Vereinfachend wird angenommen, dass für eine Multicastübertragung an die Gruppe *eine* Nachricht versendet wird und zudem, dass während der Trennungsphase durch die Anwendung n Gruppennachrichten versendet werden. Ergänzende Details zu diesen Abschätzungen finden sich in [91].

6.2.1 Aufwandsabschätzung für eine Gruppe ohne dedizierte Mobilitätsunterstützung

Zunächst soll der Kommunikationsmehraufwand für die Situation (1) – erfolgreicher Wiedereintritt eines Roboters – abgeschätzt werden. Nachrichtenverluste, Sendewiederholungen oder Detektionsfehler werden vernachlässigt. Durch die Abschätzung wird daher nur die Anzahl von Nachrichten bestimmt, die minimal nach dem Erkennen des Verbindungsverlusts eines Roboters bis zu seinem Wiedereintritt ausgetauscht werden.

Nachdem der Roboter die Verbindung zu seiner Gruppe verloren hat, wird er (1) als fehlerhaft erkannt und markiert und die Gruppe darüber mittels *einer* Multicastnachricht informiert. (2) Die anschließende Einrichtung einer neuen Sicht benötigt weitere m Multicastnachrichten. (3) An die neue Situation passt sich die Anwendung durch den Austausch von m Multicastnachrichten an.

(4) Stellt der fehlerhafte Roboter seine Netzverbindung wieder her, versucht er, wieder in die Gruppe einzutreten. Für den Wiedereintritt versendet er mindestens *eine* Unicastnachricht, welche mit *einer* weiteren bestätigt wird. (5) Das angefragte Gruppenmitglied kündigt den (Wieder-)Eintritt mittels *einer* Multicastnachricht an. (6) Daraufhin wird mittels m Multicastnachrichten eine neue Sicht innerhalb der Gruppe eingerichtet.

(7) Der wiedereintretende Roboter wird über *eine* Unicastnachricht in die Gruppe reintegriert. (8) Die kollaborative Anwendung übermittelt dem Roboter im Anschluss das fehlende Gruppenwissen durch n Unicastnachrichten und (9) wechselt mit dem Austausch von m Multicastnachrichten wieder in den Anwendungszustand vor der Fehlersituation.

Für die Situation (2) – der fehlerhafte Roboter ist nicht in der Lage die Netzverbindung wiederherzustellen – endet der Prozess mit Schritt (4).

Tabelle 6.1: Vergleich der entstanden Aufwände mit und ohne Mobilitätsunterstützung

	Wiedereintritt ohne Mobilitätsunterstützung		**Wiedereintritt mit Mobilitätsunterstützung**	
	Erfolgreich	**Erfolglos**	**Erfolgreich**	**Erfolglos**
Multicast-Nachrichten	$4m + 2$	$2m + 1$	$2m + 2$	$2m + 1$
Unicast-Nachrichten	$n + 3$	0	$n + 2$	0
reguläre Sichten	2	1	0	1
opportunistische Sichten	0	0	1	1

6.2.2 Aufwandsabschätzung für eine Gruppe mit Mobilitätsunterstützung

Die Behandlung des Verbindungsverlusts mit Mobilitätsunterstützung ähnelt zunächst der im vorangegangen Abschnitt beschriebenen Vorgehensweise. Sobald der Roboter seine Netzverbindung verloren hat, wird er als fehlerhaft erkannt und entsprechend markiert (1). Anschließend werden die anderen Gruppenmitglieder mittels *einer* Multicastnachricht informiert. Im Gegensatz zum vorherigen Ablauf wird der Roboter nicht aus der Gruppe entfernt, sondern als vermisst markiert. Zusätzlich wird ein Timer gestartet, der das erlaubte Wiedereintrittsintervall pt überwacht.

Daraufhin (2) wechselt die Gruppe in eine opportunistische Sicht. Der vermisste Roboter wird als passives Gruppenmitglied betrachtet. Jede in der Gruppe ausgetauschte Nachricht adressiert auch diesen Roboter als Empfänger. Die ausgetauschten Nachrichten werden durch die übrigen Gruppenmitglieder gespeichert. (3) Für die Einrichtung der optimistischen Sicht werden m Multicastnachrichten ausgesandt.

(4) Wenn der getrennte Roboter den Wiedereintritt versucht, kündigt er dies mit dem Versand *einer* Multicastnachricht an. (5) Mit einer Unicastnachricht wird seine Sicht aktualisiert, was er *bestätigt*. Der erfolgreiche Wiedereintritt beendet die opportunistische Sicht und (6) richtet durch den Austausch von weiteren m Multicastnachrichten eine neue reguläre Sicht in der Gruppe ein. (7) Parallel dazu wählt der wiedereingetretene Roboter ein Gruppenmitglied als Resynchronisationspunkt aus und ruft von diesem die während seiner Abwesenheit in der Gruppe ausgetauschten n Nachrichten ab, um sich zu resynchronisieren.

Sollte der fehlende Roboter nicht in der Lage sein, eine Verbindung herzustellen, läuft nach Schritt (2) der Wiedereintrittstimer pt ab. Daraufhin wird eine neue reguläre Sicht (ohne den fehlenden Roboter) in der Gruppe aufgesetzt, was m Multicastnachrichten erfordert.

6.2.3 Vergleich der Aufwände

Die Abschätzung der entstehenden Aufwände beider Ansätze ist idealisiert. Nicht betrachtet wurden beispielsweise Kommunikationsfehler oder Einflüsse der Gruppentopologie. Der damit einhergehende Fehler ist vernachlässigbar, da er in beiden Ansätzen vergleichbar ist. Die resultierenden Aufwände sind in Tabelle 6.1 zusammengestellt. Es werden jeweils Abschätzungen für den erfolgreichen und den fehlgeschlagenen Wiedereintritt in Abhängigkeit von der Anzahl der versendeten Nachrichten sowie der eingerichteten Sichten (bei verschiedenen Sichttypen) dargestellt.

Wie die Ergebnisse zeigen, sind in Situation (2) – dem Scheitern des Gruppenwiedereintritts – die Aufwände identisch. Für den erfolgreichen Wiedereintritt – Situation (1) – sind die Unterschiede ebenfalls gering. So liegt der Aufwand für den Gruppenwiedereintritt ohne eine Mobilitätsunterstützung bei $O_{Mul}(4m + 2)$ bzw. $O_{Uni}(n + 3)$. Für den

Wiedereintritt bei vorhandener Mobilitätsunterstützung sinken diese auf $O_{Mul}(2m+2)$ und $O_{Uni}(n+2)$. Beide Ansätze haben jeweils zunächst einen linearen Aufwand.

Für den konkreten Anwendungsfall der Roboter kann angenommen werden, dass für das Wiederherstellen des gemeinsamen Wissens nach einem Wiedereintritt weniger als n Nachrichten benötigt werden. Dieser Vorgang ist in beiden Ansätzen gleich „*teuer*“ und kann damit als konstant angenommen werden. Darüber hinaus sind die Unicastaufwände mit $O_{Uni}(n+3)$ bzw. $O_{Uni}(n+2)$ nahezu identisch. Bei näherer Betrachtung unterscheiden sich also beide Ansätze vor allem in den Multicastaufwänden von $O_{Mul}(4m+2)$ gegenüber $O_{Mul}(2m+2)$.

Multicastübertragungen werden oft durch die Abbildung auf mehrere Unicastübertragungen realisiert, wodurch eine Multicastnachricht eine Reihe von Unicastnachrichten in Abhängigkeit von der gewählten Gruppentopologie erzeugt. Eine Ringtopologie beispielsweise benötigt für eine Multicastübertragung an m-Gruppenmitglieder: $(m-1)$ Nachrichten.

Da mobile Endgeräte nur über begrenzte Ressourcen verfügen und die drahtlose Kommunikation den Verbrauch dominiert, werden die konstanten Aufwandsfaktoren bei der Multicastkommunikation relevant. Ein direkter Vergleich von $O_{Mul}(4m+2)$ mit $O_{Mul}(2m+2)$ zeigt, dass unabhängig von der Gruppentopologie und dem konkreten Multicastdienst der Aufwand für den zweiten Ansatz nur halb so groß ist. Die gewonnenen Ressourcen verlängern die Lebensdauer der mobilen Teilnehmer und damit ihre Zusammenarbeitsfähigkeit [91].

6.3 Mobilitätsunterstützung in *Moversight*

Beim Entwurf von *Moversight* waren für die gesuchte Mobilitätsunterstützung verschiedene Aufgaben zu lösen, unter anderem:

1. die Erkennung von fehlerhaften Peers und Partitionierungssituationen,
2. die Behandlung der temporären Nichterreichbarkeit von Peers,
3. die Erkennung wieder erreichbarer Peers,
4. ihr nahtloser Gruppenwiedereintritt, und
5. die Wiederherstellung des gemeinsamen Gruppenwissens.

Eine detaillierte Darstellung der einzelnen Teilprobleme und ihrer Lösung im Rahmen des Gesamtkonzepts ist in [192] gegeben.

6.3.1 Erkennen fehlerhafter Peers und Partitionssituationen

Voraussetzung für die nahtlose Rückkehr von Peers nach Verbindungsunterbrechungen und die Herstellung ihrer ursprünglichen Rechte und Privilegien ist das schnellstmögliche Erkennen der Fehlersituationen. Dabei schränkt die Fehler- und Partitionierungserkennung die Verfügbarkeit der Gruppe ein (Availability im CAP-Theorem).

Bisherige Gruppenkommunikationssysteme setzen für die Fehler- und Partitionierungserkennung zumeist Fehlerdetektordienste (z. B. [10, 31, 161, 209] ein. Diese schließen aus dem wiederholten Ausbleiben von Bestätigungen auf Kontrollnachrichten auf eine unterbrochene Verbindung zu den betreffenden Peers. Aus den Verbindungsinformationen aller Peers wird eine eventuell existierende Partitionierung der Gruppe erkannt. Vereinzelt werden dabei eine implizite Detektion, die anhand fehlender Nachrichten im Protokollfluss auf eine Partitionierung schließt (vgl. [55, 81]) oder eine gerüchtebasierte Detektion (beispielsweise in GEODE[2]) genutzt.

Bei Fehlerdetektoren findet eine direkte Überwachung der Peers und ihrer Verbindungen statt. Dies belastet das Netz sowie die Ressourcen der Endgeräte, da kontinuierlich Nachrichten zwischen den Peers ausgetauscht werden (müssen). Die implizite Detektion belastet Netz und Ressourcen geringer, benötigt zur Diensterbringung jedoch eine

[2]Siehe `https://cwiki.apache.org/confluence/display/GEODE/Index`

kontinuierliche Kommunikation zwischen den Gruppenmitgliedern, welche bei kollaborativen Anwendungen nicht jederzeit gegeben ist. Bei der gerüchtebasierten Detektion kann keine erfolgreiche Erkennung zugesichert werden, wodurch dieser Ansatz nicht für *Moversight* in Frage kommt.

Erkennen fehlerhafter Peers Aus diesem Grund wird in *Moversight* ein hybrider Ansatz für die Fehlerdetektion verwendet, bei dem der Fehlerdetektor mit dem TOM-Dienst kooperiert. In Phasen der aktiven Kommunikation innerhalb der Gruppe werden die ausgetauschten Gruppennachrichten und deren Bestätigungen für die Detektion genutzt. Reagiert ein Peer auf diese nicht oder nicht protokollkonform, so meldet der Transferdienst diese(n) Peer(s) dem Fehlerdetektor als vermisst – „*missed Peer*“.

In Phasen ohne aktiven Datenaustausch innerhalb der Gruppe werden durch den Fehlerdetektor explizite Kontrollnachrichten periodisch an die zu überwachenden Peers gesendet. Dabei überwacht ein Slave seinen Master, ein Master all seine Slaves sowie die anderen Master der Gruppe. Für eine Erkennung muss der verwendete Fehlerdetektor die Eigenschaften der Klasse *Eventual-Perfect-Failure Detector* [55, 61, 81] erfüllen. Detektoren dieser Klasse erkennen ab einem (unbestimmten) Zeitpunkt garantiert alle fehlerhaften Peers in der Gruppe korrekt. Für den Versand der Kontrollnachrichten stehen verschiedene Algorithmen zur Verfügung, welche das Sendeintervall der Nachrichten beispielsweise aus den Abständen der letzten n-empfangen Nachrichten ermitteln. Dazu wird das Latenzschätzungsverfahren aus Kapitel 7 verwendet.

Erkennen von Partitionierungssituationen Wird mindestens ein Peer als fehlerhaft eingestuft, so wird die Partitionierungserkennung ausgelöst. Existierende Gruppenkommunikationssysteme besitzen nicht immer einen solchen Dienst, z. B. Zookeeper [116] oder GCP [231]. Andere Protokolle bieten nur eine eingeschränkte Partitionierungsbehandlung, bei welcher Partitionierungen zwar erkannt werden, aber nur bestimmte Partitionen uneingeschränkt weiterarbeiten dürfen[3]. Als Beispiele sind hier zu nennen: ISIS [31], Phoenix [150] oder RMP [160]. Darüber hinaus existieren noch hybride Ansätze, wie GEODE. Die überwiegende Zahl von Umsetzungen ist blockierend und schränkt die Verfügbarkeit des Systems ein.

Um eine Blockierung zu vermeiden, wird die Partitionierungserkennung in *Moversight* parallel zur normalen Kommunikation ausgeführt. Dazu prüft jeder Peer, ob:

1. seine Verbindung zur Gruppe gestört ist,
2. Peers aufgrund von Fehlern ausgefallen sind oder
3. eine Netzpartitionierung vorliegt.

Mit dem Abschluss der Partitionierungserkennung besitzen die Peers einer Partition genau das Wissen, das für die Sicherung der Konsistenz (C) notwendig ist. Für die Partitionierungsbehandlung ist die Ursache der Kommunikationsunterbrechung unerheblich. Daher können der Ausfall eines Endsystems und eine Netzpartitionierung semantisch gleichbehandelt werden.

Um eine Partitionierung erkennen zu können, werden alle Peers der Gruppe unter Verwendung einer der drei folgenden iterativen Erkennungsstrategien kontaktiert:

- Die *Kontaktiere-Alle*-Strategie versucht, alle Peers außer den bereits als fehlerhaft erkannten zu kontaktieren, um deren Erreichbarkeit zu prüfen.
- Die *Master-und-Cluster-Zuerst*-Strategie kontaktiert zuerst alle Gruppenmaster und die Peers des lokalen Clusters. In einem zweiten Durchlauf werden alle übrigen Peers (außer den als fehlerhaft erkannten) abgefragt. Normalerweise führt diese Strategie zu geringeren Aufwänden und einer längeren Erkennungsdauer.

[3] Konkret die primäre Partition

- Die *rundenweise* Abfragestrategie arbeitet dreistufig. Zunächst werden alle 1-Hop-Nachbarn kontaktiert, dann alle 2-Hop-Nachbarn und zuletzt alle anderen Peers, jeweils abzüglich der bereits als fehlerhaft erkannten Peers. Diese Strategie besitzt die längste Erkennungsdauer, im Idealfall jedoch den geringsten Aufwand.

Die Erkennungsstrategien unterscheiden sich jeweils im Aufwand, ihrer Dauer sowie im Zeitpunkt, zu dem Peers der Gruppe zu kontaktieren sind. Für alle Strategien gilt: Kontaktieren sich zwei erreichbare Peers, so übermitteln diese in der Antwortnachricht alle ihnen bekannten und erreichbaren Peers. Der Anwendungsentwickler muss eine der drei Strategien unter Abwägung der Parameter Erkennungsdauer und Aufwand auswählen. Wurden alle Peers bezüglich ihres Zustands bewertet, wird durch den Partitionierungsdetektor die Konnektivität innerhalb der Gruppe bestimmt und alle nichterreichbaren Peers der Gruppe werden markiert.

6.3.2 Verwalten der Erreichbarkeit eines Peers

Bisherige Gruppenkommunikationssysteme kennen das Konzept des nahtlosen Wiedereintritts nicht. Sie verwalten daher in einer Sicht nur Peer-Eigenschaften wie Rolle, Transportadresse oder Identifikationsnummer. Da in *Moversight* ein Gruppenmitglied kurzzeitig nicht erreichbar sein kann, muss das Sichtkonzept um die Verbindungseigenschaften eines Peers erweitert werden. Daher wird ein Peer, der ein aktives Mitglied der Gruppe in einer Sicht ist, als *JOINED* markiert. Kommt es zu einer temporären Verbindungsunterbrechung, so geht der entsprechende Peer in den Zustand *PENDING*, d. h. passives oder schwebendes Mitglied, über. Diese Peers sind weiter gültige Empfänger von Gruppennachrichten. Die Nachrichten werden zwischengespeichert und den Peers beim Wiedereintritt während der Resynchronisationsphase (siehe unten) zugestellt. Kehren als *PENDING* markierte Peers nicht innerhalb einer bestimmten Wiedereintrittszeit zurück, werden sie aus der Gruppe entfernt.

Wenn die Verbindung zu einem Gruppenmitglied temporär unterbrochen ist, muss die Partitionierungssituation gemäß dem CAP-Theorem behandelt werden. Existierende Gruppenkommunikationssysteme führen dies entweder ausschließlich im Fehlerdetektor (z. B. in ISIS, Horus, JGroups oder RMP) oder über einen speziellen Sichttyp, z. B. die erweiterte Sicht von Totem bzw. die optimistische Sicht von Sussman *et al.* [204] auch in der Gruppenverwaltung aus.

Eine Sicht mit mindestens einem als *PENDING* markierten Peer unterscheidet sich jedoch semantisch von einer regulären Sicht, die einen solchen Peer nicht enthält. Die Anwendung hat zu entscheiden, ob sie eine Gruppennachricht an einen als *PENDING* markierten Peer sendet. Daher verwendet *Moversight* die optimistische Sicht von Sussman *et al.* [204], um der Anwendung diesen semantischen Wechsel zu signalisieren. Wenn der letzte *PENDING* markierte Peer aus der Gruppe entfernt ist, kehrt die Gruppe wieder in eine reguläre Sicht zurück. Diese Sicht wird auch installiert, wenn alle als *PENDING* markierten Peers erfolgreich zur Gruppe zurückkehren. Vertiefende Erläuterungen dazu in Abschnitt 6.4.

6.3.3 Zustände eines Peers

Der Zustand eines aktiven Peers innerhalb der Gruppensicht steht in Wechselbeziehung mit dessen internem Protokollzustand. Die Zustände der Gruppensicht *JOINED* und *PENDING* wurden oben bereits eingeführt. Zur Umsetzung des Protokolls werden weitere interne Zustände benötigt, welche mit ihren Zusammenhängen in Abbildung 6.5 dargestellt sind.

Die Zustände *WAITING_FOR_ROSTER* bzw. *INVITING* sind Hilfszustände, die nur für den initialen Gruppenbeitritt bzw. -aufbau benötigt werden. Möchte ein Peer aktiv aus der Gruppe austreten, so wechselt über den *LEAVING*-Zustand in den initialen Zustand *DISJOINED*.

Zu erkennen ist, dass in Abbildung 6.5 kein Zustand *PENDING* enthalten ist und mehrere Zustände hervorgehoben sind. Damit soll Folgendes verdeutlicht werden: Ein Gruppenmitglied geht aus seiner lokalen Perspektive immer davon aus, ein aktives

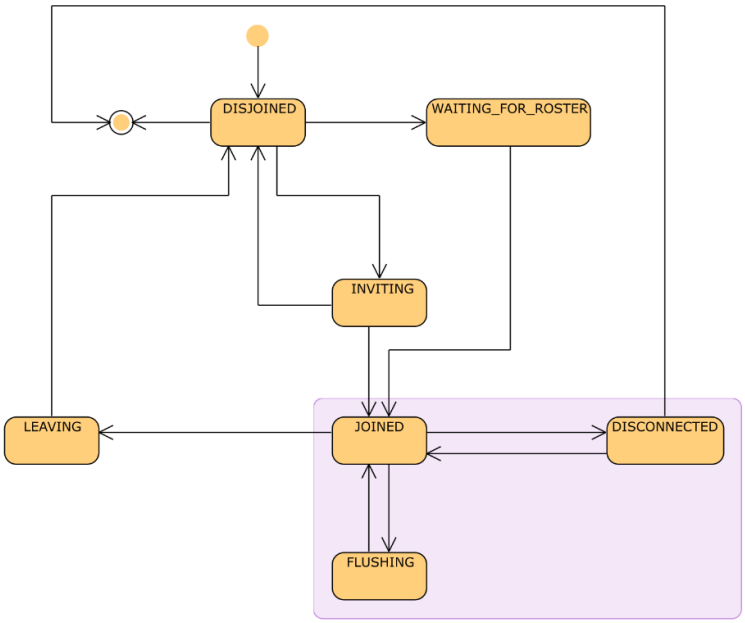


Abbildung 6.5: Zustandsautomat eines *Moversight*-Peers

Mitglied der Gruppe (*JOINED*) zu sein. Stellt der Peer jedoch fest, dass er in einer Partitionierungssituation keine weiteren Peers der Gruppe erreichen kann, wechselt er in den Zustand *DISCONNECTED*. In diesem Zustand ist der Peer sozusagen aktives Gruppenmitglied einer 1er Partition. Der Zustand *DISCONNECTED* ist somit eine semantische Spezialisierung des Zustandes *JOINED* bei einem temporären Verbindungsverlust des Peers, in welchem ein Peer die Gruppe zeitnah wieder zu erreichen versucht. Funktional schränkt dieser Zustand die erlaubten Operationen eines Peers ein, wie in im Abschnitt 6.3.4 ausgeführt wird.

Der Zustand *FLUSHING* dient dem Leeren der Warteschlangen in den Transferdiensten und spielt eine besondere Rolle bei der Sicherung der Konsistenz. Ausführlich wird dieser Zustand in den nachfolgenden Abschnitten sowie im Kapitel 8 diskutiert.

Für die Protokollverarbeitung wären die Zustände *DISCONNECTED* und *FLUSHING* nicht notwendig. Deren Einführung war bei der prototypischen Umsetzung programmiertechnisch vorteilhaft. Die notwendigen funktionalen Einschränkungen für einen Peer, welcher vollständig die Verbindung zur Gruppe verloren hat, lassen sich über einen definierten Zustand *DISCONNECTED* einfacher behandeln als eine integrierte Behandlung im Zustand *JOINED*. Ebenso können im Zustand FLUSHING die notwendigen Maßnahmen zur Konsistenzsicherung besser gebündelt werden.

Für die Behandlung der Nutzermobilität sowie für die Wartung der Topologie (siehe Kapitel 8) war es notwendig, den Zustand *JOINED* weiter zu differenzieren. Der resultierende erweiterte Zustandsautomat eines *Moversight*-Peers ist in Abbildung 6.6 dargestellt. Wiederum sind alle Zustände, welche semantisch die aktive Gruppenmitgliedschaft aus Sicht des lokalen Peers modellieren, farblich hervorgehoben.

Wie bereits ausgeführt, wird ein von der Gruppe getrennter Peer in der Gruppensicht als *PENDING* markiert, der Peer selbst befindet sich lokal jedoch im Zustand *JOINED* (bei mehreren Peers in der abgetrennten Partition) oder im Zustand *DISCONNECTED*. Der Wiedereintritt des Peers in die Gruppe und dessen Resynchronisation des gemeinsamen Wissens erfolgt in drei Phasen, welche über die Zustände *RECONNECTING*, *REINTEGRATE* und *RESYCNHRONIZE* ablaufen.

Erkennt der Peer die Partitionierungssituation, so versucht er die Gruppe wieder zu erreichen. Diese Detektion erfolgt im Zustand *RECONNECTING*. Versucht ein Peer der Gruppe wieder beizutreten, so wechselt er zunächst in den Zustand *RECONNECTING*. Ist dies erfolglos, so führen die Peers entweder die Kollaboration in der kleineren Gruppe fort, oder im Falle der 1*er*-Partition schließt der Peer die Gruppe. Wird ein

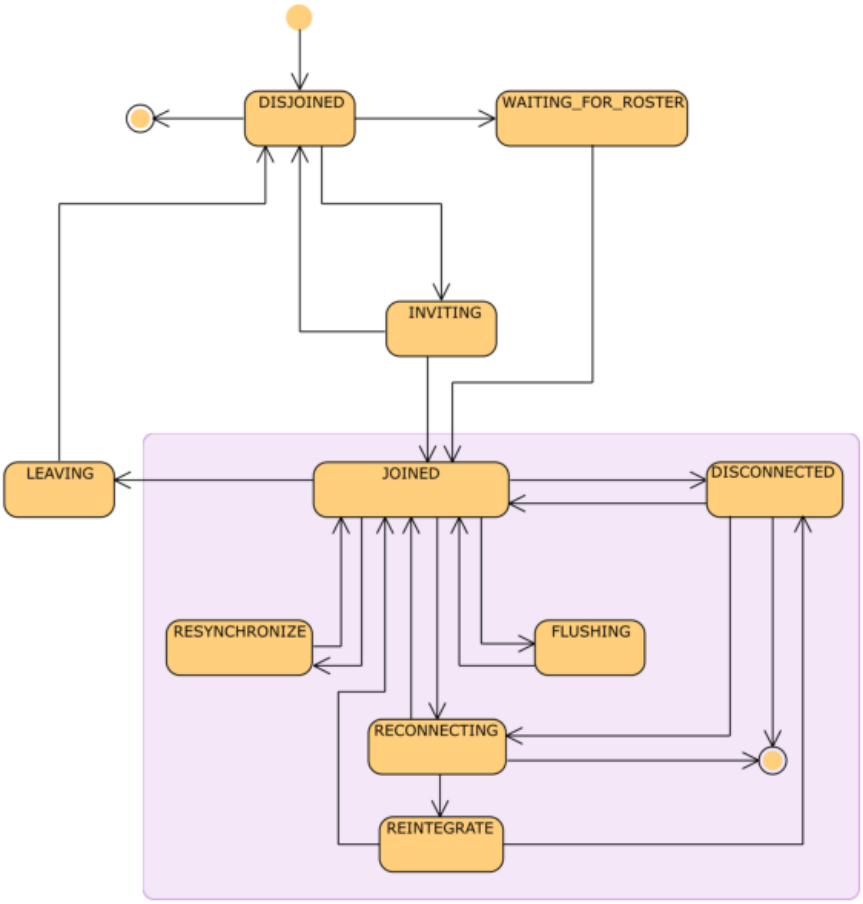


Abbildung 6.6: Erweiterter Zustandsautomat eines *Moversight*-Peers

Peer der anderen Partition erreicht, so wird über den *REINTEGRATE*-Zustand die Wiedereingliederung in die Gruppe gesteuert. Bei Erfolg dieser Operation ist der Peer im Prinzip wieder aktiver Teil der Gruppe (je nach Partitionierungssituation) oder verbleibt in seiner alten Partition. Der Mobilitätsdienst prüft in jedem Fall, ob eine weitere Detektion von nicht erreichbaren Gruppenmitgliedern notwendig ist. Nach der erfolgreichen Wiedereingliederung in die Gruppe (Zustand *JOINED*) muss noch das Wissen des Peers mit dem der Gruppe abgeglichen werden. Dies erfolgt durch den Mobilitätsdienst, welcher die notwendigen Operationen im Zustand *RESYNCHRONIZE* zusammenfasst.

6.3.4 Operationseinschränkung

Wenn mehr als die Hälfte der Gruppenmitglieder durch die Partitionierungserkennung eines Peers als erreichbar eingestuft werden, bilden diese Mitglieder die *primäre Partition* oder *Hauptkomponente* (siehe Kapitel 4 bzw. [55])[4]. In dieser Partition sind üblicherweise alle Gruppenoperationen zulässig, da hier Konsistenz (C) und Verfügbarkeit (A) entsprechend des CAP-Theorems gewährleistet sind.

Wurden hingegen weniger als die Hälfte der Gruppenmitglieder erreicht, gehört der Peer einer *sekundären Partition*[5] an. Aufgrund der optimistischen Annahmen bezüglich des nahtlosen Gruppenwiedereintritts sind in dieser Partition keine Operationen zulässig,

[4]Bei gleich großen Partitionen ist diejenige die primäre Partition, welche den Peer mit der kleinsten ID enthält.

[5]Erreicht ein Peer überhaupt keine weiteren Gruppenmitglieder (signalisiert durch den Zustand *DISCONNECTED*), so bildet er eine 1-elementige sekundäre Partition.

welche die Gruppe erweitern, d. h. *JOIN* oder *MERGE*[6]. Peers können hingegen aus beiden Partitionstypen austreten. Eine primäre Partition muss nicht in jeder Partitionierungssituation existieren. So können beispielsweise mehr als 50 % der Gruppenteilnehmer in jeweils eine getrennte Partition abgespalten werden. In diesem Falle würden alle Partitionen der Gruppe als sekundäre Partition der Operationseinschränkung unterliegen.

6.3.5 Wiederherstellen der Verbindung

Die Behandlung einer Partitionierung wird durch zwei gegensätzliche Anforderungen bestimmt: (1) das zeitnahe Zusammenführen der Partitionen und (2) die Ressourcenschonung. Für ein zeitnahes Zusammenführen muss das System möglichst schnell erkennen, dass ein vermisster Peer wieder erreichbar ist. Dazu muss zunächst definiert werden, welche Peers bzw. Partitionen diese Prüfung durchführen. Denkbar sind dafür alle Peers, ausgewählte Peers aus beiden Partitionen, Peers nur aus der primären oder Peers aus der sekundären Partition.

Die Detektion wieder erreichbarer Peers kann über einen Ping- bzw. Heartbeat-Mechanismus erfolgen, der von existierenden Gruppenkommunikationssystemen nicht unterstützt wird. Eine hohe Ping- oder Heartbeat-Rate belastet jedoch die Ressourcen der mobilen Endsysteme, weshalb ein Kompromiss zwischen diesen beiden Anforderungen gefunden werden muss.

Um bei den Peers ohne Verbindungsproblemen zusätzliche Aufwände zu vermeiden, wird in *Moversight* die Prüfung der Wiedererreichbarkeit nur durch die Peers der sekundären Partition durchgeführt. Zunächst versuchen der bzw. die getrennten Peers das Verbindungsproblem zu lösen und den Kontakt zur Gruppe wiederherzustellen. Durch die laufende Partitionierungsbehandlung wird die Verfügbarkeit (A) der Gruppe eingeschränkt. Da der Austausch von Daten innerhalb der Partitionen unbeschränkt möglich ist, wird die Konsistenz (C) des gemeinsamen Wissens zunächst nur in dieser Partition getrennt gesichert.

Der Zeitraum für das Zusammenführen der Gruppe wird durch einen Timer begrenzt. Gelingt das Wiederherstellen der Verbindung zu den anderen Gruppenmitgliedern nicht oder nicht innerhalb einer bestimmten Zeit, werden die nicht erreichbaren Peers aus der Gruppe ausgeschlossen und eine neue Sicht für die reduzierte Gruppe wird eingerichtet. Da während der Partitionierungsbehandlung Peers von der lokalen Partition getrennt werden können, wird auch in diesen Partitionen die Fehlerdetektion fortgesetzt.

Die Wiedererreichbarkeit der primären Partition wird in *Moversight* über die Nachbarschaftserkennung geprüft, die analog dem Ping-Verfahren periodisch Nachrichten aussendet, um die vermissten Gruppenmitglieder zu erreichen. Dabei entscheidet der Anwendungsentwickler, ob dies jeder Peer der sekundären Partition durchführt oder nur deren Master. Letzteres ist der Standardfall, um durch die reduzierte Anzahl von Nachrichten die Ressourcen der Endgeräte zu schonen. Diesem Ansatz liegt die Annahme zugrunde, dass die Netzverbindungen in den einzelnen Partitionen transitiv sind.

Jeder Peer, der eine Erkennungsnachricht des Nachbarschaftsdetektors empfängt, beantwortet sie. Ist der empfangene Peer ebenfalls im Zustand *RECONNECTING*, signalisiert er der Mobilitätsunterstützung, welcher Peer zu ihm Kontakt aufgenommen hat. Der über die Nachbarschaftsdetektion erreichte Peer wird *Kontaktpeer* genannt. Die Nachbarschaftsdetektion erfolgt solange, bis ein getrenntes Gruppenmitglied erreicht wurde oder der Zeitraum für die Zusammenführung der Gruppe überschritten ist.

6.3.6 Gruppenwiedereintritt

Wenn ein getrennter Peer die Gruppe erfolgreich kontaktiert (hat), kann er ihr wieder beitreten. In *Moversight* erfolgt das über die Operation *REINTEGRATE*. Sie ist thematisch eng mit der Operation MERGE verwandt ist (siehe Kapitel 9), jedoch unterscheiden sich beide Operationen durch die Art der Gruppenpartitionierung. Zu Beginn sind beim *MERGE* beide Gruppen nicht partitioniert, bei der Operation *REINTEGRATE* ist die Gruppe bereits partitioniert. Der Kontaktpeer steuert den Wiedereintritt,

[6] Siehe Kapitel 9

indem er die Gruppeninformationen der wiedereintretenden Peers aktualisiert und deren Verbindungszustand in der Gruppensicht auf *JOINED* setzt. Der Wiedereintritt kann jeweils nur für eine Partition erfolgen. Wurden mehrere Partitionen erreicht, so erfolgt deren Wiedervereinigung nacheinander.

Mit der Verteilung der *ReJoinAnnounce-Nachricht* (RA) werden neu zu versendende Gruppennachrichten zwischengespeichert (siehe Kapitel 8). Wird die RA-Nachricht durch einen Peer der sekundären Partition nicht empfangen, so wird dieser ebenso als vermisst behandelt und nicht weiter berücksichtigt[7]. Nach Erhalt der *RA*-Nachricht wechseln die Peers zur Wiedereingliederung in den Zustand *REINTEGRATING*.

Der Zielpartition wird die Kontaktaufnahme mittels einer *Reduced-Roster-Nachricht* (RRO) an den Kontaktpeer mitgeteilt. Sie enthält die Liste der aktiven Peer-IDs der sekundären Partition mit ihrer aktuellen Transportadresse und die IDs der aus der sekundären Partition ausgetretenen Peers.

Diese Daten sind notwendig, da sich während der Teilung die Clusterzuordnung, die Rollen der Peers (siehe hierzu Kapitel 8) sowie – durch einen Netzwechsel – deren Transportadresse geändert haben können. In der Zielpartition werden diese Informationen allen Peers zugestellt. Die Peers der beizutretenden Partition aktualisieren ihre Gruppenverwaltungsinformationen mit den Informationen der RRO-Nachricht[8].

Um die Reintegration auch in der sekundären Partition zu vollziehen, werden die aktualisierten Gruppendaten der Zielpartition durch den Kontaktpeer an die sekundäre Partition übermittelt und dort verteilt. Die Peers aktualisieren ihre Gruppeninformation entsprechend. Um sicherzustellen, dass die Vereinigung erfolgreich abgeschlossen wurde, sendet der Kontaktpeer im Anschluss daran eine Nachricht an die gesamte Gruppe.

Können die Gruppendaten nicht wechselseitig ausgetauscht bzw. in den Partitionen verteilt werden, wird der Wiedereintritt abgebrochen. In [192] werden die verschiedenen potentiellen Fehlersituationen und deren Behandlung im Detail diskutiert. Vereinfacht dargestellt wird bei einem Fehler die Sicht der beteiligten Peers auf die optimistische Sicht vor Beginn des Wiedereintritts zurückgesetzt.

6.3.7 Abgleich des Gruppenwissens

Nach dem erfolgreichen Wiedereintritt muss das Gruppenwissen abgeglichen werden. Dabei sind zwei Probleme lösen: (1) die Bestimmung der wechselseitig vermissten Nachrichten der zu vereinigenden Partitionen und (2) der eigentliche Nachrichtenabgleich.

Wie für die Gruppenwiedereintrittsoperation existiert auch für den Wissensabgleich in anderen Gruppenkommunikationssystemen keine vergleichbare Operation. Zum Datenabgleich zwischen verschiedenen Endgeräten gibt es jedoch zahlreiche Arbeiten und Ansätze, überwiegend aus dem Bereich der Cloudspeicher.

Die ausgetauschten Gruppennachrichten können als einzelne Dateien angesehen werden. So präsentieren beispielsweise Bao *et al.* [20] mit dem *SyncView*-Ansatz eine Dateiabstraktion, welche Dateien zwischen mehreren Endgeräten abgleicht. Dazu wird ein Client-Server-basiertes Konsistenzprotokoll benutzt, das einen Konfliktbehebungsmechanismus bietet. Liang *et al.* [141] und Lopez *et al.* [148] diskutieren vergleichbare Ansätze. Für die Gruppensicht können diese Ansätze jedoch nicht verwendet werden. Wang *et al.* [213] präsentieren einen Algorithmus für den Wissensabgleich innerhalb einzelner Dateien. Ziel ist es, mit minimalen Kommunikations- und Ressourcenaufwand inhaltliche Änderungen (Einfügungen, Änderungen und Löschungen) an ihnen abzugleichen, was den Änderungen an der Gruppensicht entspricht. Alle Ansätze sind kompliziert strukturiert. Die verwendete Dateiabstraktion passt nicht zum gemeinsamen Wissen bzw. den ausgetauschten Gruppennachrichten, da die betreffenden Datenmengen bei Wang deutlich geringer sind. Ebenso widerspricht die Client-Server-basierte Kommunikation dem P2P-basierten Ansatz kollaborativer Anwendungen.

Der in dieser Arbeit entwickelte Ansatz für den Wissensabgleich berücksichtigt dagegen die Anforderungen mobiler kollaborativer Anwendungen und hat eine geringe Komplexität. Für die Resynchronisation wechseln die wiedereingetretenen Peers in den

[7] Für eine ausführliche Diskussion möglicher Fehlersituationen sei hier auf [192] verwiesen.

[8] Im Sinne von Entfernen von ausgetretenen Peers sowie Aktualisierung von Zustand und Transportadresse der aktiven Peers der sekundären Partition.

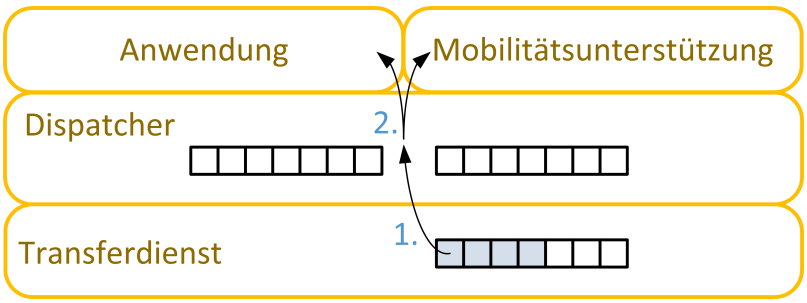


Abbildung 6.7: Normales Auslieferungsschema von *Moversight*

Zustand *RESYNCHRONIZE*. Als aktive Gruppenmitglieder bearbeiten auch die wiedereingetretenen Peers empfangene neue Gruppennachrichten protokollkonform. Parallel dazu werden die Nachrichten von der Gruppe abgefragt, welche während der Abwesenheit der Peers ausgetauscht wurden, d. h. die *vermissten Nachrichten*. Für deren Abfrage ist der Synchronisationsdienst zuständig, welcher ein Teil der Mobilitätsunterstützung ist.

Vermisste Nachrichten Jeder Peer der Gruppe speichert die Gruppennachrichten der aktuellen und der vorangegangenen Sicht. Aus dieser Nachrichtenmenge fordert der wieder eintretende Peer alle vermissten Nachrichten von einen Peer der ehemals primären Partition an. Die vermissten Nachrichten werden an ihn übertragen und entsprechend ihrer ursprünglichen Auslieferungsreihenfolge verarbeitet. Über die Konfiguration wird festgelegt, ob der Abgleich wechselseitig oder nur in Richtung der sekundären Partition erfolgt. Letzteres ist der Standardfall, um den Abgleichaufwand zu begrenzen.

Nachrichtenabgleich Für den Nachrichtenabgleich werden verschiedene Abgleichstrategien genutzt. Sie legen fest, von welchen Peers der Gruppe ein wieder eingetretener Peer die vermissten Nachrichten abruft. Im Rahmen dieser Arbeit wurde eine einfache, rein P2P-basierte Abgleichstrategie umgesetzt[9], bei der jeder wieder eingetretene Peer die fehlenden Nachrichten von seinem Master abruft. Ist der wieder eingetretene Peer selbst Master bzw. ist sein Master mit ihm zusammen zur Gruppe zurückgekehrt, wird der Master mit der kleinsten Peer-ID aus der ehemals primären Partition gewählt.

Der wiedereingetretene Peer übermittelt die Referenz der Nachricht, die als letztes vor dem Verbindungsabbruch von der Gruppe erfolgreich bestätigt wurde, an seinen Master. Der sendet ausgehend von dieser Referenz alle nachfolgend gespeicherten Nachrichten zuverlässig an den anfragenden Peer. Für die Wiederholung fehlgeschlagener Anfragen sind die wieder eingetretenen Peers verantwortlich.

Für die Wiederherstellung des konsistenten Anwendungsstands ist es jedoch notwendig, dass die vermissten Nachrichten vor den neu empfangen Gruppennachrichten an die Anwendung ausgeliefert werden. Aus diesem Grund wird während des Wissensabgleichs die Auslieferung der Gruppennachrichten geändert. Abbildung 6.7 zeigt das normale Auslieferungsschema. Bei der Auslieferung einer Gruppennachricht vom Transferdienst an die Anwendung erhält die Mobilitätsunterstützung eine Kopie dieser Nachricht. Die verwendete Abgleichstrategie entscheidet, ob diese Kopie gespeichert oder verworfen wird.

Abbildung 6.8 stellt die Auslieferung der Nachrichten während des Wissensabgleichs dar. Durch den Synchronisationsdienst werden die vermissten Gruppennachrichten von der Gruppe abgerufen (1), in die ursprüngliche Reihenfolge (B) gebracht und sukzessive an die Anwendung ausgeliefert (2). Nachrichten, welche durch die Transferdienste (3) auszuliefern sind, werden zunächst zwischengespeichert (A)[10]. Nachdem die letzte vermisste Nachricht ausgeliefert wurde, kehrt der Peer in den *JOINED*-Zustand zurück

[9] Weiterführende Ideen und Ansätze für komplexere Abgleichstrategien werden in Kapitel 10 skizziert.

[10] Einzige Ausnahme sind Kontrollnachrichten des Fehlerdetektors.

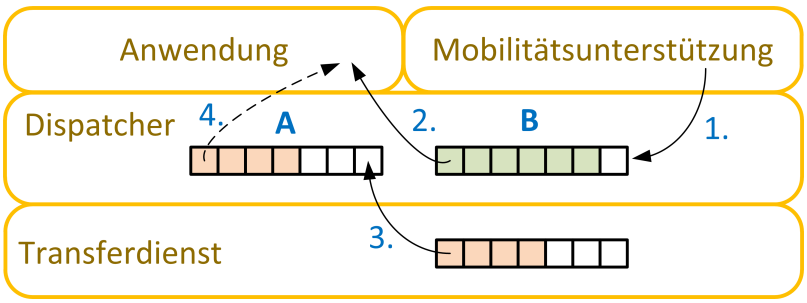


Abbildung 6.8: *Moversight* Auslieferungsschema während des Wissensabgleichs

und liefert die zwischengespeicherten Nachrichten an die Anwendung aus. Ab diesem Zeitpunkt ist die virtuelle Synchronität zwischen den wieder eingetretenen Peers und der Gruppe wiederhergestellt. Weitere Details zum Nachrichtenabgleich, insbesondere zur Bestimmung der Größe der notwendigen Warteschlangen und zur Speicherdauer der einzelnen Nachrichten sind unter anderem in [91] ausgeführt.

Gesamtablauf Die Kombination der beschriebenen Schritte zum Erkennen und Beheben von Netzpartitionierungen[11] bildet den Kern der Behandlung der Nutzermobilität in *Moversight*. Sie besteht im Wesentlichen aus der Kooperation der drei Dienste TOM-*Transfer*, *Fehlerdetektion* und *Mobilitätsunterstützung* (siehe Abbildung 6.9). Dabei steht die Operation *RECONNECT* für das Wiederherstellen der Verbindung, die Operation *REINTEGRATE* für den Gruppenwiedereintritt und die Operation *RESYNCHRONIZE* für den Abgleich des Gruppenwissens.

6.4 Mobile Optimistic Virtual Synchronity

Der beschriebene Ansatz zur Mobilitätsunterstützung entspricht nicht den bisherigen Varianten der virtuellen Synchronität, wie sie im Abschnitt 4) eingeführt wurden. Deshalb wird in diesem Abschnitt ein neues semantisches Modell für die virtuelle Synchronität vorgestellt, das insbesondere die Anforderungen der Nutzermobilität berücksichtigt. Es heißt *Mobile Optimistic Virtual Synchronity* (MOVS) (engl. *mobile optimistic virtual synchrony*) [91, 94].

Das ursprüngliche Konzept der virtuellen Synchronität wurde für stationäre Umgebungen entworfen, in denen Netzpartitionierungen selten auftreten. Weiterentwicklungen berücksichtigen partiell den Partitionierungsaspekt und berücksichtigen eingeschränkt auch mobile und nomadische Nutzer. Die verschiedenen Varianten der virtuellen Synchronität können für die Peers sehr ressourcenintensiv sein. Kehrt ein getrennter Peer nicht zur Gruppe zurück, ist ein Rollback des Gruppenwissens erforderlich. Dadurch wird das Wissen der Anwendung auf den Zeitpunkt vor dem Kontaktverlust mit dem Peer zurückgesetzt. Der erforderliche Mehraufwand für den Nachrichtenaustausch während der Abwesenheit der Peers war vergeblich. Aus diesem Grunde wurde ausgehend von der optimistischen virtuellen Synchronität eine neue Variante definiert, in der temporär nicht erreichbare Peers als passive Gruppenmitglieder betrachtet werden, die zeitnah zur Gruppe zurückkehren. Dadurch wird ein nahtloser Wiedereintritt in eine geschlossene Gruppe in Situationen mit häufigen Verbindungsabbrüchen ermöglicht, der den Aufwand für die Resynchronisation und den Gruppenwiedereintritt weitgehend auf die wiedereintretenden Peers verlagert. Es ist kein Rollback mehr erforderlich. Damit können die übrigen Gruppenmitglieder ihre Zusammenarbeit fortsetzen und werden kaum mit Resynchronisationsoperationen belastet.

[11] Siehe Abschnitt 6.3

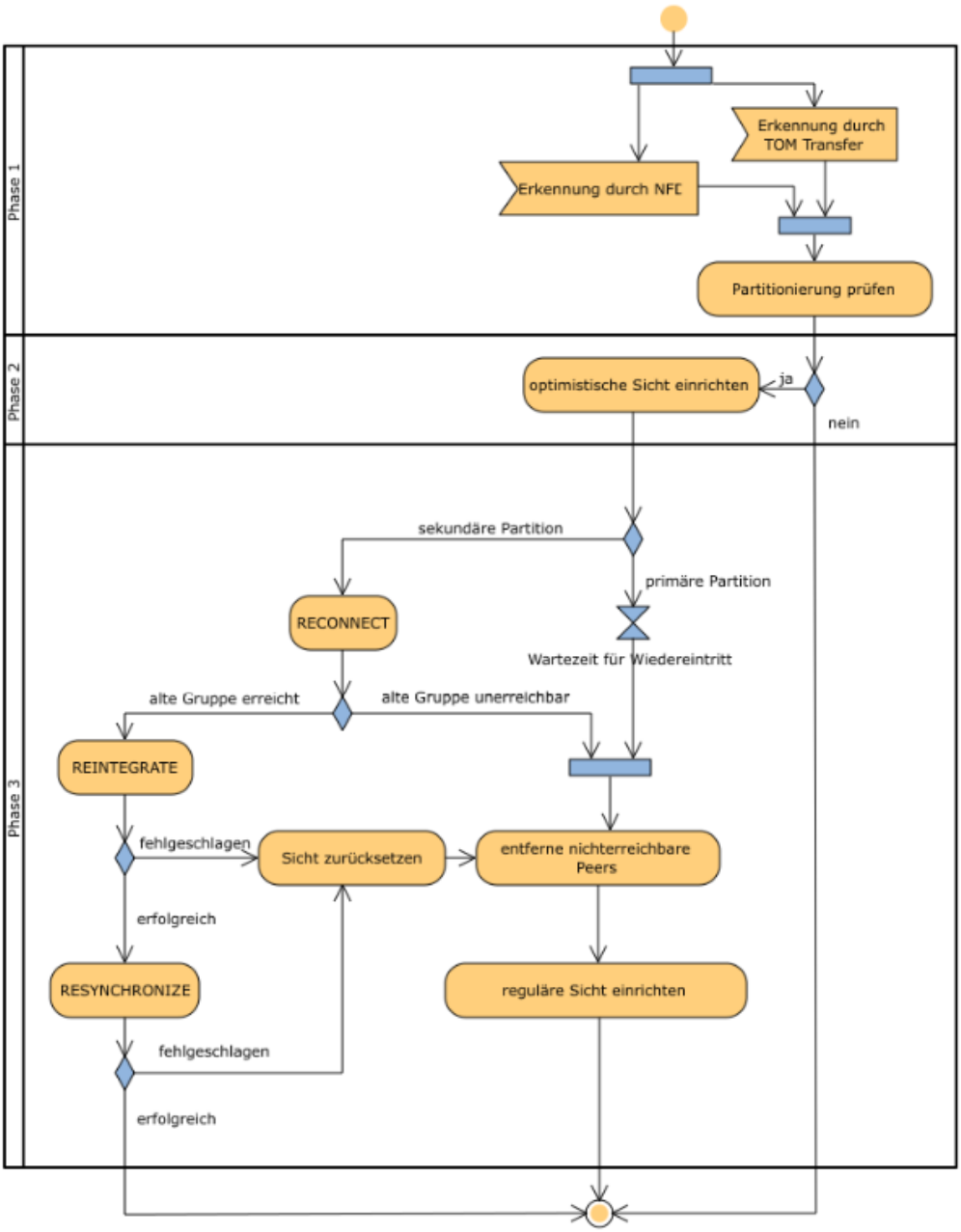


Abbildung 6.9: Ablauf Gruppenwiedereintritt

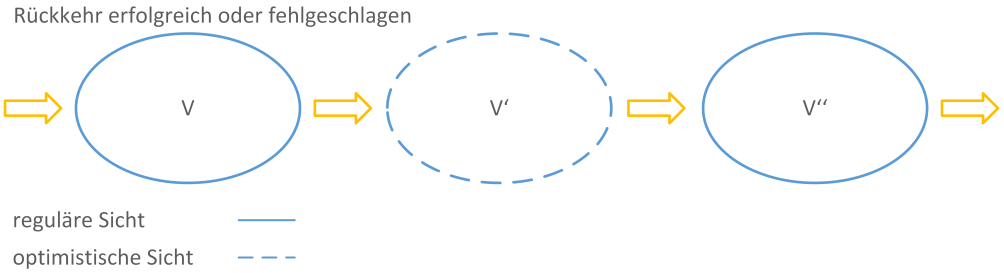


Abbildung 6.10: Sichtübergang bei der Verwendung des MOVS-Paradigmas

6.4.1 Sichtprinzip

Diesem Ansatz liegt folgendes Prinzip zugrunde: Verliert ein Peer seine Verbindung zur Gruppe, wird die Zustellung und Auslieferung der Nachrichten innerhalb der Gruppe fortgesetzt und die Gruppenverwaltung richtet vergleichbar zur optimistischen virtuellen Synchronität eine optimistische Sicht ein. Diese unterscheidet sich von einer regulären Sicht dadurch, dass sie mindestens einen pending Peer enthält. Auf eine optimistische Sicht folgt entweder eine weitere optimistische Sicht oder eine reguläre Sicht. Mit der Einrichtung der optimistischen Sicht informiert die Gruppenverwaltung die Gruppe darüber, welcher Peer vermisst wird. Kehrt der pending Peer innerhalb eines bestimmten Zeitintervalls zur Gruppe zurück, wird eine neue reguläre Sicht installiert, sofern kein weiterer Peer vermisst wird. Ist ein pending Peer nicht in der Lage, seine Verbindung mit der Gruppe (rechtzeitig) wiederherzustellen, so wird eine neue Sicht ohne diesen Peer installiert. In Abhängigkeit davon, ob noch ein Peer vermisst wird, kann dies eine optimistische oder eine reguläre Sicht sein. Abbildung 6.10[12,13] zeigt beispielhaft den Sichtwechsel beim MOVS-Paradigma.

6.4.2 Nachrichtenversand

Im Gegensatz zu anderen Varianten der virtuellen Synchronität wird bei MOVS ein pending Peer weiter als ein reguläres, aber passives Mitglied der Gruppe betrachtet. Daher werden bei einer Nachrichtenübertragung diese Peers ebenfalls als valide Empfänger angesehen, ohne sie tatsächlich in den Versand einzubeziehen.

Der erfolgreiche Versand von Nachrichten wird bei MOVS auch über Bestätigungen ermittelt. Die globale Sendebestätigung für jede Nachricht an die Gruppe wird zusätzlich genutzt, um anzuzeigen, welche Empfänger die Nachricht nicht bestätigt haben, also als pending markiert wurden, vergleichbar mit dem *safe*-Message-Konzept der erweiterten virtuellen Synchronität. Dementsprechend aktualisieren die Empfänger ihre Gruppeninformationen. Die Nachrichten selbst werden unabhängig von den mit ihnen verknüpften schwebenden Peers lokal zugestellt. Die Anwendungen bzw. höheren Dienste des Gruppenkommunikationssystems können auf die Menge der Peers, die den Empfang der Nachricht m nicht bestätigt haben, über die Funktion $m.pending$ zugreifen.

Da die Gruppenverwaltung jederzeit Kenntnis besitzt, ob es einen pending Peer gibt, können die mobile kollaborative Anwendung oder ein anderer Dienst für jede Nachricht sofort entscheiden, ob diese versandt werden soll. Ebenso kann sofort entschieden werden, ob die Nachricht bei der lokalen Zustellung verarbeitet oder verworfen wird (in Abhängigkeit von den anwendungs- oder dienstspezifischen Anforderungen). Diese Entscheidung kann im Gegensatz zur optimistischen virtuellen Synchronität durch die Anwendung allein auf Basis der empfangenen Nachricht getroffen werden. Darüber hinaus haben frühere Ereignisse in der Gruppenverwaltung keinen Einfluss auf den zukünftigen Anwendungszustand, weshalb eine Zurücknahme der optimistischen Sicht durch ein Rollback nicht mehr erforderlich ist.

[12] Bedingung fehlgeschlagen: $V''.members == V.members\ p$
[13] Bedingung erfolgreich: $V''.members == V.members$

Prinzipiell unterstützt die mobile optimistische virtuelle Synchronität beim Nachrichtenversand sowohl die Sendesicht- als auch die Gruppensichtzustellung. Dazu wird bei einer Nachricht m über $m.requiresSendingViewDelivery$ abgefragt, welche Zustellungseigenschaft für die Nachricht gefordert ist. Festgelegt wird dieses Attribut vom Sender der Nachricht.

6.4.3 Bestimmen der vermissten Nachrichten

Für die erneute Synchronisation des Wissens muss die Anzahl der vom zurückkehrenden Peer vermissten Nachrichten bestimmt werden. Als Kriterium für die Ermittlung der vermissten Nachrichten wird die Stabilität[14] der Nachrichten genutzt. Dazu werden die ausgetauschten Nachrichten vier Kategorien zugeordnet:

1. ***Global stabile*** Nachrichten – Menge der Nachrichten, welche alle Peers der Gruppe empfangen haben.
2. ***Global instabile*** Nachrichten – Menge der Nachrichten, welche aufgrund von Verbindungsabbrüchen oder Paketverlusten von einigen Peers der Gruppe nicht empfangen wurden.
3. ***Lokal stabile*** Nachrichten – Menge der von einem Peer lokal erfolgreich empfangenen und bestätigten Nachrichten.
4. ***Letzte lokal stabile*** Nachricht, die ein Peer lokal empfangen und der Gruppe global bestätigt hat.

Eine Nachricht ist bezüglich des wieder eingetretenen Peers ***lokal stabil***, sofern er nicht als schwebender Peer für diese Nachricht geführt wird. Ist sie für alle Peers der Gruppe lokal stabil, wird sie als ***global stabil*** bezeichnet. Die Menge der lokal stabilen Nachrichten eines Peers kann durch die ***letzte lokal stabile*** Nachricht repräsentiert werden. Alle ***global instabilen*** Nachrichten werden für die Wiederherstellung der Anwendungskonsistenz gespeichert. Die Zahl der vermissten Nachrichten umfasst also alle Nachrichten, die entsprechend der globalen totalen Ordnung bis zum Wiedereintritt des Peers in der Gruppe ausgetauscht wurden.

6.5 Protokollfunktionen zur Konsistenzsicherung

Zur Umsetzung der mobilen optimistischen virtuellen Synchronität war die Entwicklung von speziellen Protokollfunktionen notwendig, welche die Sicherung der Konsistenz des globalen Wissens in mobilen kollaborativen Anwendungen unterstützen.

6.5.1 Nachrichteneingangsprüfung im TOM-Transfer

Moversight sichert der Anwendung eine globale total geordnete Nachrichtenauslieferung zu, welcher in erster Linie über eine kontinuierliche Reorganisation der Warteschlange im TOM-Transfer realisiert wird. Nachrichten, welche nicht innerhalb der Gruppe zugestellt werden können, blockieren die Warteschlange und verzögern deren Reorganisation bzw. machen sie unmöglich. Eine häufige Ursache dafür sind ungültige Sichten und nicht mehr der Gruppe zugehörige oder temporär nicht erreichbare Peers.

Daher muss der TOM-Transfer für jede Nachricht, die ein Peer von der Gruppe empfängt, prüfen, ob diese zu akzeptieren oder zu verwerfen ist. Die Eingangsprüfung wird vor allem über den Zustand des Peers gesteuert (vgl. Abschnitt 6.3.3). Allgemein werden neue Nachrichten akzeptiert, wenn die Sendesichteigenschaft erfüllt ist. Ist die Sicht-ID der Nachricht größer als die des Peers, wird die Verarbeitung dieser Nachricht auf einen späteren Zeitpunkt verschoben. Sonstige Nachrichten werden über den Rückweisungsmechanismus verworfen. In den Zuständen *DISJOINED*, *WAITING_FOR_ROSTER* und *LEAVING* werden grundsätzlich alle Nachrichten abgelehnt.[15]

[14] Vergleichbar mit dem *safe*-Message-Konzept der erweiterten virtuellen Synchronität.
[15] Da der Peer noch kein bzw. nicht mehr vollständiges, aktives Gruppenmitglied ist.

Nachrichten, welche durch den lokalen Peer schon „*gesehen*“ wurden (im Sinne schon einmal empfangen und evtl. aktuell in Bearbeitung), müssen gesondert behandelt werden. Der wiederholte Empfang einer Nachricht kann ein Hinweis darauf sein, dass die zugehörige Bestätigungsnachricht verloren gegangen ist. Ist die Verarbeitung für diese Nachricht noch nicht abgeschlossen, so wird die Bestätigung wiederholt versandt, anderenfalls wird die Nachricht verworfen.

Eine besondere Bedeutung bei der Konsistenzsicherung hat der Zustand *FLUSHING*[16] (vgl. Abschnitt 6.3.3). Ein Peer wechselt in diesen Zustand, wenn eine potentiell sichtändernde Nachricht verarbeitet wird. Sofern sie bisher für den Peer unbekannt ist und ihre Sendesicht der Sicht des Peers entspricht, wird die Nachricht speziell[17] geprüft, anderenfalls verworfen.

Die Prüfung einer potentiell sichtändernden Nachricht ist abhängig vom Inhalt der Warteschlange. Ist diese leer oder enthält keine sichtändernden Nachrichten, wird die neue Nachricht akzeptiert. Wenn bereits eine sichtändernde Nachricht verarbeitet wird, so wird diejenige Nachricht verarbeitet, welche die kleinste Nachrichtenreferenz besitzt. Gegebenenfalls wird der Abbruch der Nachrichtenverarbeitung über den Rückweisungsmechanismus angezeigt (siehe folgenden Abschnitt). Nicht sichtändernde Nachrichten werden im Zustand *FLUSHING* akzeptiert, wenn es Sendewiederholungen sind. Ist die empfangene Nachricht nicht sichtändernd und nicht wiederholt versendet worden, so wird deren Verarbeitung zeitlich nach dem Sichtwechsel verschoben.

6.5.2 Der Nachrichtenrückweisungsmechanismus des TOM-Transfers

Um die Reaktionsfähigkeit von *Moversight* aufrechtzuerhalten, ist es notwendig, Übertragungen von dauerhaft unzustellbaren Nachrichten zeitnah abzubrechen. Dafür wird der Rückweisungsmechanismus des TOM-Transfers genutzt, welcher unnötige Sendewiederholungen, Wartezeiten sowie Split-Brain-Situationen[18] vermeidet. Wird zum Beispiel eine Nachricht mit einer ungültigen Sendesicht empfangen, so wird in der Gruppe eine Rückweisungsnachricht verteilt. Diese sendet der zurückweisende Peer an den Peer, von dem er die zurückgewiesene Nachricht erhalten hat. Die Master verteilen die Rückweisung invers zum Datenpfad (vgl. Abschnitt 5.2.3) weiter. Im Anschluss wird geprüft, ob die korrespondierende Nachricht lokal in der Warteschlange gespeichert ist. Dazu wird wiederum die Nachrichtenreferenz verwendet. Ist dies der Fall, so wird sie aus allen Warteschlangen entfernt und sämtliche zugehörige Timer werden gestoppt. Ist der Peer der Sender der Nachricht, wird der Anwendung ein fehlgeschlagener Sendeversuch signalisiert.

6.5.3 Parallele Sichtänderungsoperationen

Eine kollaborative Gruppe besteht aus autonom agierenden Teilnehmern, deren Handlungen sich oft beeinflussen. Diese Aktivitäten muss das Protokoll global ordnen. Ein besonderes Problem stellen dabei parallel initiierte Sichtänderungen dar, welche die gemeinsame Sicht der Gruppe zerstören können. Das Prinzip des Erkennens und Behandelns solcher Situationen durch die mobile optimistische virtuelle Synchronität zeigt das folgende Beispiel:

Eine Gruppe soll aus sechs, auf drei Cluster verteilten Mitgliedern bestehen. Je ein Master (M_1, M_2 und M_3) verwaltet einen Slave (entsprechend S_1, S_2 und S_3). Wollen nun alle Slaves aus der Gruppe austreten, kann das Protokoll ohne weitere Maßnahmen unter Umständen längere Zeit nicht reagieren bzw. kein Austrittswunsch kann erfolgreich bearbeitet werden. Jeder Master ordnet den von seinem Slave empfangenen Austrittswunsch in die Warteschlange ein und beginnt dessen Weiterverteilung[19]. Da ein Gruppenaustritt sichtändernd ist, wechseln die Master jeweils in den Zustand *FLUSHING* (siehe Abschnitt 6.3.3), in welchem weitere eintreffende sichtändernde

[16] Siehe Abschnitt 6.3 sowie Kapitel 8
[17] Der Prüfalgorithmus ist im Pseudocode im Algorithmus 11 im Anhang B dargestellt
[18] Vergleiche Kapitel 4
[19] Entsprechend des Kommunikationsschemas aus Abschnitt 5.2.2

Nachrichten zwischengespeichert werden. Erhält M_1 der Austrittswunsch von M_2, so ist zu prüfen, ob dieser (1) zu verwerfen oder (2) anzunehmen ist.

(1) Beim grundsätzlichen Verwerfen neuer potentiell sichtändernde Nachrichten würden M_1, M_2 und M_3 die Verteilung des Austrittswunsches nicht abschließen können. Die Master würden erfolglos auf die Bestätigung der versendeten Nachrichten warten. Diese Situation wird durch den Fehlerdetektor (korrekterweise) nicht als fehlerhaft bewertet[20]. Mit dem Erreichen der maximalen Anzahl von Sendewiederholungen[21] werden die Übertragungen abgebrochen und die Blockade aufgelöst. Es sind auch Situationen vorstellbar, in denen das wechselseitige Warten auf die Bestätigung für eine sichtändernde Nachricht nur einzelnen Peers betrifft. In diesem Falle würden die Zustände innerhalb der Gruppe bei den Peers variieren, was zu einem inkonsistenten Zustand des gemeinsamen Wissens führt. Daher ist ein grundsätzliches Verwerfen von potentiell sichtändernden Nachrichten im Zustand *FLUSHING* nicht möglich.

(2) Die Annahme weiterer sichtändernder Nachrichten im Zustand *FLUSHING* würde:

- die Rückkehr in den Normalzustand stark verzögern und
- das Risiko eines inkonsistenten Gruppenwissens in sich bergen.

Ursache dafür können unterschiedliche Verzögerungen oder Paketverluste sein, wodurch beispielsweise die Warteschlange bei M_1 noch gefüllt und bei M_2 bereits geleert ist. Daraufhin würde M_2 diejenige sichtändernde Operation ausführen, welche das Leeren der Warteschlangen ausgelöst hat. Anschließend würden durch den Wartungsdienst beispielsweise Peers zwischen den Clustern verschoben werden (siehe Kapitel 8). Durch die unterschiedlichen Sichten der Peers M_1 und M_2 würde M_2 eintreffende Nachrichten aus der alten Sicht verwerfen. Dieser inkonsistente Zustand der Gruppe wäre sehr schwer zu erkennen.

Aus diesem Grund wird bei der Nachrichteneingangsprüfung sichergestellt (siehe oben), dass bei parallelen potentiell sichtändernden Nachrichten sich diejenige Nachricht durchsetzt, die die kleinere Nachrichtenreferenz hat. Dadurch wird in solchen Situationen nur genau eine sichtändernde Operation ausgeführt.

6.6 Leistungsbewertung

Abschließend wird die Funktions- und Leistungsfähigkeit des vorgestellten Ansatzes zur Behandlung der Nutzermobilität demonstriert. Eine vollständige Untersuchung des Leistungsverhaltens ist in [192] gegeben. Dafür werden fünf Testfälle genutzt, die sich durch die Anzahl der von der Gruppe abgespaltenen Peers unterschieden. Zur Vereinfachung soll angenommen werden, dass die Gruppe immer in zwei Partitionen zerfällt, wobei die abgespaltene Partition später weiter zerfallen könnte. Die Testfälle sind charakterisiert durch die Abspaltung (1) eines Peers, (2) einer beliebigen Anzahl von Peers und (3) exakt der Hälfte der Peers. Testfall (2) unterteilt sich weiter in (2.1) zwei Peers werden abgespalten, (2.2) ein Viertel aller Peers wird abgespalten und (2.3) die abgespaltene Partition umfasst exakt einen Peer weniger als die andere Partition.

Das Erkennen einer Partitionierungssituation und ihre Behandlung hängt nicht von der konkreten Zahl abgespaltener Peers ab. Weiter wird angenommen, dass Testfall (1) in kollaborativen Anwendungen am häufigsten auftritt. Die konkrete Clusterstruktur der einzelnen Partitionen ist für die Beurteilung der Leistungsfähigkeit nicht relevant.

Die einzelnen Testfälle wurden für die Gruppengrößen 6, 12, 24, 48, 72 und 96 ausgeführt. Die so entstehenden Partitionen sind in Tabelle 6.2 zusammengefasst. Dabei stehen x und y für die Anzahl der Peers in der primären und sekundären Partition.

Zum Erkennen einer Partitionierungssituation mit und ohne erfolgreichem *REJOIN* werden die Operationsdauer und die Anzahl der in diesen Zusammenhang versendeten Nachrichten bestimmt. Letztere wird mit der Anzahl von Nachrichten in dem

[20] Da bei der zuverlässigen Übertragung durch den Basistransfer jeweils eine Bestätigung für jeden Übertragungsversuch generiert wird.

[21] Was bei entsprechenden Timeoutwerten mehrere Minuten dauern kann.

Tabelle 6.2: Aufteilung der Gruppe in Partitionen abhängig vom Testfall und der Peer-Anzahl

Peers	**Tesfälle**				
	1	**2.1**	**2.2**	**2.3**	**3**
6	(1,5)	(2,4)	(2,4)	(2,4)	(3,3)
12	(1,11)	(2,10)	(3,9)	(5,7)	(6,6)
24	(1,23)	(2,22)	(6,18)	(11,13)	(12,12)
48	(1,47)	(2,46)	(12,36)	(23,25)	(24,24)
72	(1,72)	(2,70)	(18,54)	(35,37)	(36,36)
96	(1,95)	(2,94)	(24,72)	(47,49)	(48,48)

üblicherweise verwendeten *LEAVE/JOIN*-Ansatz verglichen, wonach verlorene Peers ausgeschlossen und explizit neu eingeladen werden.

Für die Evaluierung wurde wiederum das Simulationsframework OMNeT++ und der Testaufbau aus Abschnitt 5.5 genutzt. Zur Anwendung kam dabei das kleine Backbonenetz mit 10 Routern. Die Verbindung zwischen den Peers und den Routern wurde idealisiert mit einer Verzögerung von 0,0001 ms und einer Bandbreite von 10 Mbit/s modelliert, die Verbindung zwischen den Routern mit einer Verzögerung von 0,001 ms sowie einer Bandbreite von 512 Mbit/s.

6.6.1 Erfolgreicher Wiedereintritt

Zunächst wurde die Dauer der Partitionierungserkennung bestimmt, d. h. die Zeit zwischen dem Erkennen einer Fehlersituation durch einen Peer der Gruppe und der erfolgreichen Installation der optimistischen Sicht bei allen Peers der Partition. Bei der Messung wurde die Dauer für die Nachbarschaftsdetektion der *RECONNECT*-Operation nicht berücksichtigt, da diese über einen Timer fest definiert ist. Die Messung wurde für jeden Peer der Gruppe durchgeführt. Anschließend wurde die gesuchte Operationsdauer über den Maximalwert der Einzelmessungen bestimmt.[22]

Die Operationsdauer für den erfolgreichen Wiedereintritt ist für die einzelnen Testfälle in Tabelle 6.3 und in Abbildung 6.11 dargestellt. Die Dauer der Partitionserkennung und des Wiedereintritts charakterisieren den Ansatz zur Behandlung der Nutzermobilität am besten.

Zu erkennen ist, dass die Ergebnisse linear abhängig von der Anzahl der Peers sind, wobei sich die Testfälle aufgrund der geringeren Anzahl von versendeten Nachrichten nur geringfügig unterscheiden.[23]

Vergleichend wurden die Testfälle mit dem *LEAVE/JOIN*-Ansatz simuliert[24]. Die Dauer des *LEAVE/JOIN*-Ansatzes bestimmt sich aus der Zeitdifferenz zwischen dem Entfernen des nichterreichbaren Peers (*LEAVE*) und dessen Wiedereintritt (*JOIN*). Die Messungen erfolgten wiederum bei allen Peers der verbleibenden Gruppe. Es wurde jeweils die maximale Zeitdifferenz gemessen. Die Ergebnisse sind in Tabelle 6.4 und in Abbildung 6.12 dargestellt.

Im Vergleich zu *Moversight* zeigt sich beim *LEAVE/JOIN*-Ansatz eine größere Spannweite ($[0,0;0,6]$ versus $[0,0;2,4]$), was hauptsächlich durch die Dauer der Operationen *LEAVE* und *JOIN* begründet ist, da der parallele Austritt der Gruppenmitglieder[25] durch die totale Nachrichtenordnung eine größere Verzögerung bei der Nachrichtenauslieferung bewirkt. Die Testfälle (1) und (2.1) unterscheiden sich aufgrund der geringen

[22] In [192] ist eine Ausführliche Darstellung der Messungen gegeben.
[23] Ausführlich in [192] dargelegt.
[24] Siehe Abschnitt 6.2
[25] Ausgelöst durch den gleichzeitigen Ausschluss der nichterreichbaren Peers aus der Partition.

Tabelle 6.3: Dauer der Partitionierungserkennung mit anschließenden Wiedereintritt in Sekunden

Peers	**Testfälle**				
	1	**2.1**	**2.2**	**2.3**	**3**
6	0,0077	0,0102	0,0102	0,0102	0,0104
12	0,0121	0,0144	0,0158	0,0159	0,0154
24	0,0171	0,0199	0,0212	0,0128	0,0126
48	0,0284	0,0314	0,0346	0,0326	0,0326
72	0,0388	0,0419	0,0448	0,0440	0,0461
96	0,0488	0,0519	0,0577	0,0552	0,0551

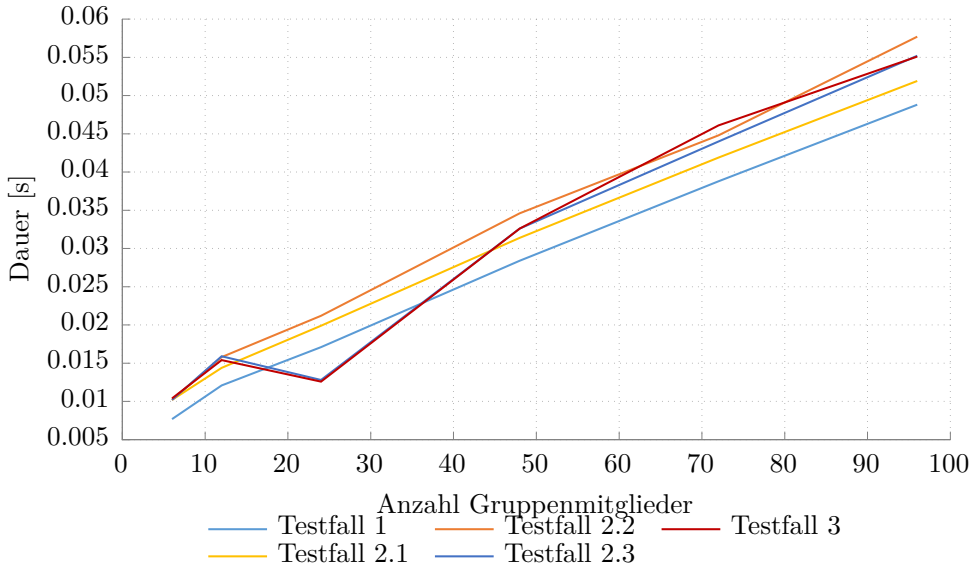


Abbildung 6.11: Dauer der Partitionierungserkennung mit Wiedereintritt

Tabelle 6.4: Dauer *LEAVE* mit anschließendem erneuten *JOIN* in Sekunden

Peers	**Testfälle**				
	1	**2.1**	**2.2**	**2.3**	**3**
6	0,0064	0,0081	0,0081	0,0081	0,0120
12	0,0092	0,0096	0,0183	0,0886	0,1081
24	0,0108	0,0233	0,0730	0,1286	0,1481
48	0,0156	0,0281	0,1535	0,6546	0,6743
72	0,0195	0,0320	0,8991	0,9981	1,4172
96	0,0226	0,0360	1,7326	2,1364	2,3413

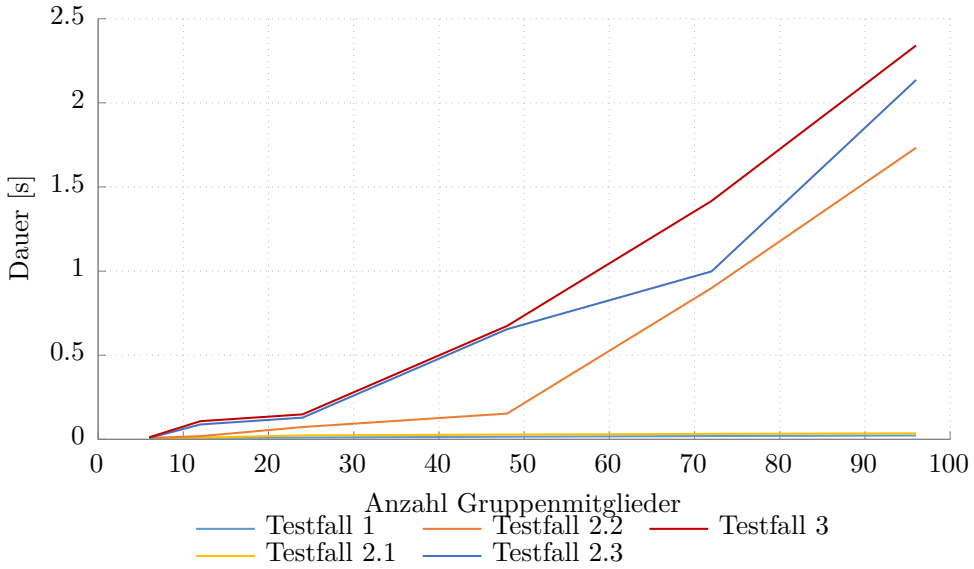


Abbildung 6.12: Dauer der Partitionierungserkennung und Wiedervereinigung mit dem *LEAVE/JOIN*-Ansatz

Anzahl nicht erreichbarer Peers in den Ergebnissen deutlich von den übrigen Testfällen. Allgemein ist festzustellen, dass mit zunehmender Größe der sekundären Partition die Leistungsfähigkeit des *LEAVE/JOIN*-Ansatzes sinkt und im direkten Vergleich zu *Moversight* die bis zu 4, 3-fache Operationszeit benötigt[26]. Für die Testfälle (1) und (2.1) ist der *LEAVE/JOIN*-Ansatz bei kleinen Gruppen jedoch effizienter.

Die Leistungsfähigkeit des *Moversight*-Ansatzes verdeutlicht folgender Extremfall: In einer Gruppe mit 96 Peers sollen gleichzeitig alle Verbindungen ausfallen, wobei jeder Peer eine eigene Partition bildet. Die Gruppe zerfällt somit in 96 sekundäre Partitionen. Sie können mit dem *Moversight*-Ansatz auf zwei Wegen wieder vereinigt werden. (1) Im Idealfall erreichen sich jeweils zwei Partitionen, um sich zusammenzuschließen. Die resultierenden 48 Partitionen erreichen sich wiederum paarweise und vereinigen sich zu Teilgruppen mit je vier Peers. Dieser Ablauf wiederholt sich, bis die ursprüngliche Gruppe wiedervereint ist. Der Ablauf dauert 0,2733 s. (2) Die Wiedervereinigung der einzelnen sekundären Partition kann aber auch seriell erfolgen, d. h. zunächst erreichen sich zwei Einerpartitionen, welche sich vereinigen. Daraufhin vereinigen sich die Zweierpartition und eine Einerpartition, die daraufhin entstehende Dreierpartition mit einer weiteren Einerpartition. Dieser Ablauf wiederholt sich bis zur Zielgruppengröße von 96 Peers und dauert insgesamt 1,3208 s. Damit dauert der erfolgreiche Wiedereintritt in diesem Szenario maximal 1,3208 s. Beim *LEAVE/JOIN*-Ansatz müssen alle 95 Peers wieder in die Gruppe eingeladen werden, sobald sie wieder erreichbar sind. Dafür werden mindestens 1,8918 s benötigt.

6.6.2 Fehlgeschlagener Wiedereintritt

Beim fehlgeschlagenen Wiedereintritt kann die Teilgruppe nach Wiederaufnahme der Kommunikation die Operationen aufgrund von Paketverlusten nicht erfolgreich abschließen. Dabei können die Paketverluste sowohl innerhalb der Partitionen als auch zwischen diesen auftreten. Für die definierten Testfälle wurden zum einen wiederum die Dauer der Partitionierungserkennung und zum anderen die Dauer für das Anzeigen des fehlgeschlagenen Wiedereintritts gemessen. Die Dauer der Partitionierung wird erneut nicht berücksichtigt. Die Messergebnisse in Abbildung 6.13 in Tabelle 6.5 zeigen, dass

[26] Die *Moversight*-Partitionserkennung mit anschließenden *REJOIN* dauert zwischen 7,7 ms und 55 ms, der *LEAVE/JOIN*-Ansatz zwischen 6 ms und 2,4 s

Tabelle 6.5: Dauer der Partitionierungserkennung und Senden der RF-Nachricht in Sekunden

Peers	**Testfälle**				
	1	**2.1**	**2.2**	**2.3**	**3**
6	0,0025	0,0021	0,0021	0,0021	0,0016
12	0,0042	0,0038	0,0040	0,0036	0,0034
24	0,0054	0,0054	0,0050	0,0044	0,0046
48	0,0085	0,0092	0,0082	0,0073	0,0072
72	0,0112	0,0118	0,0104	0,0102	0,0104
96	0,0137	0,0143	0,0137	0,0129	0,0120

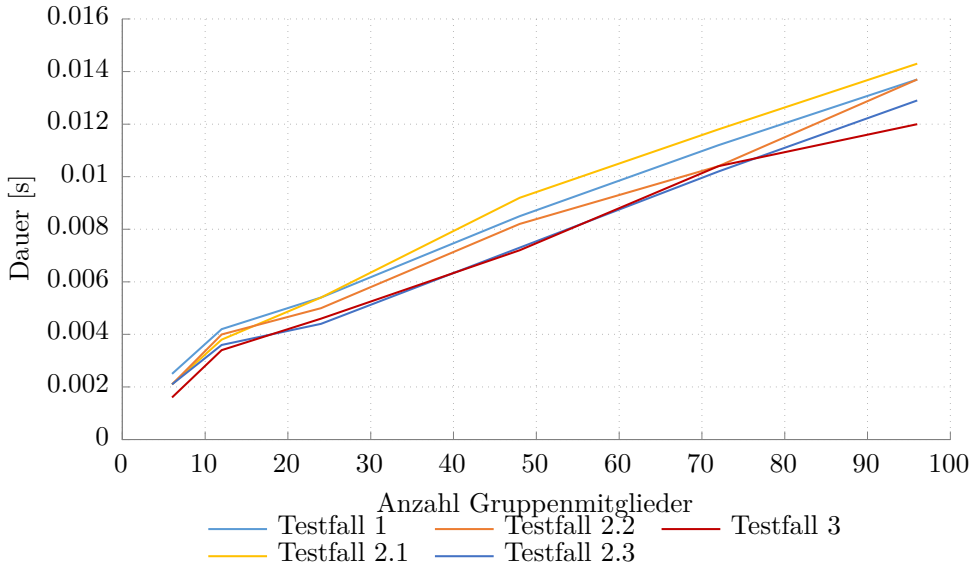


Abbildung 6.13: Operationsdauer bei einem fehlgeschlagenen Wiedereintritt

der Verlauf vergleichbar mit den Ergebnissen des erfolgreichen Wiedereintritts ist[27]. Im Gegensatz dazu ist die Laufzeit der Testfälle mit wenigen abgespaltenen Peers (1 und 2.1) jedoch größer als bei denen mit vergleichbarer Partitionsgröße, zurückzuführen auf die unterschiedlichen Aufwände für die erfolglose *REINTEGRATE*-Operation sowie für den Versand der Abbruchnachrichten. Für den Vorgang wurde eine Gesamtdauer von 2,5 bis 12 ms bestimmt.

Der *LEAVE*/*JOIN*-Ansatz degeneriert im Fehlerfall zu einer *LEAVE*-Operation, deren Dauer vom Austritt des ersten Peers bis zum Austritt des letzten nicht erreichbaren Peers bestimmt wird. Deshalb wird hier mit dem maximalen Messwert für die *LEAVE*-Operation verglichen. Die erzielten Ergebnisse sind in Abbildung 6.14 und in Tabelle 6.6 zusammengestellt. Die Operationsdauer liegt zwischen 2,9 ms und 1,4 ms, was deutlich langsamer ist als *Moversight*. Bei den Testfällen (1) und (2.1) wird die benötigte Operationsdauer fast vollständig durch die Partitionierungserkennung dominiert.

[27] Wiederum ist in [192] die Darstellung der Einzelmessungen gegeben.

Tabelle 6.6: Dauer *LEAVE* in Sekunden

Peers	**Testfälle**				
	1	**2.1**	**2.2**	**2.3**	**3**
6	0,0029	0,0041	0,0041	0,0041	0,0062
12	0,0042	0,0046	0,0079	0,0438	0,0533
24	0,0050	0,0075	0,0172	0,0288	0,0323
48	0,0075	0,0098	0,0352	0,4263	0,4360
72	0,0094	0,0118	0,7189	0,6479	1,0579
96	0,0110	0,0141	1,0375	1,4046	1,3898

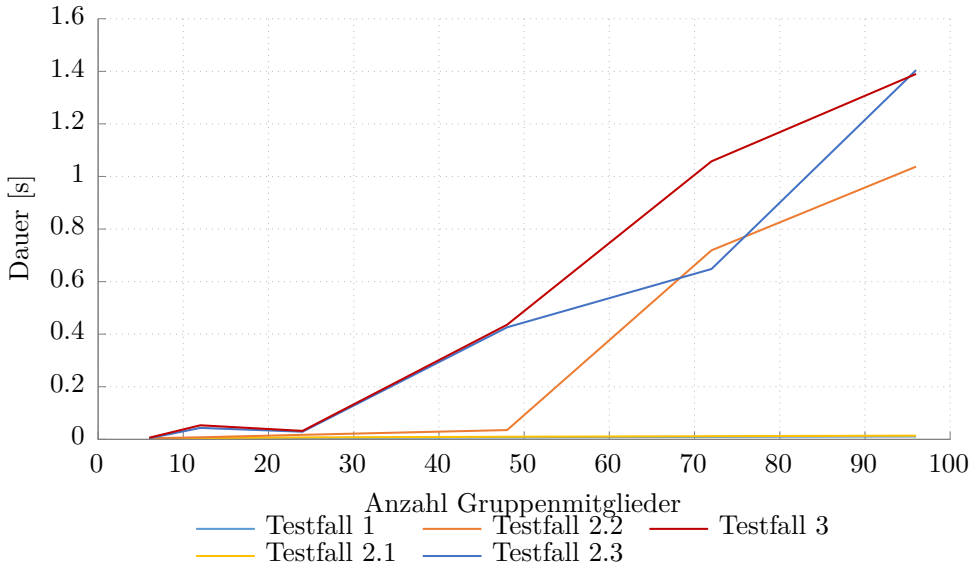


Abbildung 6.14: Operationsdauer eines fehlgeschlagenen Wiedereintrittes beim *JOIN/LEAVE*-Ansatz

6.6.3 Anzahl versendeter Nachrichten

Die Effizienz des Ansatzes kann auch über die Anzahl der ausgetauschten Nachrichten beurteilt werden [192]. Maßgeblich sind dabei die ursprüngliche Gruppengröße (n) sowie die Größe der beiden entstehenden Partitionen (n_1 und n_2)[28]. Dabei wird wieder zwischen einem erfolgreichen und einem fehlergeschlagenen Wiedereintritt unterschieden.

Erfolgreicher *REJOIN* Es wird davon ausgegangen, dass die Nachbarschaftsdetektion eine Minute dauert. Danach können die Partitionen wieder miteinander kommunizieren. Innerhalb dieses Zeitraumes wird im Abstand von jeweils 2 s von den abgespaltenen Peers eine Detektionsnachricht versendet. Daraus ergibt sich die Anzahl aller versandten Nachrichten wie folgt:

$$12n + 27n_1 - 7$$

Abbildung 6.15 stellt die Anzahl der versandten Nachrichten (zwischen 146 und 3710) für die verschiedenen Testfälle und -läufe dar. Es wird deutlich, dass deren Anzahl

[28] Wobei gilt: $n = n_1 + n_2$

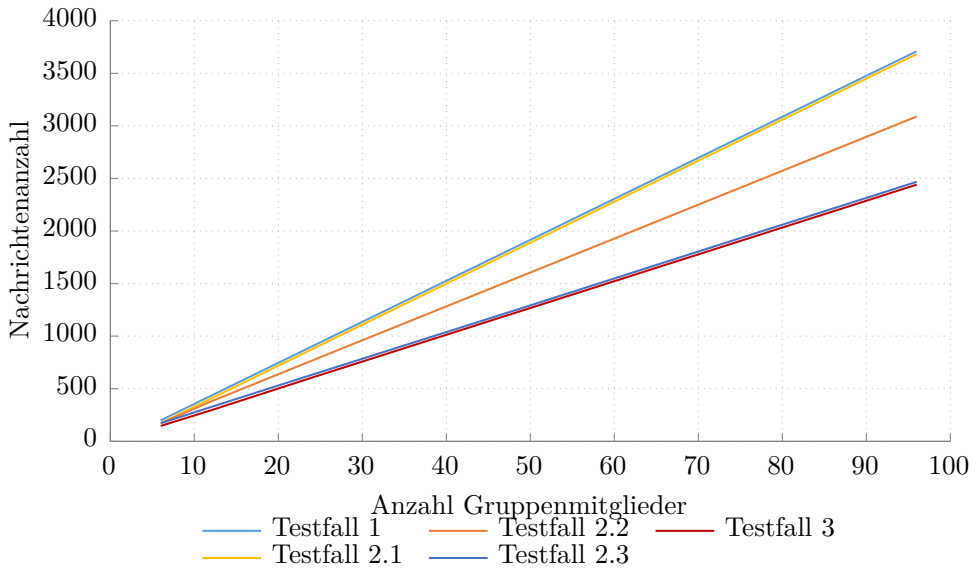


Abbildung 6.15: Anzahl der versendeten Nachrichten mit dem *Moversight*-Ansatz

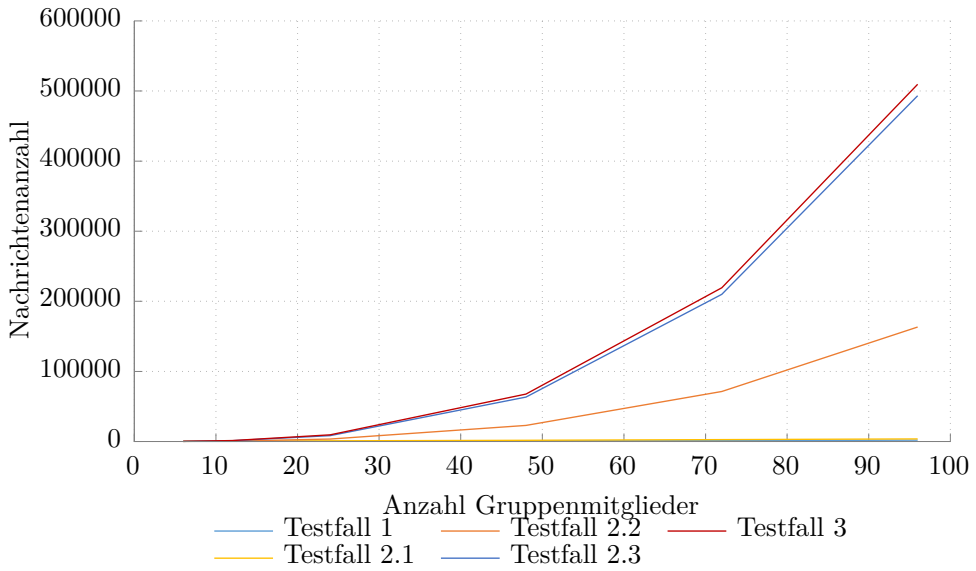


Abbildung 6.16: Anzahl der versendeten Nachrichten mit dem *LEAVE*/*JOIN*-Ansatz

von der Größe der Partition n_1 abhängig ist: Je kleiner die abgetrennte Partition ist, desto mehr Detektionsnachrichten sind je abgetrenntem Peer zu versenden.

Für die Anzahl der versendeten Nachrichten für den *LEAVE*/*JOIN*-Ansatz ergeben sich nach [192], Werte zwischen 101 und 509548 Nachrichten (siehe Abbildung 6.16). Im Testfall (1) und (2.1) werden durch den *LEAVE*/*JOIN*-Ansatz wenige bzw. ähnlich viel Nachrichten wie in *Moversight* benötigt. Somit ist für bis zu zwei nicht erreichbarere Peers der *LEAVE*/*JOIN*-Ansatz hinsichtlich der versendeten Nachrichten effizienter. Die hohe Anzahl von Nachrichten bei den übrigen Testfällen ist durch die zuverlässige, total geordnete Multicastkommunikation der *LEAVE*- und *JOIN*-Operationen begründet. *Moversight* benötigt hier nur vier Multicastnachrichten.

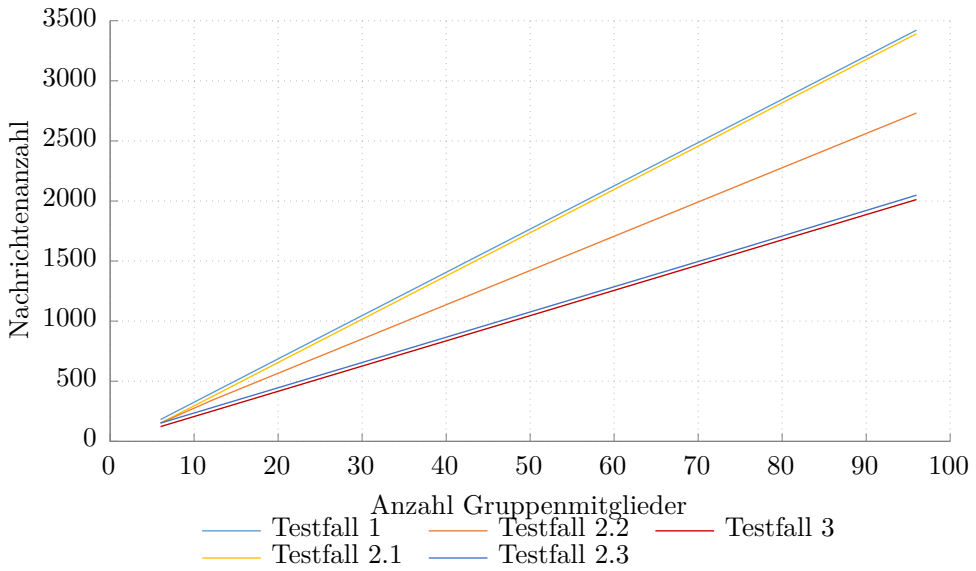


Abbildung 6.17: Anzahl der versendeten Nachrichten mit dem *Moversight*-Ansatz bei einem fehlgeschlagenen Wiedereintritt

Fehlgeschlagener Wiedereintritt Für diesen Fall wurden während der Testläufe zwischen 122 bis 3422 Nachrichten versendet, nach Formel [192]:

$$6n - 30n_1 + 4$$

Der Funktionsverlauf ist in Abbildung 6.17 dargestellt. Beim *LEAVE/JOIN*-Ansatz ist in diesem Szenario nur die Anzahl von Multicastnachrichten für den Gruppenaustritt zu berücksichtigen. Nach [192] ergibt sich diese zwischen 111 und 352527 Nachrichten je Gruppengröße und Testfall, wie Abbildung 6.18 zeigt. Erneut sind die Testfälle (1) und (2.1) des *LEAVE/JOIN*-Ansatzes effizienter als *Moversight*. Die meisten Nachrichten entfallen hier wieder auf die Nachbarschaftsdetektion. Jedoch zeigen die Ergebnisse, dass auch bei einem erfolglosen Wiedereintritt der vorgestellte Ansatz insgesamt effizienter ist als *LEAVE/JOIN*.

Zusammenfassend kann festgestellt werden, dass das *Moversight*-Konzept zur Behandlung der Nutzermobilität eine Partitionierung der Gruppe zuverlässig erkennt und den nahtlosen Wiedereintritt der temporär getrennten Gruppenmitglieder ermöglicht. Es zeigte sich, dass ein Gruppenkommunikationssystem mit integrierten Wiedereintrittsoperation im Vergleich zu anderen Konzepten effizienter mit den Ressourcen mobiler Endgeräte umgeht. Bestehende Gruppenkommunikationssysteme wie ISIS, Transis, GCP oder JGroups bieten keine solche Operation.

Die Leistungsfähigkeit dieser Operation wurde ausgehend von der Definition zweier Testszenarien und verschiedener Testfälle mit unterschiedlichen Gruppengrößen dargestellt. Vor allem in der Erkennung und Behandlung von sekundären Partitionen mit mehr als zwei Mitgliedern ist das vorgestellte Konzept sehr effizient. Beim *LEAVE/JOIN*-Ansatz müssen die wiedereintretenden Peers die Änderungen während ihrer Abwesenheit auf Anwendungsebene abgleichen. Das führt im Allgemeinen zu sogenannten Application-State-Transfers[29], welche ressourcenintensiv sind. Der Aufwand der *RESYNCHRONIZE*-Operation ist zumeist geringer als für einen vollständigen Application-State-Transfer.

[29] Siehe Abschnitt 6.2 bzw. [33]

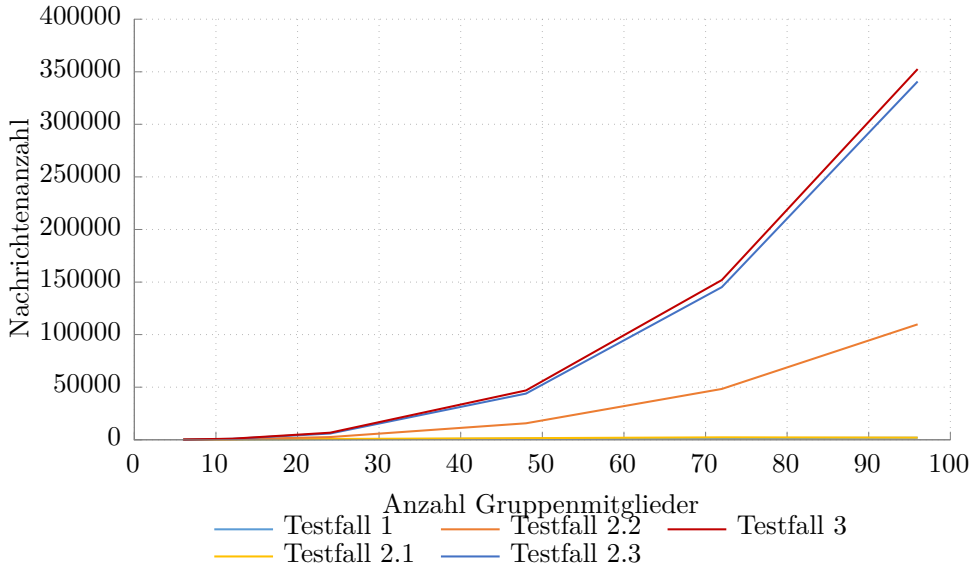


Abbildung 6.18: Anzahl der versendeten Nachrichten mit dem *LEAVE*/*JOIN*-Ansatz bei einem fehlgeschlagenen Wiedereintritt

Kapitel 7

Verteilte Latenzschätzung

Durch die präzise Abschätzung der bei einer Nachrichtenübertragung zwischen zwei Kommunikationspartnern entstehenden Verzögerung kann die verfügbare Kapazität der Kommunikationsverbindung optimal genutzt werden [195]. Solche Abschätzungen werden von einer Reihe von Anwendungen benötigt, um überflüssige Übertragungswiederholungen und lange Warteperioden bei der Fehlerbehandlung zu vermeiden [121]. Gruppenorientierte Ansätze wie SpoVNet [212] oder Zookeeper [116] definieren beim Nachrichtenaustausch für die Übertragungsverzögerung statische Obergrenzen. Je nach Last- und Verkehrssituation können hierdurch die Reaktionsfähigkeit des Systems verzögert bzw. verzögerte Übertragungen fälschlicherweise als fehlerhaft eingestuft werden. Daher gibt es für die Festlegung derartiger Obergrenzen unter den Gesichtspunkten Ressourcennutzung und Reaktionsfähigkeit keine optimale Lösung. Neben der Nachrichtenübertragung wird die Latenz- oder Verzögerungsschätzung auch für die Ausfallerkennung von Kommunikationspartnern bei der Netzfehlerdetektion [47, 50] genutzt. In beiden Fällen wird dazu üblicherweise die Umlaufszeit bzw. *Round Trip Time* (RTT) bestimmt. Dazu wird üblicherweise eine lokale Uhr genutzt. Die dabei eingesetzten Verfahren sind beispielsweise in [50, 121] und [110] beschrieben.

In mobilen kollaborativen Anwendungen werden globale Ereignisse oft von den Peers mittels Timer überwacht. Für synchronisierte Aktionen müssen die Timer auf der gleichen Zeitabschätzung basieren. Ein Beispiel dafür wäre der Eintritt eines neuen Peers in eine kollaborative Gruppe. Um hier das globale Wissen zu sichern, muss jeder Peer lokal der Gruppe einen neuen Peer hinzufügen, wenn die Einladung durch den eingeladenen Peer angenommen wird. Der Beitrittsprozess wird bei jedem Peer von einem Timer überwacht, dessen Laufzeit innerhalb der Gruppe konsistent abzuschätzen ist. Ist der Beitritt bis zum Timeout nicht abgeschlossen, brechen alle Peers den Beitrittsprozess ab und tragen den eingeladenen Peer wieder aus. Die Nutzung der lokalen Zeit wäre in asynchronen verteilten Systemen für eine solche globale Schätzung nicht geeignet. Aufgrund der Drift der lokalen Uhren würde die individuelle Bestimmung der Verzögerung für eine Gruppenübertragung bei n-Gruppenmitgliedern zu ebenso vielen unterschiedlichen Ergebnissen führen [136].

Die Asynchronität zwischen den Teilnehmern kann aufgrund des total geordneten Nachrichtenaustausches und der Sichtorientierung vernachlässigt werden. Deshalb kann z. B. ein Sichtwechsel oder die Auslieferung einer Gruppennachricht statt eines konkreten Zeitpunkts als Auslöser für Aktionen wie einen Rollenwechsel genutzt werden. Dadurch wird die Abhängigkeit der kollaborativen Anwendung von einer präzisen, systemweiten Zeit reduziert. Um beispielsweise für den Gruppeneintritt oder eine Nachrichtenübertragung die benötigten Timer einzustellen, muss das Timeoutintervall lokal bestimmt werden. Dafür nutzen bisherige Latenzschätzverfahren die lokale Uhr.

Die Aufgabe einer globalen Latenzschätzung besteht im Gegensatz dazu darin, die lokale Zeit durch eine globale Gruppenzeit zu ersetzen. Diese wird für jeden TOM-Transfer jeweils durch den primären Master bestimmt und ist der gesamten Gruppe über das globale Wissen bekannt. Auf dieser Basis kann dann die RTT einer Nachricht abgeschätzt werden. Wegen der begrenzten Ressourcen mobiler Endgeräte ist es notwendig, diese Abschätzung ohne zusätzliche Kommunikation lokal vorzunehmen. In

diesem Kapitel wird ein Ansatz für eine verteilte globale Latenzabschätzung in mobilen kollaborativen Anwendungen vorgestellt, der für das Gruppenkommunikationsprotokoll *Moversight* entwickelt wurde [93]. Er ist generell auf alle geschlossenen Gruppenkommunikationssysteme anwendbar, sofern diese ein gemeinsames globales Wissen verwalten und die Nachrichtenübertragung global total geordnet ist (vgl. Abschnitt 5.4).

7.1 Zeitdienst

Die globale Koordination der Timeoutintervalle ist wesentlich für die Sicherung des globalen Wissens. So hängt beispielsweise bei einer TOM-Übertragung der Nachrichtenversand durch den primären Master vom Transferzustand der sekundären Master ab. Eine Nachricht kann erst global bestätigt werden, wenn die Sendevorgänge bei allen sekundären Mastern abgeschlossen sind. Somit beeinflusst die Verzögerung in den Clustern die Gesamtverzögerung. Die Dauer der Nachrichtenverteilung im sekundären Cluster kann durch den primären Master jedoch nicht direkt bestimmt werden. Aus diesem Grund stellt *Moversight* mit dem *Zeitdienst* den lokalen Peers systemweit verschiedene Funktionen für die Schätzung der Nachrichtenverzögerungen zur Verfügung. Es können Nachrichtenverzögerungen zwischen:

1. zwei Peers der Gruppe,
2. zwischen einem Peer und seinem Cluster oder
3. einen Peer und der gesamten Gruppe

bestimmt werden (siehe Abbildung 7.1). Die Abschätzungen (1) und (2) sind mit den verwendeten Verfahren beim *Transmission Control Protocol* (TCP) [121] vergleichbar. Der Zeitdienst enthält für jede Funktion einen Schätzer. Das ist eine Komponente des Gruppenkommunikationssystems, welche auf Basis der gemessenen Verzögerungen von Übertragungen mithilfe eines Algorithmus die wahrscheinliche Verzögerung der nächsten Übertragung abschätzt. Ein Schätzer ist dabei jeweils einer der drei zuvor genannten Arten von Nachrichtenverzögerungen zugeordnet. Die dafür notwendigen Messungen der Nachrichtenverzögerungen stammen aus den einzelnen Transferdiensten. Der eigentlich verwendete Schätzalgorithmus ist über eine Metrik im Schätzer gekapselt und kann über deren Konfiguration angepasst werden.

Mittels des *Peer-RTT-Schätzers* wird die Übertragungsverzögerung zwischen direkten Kommunikationspartnern auf Basis der gemessenen RTT einer zuverlässigen Übertragung des Basis-Transfers bestimmt. Slaves schätzen also die Verzögerung einer Übertragung zu ihren jeweiligen Mastern, diese wiederum ermitteln die Verzögerungen von Übertragungen zu ihren Slaves und den übrigen Mastern.

Über den *Cluster-RTT-Schätzer* wird die Dauer einer Nachrichtenverteilung durch den Master basierend auf einer TOM-Übertragung innerhalb seines Clusters ermittelt. Der zu bestimmende Zeitraum beginnt mit dem Versenden einer Nachricht an den ersten Slave des Clusters durch den Master. Das Ende wird markiert durch das Eintreffen der letzten LT-Nachricht von einem Slaves dieses Clusters. Wegen dieser Abhängigkeit wird diese Schätzung nur von einem Master durchgeführt.

Der *Gruppen-RTT-Schätzer* bestimmt die vermutete Verzögerung für eine Gruppennachricht. Die konkrete Ermittlung der Verzögerung wird nachfolgend im Abschnitt 7.2 vorgestellt. Vereinfacht kann man die Verzögerung einer Gruppennachricht verstehen als den Zeitraum zwischen dem Versenden der Nachricht durch den primären Master und dem Eintreffen der letzten LT-Nachricht aus der Gruppe. Die Verteilung der GT-Nachricht innerhalb der Gruppe ist somit nicht Teil der Abschätzung. Die Verzögerung, auf der diese Schätzung basiert, wird durch den TOM-Transfer bestimmt. Die gemessene Gruppen-RTT wird über die GT-Nachricht an alle Peers der Gruppe verteilt. Damit kumuliert der Gruppen-RTT-Schätzer die Verzögerungen aller Gruppennachrichten in einer globalen Schätzung. Allen Schätzern gemein sind folgende Rahmenbedingungen:

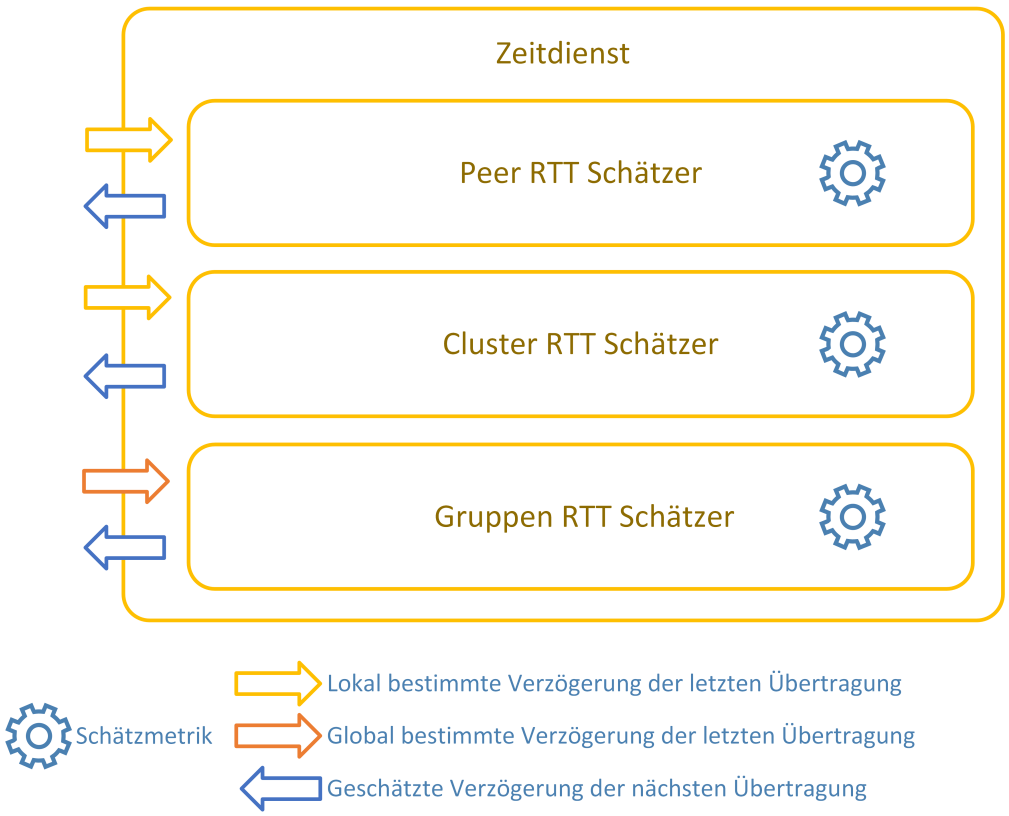


Abbildung 7.1: Aufbau des Zeitdienstes

Ignorieren von Wiederholungen Ist es notwendig, dass für eine Nachrichtenübertragung der Transfer der Gruppennachricht wiederholt wird (nicht deren zugehörige LT-Nachricht), so wird diese Übertragung für die Schätzung nicht berücksichtigt, da sonst das Schätzergebnis verfälscht wird [121].

Gleitendes Schätzfenster Um eine zeitnahe Anpassung der Schätzung an sich ändernde RTT-Werte sicherzustellen, werden jeweils nur die letzten n-Übertragungen berücksichtigt. Die konfigurierbare (Übertragungs-)Fenstergröße liegt für mobile kollaborative Anwendungen üblicherweise zwischen 100 und 1000 Werten [93].

Konfiguration der Schätzer Prinzipiell können die Schätzer unabhängig voneinander konfiguriert werden. Neben der bereits erwähnten Fenstergröße ist für jeden Schätzer ein Standardschätzwert anzugeben, der verwendet wird, wenn keine oder nicht genügend RTT-Werte vorrätig sind, um eine Schätzung abzugeben [93].

Kontinuierliche Rücksetzung Bei jedem Sichtwechsel kann eine Neuordnung der Topologie vorgenommen werden (vgl. dazu Kapitel 8). Ebenso kann sich der Aufbau der Sichten deutlich unterscheiden. Daher ist nicht gesichert, dass die Schätzung über einen Sichtwechsel stabil und richtig bleibt. Folglich ist über die Konfiguration zu definieren, ob das Übertragungsfenster bei jedem Sichtwechsel zurückzusetzen ist oder die in der alten Sicht gesammelten RTT-Werte weiter genutzt werden.

7.2 Latenzmodell

Das für die Latenzabschätzung verwendete Modell ist eine Weiterentwicklung des Single-Bottelneck-Modells von Gunawardena *et al.* [106]. Darin werden die unterschiedlichen Übertragungskapazitäten der Einzelverbindungen vernachlässigt. Stattdessen wird davon ausgegangen, dass die Verzögerung in erster Linie durch die Warteschlangen – die

Bottelnecks – der Zwischensysteme erzeugt wird. Weiter wird angenommen, dass neben der Ausbreitungsverzögerung nur der „*größte Engpass*“ die Gesamtverzögerung für eine Verbindung maßgeblich bestimmt. Für das hier verwendete Latenzmodell wird diese Annahme auch für Verbindungen zwischen Gruppenteilnehmern außerhalb eines lokalen Netzes getroffen. Ferner werden die Engpässe der einzelnen Teilstrecken dieser Verbindung zusammen betrachtet. Darauf aufbauend kann die Nachrichtenverteilung innerhalb der Gruppe modelliert werden, die aus einer Reihe von Einzelübertragungen besteht. Die Dauer einer bestätigten Übertragung zwischen einem Sender und einem Empfänger wird als RTT^P bezeichnet. Wenn die Quelle zum Zeitpunkt t_i eine Nachricht m_i sendet, so trifft die Bestätigung der erfolgreichen Übertragung beim Sender zum Zeitpunkt t_j ein. Formel 7.1 zeigt die daraus folgende Definition für RTT^P.

$$RTT^P = t_j - t_i \tag{7.1}$$

Da das globale Wissen nicht für die Bestimmung von RTT^P benötigt wird, kann ein beliebiger Latenzschätzungsalgorithmus verwendet werden. Für Gruppennachrichten ist dieser Ansatz jedoch nicht anwendbar. Die benötigte minimale Zeit RTT^G für die Zustellung einer einzelnen bestätigten Nachricht an die Gruppe (entspricht einer Gruppen-RTT) ist unter Verwendung einer Übertragung mit der MOVS-Semantik durch den TOM-Transfer über Formel 7.2 definiert.

$$\begin{aligned} RTT^G_{min} = t(s,m) + \sum_{i=1}^{n} t(m, om_i) \\ + max(\forall\ m \in G \sum_{i=1}^{n} t(m, s_i)) \end{aligned} \tag{7.2}$$

Dabei ist $t(s,m)$ die maximale Übertragungsverzögerung zwischen einer Quelle und deren Master (vergleiche: primäre Master, Abschnitt 5.2.2). Ist der Sender der Nachricht selbst Master, so gilt $t(s,m) = 0$. Die Summe aller $t(m, om_i)$ ergibt die Übertragungsverzögerung zwischen dem Master (m) und jedem anderen Master der Gruppe (om_i). Der letzte Term entspricht der maximalen Verzögerung bei der Übertragung einer Nachricht vom Master an dessen Slaves.

Dezentral kann RTT^G_{min} ohne Rückkopplung nicht durch den Sender der Nachrichten bestimmt werden. Deshalb ist die Definition von RTT^G_{min} anzupassen. Da der primäre Master für die erfolgreiche Übertragung der Gruppennachricht verantwortlich ist und der sendende Slave im weiteren Übertragungsverlauf nur als Empfänger agiert, kann die Sendeverzögerung für die Übermittlung der Nachricht vom sendenden Slave zum primären Master ignoriert werden (vgl. Abschnitt 5.2.2). Vereinfachend wird hier angenommen, dass eine Nachricht immer vom primären Master versendet wird. Der aus dieser Annahme resultierende Fehler ist in der Gesamtbetrachtung gering und wird über später eingeführte Sicherheitsabstände bei der Schätzung korrigiert. Folglich muss nur die Verteilung der Nachricht innerhalb der Gruppe betrachtet werden, woraus sich $RTT^G_{min,redef}$ (gem. Formel 7.3) ergibt.

$$\begin{aligned} RTT^G_{min,redef} = \sum_{i=1}^{n} t(m, om_i) \\ + max(\forall\ m \in G \sum_{i=1}^{n} t(m, s_i)) \end{aligned} \tag{7.3}$$

Gunawardena *et al.* [106] haben mit Hilfe des Single-Bottleneck-Modells nachgewiesen, dass die Verzögerungen bzw. Umlaufzeiten in drahtlosen Netzen einer Gammaverteilung folgen. Ihre Ergebnisse zeigen, dass mit wachsendem Hintergrundverkehr die Gammaverteilungsparameter α und θ (Lage (engl. *position*) und Form (engl. *shape*)) an Bedeutung gewinnen. Aus Sendersicht ist die Ursache der Paketverluste unerheblich. Nach Gunawardena *et al.* können aber sowohl eine geringe Verbindungsqualität als auch eine Netzüberlastung Neuübertragungen auslösen. Entsprechend folgern sie, dass Verzögerungen, welche durch:

- überlastete Netzkomponenten,
- schlechte Verbindungsqualität oder
- sonstige Nachrichtenverluste

entstehen, als eine einzelne Verzögerung (engl. *Single-Bottleneck*) zu modellieren sind.

7.3 Latenzschätzung

Im Jahre 1988 veröffentlichte van Jacobson eine der wichtigsten Methoden zur Latenzschätzung, welche in die Version 4BSD des TCP-Protokolls einging [121]. Dieser Ansatz nutzt die lokale Zeit zur RTT-Schätzung, indem für jede Nachrichtenübertragung des aktuellen TCP-Übertragungsfensters die Round Trip Time gemessen wird. Über Formel 7.4 wird die Varianz der Messungen bestimmt. Dabei bezeichnet:

- var die Varianz der RTT-Reihe,
- $srtt$ die geglättete RTT und
- β einen Gewichtsfaktor.

Die Variable at entspricht dem Zeitpunkt des Nachrichtenempfangs (engl. *arrival time*). Die geglättete RTT ist in Formel 7.5 nach [121, 171, 180] definiert. Wie β ist α ein Gewichtungsfaktor, über den der Einfluss der Einzelschätzung auf das Gesamtergebnis gesteuert wird, wobei typischerweise $\alpha = 0,125$ und $\beta = 0,25$ ist[1].

$$var = (1 - \beta) * var + \beta * |srtt - at| \tag{7.4}$$

$$srtt = (1 - \alpha) * srtt + \alpha * at \tag{7.5}$$

Basierend auf der geglätteten RTT schätzt van Jacobson die Latenz der nächsten Übertragung über Formel 7.6, wobei:

- G die Granularität[2] und
- K wiederrum ein Gewichtungsfaktor ist[3].

Darauf aufbauend wurden weitere Ansätze für die Latenzschätzung entwickelt.

$$timeout = srtt + max(G, K * var) \tag{7.6}$$

So schlugen Chen *et al.* [50] ein Verfahren vor, welches auf einer wahrscheinlichkeitstheoretischen Analyse des Netzverkehrs basiert. Es nutzt zur Abschätzung der RTT das arithmetische Mittel eines Übertragungsfensters (üblicherweise 100 bis 1000 Nachrichten) sowie einen Sicherheitsabstand. Dieser dient der Stabilisierung der Schätzung und kann als Sicherheitspuffer für den Ausgleich von Schwankungen der Verzögerungen verstanden werden. Ohne diesen würden kleinste RTT-Änderungen potentiell zu einer Neuübertragung oder im schlimmsten Fall zum Abbruch des Sendevorgangs führen. Der Sicherheitsabstand bewirkt, dass nach Ablauf der geschätzten RTT das Übertragungsprotokoll noch eine gewisse Zeit wartet, bis eine Fehlerbehandlung eingeleitet wird. Demnach ergibt sich die geschätzte Verzögerung der nächsten Übertragung rtt' durch Formel 7.7.

$$rtt' = rtt + mean + \alpha \tag{7.7}$$

Dabei sind:

- rtt die Round-Trip-Time der letzten Nachricht,

[1] Detaillierte Erläuterungen zur Wahl von α und β sind zu finden bei van Jacobson [121] und im RFC 6298 [171].

[2] im Sinne von Genauigkeit, definiert als $10e - 6$

[3] definiert als 4

- *mean* das arithmetische Mittel des aktuellen Übertragungsfensters und
- α ein konstanter Sicherheitsabstand.

Der feste Sicherheitsabstand ist der größte Nachteil am Verfahren nach Chen. Er wird nur einmalig bestimmt und nicht dynamisch an Änderungen der Netzsituation angepasst. In mobilen Umgebungen adaptiert sich rtt' nur sehr langsam an die aktuelle Netzsituation, was auf die üblicherweise verwendete Übertragungsfenstergröße von 1000 Nachrichten zurückzuführen ist.

Im Kontext der adaptiven Fehlerdetektoren wurde von Bertier *et al.* [28] ein Schätzverfahren vorgeschlagen, welches ebenfalls auf dem Prinzip von van Jacobson basiert. Hier wird auf Basis der geschätzten RTT ein adaptierbarer Sicherheitsabstand berechnet und der allgemeinen Latenzschätzung hinzugefügt. Die Abhängigkeit des Bertier-Algorithmus von der lokalen Uhr kann durch eine „*global bestimmte*“ Umlaufzeit ersetzt werden, die Teil des globalen Wissens ist. Untersuchungen haben jedoch gezeigt, dass die Bertier-Schätzung für Anwendungsszenarien, in denen die Latenz nicht der Normalverteilung unterliegt, nicht geeignet ist. Wie oben ausgeführt, entspricht die Latenz in drahtlosen Netzen einer Gammaverteilung. Die Auswirkungen nicht normalverteilter Eingangswerte auf den Bertier-Schätzer werden in Abschnitt 7.5 dargestellt. Es kommt bei diesem Verfahren zu zahlreichen Unter- und Überschätzungen der RTT, wodurch jede Schätzung fehlerbehaftet sein kann.

Hayashibara *et al.* [110] schlugen den φ-Fehlerdetektor vor. Statt von einer binären Schätzung über den Zustand eines Peers auszugehen (im Sinne von: auf dessen Fehlerfreiheit vertrauen/diese zu vermuten oder nicht), gibt dieser Fehlerdetektor einen Wert auf einer kontinuierlichen Skala aus. Der φ-Detektor passt die Ergebnisskala dynamisch an die sich ändernden Netzbedingungen an. Das Ergebnis kann als variabler Sicherheitsabstand mit dem arithmetischen Mittel der RTT des Übertragungsfensters zu einer Latenzschätzung für die nächste zu übertragende Nachricht kombiniert werden. Da der φ-Fehlerdetektor nur für normalverteilte Eingabewerte definiert ist, kann er zur Latenzschätzung in mobilen Umgebungen nicht genutzt werden.

In der Literatur sind nur wenige Ansätze für eine Latenzschätzung in reinen P2P-Anwendungen zu finden. Einige Ansätze (z. B. [46, 109, 125]) modellieren die Latenz als Abstandsfunktion innerhalb eines virtuellen Koordinatensystems [65]. Diese Ansätze sind jedoch sehr komplex und zumeist auch eher auf drahtlose Sensornetze bezogen. Der Aufwand für diese Ansätze steigt mit der Anzahl der Peers, da die Koordinaten als n Tupel modelliert sind, wobei n die Anzahl der Gruppenmitglieder ist. Diese Ansätze haben eine mittlere Fehlerrate von ca. 25 %[109]. Damit weicht in einem Viertel aller Schätzungen die virtuelle Entfernung deutlich von der Entfernung im realen Netz ab. Gleiches gilt damit auch für die geschätzte Latenz. Diese Ansätze sind aufgrund ihrer Komplexität und der damit einhergehenden Fehlerquote kaum für eine Nutzung im Rahmen von mobilen kollaborativen Anwendungen geeignet. Deshalb wird für die Latenzabschätzung in mobile kollaborative Anwendungen ein deutlich einfacherer Ansatz gewählt, welcher auf der gemessenen Umlaufzeit basiert.

7.4 Ein verteilter Latenzschätzungsansatz

Der Grundgedanke des Ansatzes besteht darin, die Abhängigkeit der Schätzung von einer lokalen Uhr wie in den zuvor diskutierten Ansätzen bei gleichzeitiger Wahrung der Funktionalität für Normal-, Gamma- und Exponentialverteilungen zu beseitigen. Der verwendete Schätzalgorithmus wird in einen eigenständigen Funktionsbaustein gekapselt – den Schätzer – der die gemessene Übertragungsverzögerung und Konfigurationsparameter als Eingabe erhält und als Ausgabe/ als Ergebnis eine Schätzung für die Verzögerung der nächsten Nachrichtenübertragung liefert.

Die Abhängigkeit von der lokalen Uhr entsteht durch die Notwendigkeit, für jede Nachricht die Ankunftszeit und die Round Trip Time zu bestimmen. In verteilten asynchronen Systemen weichen jedoch die Uhren stets voneinander ab (engl. *drift*). Lokale RTT-Messungen führen damit zu individuellen RTT-Werten innerhalb der Gruppe.

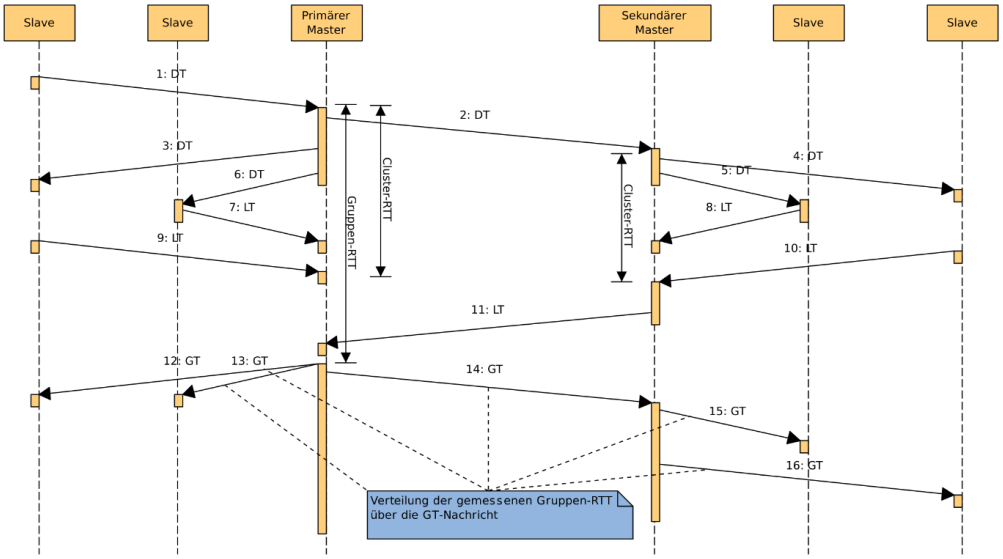


Abbildung 7.2: Darstellung des Schätzprinzips

Dadurch schätzt jeder Peer die Latenz verschieden ein, was zu unterschiedlichen Reaktionen führen kann und die Gruppensynchronität aufhebt. Für die Latenzabschätzung muss daher die lokale Uhr durch die Gruppen-RTT RTT^G, d. h. durch die minimale Zeit für die bestätigte Zustellung einer Nachricht an die Gruppe aus dem globalen Wissen ersetzt werden.

Wie im Abschnitt 5.4 dargestellt, ist ein Master in *Moversight* für die Bestätigung der Nachricht an die Gruppe verantwortlich. Die Bestätigung enthält die virtuelle logische Zeit, welche für diese Nachricht als Empfangszeit bestimmt wurde. Im hier vorgestellten Ansatz ermittelt der für die Nachrichtenverteilung verantwortliche Master neben der virtuellen Empfangszeit lokal auch die RTT^G. Diese ist nach Formel 7.3 definiert als die Zeitspanne zwischen der Verteilung der Nachricht an die Gruppe und dem Empfang der Empfangsbestätigungen von allen Mastern und Slaves des lokalen Clusters. Die Verzögerung, welche durch die Verteilung der Bestätigung innerhalb der Gruppe entsteht, kann mangels Rückkopplung nicht abgeschätzt werden. Der dadurch entstehende Schätzfehler wird durch einen erhöhten Sicherheitsabstand korrigiert, welcher ebenso in die Latenzschätzung einfließt [28].

Die Gruppen-RTT wird über die GT-Nachricht für diese Nachricht in der Gruppe verteilt (vgl. Kapitel 5.2). Die Bestätigungsnachricht gibt auch an, ob ein Master die zu übertragende Nachricht aufgrund von Paketverlusten o. ä. wiederholt an den Empfänger übertragen musste. In diesem Fall wird der RTT^G-Wert durch den Schätzer nicht berücksichtigt, da die (erhöhten) Werte die Schätzung verfälschen würden.

Abbildung 7.2 stellt dieses Schätzprinzip mit Hilfe einer Gruppe von sechs Peers, verwaltet in zwei Clustern, für die Übertragung einer Gruppennachricht dar. Die initiale Übertragung (1:DT) vom Slave an den primären Master wird ignoriert und RTT^G für den Zeitraum der Übertragungen 2:DT bis 11:LT ermittelt. Im Anschluss wird RTT^G über die GT-Nachricht in der Gruppe verteilt. Beim Empfang dieser Nachricht durch die einzelnen Peers wird der RTT^G der GT-Nachricht entnommen und dem lokalen Schätzer übergeben, der unter Verwendung der im folgenden Abschnitt dargestellten Algorithmen die Latenz der nächsten Gruppennachricht ermittelt.

Die Bestimmung der Gruppen-RTT nach Formel 7.8 leitet sich aus dem Nachrichtenverteilungsprinzip von *Moversight* und der Formel 7.3 ab. Dabei ist der Ausdruck $\forall m \in G: \ t(om, m)$ definiert über Gleichung 7.9.

$$RTT^G = max|t(m, om)| + max|t(om, m)| \tag{7.8}$$

$$\begin{aligned} t(om, m) = max|\forall s \in C: \ max|t(m, s)|| \\ + max|\forall s \in C: \ max|t(s, m)|| \end{aligned} \tag{7.9}$$

Der verantwortliche Master (m) sendet die Nachricht an alle anderen Master (om). Der Term $max|t(m, om)|$ definiert die dabei entstehenden Verzögerung. Anschließend sammelt der verantwortliche Master die Bestätigungen jedes Clusters. Diese beinhalten die logische Zustellzeit für die einzelnen Cluster. Daraus wird die globale logische Zustellzeit bestimmt. Jeder andere Master om bestätigt die Nachricht für den Cluster C, nachdem er sie erfolgreich in seinem Clusters verteilt hat. Daher beschriebt $t(om, m)$ die Verzögerung für die übrigen Master, welche dadurch entsteht, dass sie die Nachricht innerhalb ihres Clusters an die einzelnen Slaves s verteilen und auf deren Reaktion warten. Entsprechend der Definition von RTT^G kann die Latenz für die nächste Nachricht RTT^G_{i+1} auf Basis der vorrausgegangenen w Nachrichten abgeschätzt werden (dem Übertragungsfenster) $RTT^G_i, \ldots, RTT^G_{i-w-1}$ über die Formel 7.10.

$$RTT^G_{i+1} = sa(RTT^G_i, RTT^G_{i-1}, \ldots, RTT^G_{i-w-1}) + \alpha \tag{7.10}$$

Der Sicherheitsabstand α und die geglättete Durchschnittsfunktion sa werden wie folgt bestimmt: Nach Gunawardena *et al.* [106] und Xu *et al.* [223] sind $mean$, $median$, und $max(mean, median)$ die drei am häufigsten genutzten Glättungsfunktionen. Ebenso verbreitet sind Schätzungen unter Nutzung von Glättungsfunktionen ersten und zweiten Grades. In [93] wurde die Schätzgenauigkeit für diese fünf Schätzer untersucht. Die drei genauesten Schätzer werden nachfolgend vorgestellt.

Mean-Schätzer. Der Mean-Schätzer (engl. *Mean Estimator*) nutzt zur Abschätzung der Latenz das arithmetische Mittel ($mean$) der Verzögerungen der letzten Übertragungen. Seine Funktionsweise ist im Algorithmus 1 dargestellt.

Algorithmus 1 Mean-Schätzer

$window \leftarrow window \cup RTT^{G,m}_i$
$error \leftarrow RTT^{G,m}_i - ea$
$ea \leftarrow mean(window)$
$\sigma \leftarrow variance(window)$
$wd \leftarrow \phi * \sigma$
if $error > 0$ **then**
  $\alpha \leftarrow wd + error$
else if $error = 0$ **then**
  $\alpha \leftarrow wd$
else
  $r \leftarrow \gamma * error$
  if $r < wd$ **then**
    $\alpha = wd - r$
  else
    $\alpha = 0$
  end if
end if
$RTT^{G,e}_{i+1} \leftarrow ea + \alpha$

Um $RTT^{G,e}_{i+1}$ abzuschätzen, wird die gemessene Round Trip Time der zuletzt empfangen Nachricht $RTT^{G,m}_i$ zum Übertragungsfenster ($window$) hinzugefügt und der Fehler der letzten Schätzung ($error$) bestimmt. Das Übertragungsfenster speichert die gemessenen RTT-Werte für die n letzten Übertragungen.

Für die Abschätzung der Latenz (ea) wird zunächst das arithmetische Mittel ($mean$) des Übertragungsfensters ($window$) bestimmt. Im Anschluss wird der Sicherheitsabstand α über die Standardabweichung ($sigma$) des Übertragungsfensters ermittelt. Um die gewichtete Standardabweichung (engl. *weighted standard deviation*) (wd) zu berechnen, wird σ über den Parameter ϕ angepasst.

- Ein Schätzfehler größer als Null bedeutet, dass die Schätzung von $RTT^{G,e}_i$ zu gering gewesen ist. In diesem Fall ergibt sich α aus der Summe von $error$ und wd.

- War die Schätzung korrekt, dann ist α genau wd.
- Alle anderen Fälle sind Überschätzungen. Um zu aggressive Anpassungen zu vermeiden (im Sinne von negativen Schätzungen), wird in diesem Fall die Reduktion r berechnet. Diese ergibt sich, indem der Schätzfehler über den Wert γ angepasst wird.
- Ist r geringer als wd, so wird α bestimmt über $wd - r$.
- Anderenfalls wird α auf Null gesetzt.

Die neue Schätzung $RTT_{i+1}^{G,e}$ wird bestimmt durch ea plus α.

Median-Schätzer. Der Median-Schätzer ähnelt dem vorrangegangenen. Der Unterschied zwischen beiden besteht in der Nutzung des Medians[4] (engl. *median*) statt des arithmetischen Mittels, um den Wert von ea und sowie den Wert σ des Übertragungsfensters zu bestimmen.

Max-Schätzer. Der Max-Schätzer ist eine Weiterentwicklung der beiden vorangegangen Schätzer. Hier werden zunächst sowohl das arithmetische Mittel als auch der Median des Übertragungsfensters bestimmt. Im Anschluss wird der größere der beiden Werte als Schätzung für ea gewählt. Das Maximum beider Werte wird ebenso für die Bestimmung von σ des Übertragungsfensters genutzt.

7.5 Vergleich der Algorithmen

Um die Leistungsfähigkeit der vorgeschlagenen Algorithmen zu prüfen, wurden diese in [93] untereinander sowie mit dem Algorithmus von Bertier *et al.* [28] verglichen. Letzterer wurde zum Vergleich herangezogen, da er in der einschlägigen Literatur häufig genutzt wird. Für die Evaluation wurde seine Abhängigkeit von der lokalen Uhr durch entsprechende Anpassungen eliminiert. Die Evaluation wird im Einzelnen in [93] beschrieben. Die vorliegende Arbeit konzentriert sich auf die Darstellung der Evaluationsmethodik und deren Ergebnisse.

Wie ausgeführt, folgen die Latenzen in drahtlosen Netzen der Gammaverteilung. Bei der Evaluation wurden drei Kurven dieser Verteilung genutzt: I, II und III genannt. Die für die Kurven verwendeten Gestalt- und Skalierungsparameter α und θ sind in Tabelle 7.1 zusammengefasst. Diese Kurven wurden wiederum mit drei Arten von simulierten Gruppen evaluiert:

a) mit einer Gruppe mit vorrangig stationären Teilnehmern,

b) mit einer Gruppe mit einer etwa gleichen Anzahl an mobilen und stationären Teilnehmern und

c) mit einer Gruppe aus überwiegend mobilen Teilnehmern.

Die Verzögerung bei der Übertragung von Gruppennachrichten nimmt von Gruppe a nach c zu, begründet durch den steigenden Anteil mobiler Teilnehmer. Um dies in der Evaluation abzubilden, wurden die Kurven I, II und III über eine Bandfilterung angepasst. Die genutzten Filter zeigt Tabelle 7.1. Sie werden ebenfalls a, b und c genannt. Die untere Schranke eines Filters wird mit f_L bezeichnet, die obere Schranke entsprechend f_H.

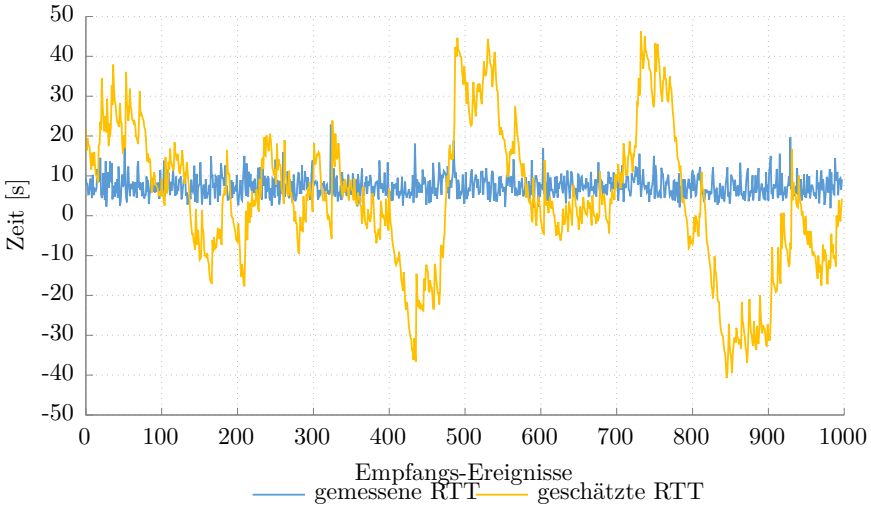
Das Kreuzprodukt der Kurven I bis III und der Filter a bis c wurden als Testmengen $Ia, Ib, \ldots IIIc$ in der Evaluation verwendet. Aus den Testmengen wurden jeweils 1000 Elemente als Eingangsvariablen für die Schätzer zufällig gewählt. Jedes Element repräsentiert dabei eine gemessene Verzögerung (RTT) für eine Nachrichtenübertragung.

In [93] wurden die Kurven Ia bis $IIIc$ mit einem Übertragungsfenster der Größe 10 und 100 jeweils für die Schätzungsparameter γ und ϕ gleich $1,0$ evaluiert. Für

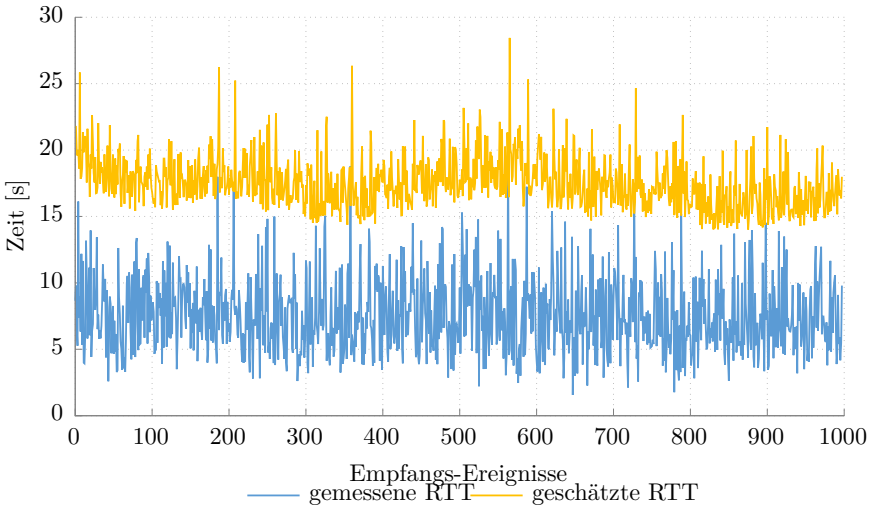
[4] Im Sinne von Zentralwert einer Menge, wenn diese der Größe nach geordnet wird. Beispielsweise für die Menge $10, 2, 4, 2, 1$ ist der Median 2.

Tabelle 7.1: Für die Simulation genutzte Gammaverteilung und Bandpassfilterparameter

γ-Verteilung	α	θ	**Filter**	f_L	f_h
I	1.0	2.0	a	0	2
II	9.0	0.5	b	0	10
III	7.5	1.0	c	0	100



(a) Ergebnisse der Bertier-Schätzung



(b) Ergebnisse der Max-Schätzung

Abbildung 7.3: Evaluationsergebnisse der Kurve $IIIc$ mit $window\ size = 100$

den Parameter β der Bertier-Schätzung wurde ebenfalls der Wert eins verwendet. Die Schätzgüte der Algorithmen wurde über den Mean-Absolut-Error (MAE) und die Standardabweichung des Schätzfehlers (σ_e) ermittelt.

Abbildung 7.3a zeigt die Resultate der Bertier-Schätzung für die Kurve $IIIc$ im Detail. Unabhängig von der reinen statistischen Analyse sind deren Ergebnisse jedoch wie oben ausgeführt z. B. für die Timereinstellung oder die maximale Bandbreitennutzung in mobilen kollaborativen Anwendungen nicht nutzbar. Besser geeignet dafür ist der Max-Schätzer, dessen Verhalten in Abbildung 7.3b beispielhaft dargestellt ist. Er besitzt

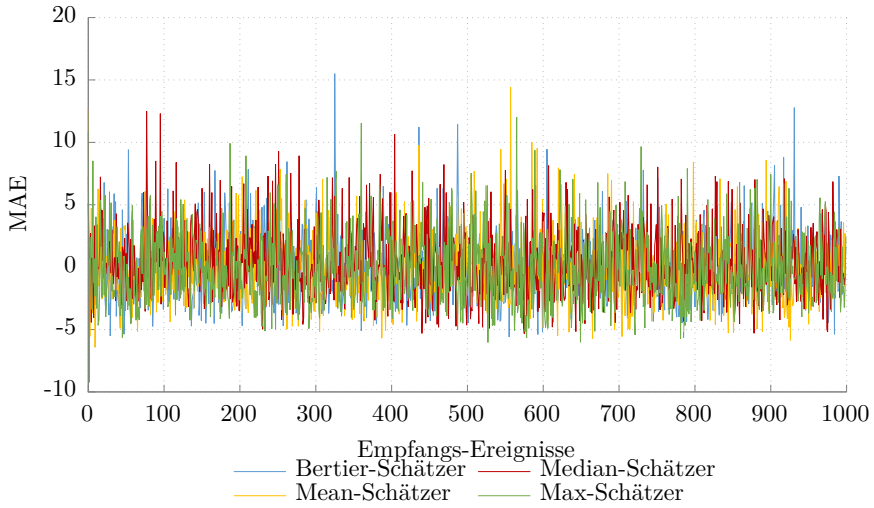


Abbildung 7.4: Vergleich der MAE-Werte für Kurve $IIIc$

eine stabile untere Schätzschranke.

Abbildung 7.4 vergleicht die Mean-Absolut-Error-Werte der drei hier vorgestellten Schätzer für Kurve $IIIc$ bei einem Übertragungsfenster von 100. Bezüglich dieses Wertes haben die verglichenen Algorithmen vergleichbare Eigenschaften. Der Unterschied im Verlauf der Kurven aus Abbildung 7.3a und 7.4 ist der Grund für die guten MAE-Ergebnisse der Bertier-Schätzung: die Unter- und Überschätzungen kompensieren einander.

Aus den Ergebnissen lässt sich nicht eindeutig ableiten, welcher der beste Algorithmus zur Schätzung der Latenz ist. Dennoch zeigen die Ergebnisse in [93] einen klaren Trend:

- Für die Kurven $IIIb$ und $IIIc$ hat der Bertier-Schätzer bei einem Übertragungsfenster von 10 statistisch die besten Eigenschaften, was aber für reale kollaborative Anwendungen zu gering ist.
- In fünf von achtzehn Testfällen zeigten der Max-, der Median- und der Mean-Schätzer die besten Ergebnisse.
- Der Max-Schätzer zeigt jeweils das zweitbeste Ergebnis, wenn der Mean- bzw. der Median-Schätzer das beste Ergebnis erzielen. Da der Max-Schätzer diese beiden Schätztypen nutzt, war dies zu erwarten.

Folglich ist die Schätzgüte des Max-Schätzers im Durchschnitt die stabilste der vorgestellten Schätzer, da seine Eigenschaften in den einzelnen Testfällen immer die besten Ergebnisse aufwiesen oder wenigstens nahe bei den besten Ergebnissen lagen. Aus diesem Grunde bietet es sich an, diesen Schätzer zur Bestimmung der Timerintervalle in mobilen kollaborativen Anwendungen zu verwenden.

Kapitel 8

Dynamische Optimierung der Clustertopologie

Die durch eine mobile kollaborative Anwendung gebildete Gruppe ist während ihrer Existenz selten statisch. Durch Änderungen auf verschiedene Protokollebenen wandelt sich ihre Struktur kontinuierlich. Die dadurch entstehenden Gruppentopologien sind nicht immer optimal und müssen deshalb regelmäßig angepasst werden. Diese Aufgabe übernimmt der Wartungsdienst von *Moversight*.

Auf Anwendungsebene ändern Gruppenein- und -austritte sowie Veränderungen im Kommunikationsverhalten (i.S.v. Sendefrequenz, Quellen, Senken) fortlaufend die Gruppenzusammensetzung. Auf der Netz- und der Verbindungsebene können durch Effekte wie Churns und eine fehlende Netzabdeckung die bestehenden Kommunikationsverbindungen unterbrochen werden. Die begrenzten Ressourcen und Fähigkeiten in den Endgeräten der Teilnehmer, vor allem bei der Energieversorgung, können Kommunikationsverbindungen unterbrechen bzw. ihre Servicequalität reduzieren. Bestehende Kommunikationsverbindungen werden ebenso durch Veränderungen der Netzeigenschaften, wie Bandbreite, Latenz, Paket- und Bitfehlerrate kontinuierlich beeinflusst. Dadurch erhöht sich die Wahrscheinlichkeit von Kommunikationsfehlern in der Gruppe.

Moversight nutzt eine vollständig vermaschte Clustertopologie für die Teilnehmerverwaltung. Innerhalb eines Clusters übernimmt ein Peer entweder die Rolle des Masters oder eines Slaves. Die bestmögliche Clustertopologie ist charakterisiert durch eine optimale Anzahl von Peers pro Cluster, eine homogene Verteilung der Peers auf die einzelnen Cluster bzgl. ihrer Ressourcen und Mobilitätseigenschaften sowie einen leistungsfähigen Master. Als Folge der sich ändernden Eigenschaften der Kommunikationsverbindungen und durch Gruppenein- und -austritte kann die Clustertopologie destabilisiert und die Qualität der Zusammenarbeit beeinträchtigt werden. Daher ist es notwendig, vorbeugende Wartungsmaßnahmen zu treffen und kontinuierlich die Peer-to-Peer-Struktur an die sich ändernden Bedingungen anzupassen. Die dazu zu ergreifenden Massnahmen werden im Wartungsdienst von *Moversight* zusammengefasst, dessen Grundzüge nachfolgend umrissen werden. Eine ausführliche Beschreibung dieses Dienstes, insbesondere seine Struktur, seine Funktionen, sein Leistungsverhalten sowie dabei auftretende Synchronisationsprobleme ist in [203] und [95] zu finden.

8.1 Anforderungen und Rahmenbedingungen für einen Wartungsdienst

Im Bereich der Kommunikationsprotokolle ist der Begriff der Wartung semantisch mit dem Netzmanagement entsprechend des OSI/ISO Netzmanagementmodells verknüpft [119, 120]. Dieses hat sich jedoch in der Praxis nicht durchgesetzt. Auch ist im Kontext der mobilen kollaborativen Anwendungen eine Wartung der Zwischensysteme oder Kommunikationsverbindungen häufig nicht möglich [24, 128]. Die Realisierung eines Wartungsdienstes in Anlehnung an das OSI/ISO Netzmanagement wäre aufgrund der Komplexität in mobilen Endgeräten nicht möglich [27]. Darüber hinaus muss der

Aufwand für den Wartungsdienst innerhalb der Gruppe gleichmäßig verteilt werden. Die wesentlichen Anforderungen an die Realisierung des Wartungsdienstes sind:

1. Aufrechterhaltung des globalen Wissens und der MOVS-Eigenschaften (vgl. Kapitel 5),
2. Kontinuierliche Optimierung der Kommunikationsverbindungen unter Berücksichtigung der verfügbaren Ressourcen jedes Peers,
3. Isochrone, lokale Diensterbringung ohne Abhängigkeiten von weiteren externen Diensten

8.1.1 Aufrechterhaltung des globalen Wissens und der MOVS-Eigenschaften

Der Wartungsdienst muss gewährleisten, dass die Eigenschaften der Mobile Optimistic Virtual Synchronity stets erhalten bleiben[1]. Vier Eigenschaften sind dabei von besonderer Relevanz:

1. Globale Ordnung der Sichten,
2. Übereinstimmung der Sichten,
3. Selbstinklusion,
4. Lebendigkeitseigenschaften.

1. Die globale Ordnung der Sichten (engl. *View Order*) muss auch während einer Wartung weiter bestehen, wobei zusätzliche Sichten erzeugt werden können. Die Wartungsoperationen müssen sich also bei allen fehlerfreien Gruppenmitgliedern auf die gleiche Sicht auswirken und zu einem identischen Ergebnis führen. Insbesondere ist das gemeinsame Wissen bezüglich des Gruppenaufbaus zu sichern.

2. Als Konsequenz aus (1) folgt, dass Optimierungen der Topologie nicht die Übereinstimmung der Sichten innerhalb der Gruppe verletzten dürfen. Nach Fischer *et al.* [83] ist es bereits mit einem fehlerhaft agierenden Peer innerhalb der Gruppe unmöglich, Konsens zwischen den Peers herzustellen. Deshalb muss gewährleistet werden, dass fehlerhaft agierende Peers die Gruppe nicht beeinflussen können.

3. Für den Wartungsdienst muss die Selbstinklusionseigenschaft aus Kapitel 4 gelten. Sie besagt, dass die Wartung nur dann ausgeführt werden kann, wenn der ausführende Peer Bestandteil der Ergebnissicht ist.

4. Ebenso muss bei einer Wartung die Zustellungslebendigkeit aufrechterhalten werden (vgl. Anhang B), so dass ein Peer auch nach einer Wartung noch als fehlerfreies Mitglied der Gruppe agiert und Nachrichten an alle anderen fehlerfreien Peers (einschließlich sich selbst) versenden kann.

8.1.2 Kontinuierliche ressourcenbewusste Optimierung

Die verfügbaren Ressourcen eines Endgerätes bestimmen die maximal erreichbare Kooperationsfähigkeit eines Peers. Werden unter idealen Bedingungen nur die kommunikationsspezifischen Verbräuche betrachtet, so wird der Energieverbrauch nach [43, 44, 58, 169, 206] vorrangig bestimmt durch:

- das verwendete Kommunikationsmedium,
- die technischen Eigenschaften des mobilen Endgerätes,

[1] Formale Ergänzungen dazu sind im Anhang D ausgeführt.

- die angewendeten Maßnahmen zur Reduzierung des Kommunikationsaufwands (z. B. Art der Antenne, Anwendungsdesign, Topologie),
- den konkreten Anwendungsfall einschließlich des verwendeten Protokollstacks,
- die Bewegungsgeschwindigkeit des mobilen Endgerätes,
- das Kommunikationsschema (z. B. beteiligte Peers, Pfadlänge, Gruppengröße, gleichzeitige Sendevorgänge, Sendefrequenzen),
- die Prozessausführungszeit.

Von der kollaborativen Anwendung können nur einzelne dieser Faktoren direkt beeinflusst werden. Neben einer reduzierten und sparsamen Kommunikation gilt es vor allem, die Anzahl der durch einen Peer zu verarbeitenden Nachrichten zu reduzieren. Insbesondere die Master haben einen erhöhten Energieverbrauch, da sie im Vergleich zu einem Slave mehr Kommunikationsverbindungen parallel aufrechterhalten müssen und eine größere Anzahl von Nachrichten verarbeiten. Der kommunikationsbezogene Energieverbrauch eines Peers ist somit abhängig von dessen Rolle innerhalb der Gruppe. Daher ist es vorteilhaft, diese periodisch zu wechseln und bei der Rollenvergabe die jeweilige Ressourcensituation der einzelnen Peers zu berücksichtigen. Gruppenein- und -austritte bilden dabei im Laufe einer kollaborativen Sitzung natürliche Optimierungszeitpunkte. In Phasen ohne eine Änderung der Gruppenzusammensetzung sind andere Trigger zu identifizieren, welche eine Optimierung der Topologie auslösen.

8.1.3 Isochrone, lokale Diensterbringung

Die dezentrale Ausführung der Wartung muss gewährleisten, dass die lokalen Optimierungen jeweils eine konsistente Clustertopologie sicherstellen. Die Horizonte, entlang derer Ereignisse aufgrund von Kommunikationsverzögerungen bei der Übertragung innerhalb der kollaborativen Gruppe auftreten, bilden Isochrone, d. h. Linien gleichen Zeitpunktes des Eintritts eines Ereignisses. Abbildung 8.1 stellt exemplarisch die isochrone Verteilung der Sichten V, V' und V'' in einer Gruppe mit vier Peers dar. Aufgrund der ihm inhärenten Asynchronität des Gruppenkommunikationssystems haben die Peers p_1 und p_2 die Sicht V installiert, die übrigen die Sicht V'. Die jeweilige Sicht der einzelnen Peers wird in der Abbildung über den roten Punkt dargestellt. Die Installation einer Sicht ist dabei unabhängig von der Zeit, sondern nur von den durch den Peer verarbeiteten Ereignissen.

Um die Konsistenz der Gruppe zu wahren, muss die Wartung bei allen Peers in derselben Sicht stattfinden. Neben dem Sichtwechsel können auch auslösende Trigger für die Optimierung genutzt werden, z. B.:

- die Anzahl der ausgelieferten Nachrichten,
- die aktuelle logische Zeit oder
- die Ressourceneigenschaften der Peers.

Die Fehlererkennung wird durch den Netzfehlerdetektor durchgeführt. Daher kann verallgemeinert das Ziel der Wartungsmaßnamen als eine Optimierung der clusterbasierten Topologie beschrieben werden, welche vorrangig der Ressourceneinsparung dient.

8.2 Ansätze zur Clusterwartung

Die Kommunikationstopologie ist eine wesentliche Eigenschaft eines verteilten Systems, seine kontinuierliche Anpassung ein klassisches Optimierungsproblem. Es gibt bereits verschiedene Ansätze für die optimale Platzierung von Peers in einer verteilten Topologie [3, 166]. Diese Protokolle werden hier Clusterprotokolle genannt. Sie können in (1) zentralisierte, (2) verteilte und (3) hybride Protokolle unterteilt werden. Sie beziehen

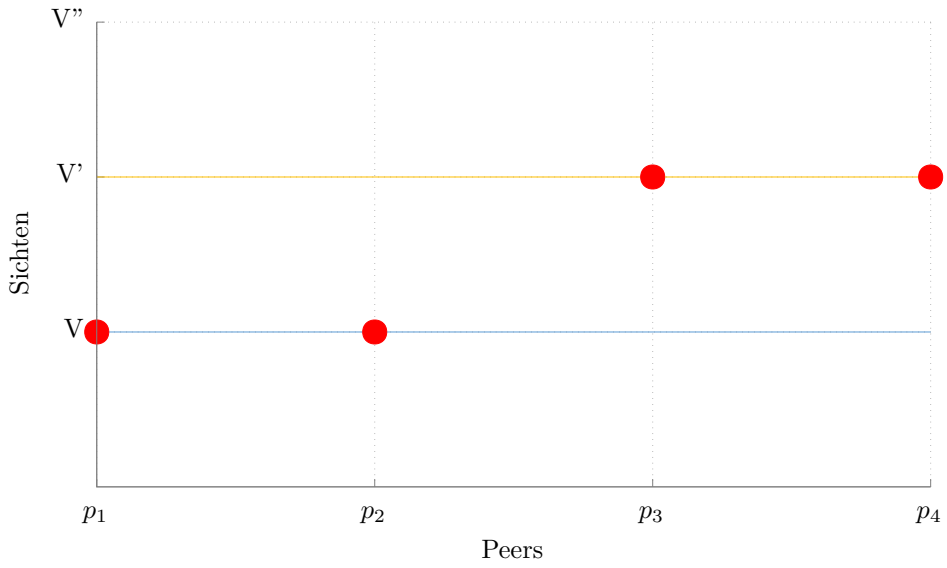


Abbildung 8.1: Exemplarische Darstellung der isochronen Verteilung der Sichten V, V' und V'' in einer Gruppe mit vier Peers

sich jedoch meist auf MANETs und Sensornetze, weniger auf mobile kollaborative Anwendungen. Drei Beispiele solcher Clusterprotokolle soll hier vorgestellt werden.

SONDe [104] ist ein Beispiel für eine Klasse von Protokollen, welche Cluster hauptsächlich auf Basis der Anzahl logischer Hops zwischen dem Master und einem Slave bilden. Diese Protokolle stammen zumeist aus dem LAN- und WAN-Bereich. Dabei bestimmt eine konstante oberen Schranke h die maximale Anzahl von Hops bis zum Master. Jeder Peer beobachtet kontinuierlich seine h-Hop Umgebung und entscheidet, ob er als Master oder Slave fungiert. Die Clustertopologie ändert sich dynamisch in Abhängigkeit von der Knotenumgebung. Die entstehende Topologie ist instabil und schlecht ausbalanciert, was zu Single-Point-of-Failure-Situationen bzw. einer frühzeitigen Ressourcenerschöpfung von mobilen Peers führen kann. Daher ist diese Art von Protokollen nicht geeignet für mobile kollaborative Anwendungen.

Protokolle wie *NONSTOP* [139] und andere [23, 72, 89] ordnen die Cluster aufgrund der Peer-Eigenschaften. NONSTOP zielt in erster Linie auf die Vermeidung von Netzpartitionierungen, ausgelöst durch die Mobilität der Anwendungsnutzer. Teilnehmer mit vergleichbaren Eigenschaften werden in einem Cluster zusammengefasst. Die Werte der gewählten Teilnehmereigenschaften (beispielsweise deren Position) werden per Piggybacking in der Nachbarschaft verteilt. Basierend auf Prädiktionstechniken schätzt NONSTOP Ort und Zeitpunkt ab, wo und wann die nächste Partitionierung des Netzes auftreten kann. Daraufhin wählt es im Voraus neue Master, um das Entstehen der Partitionierung zu vermeiden. Auch hier variiert die Anzahl von Peers pro Cluster und deren Belastung über die Zeit. Diese Art von Protokollen erreichen eine Masterverfügbarkeit von 80 %, womit eine kontinuierliche Synchronität der Gruppe nicht gewährleistet werden kann.

Sensornetzprotokolle wie *Low-Energy Adaptive Clustering Hierarchy* (LEACH) [112] oder die Scheduling Plattform von Frincu *et al.* [88] nutzen statistische Techniken für die Clusterbildung. Rundenbasiert selektiert LEACH P-Master, wobei P ein vordefinierter Wert ist. Durch den verteilten Auswahlalgorithmus werden die aktuell verfügbaren Energieressourcen eines Peers als Kriterium verwandt. Die übrigen Peers werden demjenigen Cluster als Slave zugeordnet, dessen Master sie am nächsten sind. Für die Kommunikation innerhalb des Cluster wird ein Zeitschlitzverfahren genutzt. Der statistische Ansatz reduziert zwar die Komplexität des Algorithmus für die Clusterbildung, die Zuordnung der Slaves zu den Clustern ist in der Regel jedoch instabil und führt zu dynamischen Topologien, welche die Gruppensynchronität nicht sichern können.

8.3 Ein Ansatz für eine verteilte Optimierung von Clustertopologien

Die Bestimmung der optimalen Clustertopologie für die kollaborative Gruppe ist nach Agarwal und Procopiuc ein Problem der Klasse NP[2] [5]. Da kein effizienter Optimierungsalgorithmus bekannt ist, könnten Verfahren der kombinatorischen Optimierung verwendet werden, wie das *simulated annealing* [135], die lokale Suche [2], die Tabu-Suche [102] oder evolutionäre Algorithmen [13].

Nach Aigner [7] kann über die *Bellsche Zahl* $B(N)$ die Anzahl aller möglichen Teilnehmerkombinationen und damit der Aufwand dieser Verfahren bestimmt werden. Stubbe [203] zeigt deren Bestimmung über die zweite Stirlingzahl im Kontext einer clusterbasierten kollaborativen Gruppe mit bis zu 100 Mitgliedern. Dazu werden alle möglichen Kombinationen einer Aufteilung von n-vielen Gruppenteilnehmern auf k-viele Cluster bestimmt. Für eine kollaborative Gruppe mit bis zu 100 Teilnehmern ergibt dies $B(100) = 4,7585 * 10^{115}$ mögliche Teilnehmerkombinationen. Wegen dieser großen Anzahl ist die wiederholte Verwendung der genannten kombinatorischen Verfahren nicht sinnvoll. Ein „*Durchprobieren*“ aller möglichen Lösungen übersteigt die Fähigkeiten und Ressourcen mobiler Endgeräte. Aus diesem Grund wird nur für ausgewählte Eigenschaften die optimale Clustertopologie ermittelt. Mögliche Eigenschaften sind dabei die aktuelle Clustergröße, der aktuelle Ressourcenwert der Peers und deren ID.

8.3.1 Optimierung der Heterogenität im Clusternetz

Die Unterschiede der Peers hinsichtlich ihrer Eigenschaften und Kommunikationsverbindungen können als Maß der Heterogenität des Clusternetzes genutzt werden [166]. Die tatsächliche Abweichung von bestimten Eigenschaften wie der Anzahl von Mitgliedern je Cluster oder die Ressourcenverteilung zwischen Clustern und innerhalb eines Clusters gibt die Heterogenität der aktuellen Netzsituation gegenüber einem definierten Referenzwert an. Der Referenzwert als solches kann vielfältig bestimmt werden, beispielsweise über den Durchschnitt oder das Optimum einer gegebenen Verteilung. Auf dieser Basis kann daraufhin das Netz optimiert werden. Dazu wird für jeden Cluster der aktuelle normierte Eigenschaftswert analog zur Bestimmung des Referenzwerts ermittelt und markiert als:

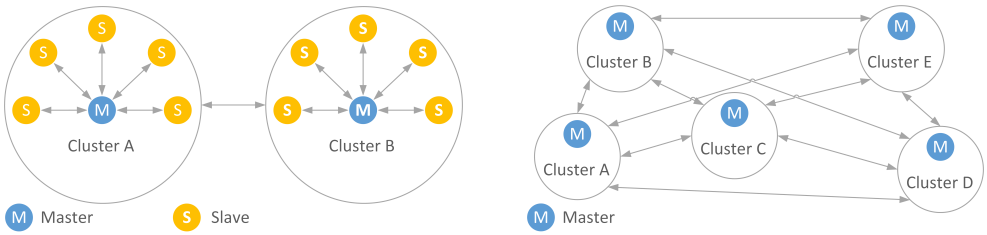
- *überlastet* – wenn der Eigenschaftswert des Clusters größer ist als der Referenzwert,
- *normal* – wenn Referenzwert und Eigenschaftswert des Clusters (nahezu) gleich sind, oder
- *entlastet* – wenn der Referenzwert oberhalb des Clustereigenschaftswertes liegt.

Semantisch betrachtet verwaltet beispielsweise ein überlasteter Cluster mehr Mitglieder als optimal, und ein entlasteter Cluster weniger. Ein als normal markierter Cluster bedarf keiner Optimierung. Auf Basis der Markierungen kann die aktuelle Verteilung bezüglich einer Normalverteilung für die gewählte Eigenschaft bestimmt und beispielsweise der optimale Cluster für ein der Gruppe beitretendes Mitglied ermittelt werden.

Der hier vorgestellte Ansatz ist allgemeingültig und kann für iterative Optimierungen bezüglich einer frei gewählten normierbaren Topologieeigenschaft genutzt werden. Um beispielsweise die Last eines Clusters, d. h. die Anzahl der Peers je Cluster zu optimieren, muss wie folgt vorgegangen werden:

1. Ermitteln des Referenzwerts für alle Cluster (beispielsweise über die durchschnittliche Anzahl von Peers pro Cluster).
2. Festlegen der zulässigen Varianz des Referenzwertes in Abhängigkeit vom jeweiligen Anwendungsszenario. Diese Definition stellt immer eine Abwägung dar: Eine kleine Varianz erhöht die Anzahl von Optimierungsiterationen, eine große Varianz

[2] NP für nichtdeterministisch polynomielle Zeit



(a) Clustertopologie mit zu wenigen Clustern

(b) Clustertopologie mit zu viel Clustern

Abbildung 8.2: Beispiele für nichtoptimale Clustertopologien

reduziert den Optimierungsaufwand, verursacht aber potentiell einen höheren Kommunikationsaufwand.

3. Bestimmen der Last der einzelnen Cluster und ihre Markierung entsprechend des Referenzwerts.

4. Optimieren der Topologie durch Verschiebung von Peers zwischen den Clustern und anschließende Rollenzuweisung.

5. Erneutes Ermitteln der Last der beiden beteiligten Cluster und Wiederholen des Verfahrens für den nächsten Peer. Der Algorithmus stoppt, sobald die Last der einzelnen Cluster „*nahe*“ dem Referenzwert ist.

8.3.2 Optimierung der Anzahl der Cluster

Einen wesentlichen Einfluss auf die Latenz einer Gruppennachricht hat die die Anzahl der im Overlay vorhandenen Cluster [72]. Die Abbildungen 8.2a und 8.2b zeigen exemplarisch Extremfälle einer möglichen voll vermaschten Clustertopologie. Durch die geringe Anzahl von Clustern in Abbildung 8.2a müssen die Master (M) in der Topologie eine hohe Anzahl von Verbindungen zu den Slaves (S) verwalten. Zusätzlich haben sie eine weitere Verbindung zu jedem anderen Cluster. Das erhöht die Verzögerungen bei der Nachrichtenübertragung und wandelt bei zunehmender Gruppengröße die Eigenschaften der gesamten Topologie in die einer Sterntopologie.

Demgegenüber steigt bei dünn besetzten Clustern die Anzahl von Verbindungen zwischen den Clustern (siehe Abbildung 8.2b). Wie im Abschnitt 6.3.1 ausgeführt, ist der *Netzfehlerdetektor* für das Erkennen und Behandeln von Peer-Ausfällen im Overlay zuständig. Durch dessen Überwachungsschemata würden sich in einer dünn besetzten Clustertopologie wie in Abbildung 8.2b alle Teilnehmer gegenseitig überwachen. Das kann das Netz stark belasten und viele Ressourcen verbrauchen.

In [203] und [95] wird ausführlich die Bestimmung der optimalen Anzahl von Clustern in einer Clustertopologie diskutiert. Die Herleitung geht von der Annahme aus, dass die N Gruppenteilnehmer gleichmäßig auf eine optimale Anzahl von Cluster verteilt werden sollen, wobei das entstehende Clusternetz als Graph mit N Knoten und E Kanten betrachtet wird. Jedes Clusternetz hat dabei M Master, die untereinander voll vermascht sind. Mit der Gesamtverbindungszahl E (Formel 8.1) sowie der Annahme, dass jeder Cluster genau einen Master besitzt, wird die Anzahl von Kanten im Clusternetz E in Abhängigkeit von der Clusteranzahl C, $e(C)$ bestimmt (siehe Formel 8.2).

$$E = (M^2 - M)/2 + N - M \tag{8.1}$$

$$e(C) = (C^2 - C)/2 + N - C \tag{8.2}$$

Abbildung 8.3 stellt den Verlauf dieser Funktion für eine Gruppe von 16 Peers dar. Das Minimum dieser Funktion ist mit $1,5$ unabhängig von der Gruppengröße (vgl. Abbildung 8.4). Wird die globale Anzahl an Verbindungen betrachtet, ist folglich ein „*degeneriertes*“ Clusternetz aus ein bzw. zwei Clustern optimal. Die Cluster wären

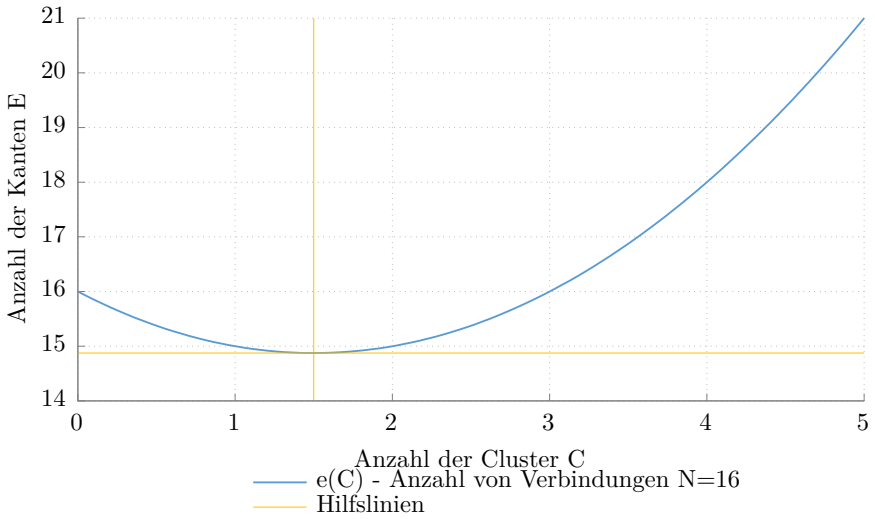


Abbildung 8.3: Verlauf von $e(C)$ für $N = 16$

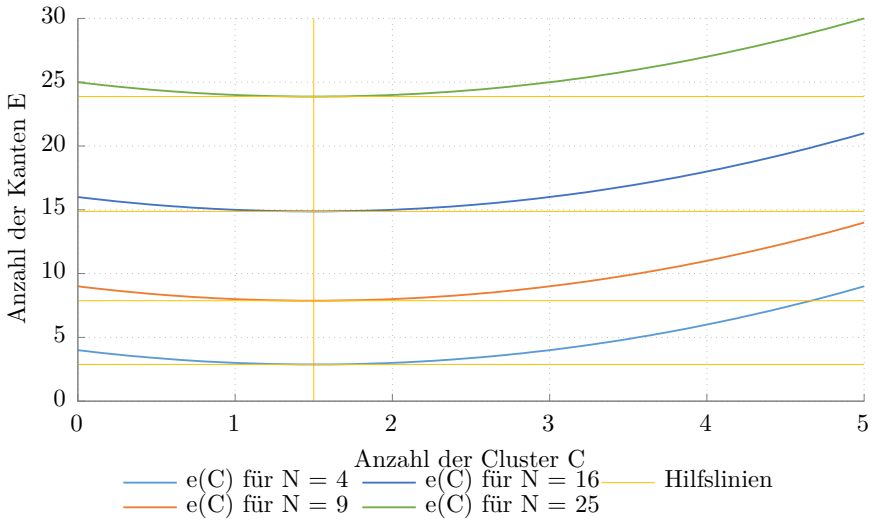


Abbildung 8.4: Verlauf von $e(C)$ für verschiedene Gruppengrößen

jedoch sehr dicht besetzt, was den Verwaltungsaufwand der Master erhöht. Daher ist es vorteilhaft, nur die Verbindungszahl der einzelnen Teilnehmer zu optimieren, wodurch die Anzahl von Übertragungen je Master und damit der Ressourceneinsatz reduziert werden [3].

Jeder Slave hat genau eine Verbindung zu seinem Master, während die Anzahl der Verbindungen bei den Mastern je nach Anzahl der Cluster und Slaves variiert. Daher kann für einen Master die durchschnittliche, gewichtete Verbindungsanzahl nach Formel 8.3 definiert werden. Dabei stellt α den Gewichtungsfaktor für Verbindungen zwischen den Mastern und β den für die Slaves dar.

$$E_{m,d,w} = \alpha(C-1) + \beta((N-C)/C-1) \tag{8.3}$$

Unter der Annahme, dass der Ressourcenverbrauch eines Peers sinkt, je weniger Verbindungen er zu verwalten hat, wird in [95] die Gleichung 8.3 minimiert. Aus $C \in N$ folgte als gültige Lösung $C_1 = \sqrt{\beta * N/\alpha}$. In [95, 203] wird gezeigt, dass eine Topologie, die gegenüber der optimalen Clusteranzahl zu wenige Cluster besitzt, einen größeren Optimierungsgewinn erzielt als eine Topologie mit einer Clusterzahl oberhalb

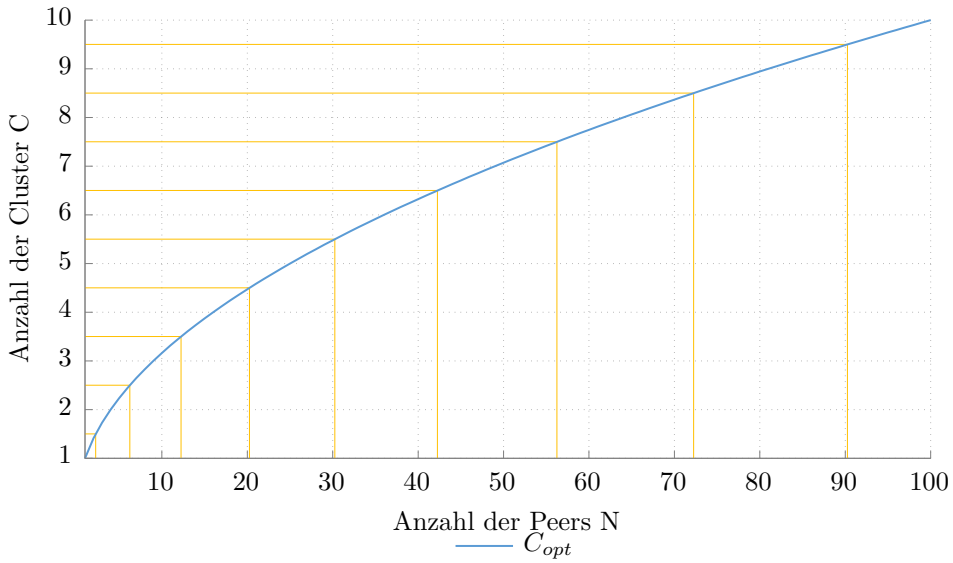


Abbildung 8.5: Verlauf der Funktion C_{opt} für $N \in 1, \ldots, 100$

des Optimums. Der Grund dafür besteht in der sinkenden Anzahl von Verbindungen pro Master.

Für eine *Moversight*-Gruppe mit maximal 100 Teilnehmern existiert jedoch kein signifikanter Unterschied zwischen Master-Master- und Master-Slave-Verbindungen. Daher können die Gewichte α und β jeweils eins gesetzt werden. Folglich bestimmt Formel 8.4 die optimale Clusteranzahl für die betrachtete Topologie. Der Verlauf der Funktion C_{opt} bis zu einer Gruppengröße von einhundert ist in Abbildung 8.5 dargestellt.

$$C_{opt} = \sqrt{N} \tag{8.4}$$

8.3.3 Optimierung der Masterwahl

Ein guter Master zeichnet sich durch ausreichende Ressourcen und eine stabile Netzverbindung aus [3]. Seine Wahl kann als Teil einer Protokollstrategie umgesetzt werden, die zentral bzw. dezentral realisierbar ist. Für mobile kollaborative Anwendungen wird eine dezentrale, P2P-basierte Umsetzung bevorzugt. Aus der Vielzahl von möglichen Strategien werden die wichtigsten hier kurz angerissen:

Round-Robin-Strategien Diese Strategien ordnen den Peers eines Clusters rundenweise iterativ die Rolle des Masters zu, unabhängig von ihren konkreten Eigenschaften. Diese Strategien werden aufgrund des begrenzten Selektionsaufwandes vor allem bei drahtlosen Sensornetzen eingesetzt, wie beispielsweise in Heinzelman *et al.* [112] vorgestellt.

Da mobile kollaborative Anwendungen sichtbasiert arbeiten, kann jeder Peer lokal diese Selektionsstrategie ausführen, ohne zusätzlichen Kommunikationsaufwand zu verursachen. Round-Robin-Strategien verteilen die Last fair über den Cluster und vermeiden, dass einige Peers häufiger als Master agieren müssen. Von Nachteil ist, dass auch Peers als Master agieren, welche für diese Rolle nicht geeignet sind [197].

Ressourcenbasierte Strategien Die Wahl des Masters entsprechend des aktuellen Ressourcenwertes eines Peers ist das Grundprinzip der ressourcenbasierten Strategien [112]. Dadurch können zu „*schwache*“ oder ungeeignete Peers von vornherein von der Wahl ausgeschlossen und somit nur der „*am besten*“ geeignete Peer als Master ausgewählt werden. Diese Strategien können rundenbasiert umgesetzt werden.

Prinzipiell kann jede Ressource bzw. eine Kombination verschiedener Ressourcen als Ressourcenwert verwendet werden, sofern eine lokale Funktion existiert, welche den aktuellen Zustand der Ressource oder Ressourcenkombination auf einen konkreten normierten Wert – den *Ressourcenwert* – abbildet. Er wird der Gruppe als globales Wissen bereitgestellt. Beispielsweise kann der aktuelle Ladezustand der Batterie eines Endgeräts in Prozent auf den Ressourcenwert abgebildet werden. Endgeräten, welche eine konstante Stromversorgung besitzen, wird entsprechend der maximale Ressourcenwert zugeordnet. Analog lassen sich die Geräteklassen (Smartphone, Laptop, Sensorknoten, etc.) und deren Rechenkraft (im Sinne von *computation power*) in die Abbildung mit einbeziehen. Ebenso kann die Zeit berücksichtigt werden, in welcher ein Peer als Master agiert hat. Um den erhöhten Ressourcenverbrauch eines Masters zu berücksichtigen, verteilt eine faire Rollenselektionsstrategie die entstehenden Lasten gleichmäßig entsprechend den individuellen Möglichkeiten auf alle Peers des Clusters.

Ebenso können die Eigenschaften der Netzverbindung als Selektionskriterium genutzt werden. Durch die variable und fluktuierende Natur mobiler Kommunikationsnetze müssen jedoch die einzelnen Werte der Eigenschaften wie Latenz, Datenrate oder Signalstärke über die Zeit geglättet werden, bevor sie für die Berechnung des Ressourcenwertes genutzt werden.

Schließlich besteht die Möglichkeit, mehrere Eigenschaftswerte auf einen einzelnen Ressourcenwert abzubilden. Entsprechende Ansätze – wie beispielsweise eine vektorielle Abbildung – werden unter anderem von Basagni [23] diskutiert.

Auktionsbasierte Strategien Diese Strategien sind in ihrer Umsetzung komplexer als die bisher genannten [159]. Jeder Gruppenteilnehmer muss sein Gebot für die Masterrolle in der Gruppe bekannt geben. Das kann wiederum über einen verteilten [56, 225] oder über einen zentralisierten [29] Ansatz erfolgen. In beiden Fällen ist es notwendig, rundenweise die Gebote der einzelnen Peers zu ermitteln. Dabei muss die Fairness zwischen den Peers für jedes Gebot gewährleistet werden, um sicher zu stellen, dass jeder Peer potentiell Master werden kann. Dies ist ein bekanntes P2P-Problem, welches in Ansätzen auch bei den anderen Strategien auftritt. Die Umsetzung eines verteilten sichtbasierten Ansatzes, welcher diese Rahmenbedingungen sichert, ist sehr aufwendig.

Im Vergleich der vorgestellten Strategien bieten die ressourcenbasierten Strategien den besten Kompromiss zwischen Kommunikationssaufwand und Güte der Masterwahl. Round-Robin-basierte Strategien sind einfach umzusetzen, bergen jedoch das Risiko, Peers als Master zu bestimmen, welche wegen ihrer Eigenschaften nicht für diese Rolle geeignet sind. Auktionsbasierte Strategien erfordern in einem dezentralen System den globalen, zuverlässigen Austausch der Gebote sowie die Sicherung der Fairness für alle Teilnehmer. Dies ist kompliziert und erzeugt wiederkehrend einen erhöhten Kommunikationsaufwand. Die Verlässlichkeit der Ressourceninformationen muss bei den ressourcenbasierten Strategien ebenso sichergestellt werden wie die Abgabe der Gebote bei den auktionsbasierten Strategien. Jedoch können Ressourceninformationen kontinuierlich beispielsweise über Piggy-back-Verfahren innerhalb der Gruppe verteilt werden, was den Kommunikationsaufwand gegenüber den auktionsbasierten Strategien senkt. Auf Basis dieser Informationen können gegenüber den Round-Robin-Strategien geeignetere Peers als Master gewählt werden. In der Gesamtbetrachtung bietet es sich daher an, eine ressourcenbasierten Strategie für die Masterwahl zu nutzen.

8.4 Ein verteilter Ansatz zur Optimierung von Clustertopologien für mobile kollaborative Anwendungen

Der hier vorgeschlagene Ansatz für einen verteilten Wartungsdienst besteht aus drei Teilen:

1. einer dynamischen Peer-Platzierungsstrategie,
2. einem unabhängigen Rollenwartungsdienst und
3. einem Ressourcenupdatehilfsdienst.

8.4.1 Dynamische Peer-Platzierung

Die Zuordnung der einzelnen Peers zu den Clustern realisiert die Gruppenverwaltung mit Hilfe einer Platzierungsstrategie. Die im Abschnitt 5.3 vorgestellten einfachen Strategien gehen vor allem auf eine Platzierung basierend auf statischen Eigenschaften der Gruppe bzw. der einzelnen Peers zurück. Im Rahmen der Entwicklung des Wartungsdienstes wurden diese um die DCS erweitert, welche die Platzierungsentscheidungen unter Nutzung einer Platzierungsmetrik trifft. Für Details und weiterführende Informationen zum DCS-Algorithmus über die folgende Darstellung hinaus sei auf [95] verwiesen.

DCS ist verantwortlich für die Verwaltung der Cluster in der Topologie. Die Verteilung der Peers über die Cluster wird durch eine Metrik gesteuert. Als Metrik wird hier eine Kontrollfunktion bezeichnet, welche anhand von bestimmten Eigenschaften die Peers und die Cluster beurteilt. So kann mit Hilfe einer Ressourcenwert-Metrik beispielsweise der Peer mit den meisten verfügbaren Ressourcen im Cluster oder in der Gruppe ermittelt werden. Ändert sich die Zusammensetzung der Gruppe, so wird durch den DCS-Algorithmus eine optimale Verteilung der Peers innerhalb der Topologie gesichert. Beispielsweise wird beim Gruppeneintritt eines Peers zunächst die notwendige Anzahl von Clustern nach Formel 8.4 bestimmt und die Topologie entsprechend angepasst. Existieren zu viele Cluster, werden zwei Cluster verschmolzen. Anderenfalls wird, sofern nötig, ein neuer Cluster erzeugt. Anschließend wird der neue Peer dem am wenig belasteten Cluster zugewiesen.

Der DCS-Ansatz ist allgemeingültig und besteht aus einem dreistufigen iterativen Algorithmus (siehe Algorithmus 2). Zuerst wird die Anzahl von Optimierungsrunden bestimmt. Die maximale Anzahl der Optimierungsrunden ist durch die Anzahl der aktuellen Gruppenmitglieder nach oben beschränkt. Anschließend wird die Topologie optimiert, indem Peers von den überlasteten Clustern in die entlasteten Cluster verschoben werden. Dieser Schritt wird *Vorwärtsoptimierung* (engl. *forward balancing*) genannt. Ist die für die Optimierung verwendete Eigenschaft spezifisch für jeden Peer, dann ist ein dritter Schritt notwendig – die *Rückwärtsoptimierung* (engl. *backward balancing*) – um durch die Optimierung überlastete Cluster zu entlasten.

Algorithmus 2 Die Funktion *balance*

```
1: max_rounds ⇐ N − C
2: balancingForward(max_rounds)
3: if property is peer specific then
4:     balancingBackward(max_rounds)
5: end if
```

Für die Optimierung der Topologie auf Basis der Clustergrößeneigenschaft wird zunächst der Grad der Clusterheterogenität über den Referenzwert der Topologie bestimmt (vgl. Abschnitt 8.3.1). Bei Verwendung der Clustergröße wird dieser über die durchschnittliche Anzahl von Peers pro Cluster ermittelt. Anschließend wird eine zulässige Abweichung von diesem Wert durch den Anwendungsentwickler definiert. Als normal gilt jeder Cluster, dessen Anzahl von Peers innerhalb der Spanne Referenzwert $+/-$ zulässige Abweichung liegt. Für die Optimierung wird die Belastung jedes Clusters bestimmt. Algorithmus 3 zeigt den Ablauf der Vorwärtsoptimierung.

Pro Runde wird der leistungsstärkste Peer von dem am stärksten überlasteten (Zeile 3, *most overloaded cluster*) in den am wenigsten belasteten Cluster (Zeile 4, *least loaded cluster*) verschoben (Zeile 7). Im Beispiel wird ein beliebiger Peer deterministisch im Quellcluster ausgewählt (Zeile 6). Ist die zu optimierende Eigenschaft spezifisch für einen Peer (beispielsweise der aktuelle Ladezustand), wird statt eines beliebigen Peers der leistungsstärkste Peer im Sinne dieser Eigenschaft ausgewählt. Als nächstes wird der schwächste Peer aus den am wenigsten belasteten in den am meisten überlasteten Cluster verschoben (Zeile 7). Anschließend werden die Last der einzelnen Cluster sowie der Referenzwert neu berechnet (Zeilen 8 und 9). Dieser Algorithmus wird für maximal $N - 1$ Peers wiederholt. Im Ergebnis ist die Anzahl der Peers je Cluster so nah wie möglich am Referenzwert.

Algorithmus 3 Die Funktion $forwardBalancing$

```
1: for i = 0 to max_rounds do
2:     if exists overloaded cluster then
3:         srcC ⇐ getMostOverLoadedCluster()
4:         dstC ⇐ getLeastLoadedCluster()
5:         if isBalancingPossible(srcC, dstC) is true then
6:             moving_peer ⇐ srcC.getPeer()
7:             memberRegister.movePeer(moving_peer, dstC)
8:             calculateClusterLoad()
9:             determineReferenceValue()
10:        end if
11:    end if
12: end for
```

Bei der Initialisierung der Metrik wird die aktuelle Belegung der einzelnen Cluster bewertet. Die Cluster und Peers werden markiert. Wurden die Peers verschoben, wird in jedem Cluster der leistungsstärkste Peer als Master gewählt. Es werden nachfolgende Platzierungsmetriken definiert:

1. Clustergrößen-Metrik,
2. Ressourcenwert-Metrik,
3. gemischte Metrik.

Clustergrößen-Platzierungsmetrik. Bei der Clustergrößen-Metrik wird der Referenzwert nach Gleichung 8.5 bestimmt. Ziel dieser Metrik ist die Balancierung der Anzahl von Peers je Cluster. Für diese Metrik ist der am meisten überlastete Cluster derjenige, welcher die größte Anzahl von Peers verwaltet. Als Master wird der Peer mit der kleinsten ID gewählt. Dies basiert auf der Annahme, dass die ID eines Peers um so kleiner ist, je länger er in der Gruppe ist. Das birgt jedoch das Risiko, dass auch ungeeignete Peers gewählt werden. Verwendet wird dieses Kriterium beispielsweise beim Flugraumüberwachungsprogramm *PHIDEAS* [18] oder in JGroups [161].

$$R_{Netz} = N/C \tag{8.5}$$

Abbildung 8.6 stellt das Ergebnis einer solchen Balancierung graphisch dar: Die Gruppe besteht aus sieben Peers ($p_i, i \in 1, 7$), welche auf drei Cluster verteilt sind. Die Wahl des Masters in den Clustern A und B ist bezüglich der verfügbaren Ressourcen jedoch nicht optimal, da sie den Referenzwert ihres Clusters sowie den des Gesamtnetzes unterschreiten.

Ressourcenwert-Platzierungsmetrik Die Ressourcenwert-Metrik balanciert die Clusterbelegung auf Basis der durchschnittlich verfügbaren Ressourcen pro Peer. Der Referenzwert des Netzes R_{Netz} wird hier nach Gleichung 8.6 berechnet, wobei RW_i der Ressourcenwert des Peers P_i ist.

$$R_{Netz} = \frac{SUM_{i=1}^{N} RW_i}{N} \tag{8.6}$$

Um eine optimale Ressourcenverteilung zu erreichen, werden Peers mit einem höheren Ressourcenwert in Cluster verschoben, welche einen geringeren durchschnittlichen Ressourcenwert besitzen als der Referenzwert des Netzes. Umgekehrt werden Peers mit einem geringeren Ressourcenwert als der Referenzwert in einen Cluster verschoben, welcher einen höheren durchschnittlichen Ressourcenwert als R_{Netz} besitzt. Die Balancierung ist für höhere und niedrigere Ressourcenwerte im Vergleich zu R_{Netz} durchzuführen. Abbildung 8.7 stellt das Ergebnis der Balancierung für die gleiche Gruppensituation wie oben dar.

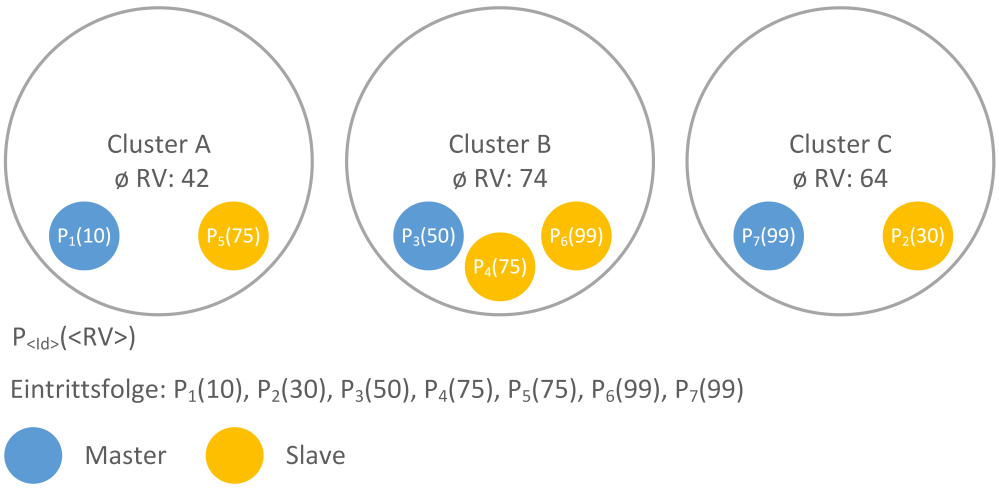


Abbildung 8.6: Beispiel für eine Topologie, balanciert unter Nutzung der Clustergrößen-Platzierungsmetrik

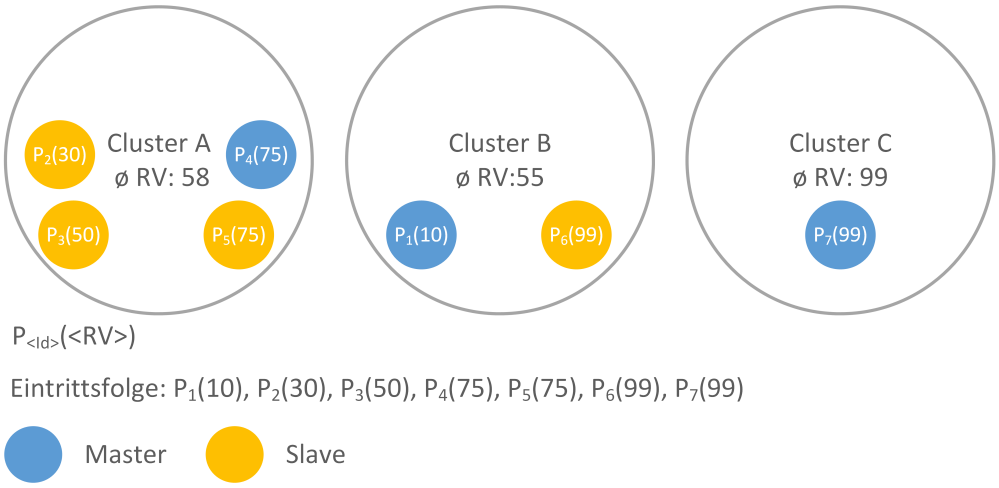


Abbildung 8.7: Beispiel für eine Topologie, balanciert unter Nutzung der Ressourcenwert-Platzierungsmetrik

Die geeignetsten Peers je Cluster wurden als Master bestimmt. Ungünstig ist jedoch die inhomogene Zusammensetzung der Cluster. Zudem können Pendeleffekte auftreten, d. h. es kann zu einer wechselseitigen Verschiebung eines oder mehrerer Peers zwischen zwei Clustern kommen.

Gemischte Platzierungsmetrik Diese Metrik kombiniert die beiden vorangegangenen Metriken. Die Peers werden entsprechend der Clustergröße zugeordnet. Die Master werden anhand der verfügbaren Ressourcen bestimmt. Das Ergebnis einer Balancierung für die zuvor genutzte Beispielgruppe zeigt Abbildung 8.8. Sie zeigt eine gleichmäßige Verteilung der Peers und eine günstigere Masterauswahl.

8.4.2 Rollenwartungsdienst

Die Dynamische Peer-Platzierung balanciert das Overlay immer dann, wenn ein Peer der Gruppe beitritt oder diese verlässt. In den Phasen ohne Änderungen der Gruppenzusammensetzung obliegt es dem *Rollen-Wartungsdienst* (RMS) (engl. *Role-Maintenance-Service*), „*schlechte*“ Master auszutauschen. Der RMS ist ebenfalls dezentral organisiert. Er operiert lokal ohne zusätzlichen Kommunikationsaufwand und ist global synchroni-

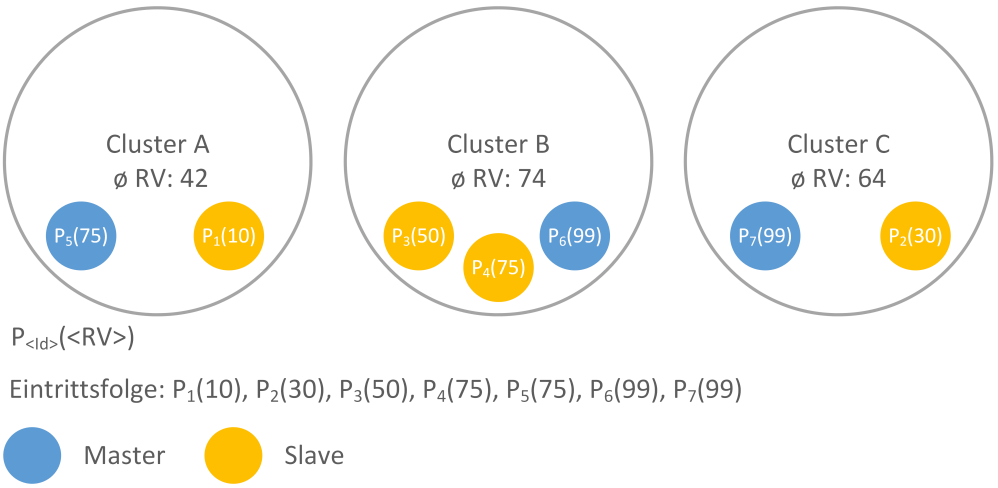


Abbildung 8.8: Beispiel für eine Topologie, balanciert unter Nutzung der gemischten Platzierungsmetrik

siert. Der Entwurf des Dienstes geht von einem kontinuierlichen Nachrichtenaustausch in der Gruppe aus.

Wird eine Gruppennachricht durch einen Peer in die Warteschlange eingereiht, so überprüft der RMS mit Hilfe der Rollenwechselzeitpunkt-Metrik, ob eine Optimierung der Clustertopologie notwendig ist. Ist dies der Fall, wird die Kollaboration auf Transportebene kurzfristig unterbrochen, indem eintreffende, nicht zur aktuellen Sicht gehörige Nachrichten zwischengespeichert werden. Die Anwendungsnutzer können weitere Nachrichten versenden, deren Zustellung wird jedoch verzögert. In dieser Phase versandte Nachrichten haben also eine höhere Latenz. Im Abschnitt 8.5 wird darauf vertiefend eingegangen.

Ist die letzte in Übertragung befindliche Nachricht der aktuellen Sicht lokal ausgeliefert, bestimmt der RMS anhand des Ressourcenwerts pro Cluster einen neuen Master. Damit das Overlay bezüglich sich eventuell in Übertragung befindlicher Nachrichten stabil bleibt, werden keine Peers zwischen den Clustern verschoben. Im Anschluss wird die Kollaboration fortgesetzt und eventuell zwischengespeicherte Nachrichten werden verarbeitet. Die Optimierung erfolgt unabhängig von der gewählten Platzierungsstrategie der Gruppenverwaltung.

Um den Zeitpunkt einer Optimierung zu bestimmen, wurden in [95] eine Reihe von Metriken zur Bestimmung des Rollenwechselzeitpunkts vorgeschlagen. Sie basieren auf drei Kriterien:

1. auf der Anzahl der in der Gruppe verarbeiteten Multicastnachrichten,
2. auf der vergangenen logischen Zeit seit der letzten Optimierung der Clustertopologie und
3. auf dem verfügbaren Ressourcenwert der einzelnen Peers.

Kriterium (1) wird weiter unterteilt in:

a. Anzahl der durch einen Master versandten Nachrichten,
b. Anzahl der von Peers eines Clusters versandten Nachrichten und
c. Anzahl der von der gesamten Gruppe versandten Nachrichten.

Dabei ist die Definition der Schwellwerte für die jeweiligen Metriken schwierig. Für die statische Definition ist Wissen über den konkreten Anwendungsfall und das zu erwartende Nachrichtenaufkommen notwendig. Eine eingeschränkte dynamische Definition ist ebenso denkbar, aber nicht weiter untersucht worden.

Das Wechselzeitpunktkriterium der logischen Zeit betrachtet die Anzahl logischer Zeitschritte oder Ereignisse in der Gruppe. Diese werden durch den verwendeten total geordneten Multicastdienst bestimmt. Da die logische Zeit vor allem durch die Anzahl der verarbeiteten Nachrichten beeinflusst wird, hängt die Wahrscheinlichkeit für einen Rollenwechsel somit von der Gruppengröße bzw. der Anzahl der ausgetauschten Multicastnachrichten ab.

Hat sich der Ressourcenwert eines Masters über einen festgelegten, logischen Zeitraum um einen definierten Wert reduziert bzw. unterschreitet dieser ein gegebenes Minimum, so wird durch die Ressourcenwert-Metrik ein Rollenwechsel initiiert. Eine ungünstige Definition des Schwellwerts für den Rollenwechsel kann dazu führen, dass nach einem Rollenwechsel auch für weitere Cluster ein Rollenwechsel notwendig wird. Um häufige Blockierungen auf Transportebene zu vermeiden, wurde daher eine weitere ressourcenbasierte Metrik definiert, welche direkt nach einem Rollenwechsel mit einem verringerten Schwellwert prüft, ob weitere Rollenwechsel notwendig sind. Ebenso wird geprüft, ob der neu gewählte Master geeignet ist oder für diesen sofort ein weiterer Rollenwechsel initiiert werden würde, was zu verhindern wäre.

8.4.3 Modellierung des Ressourcenwertes

Wie erläutert, können verschiedene Parameter wie Ladezustand, Mobilitätstyp, Geräteklasse, Netzanbindung usw. auf einen *Ressourcenwert* (RV) (engl. *resource value*) abgebildet werden. Eine Abbildung aller Parameter auf einen einzigen Wert ist nicht immer sinnvoll, da insbesondere der Mobilitätstyp des Peers (*stationär*, *nomadisch* und *mobil*) durch einzelne Metriken häufig abgefragt wird.

Für die Umsetzung des Wartungsdienstes in Moversight wurde daher der Ressourcenwert RV als ein 3*er* Tupel definiert:

$$\begin{aligned} RV =& (MOBILITÄTS\_TYPE, Ressourcenfunktionswert, Konsumrate) \\ & MOBILITÄTS\_TYPE \in \{M, S, N\} \\ & Ressourcenfunktionswert \in \{rfw \in \mathbb{R}^+ |\ 0 <= rfw <= 100\} \\ & Konsumrate \in \mathbb{R} \end{aligned} \tag{8.7}$$

wobei beim Mobilitätstyp M für mobil, S für stationär und N für nomadisch steht. Der Ressourcenfunktionswert ist das Ergebnis der Abbildung des aktuellen Akkuladestands auf das Intervall $[0, 100]$. Im einfachsten Fall entspricht dies dem prozentualen Ladungsstand des Akkumulators. Stationären Peers, die eine konstante Stromversorgung besitzen, wird der Ressourcenfunktionswert einhundert zugeordnet, anderenfalls deren aktueller Ladungszustand. Die Konsumrate als drittes Element des Tupels spiegelt die Verbrauchs- bzw. Laderate des Peers bezogen auf ein definiertes Zeitintervall wieder. Gespeichert wird der Ressourcenwert eines jeden Peers im Gruppenregister als Teil der Peer-Eigenschaften.

8.4.4 Ressourcenprofiler

Der *Ressourcenprofiler* schätzt auf Basis des globalen Wissens die Ressourcenwerte der Gruppenteilnehmer ab. Auch dieser Dienst ist dezentral definiert und operiert lokal. Das Schätzprinzip und die Verwendung des globalen Wissens sichern die gruppenweite Synchronisation der Schätzung. Weicht die Gruppenschätzung zu weit von der tatsächlichen, lokalen Ressourcensituation des Peers ab, wird durch diesen Dienst das globale Wissen bezüglich des Peers aktualisiert. Den prinzipiellen Aufbau des Dienstes zeigt Abbildung8.9. Der Ressourcenprofiler besteht aus drei Komponenten:

1. dem Gruppenprofiler,
2. dem lokalen Profiler und
3. einer Updatefunktion.

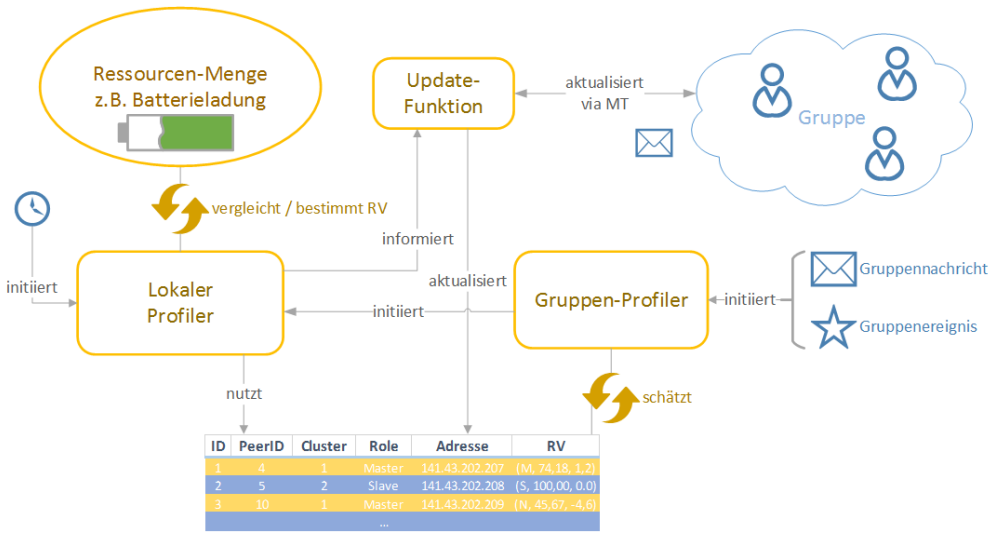


Abbildung 8.9: Prinzipieller Aufbau des Ressourcenprofilerdienstes

Der *Gruppenprofiler* schätzt lokal die Ressourcensituation der Gruppe ein. Die dazu notwendigen Peer-Eigenschaften werden zunächst initial bei der Gruppenverwaltung abgefragt und als Peer-Ressourcenwerttabelle gespeichert. Die Schätzung wird jeweils nach einem erfolgreichen Gruppenein- oder -austritt oder bei der lokalen Zustellung einer Gruppennachricht durchgeführt. Die Schätzung erfolgt auf Basis der Ressourcenwerttabelle, indem vom gespeicherten Ressourcenfunktionswert die Konsumrate subtrahiert wird. Eine Entladung des Akkus ist als eine positive Konsumrate modelliert, der Ladevorgang mit einer negativen Rate. Da alle fehlerfreien Peers innerhalb Gruppe die gleichen Nachrichten zu den gleichen logischen Zeitpunkten zustellen, ist die lokale Schätzung ohne zusätzliche Kommunikation gruppenweit synchronisiert.

Der *lokale Profiler* gleicht die gruppenweite Schätzung für den lokalen Peer mit dem tatsächlichen Ressourcenfunktionswert ab. Dazu werden dieser und die Konsumrate für die Periode der letzten Ausführung bestimmt. Ebenso wird geprüft, ob sich der Mobilitätstyp geändert hat. Bei Änderungen des Mobilitätstyps bzw. einer signifikanten Abweichung des tatsächlichen Ressourcenfunktionswertes vom geschätzten wird der aktuelle Ressourcenwert an die Updatefunktion des Dienstes weitergegeben. Der lokale Profiler wird entweder vom Gruppenprofiler oder periodisch über einen Timer initiiert.

Die *Updatefunktion* aktualisiert den Ressourcenwert des lokalen Peers im globalen Wissen. Damit das Update bei allen Empfängern in der gleichen Sicht verarbeitet wird, gilt für das Update die Same-View-Delivery-Eigenschaft. Der Versand kann grundsätzlich in zwei Varianten ausgeführt werden:

1. jeder Peer aktualisiert eigenständig seinen Ressourcenwert im globalen Wissen oder

2. der Master aktualisiert für sich und seine Slaves die Ressourcenwerte im globalen Wissen.

Variante (1) ist einfach umsetzbar, da die lokalen Ressourcenwerte direkt in der Gruppe verteilt werden können. Der Kommunikationsaufwand steigt jedoch mit der Gruppengröße. Da die Schätzung isochron über die Zustellung einer Gruppennachricht ausgelöst wird, kann dies im ungünstigsten Fall zu einem parallelen Update durch alle Gruppenteilnehmer führen. Löst ein Update einen Masterwechsel aus, würden durch die Installation der neuen Sicht alle weiteren Updates ungültig und müssten erneut von den Gruppenteilnehmern versendet werden.

Bei Variante (2) übermittelt der Slave das Update zunächst unzuverlässig an seinen Master und wartet über eine gewisse Zeit auf die Aktualisierung des globalen Wissens. Der Master sammelt eingehende Updates von seinen Slaves oder seinem lokalen Profiler

und versendet diese als Clusterupdate über den total geordneten Multicastdienst. Dadurch wird die Zahl der parallelen Multicastübertragungen im Vergleich zu Variante (1) deutlich gesenkt. Es verlängert sich jedoch die Zeitspanne, bis Ressourcenwertupdates global wirksam werden. Aufgrund des geringeren Kommunikationsaufwandes wurde diese Variante realisiert.

Algorithmus 4 Die Funktion *sendUpdate*

```
 1: function sendUpdate(peerResources)
 2:     if getLocalState() == JOINED then
 3:         if isLocalPeerMaster() then
 4:             resourceMap[getLocalID()] ← peerResources
 5:             sendResourceAnnounce()
 6:         end if
 7:     else
 8:         now ← GenericTime :: currentTime()
 9:         if now > (updateSendTime + UPDATE_THRESHOLD) then
10:             ru.sourceID ← getLocalID()
11:             ru.resource ← peerResources
12:             sendTo(ru, getLocalMasterID())
13:         end if
14:     end if
15: end function
```

Algorithmus 4 stellt das Prinzip der Updatefunktion dar. Die Slaves senden per *Resource-Update-Nachricht* (RU) ihren aktuellen Ressourcenwert an den Master (Zeile 14). Der Versand weiterer Nachrichten wird für die Zeitspanne von $UPDATE\_THRESHOLD$-Sekunden unterdrückt. Der Master aktualisiert mit dem empfangen Wert seine lokale Ressourcenwerttabelle. Bei einem Unterschied von mehr als $MAX\_DELTA\_RESOURCES$ des aktuellen Master-Ressourcenfunktionswertes zu dem Wert im globalen Wissen wird eine *Resource-Value-Announce-Nachricht* (RVA) total geordnete an die Gruppe versendet. Die RVA-Nachricht umfasst die aktuellen Ressourcenwerte des sendenden Masters und seiner Slaves.

8.4.5 Proaktiver Rollenwechsel

Für einen Master kann es wünschenswert sein, seine Rolle proaktiv zu wechseln. Beispiel: Ein nomadischer Peer wechselt von einer stationären Phase in eine mobile und der aktuelle Ladezustand seines Akkus ist ungenügend. Solch eine Situation kann durch ein Mobility Management Framework (vgl. Kapitel 2) erkannt werden.

Für einen proaktiven Rollenwechsel sendet der Master eine so genannte *Role-Switch-Announce-Nachricht* (RSA) mit seiner Peer-ID total geordnet an die Gruppe. Dabei müssen die Sendesicht- und Gruppensichtzustellungseigenschaft erfüllt sein. Wird die RSA-Nachricht lokal zugestellt, so erfolgt der Rollenwechsel gemäß Algorithmus 5. Dieser prüft zunächst, ob der Peer – identifiziert über die ID aus der RSA-Nachricht – aktuell tatsächlich ein Master ist. Ist dies der Fall, wird das aktuelle Gruppenregister ausgelesen und für den Cluster des betreffenden Peers ein neuer Master unter Nutzung der Rollenwechselzeitpunkt-Metriken bestimmt. War dies erfolgreich, wird der Rollenwechsel durchgeführt, anderenfalls bleibt die Topologie unverändert.

8.5 Synchronisationsprobleme

Eine Wartungsoperation kann sowohl die Rolle als auch die Clusterzuordnung der Peers ändern. Im Rahmen der Wartung ist es daher wichtig, dass die Eigenschaften des Gruppenkommunikationssystems gesichert werden (vgl. Abschnitt 4). Im Besonderen heißt dies, dass alle fehlerfreien Peers der Gruppe die Wartung synchron und mit dem gleichen Ergebnis durchführen. Ein abweichendes Wartungsergebnis bei einem Teil der Peers würde die Gruppe aufgrund der unterschiedlichen Sichten partitionieren.

Algorithmus 5 Die Funktion $forceRoleSwitch$

```
1:  function forceRoleSwitch(masterId)
2:      masterIDList ← getMembershipService().getMasterIDList()
3:      if masterIDList.contains(masterId) == FALSE then return
4:      end if
5:      memberRegister ← getMemberRegister()
6:      cID ← memberRegister.findPeer(masterId)
7:      newMasterID ← memberRegister.determinePossibleNewMaster(cID)
8:      if newMasterID != masterId then
9:          memberRegister.selectMaster(cID, newMasterID)
10:         refreshMetric()
11:     end if
12: end function
```

Aufgrund der Übertragungslatenz können sich jedoch auch die Sichten fehlerfreier Peers für kurze Zeitabschnitte unterschieden. In [203] werden drei wesentliche Fehlerquellen identifiziert:

- Empfang von Nachrichten aus zukünftigen Sichten,
- Empfang von Nachrichten aus vergangenen Sichten und
- parallele Sichtänderungen.

Alle drei Fehlerquellen können die Konsistenz der Gruppe zerstören, wenn ihnen nicht mit aktiven Maßnahmen begegnet wird. Die damit zusammenhängenden Synchronisationsprobleme werden in [203] und [95] ausführlich diskutiert. Details dazu werden auch in Anhang D.2 gegeben.

8.5.1 Nachrichten aus zukünftigen Sichten

Zur Verdeutlichung sollen die Synchronisationsprobleme an einer Gruppe mit zwei Peers p_1 und p_2 gezeigt werden, wobei p_1 der Master von p_2 ist. Angenommen, eine durch p_1 versendete Nachricht löst einen Rollenwechsel aus. Ist die globale GT-Nachricht (siehe Kapitel 5) dieser Nachricht für p_2 verzögert, so kann p_1 bereits in eine neue Sicht und eine neue Rolle wechseln, obwohl p_2 noch in der alten Sicht verbleibt. Eine direkt im Anschluss versendete weitere Nachricht von p_1 wird durch p_2 solange nicht akzeptiert, bis p_2 seine Sicht aktualisiert. Bei ungenügenden Netzbedingungen kann aufgrund von Paketverlusten die Synchronisation der Gruppe zerstört werden. Diese Gefahr kann wie folgt beseitigt werden:

1. Vergrößerung der Timeoutwerte und der Anzahl der erlaubten Wiederholungen für jede Nachrichtenübertragung oder
2. Puffern von Nachrichten aus zukünftigen Sichten während des Sichtwechsels.

Ansatz (1) erhöht den Kommunikationsaufwand sowie den Ressourcenverbrauch der Peers. Ebenso steigt das Risiko, dass der Fehlerdetektor eine falsche Annahme über den Zustand des Peers trifft. Daher ist Ansatz (2) zu präferieren. Nachrichten aus zukünftigen Sichten werden im so genannten Next-View-Buffer gespeichert. Wechselt ein Peer in eine neue Sicht, so wird geprüft, ob bereits Nachrichten für diese Sicht empfangen wurden. Ist dies der Fall, so werden diese ausgelesen und der gleichen Behandlung wie regulär empfangene Nachrichten zugeführt. Über dieses Konzept kann der Nutzer ohne Unterbrechung Nachrichten an die Gruppe senden und von dieser empfangen. Deren Auslieferung wird jedoch verzögert, bis der letzte Peer der Gruppe in die aktuelle Sicht gewechselt ist.

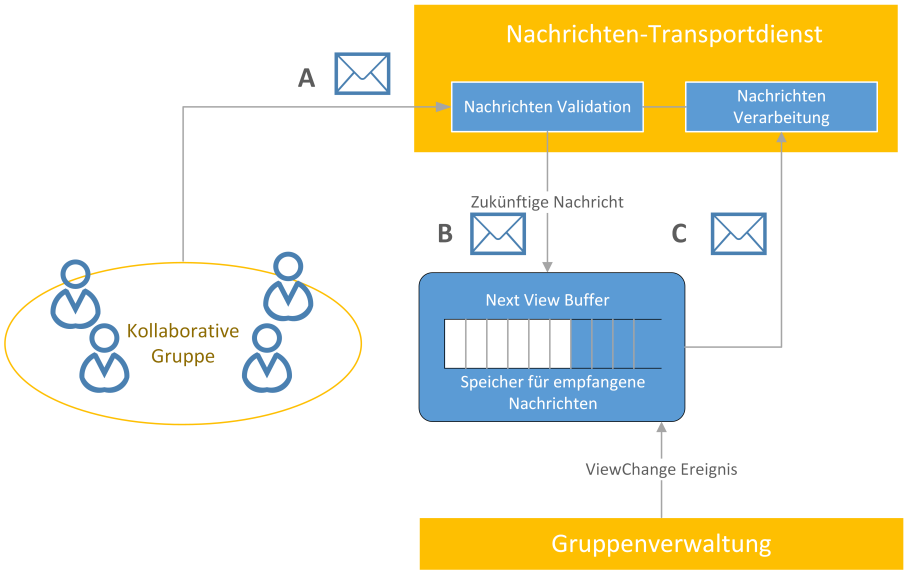


Abbildung 8.10: Funktionsprinzip NVB bei Nachrichten aus zukünftigen Sichten

8.5.2 Nachrichten aus vergangenen Sichten

Ein weiteres Synchronisationsproblem entsteht, wenn p_2 in einer alten Sicht eine Nachricht an p_1 sendet, während p_1 aufgrund einer Wartungsoperation in eine neue Sicht gewechselt ist. Diese Nachricht wird durch p_1 verworfen, da sie zu einer vergangenen Sicht gehört. Dadurch wird die Nachrichtenverarbeitung verzögert und es werden eine Reihe von Sendewiederholungen ausgelöst, da p_2 eine Bestätigung für die Nachricht benötigt. Durch die Verzögerung in der Nachrichtenverarbeitung kann p_2 nicht in die neue Sicht wechseln. In Kombination mit anderen, parallelen Gruppenoperationen und Übertragungslatenzeffekten kann auch hier die Gruppensynchronisation zerstört werden.

Um dieses Problem zu beseitigen, muss der Versand von Nachrichten durch den lokalen Peer solange verzögert werden, wie Nachrichten verarbeitet werden, die einen Sichtwechsel auslösen können. Ob eine Nachricht potentiell einen Sichtwechsel auslöst, kann über die Eigenschaft *isViewChangeable()* der Gruppennachrichten abgefragt werden. Wurden die potentiell sichtändernden Nachrichten verarbeitet, können die verzögerten Nachrichten versendet werden. Die Anwendung wird während des Versendens und Empfangens von Nachrichten durch die Wartung nicht blockiert. Die Optimierung resultiert nur in einer erhöhten Latenz der Übertragung der Nachrichten, die während einer Wartung ausgetauscht werden.

8.5.3 Next-View-Buffer

Nachrichten, welche durch einen Peer empfangen werden, jedoch den Sichtbedingungen nicht entsprechen, sind zur Sicherung der MOVS-Eigenschaften zu verwerfen. Roman *et al.* [184] schlagen vor, empfangene Nachrichten aus zukünftigen Sichten zu speichern, statt diese zu verwerfen. Dadurch können diese Nachrichten nach einem Sichtwechsel lokal verzögert zugestellt werden.

Für die Implementierung des dazu notwendigen Zwischenspeichers wird ein sogenannter *Next-View-Buffer* (NVB) genutzt, welcher Teil des Multicastdienstes ist. Das Prinzip ist in Abbildung 8.10 dargestellt. Die empfangenen Nachrichten aus den zukünftigen Sichten (A) werden nach der Prüfung entsprechend ihrer Sicht im NVB gespeichert und je Sicht wiederum in FIFO-Ordnung. Die verwendeten Prüfbedingungen werden weiter unten diskutiert. Verkündet die Gruppenverwaltung die Einrichtung einer neuen Sicht V' (B) (über ein *ViewChange*-Event), werden alle im NVB zur Sicht V' gespeicherten Nachrichten ausgelesen und wie reguläre empfangene Nachrichten verarbeitet (C).

Um das Problem der Nachrichten aus vergangenen Sichten zu lösen und auf eine Erweiterung des Sichtenmodells vergleichbar dem Ansatz der *Erweiterte Virtuelle Syn-*

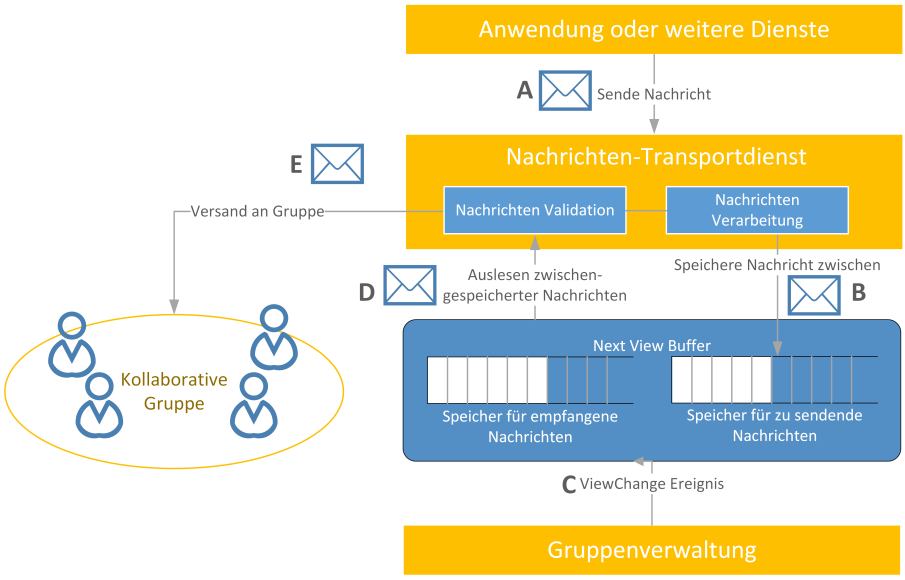


Abbildung 8.11: Speicherung zu versendender Nachrichten im Zustand *FLUSHING*

chronität (EVS) verzichten zu können, ist der Sichtwechsel so lange zu verschieben, bis alle anstehenden Nachrichten ausgeliefert wurden. Dazu werden neue Nachrichten verzögert und zwischengespeichert. Der Multicastdienst prüft die zu versendende Nachricht, ob diese sichtändernd sein könnte. Ist dies der Fall, wird in den *FLUSHING*-Zustand gewechselt (siehe Abschnitt 5). Nachfolgend zu versendende Nachrichten werden solange zwischengespeichert, bis die Warteschlange des Multicastdienstes geleert wurde.

Dafür wurde der Next-View-Buffer, wie in Abbildung 8.11 dargestellt, um einen zweiten Puffer erweitert. Soll eine Nachricht während des *FLUSHING*-Zustandes versendet werden (A), wird zunächst die Sendebedingung durch den Multicastdienst validiert. Möglicherweise sichtändernde Nachrichten werden in die NVB-Sendewarteschlange eingereiht (B) und nicht direkt zugestellt. Sobald die Gruppenverwaltung den erfolgreichen Sichtwechsel signalisiert (C), werden die gespeicherten Nachrichten ausgelesen (D) und an die Gruppe versandt (E).

Dieser Vorgang ist für den Dienstnutzer vollständig transparent, da die Entscheidung über eine Einlagerung oder den zeitnahen Versand durch den Multicastdienst getroffen wird. Sie ist jedoch mit einer erhöhten Nachrichtenlatenz verbunden.

Für die korrekte Verarbeitung parallel versandter, potentiell sichtändernde Nachrichten im *FLUSHING*-Zustand musste das Auslieferungsverhalten des Multicastdienstes modifiziert werden (abgebildet im Algorithmus 6). Im Zustand *FLUSHING* wird vor der Auslieferung einer sichtändernden Nachricht geprüft, ob alle nachfolgenden Nachrichten ebenfalls ausgeliefert werden können. Anderenfalls bleibt der Dienst im Zustand *FLUSHING* und verzögert die Auslieferung der Nachrichten.

8.6 Test und Leistungsanalyse

Um die Anwendbarkeit des Ansatzes zu beweisen, wurde das *Moversight*-Protokoll in [95] getestet, erneut mithilfe des OMNeT++ [185] Frameworks. Das Testprinzip war inspiriert durch Abbasi und Younis [3]. Es legt den Schwerpunkt auf die Fehlerfreiheit bzw. die Korrektheit der Lösung und die maximal erreichbare Anwendungslaufzeit der Gesamtgruppe. Die Leistungsanalyse in [95] bestand aus 250 Testfällen, wobei in jeden Testfall eine definierte Gruppe erstellt und der Testfall in einem simulierten drahtlosen Netz ausgeführt wurde.

Der vorgestellte Ansatz wurde dabei mit dem Prinzip des LEACH-Protokolls verglichen. Das LEACH-Protokoll wurde gewählt, da es – aus Sicht des Autors – das

Algorithmus 6 Die Funktion *sortAndDeliver*

```
 1: function sortAndDeliver
 2:
 3:     queue.sort()
 4:     while queue.size() > 0 AND queue.front().isDeliverable() do
 5:         pdu ← queue.front().getMessage()
 6:         if getLocalSubState() == FLUSHING then
 7:             if pdu.isViewChangeable() then
 8:                 for i < queue.size() do
 9:                     if queue.get(i).isDeliverable() == FALSE then return
10:                     end if
11:                 end for
12:             end if
13:         end if
14:         deliverMessage(pdu)
15:         dequeue(pdu);
16:     end while
17: end function
```

bedeutendste Clusterprotokoll ist, dessen Funktionsprinzip in reinen P2P-Netzen mit geschlossenen Gruppen und globalem Wissen angewendet werden kann.

Das für den Test simulierte Netz bestand aus bis zu 20 Zugangspunkten, welche untereinander über ein Ethernet-Netz verbunden waren. Die simulierten Endgeräte kommunizierten drahtlos und nutzten das *Moversight*-Protokoll oberhalb von UDP/IP. Sie wurden unter Nutzung des DHCP-Protokolls [71] konfiguriert. Die Endgeräte waren mit einer Batterie mit einer nominalen Kapazität von 10 kJ ausgestattet. Der drahtlose Teil des Netzes wurde unter Nutzung des OMNeT++ INET Frameworks definiert und simulierte ein IEEE 802.11g-Netz mit 54 MB/s bei 2,4 GHz. Der Energieverbrauch der drahtlosen Netzschnittstelle ist charakterisiert durch folgende Einstellungen:

Off	=	0,0 W
Sleep	=	0,0495 W
Idle	=	0,6696 W
Reception	=	1,0791 W
Transmission	=	1,7787 W

Die Tests durchliefen zwei Phasen, in denen jeweils die Dynamische Clustering-Strategie und der Rollen-Wartungsdienst getestet wurden. In jedem Test wurde die Gruppengröße von zehn bis einhundert Peers (jeweils in Zehnerschritten) in Kombination mit insgesamt 23 verschiedenen Wartungsstrategien simuliert. Die einzelnen Wartungsstrategien setzten sich jeweils wie folgt zusammen:

- einer Peer-Platzierungsstrategie (einer sequentiellen, einer parallelen und der DCS Strategie),
- sechs DCS-Metriken
- sieben Rollenwechselzeitpunkt-Metriken, welche den Rollenwechsel (gruppenweit, je Master, innerhalb eines Clusters) in Abhängigkeit von den versendeten Nachrichten, der verstrichenen virtuellen Zeit und dem Ressourcenwert (schwellwertbasierte Metrik, schwellwertbasierte Prüfung der Ressourcen aller Master) entweder nie durchführen oder diesen einleiten.

Details zu diesen Wartungsstrategien sind in [95] ausgeführt. Um die Leistungsfähigkeit der Lösung nachzuweisen, wurde diese mit zwei Clusterprotokollen verglichen, welche eine Round-Robin-Optimierung der Clustertopologie entsprechend des Prinzips von LEACH [112] umsetzen. Die LEACH-Umsetzung *L-S* verwendet eine statische Anzahl von Mastern, die Variante *L-D* eine dynamische Anzahl von Mastern und Clustern,

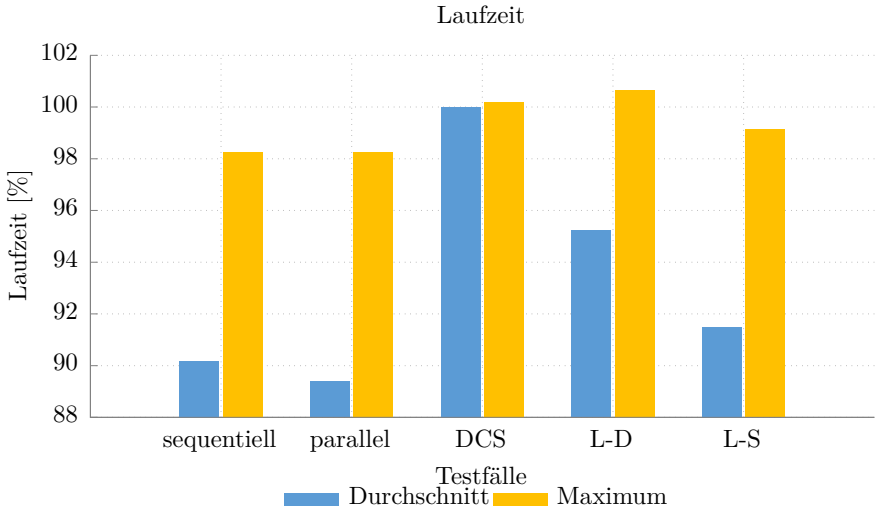


Abbildung 8.12: Ergebnisse der Testläufe, entsprechend der verwendeten Platzierungsstrategie kumuliert

wobei bei L-D die Masterrolle ebenfalls mittels einer Round-Robin-Strategie innerhalb des Clusters verteilt wird. Diese beiden Varianten entsprechen dem Prinzip der Dynamische Clustering-Strategie. Zu berücksichtigen ist dabei, dass die Leistungsfähigkeit von LEACH direkt von der zuvor fest definierten Anzahl von Clustern abhängt. Ist die Anzahl von definierten Mastern nicht für den Anwendungsfall passend, ist anzunehmen, dass die Leistungsfähigkeit von LEACH stark sinkt. Insgesamt wurden bei der Leistungsanalyse über 250 Testfälle untersucht. Für jeden Test wurde die Simulationszeit gemessen.

Die Ergebnisse zeigen, dass die relative maximale Laufzeit der einzelnen Testläufe um 10,7 % variiert und die Bandbreite der relativen durchschnittliche Laufzeit zwischen 96,92 % und 104 % liegt. Insgesamt waren die Ergebnisse des DCS-Ansatzes die stabilsten, d. h. die maximale Laufzeit aller Tests war ähnlich der durchschnittlichen Laufzeit aller Tests dieser Parameterkombination. Somit wurden mit der gemischten DCS-Metrik die besten Ergebnisse erzielt. Die instabilsten Ergebnisse lieferten die sequentielle sowie die parallele Peer-Platzierungsstrategien. Die Ergebnisse der LEACH-Varianten L-D und L-S waren ebenfalls (mit einer Schwankungsbreite von 5 % beziehungsweise 8 % um den Mittelwert) instabil.

Abbildung 8.12 zeigt die kumulierten Ergebnisse der Tests für die durchschnittliche und die maximale Laufzeit der sequentiellen, der parallelen und der DCS Strategie sowie der LEACH-artigen Varianten L-D und L-S im Detail:

1. Die durchschnittliche Laufzeit der DCS-Strategie übersteigt die Ergebnisse der sequentiellen und der parallelen Strategie im Durchschnitt um ca. 2 %.

2. Die durchschnittlichen Laufzeiten von L-D und L-S sind vergleichbar mit denen der DCS-Strategie.

3. Die maximale Laufzeit des DCS-Ansatzes ist um 10 % besser als die beste der sequentiellen Strategie, 11 % besser als die beste der parallelen Strategie, 5 % besser als die maximale Laufzeit von L-D und 9 % besser als die maximale Laufzeit von L-S.

Bei der Untersuchung der Laufzeit entsprechend der verwendeten Platzierungsmetriken zeigte die gemischte Metrik die besten Ergebnisse vor der Ressourcenwert-Metrik (Abbildung 8.13). Die durchschnittliche Laufzeit von L-D ist vergleichbar mit der gemischten Metrik, begründet durch das gleiche Clustererzeugungsschema und 3 % besser als die durchschnittliche Laufzeit von L-S.

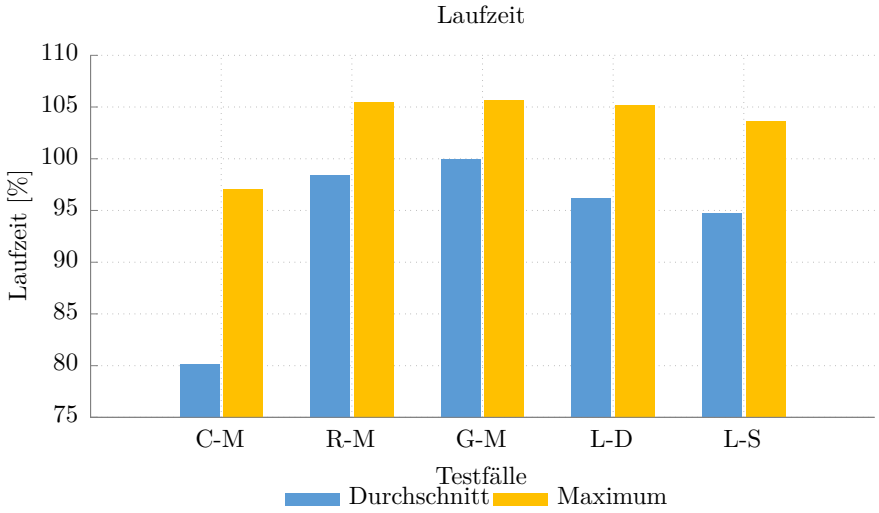


Abbildung 8.13: Ergebnisse der Testläufe, entsprechend der verwendeten Platzierungsmetriken kumuliert

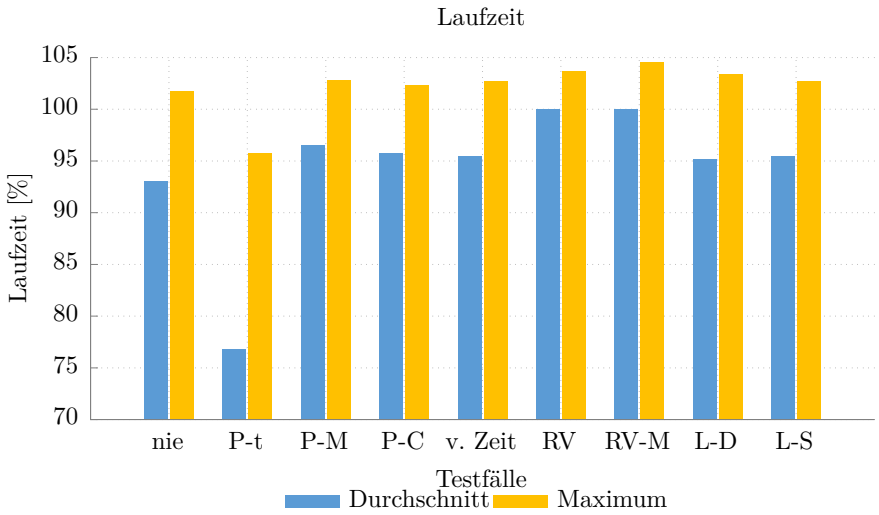


Abbildung 8.14: Ergebnisse der Testläufe, entsprechend der Rollenwechselzeitpunkt-Metriken kumuliert

Die kumulierten Ergebnisse der Rollenwechselzeitpunkts-Metriken in Abbildung 8.14 zeigen folgendes:

1. Bei den Rollenwechselzeitpunkts-Metriken wird die geringste Laufzeit von der Rollenwechselzeitpunkts-Metrik erreicht, welche nie einen Rollenwechsel auslöst, gefolgt von der verstrichenen virtuellen Zeit-Metrik.

2. Die beste Laufzeit erreichte die Ressourcenwert-Metrik, welche alle Master prüft.

3. Die geringste durchschnittliche Laufzeit wurde von der RMS-Metrik erreicht, welche den Rollenwechsel über die Anzahl der versendeten Nachrichten (PDUs) in der Gruppe steuert.

Kapitel 9

Teilung und Vereinigung von Gruppen – Die Dienste Split und Merge

Gegenstand dieses Kapitels ist die Teilung und Vereinigung von Gruppen aus Anwendersicht. Die Operationen *SPLIT* – die aktive Aufteilung der kollaborativen Gruppe in zwei unabhängige Teilgruppen – und *MERGE* – die (Wieder-) Vereinigung mehrerer Gruppen zu einer gemeinsamen Gruppe – sind im Kontext der Gruppenkommunikationsprotokolle nicht eindeutig definiert. In [98] wird eine umfassende Klassifizierung beider Operationen in Gruppenkommunikationsprotokollen und P2P-Systemen gegeben.

Mit mobilen kollaborativen Anwendungen sollen Entscheidungsprozesse der Gruppenmitglieder abgebildet werden. Bei diesen Prozessen besteht die Möglichkeit, dass sich die kooperierende Gruppe aufteilt, um Teilprobleme unabhängig voneinander zu bearbeiten. Später können sich die Teilgruppen wieder in einer gemeinsamen Gruppe zusammenfinden und die Ergebnisse austauschen. Es ist ebenso möglich, dass eine externe Expertengruppe an einer Entscheidung beteiligt werden soll. Für das Roboterbeispiel aus Kapitel 2.2 kann dies beispielsweise bedeuten, dass ein Teil der Gruppe den nördlichen Teil eines Gebiets untersucht, der andere den südlichen. Für diese unabhängigen Teilaufgaben ist es nicht notwendig, das diesbezügliche Wissen in der gesamten Gruppe konsistent zu halten. Mit der Reduktion der Kommunikationspartner können durch eine Aufteilung zusätzlich Ressourcen gespart werden.

In der Literatur wird bzgl. der Teilung und Vereinigung von Gruppen auch von Partitionierung gesprochen. Zumeist wird die Partitionierung auf der Netz- und/oder auf der Anwendungsebene betrachtet. Eine Partitionierung der Gruppe aus Anwendungssicht bedeutet, dass die geschlossene Gruppe als Einheit gesehen wird. Diese ist semantisch von der Netzpartitionierung zu unterscheiden, welche die Gruppe zumeist aus Sicht des Netzes und dessen Elementen betrachtet. Hier reagieren die Systeme auf Fehler in der Netzschicht (zumeist Verbindungsunterbrechungen).

Die beiden in dieser Arbeit vorgestellten Dienste sind gekennzeichnet durch eine anwendungsgetriebene Auslösung. Die Ausführung der Operationen ist also keine Reaktion auf Fehlerzustände bzw. Änderungen auf der Netzebene, sondern davon unabhängig. Operationsgegenstand ist die Gruppe mit ihren Gruppenverwaltungsdaten (Sichten, Gruppenmitglieder, Clusterstruktur) und deren globales Wissen. Die dabei erforderliche Synchronisation sichert das gemeinsame Wissen. Damit wird erreicht, dass alle Gruppenmitglieder den gleichen Kenntnisstand besitzen und keine Nachrichten aus vergangenen Gruppenzusammensetzungen zugestellt werden. Vereinfacht gesagt: Das Wissen der Gruppe sowie die Eigenschaften ihrer Teilnehmer sollen erhalten bleiben (im Sinne von: ein privilegierter Teilnehmer bleibt auch nach einer Aufspaltung privilegiert). Bei der Netzpartition steht die Sicherung der Nachrichtenübertragung im Fehlerfall im Mittelpunkt und soll beispielsweise folgende Fragen beantworten:

- Welche Übertragung wurde noch erfolgreich abgeschlossen?
- Wie lautet die neue Transportadresse des Peers?

Somit dient die Synchronisation hier eher der Festlegung eines wohldefinierten Startzustands, über den wieder eine fehlerfreie Kommunikation möglich ist. Während des Fehlerzustands kann es aber unter Umständen zu Verlusten im globalen Wissen kommen.

9.1 Das CAP-Theorem im Zusammenhang mit Split und Merge

In der Ausgangssituation sind die zu teilenden oder zu vereinigenden Gruppen nicht partitioniert. Die Aufspaltung oder Zusammenführung der Gruppen beschreibt nun jeweils eine Situation, welche in Abschnitt 6.1 mit dem CAP-Theorem illustriert ist. Dieses Theorem befasst sich mit der Vereinbarkeit der Eigenschaften Verfügbarkeit (A) und Konsistenz (C) im Falle einer Partitionierung (P). Man kann also sagen: *SPLIT* erzeugt eine Partitionierung und *MERGE* löst diese auf. Die Gruppe(n) wechseln von einem nichtpartitionierten Zustand in einen partitionierten und wieder zurück. Die Aufrechterhaltung des globalen Wissens (C) in dieser Transition ist die wichtigste Anforderung an die beiden Dienste. Während der Operation wird die Verfügbarkeit (A) der Gruppe eingeschränkt.

Teilen von Gruppen Der *Splitdienst* spaltet eine Teilmenge der aktuellen Gruppenmitglieder von der Gruppe ab, die fortan unabhängig von der Restgruppe existiert. Bestehende Gruppenkommunikationssysteme [9, 10, 154, 231], bieten keine vergleichbare Funktion. Bei diesen Systemen müssen die Teilnehmer aus der ursprünglichen Gruppe explizit austreten und anschließend einer neuen Gruppe beitreten. Das bisherige gemeinsame Wissen geht dabei für diese Mitglieder verloren. Das Funktionsprinzip des Splitdienstes ist in Abbildung 9.1 dargestellt. Die Operation wird durch eine aktive Entscheidung der Gruppe ausgelöst. Auf der rechten Seite der Abbildung ist eine Gruppe mit der Sicht V abgebildet, welche in zwei Gruppen mit den Sichten V' und V'' aufgespalten werden soll. Auf der linken Seite ist die TOM-Warteschlange eines Peers vor und nach der Operation dargestellt, welcher der Gruppe mit Sicht V angehört. Die getrennten Gruppen sind im Gegensatz zu einer Netzpartition uneingeschränkt arbeitsfähig und verfügbar.

Damit sich das gemeinsame Wissen konsistent in den getrennten Gruppen weiterentwickeln kann, müssen die Warteschlangen der Transferdienste bei der Gruppenaufspaltung ebenso angepasst werden. In der Abbildung wurde die Gruppe ohne solche Anpassungen aufgespalten, wodurch sich noch Nachrichten aus Sicht V in der Warteschlange befinden. Dieser Ansatz zur Aufteilung der Gruppe zieht die CAP-Eigenschaft Verfügbarkeit und Partitionstoleranz der Konsistenz vor. Die Inkonsistenz kann durch Paketverluste vor der Ausführung der Aufspaltung entstehen. Wurde eine Nachricht beispielsweise von einem Master versendet, welcher sich nicht in der abgespaltenen Teilgruppe befindet und gingen zugehörige GT-Nachrichten (globale Bestätigungen im TOM-Transfer) verloren, so kann diese Nachricht nicht mehr zugestellt werden. Dadurch wäre das globale Wissen in der Gruppe V' inkonsistent.

Ein nachträglicher Abgleich der Warteschlangen zwischen allen Peers der Gruppe V' könnte die Konsistenz wiederherstellen. Abgleich bezeichnet hier den gegenseitigen Austausch aller in der Warteschlange befindlichen Nachrichten, was einen hohen Kommunikationsaufwand bedeutet und die Verfügbarkeit einschränkt. Ziel des Splitdienstes ist es daher, Inkonsistenzen zu verhindern ohne dass die Warteschlangen der Peers abgeglichen werden müssen. Wird beispielsweise die Robotergruppe aus dem Anwendungsszenarium 2.2 geteilt, reduziert sich in den beiden entstehenden Teilgruppen die Anzahl der Peers, zwischen denen der TOM-Transfer die Konsistenz (C) sichern muss. Damit reduziert sich für die jeweiligen Peers der Übertragungsaufwand und verbessert deren Ressourcensituation.

Vereinigen von Gruppen Für den *Mergedienst* gelten ähnliche Synchronisationsprobleme wie für den Splitdienst. Vereinzelte Ansätze wie [188, 198, 222] bieten ähnliche Funktionen. Mit ihnen werden Teilgruppen über Brückenverbindungen (im Sinne von

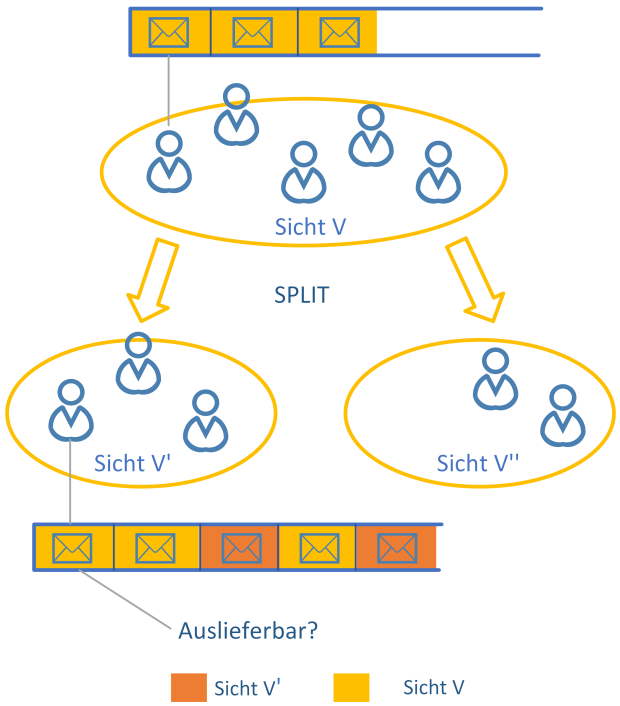


Abbildung 9.1: Aufspaltung der Gruppe in zwei unabhängige Teilgruppen

Bridging oder *Proxy*-Peers) lose gekoppelt, ohne dass dabei eine gemeinsame Gesamtgruppe entsteht. Diese Ansätze synchronisieren im Allgemeinen nur die Sichten der Gruppenteilnehmer. Bei der Zusammenführung zweier Gruppen ist die Behandlung des ehemals getrennten gemeinsamen Wissens sowie eine einheitliche Benennung bzw. Identifizierung der Gruppenmitglieder von zentraler Bedeutung. Ein Verzicht auf Synchronisationsmaßnahmen führt auch hier zu einem inkonsistenten globalen Wissen.

Abbildung 9.2 zeigt das Funktionsprinzip des Mergedienstes. Wie in der vorigen Abbildung sind auf der rechten Seite die einzelnen Gruppen dargestellt, auf der linken die Warteschlangen ausgewählter Peers. In diesem Beispiel werden die Gruppen mit den Sichten V' und V'' zu einer Gruppe mit der Sicht V vereinigt. Die Peers der Ausgangsgruppen haben jeweils nur Nachrichten aus ihrer aktuellen Sicht in der Warteschlange des TOM-Transfer. Werden die Gruppen ohne weiterführende Synchronisationsmaßnahmen vereinigt, so ergibt dies einen inkonsistenten Zustand der Warteschlangen, wie für die Gruppe mit Sicht V dargestellt. Ob dieser Zustand innerhalb der Anwendung gültig ist, hängt vom konkreten Anwendungskontext ab. Soll diese Situation verhindert werden, müssen die Warteschlangen beider Gruppen vereinheitlicht werden. Im folgenden Abschnitt werden verschiedene Ansätze dafür besprochen. Allen ist gemein, dass sie die Verfügbarkeit (A) der Gruppe(n) während des Abgleichs der Warteschlangen teilweise einschränken. Damit werden auch beim *MERGE* die CAP-Eigenschaften Konsistenz (C) und Partitionstoleranz (P) der Verfügbarkeit (A) vorgezogen. Im Kontext der mobilen Kommunikation ist es jedoch wichtig, dass die Operationsdauer und damit die Einschränkung der Verfügbarkeit gering gehalten werden, da es jederzeit zu Verbindungsabbrüchen kommen kann. Im Roboterbeispiel wechseln die beiden Teilgruppen (um sich zu vereinigen) in einen Zustand, der die Verfügbarkeit der beiden Teilgruppen einschränkt. Dadurch können beispielsweise keine weiteren Roboter in die Gruppe eintreten und der Versand neuer Nachrichten wird verzögert.

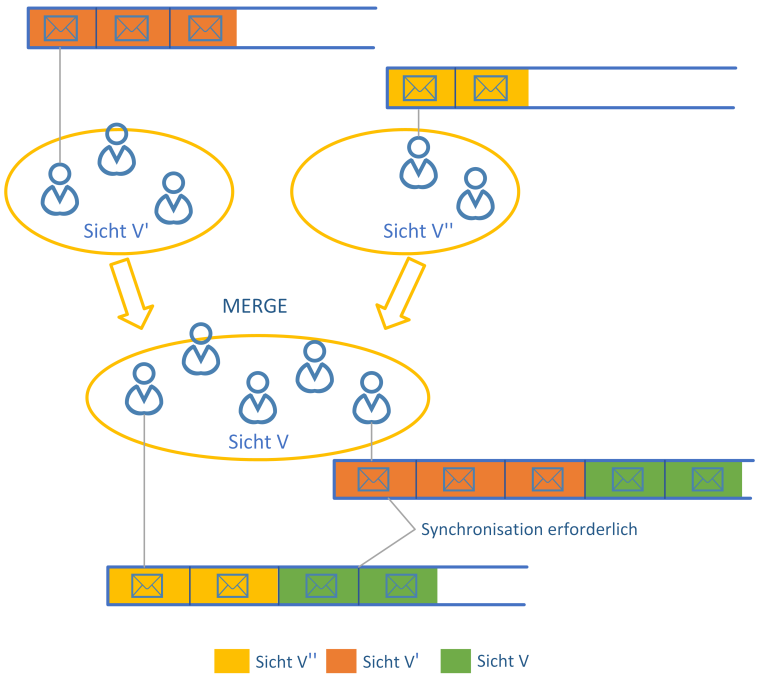


Abbildung 9.2: Vereinigung zweier bis dato unabhängiger Gruppen ohne weiterführende Synchronisation

9.2 Synchronisationsansätze im Kontext einer *SPLIT*- oder *MERGE*-Operation

Da *SPLIT* und *MERGE* sichtändernde Operation sind, muss sichergestellt werden, dass das Paradigma der mobilen optimistischen virtuellen Synchronität durch die beiden Operationen nicht verletzt wird. Daher muss nicht nur die Gruppenzusammensetzung geändert, sondern auch die Warteschlange des TOM-Transfer angepasst werden. Im Besonderen geht es dabei um die Frage, wie Nachrichten aus einer vergangenen Sicht behandelt werden.

Bei direkter Anwendung der Sendesichtzustellung würden Nachrichten aus einer vergangenen Sicht verworfen werden. Dieses Verhalten kann vergleichbar mit den Synchronisationsproblemen in Kapitel 8 die Reaktionsfähigkeit des Protokolls deutlich einschränken und zu einem inkonsistenten gemeinsamen Wissen führen. Auch bei Verwendung der Gruppensichtzustellung können Inkonsistenzen entstehen. Daher ist eine Synchronisation zwischen den Teilnehmern notwendig, bevor eine Vereinigung oder Trennung durchgeführt wird. Für die mögliche Synchronisation der betroffenen Peers sind drei Ansätze möglich [98]:

- Zwischenspeichern und Abgleichen,
- Leeren der Warteschlangen, und
- Partielle Sichten.

Die Ansätze basieren zumeist auf Konzepten zur Behandlung von Netzpartitionen und wurden auf die Bedürfnisse der Dienste Split und Merge angepasst.

Zwischenspeichern und Abgleichen In [162] werden Nachrichten während einer Netzpartition zwischengespeichert, ebenso in [156] und [37]. Ersterer führt die Teilgruppen nach einer Netzpartitionierung zusammen. Dazu werden die zu versendenden Nachrichten in den geteilten Komponenten als Transaktion zusammengefasst, per Zustandstransfer nach der Gruppenvereinigung ausgetauscht und der Zustand der einzelnen

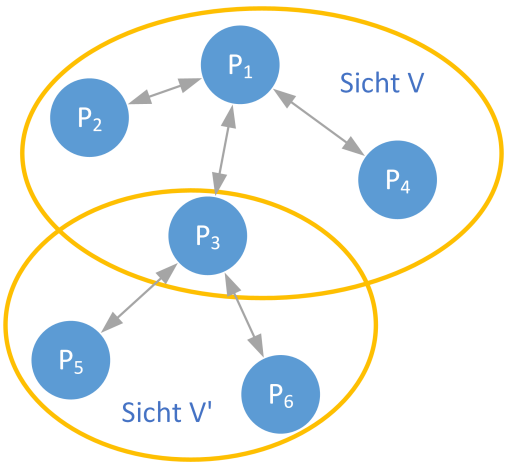


Abbildung 9.3: Partielle Sichten (nach [73])

Peers abgeglichen. In [37] dagegen wird über die erfolgreich ausgetauschten Nachrichten Buch geführt. Nach einem *MERGE* werden nur die fehlerhaften Übertragungen wiederholt. Dieser Ansatz sichert die Konsistenz während des Abgleichs, indem keine weiteren Gruppenoperationen zugelassen werden (im Sinne von: die Verfügbarkeit wird eingeschränkt).

Leeren der Warteschlange In [152, 183, 184] werden zunächst alle aktuellen Nachrichten zugestellt, bevor ein Sichtwechsel erfolgt. Dieses Verfahren wird *Leeren* (*Flushing*) genannt. Ein vergleichbarer Ansatz ist der Next-View-Buffer aus Kapitel 8, welcher in der TOM-Warteschlange existierende Nachrichten ausliefert und Nachrichten aus zukünftigen Sichten zwischenspeichert bzw. den Versand von Nachrichten während eines Sichtwechsels unterbindet. In Verbindung mit dem *Next-View-Buffer* (NVB) ist die Verfügbarkeit der Gruppe nur teilweise eingeschränkt. Die Roboter können Nachrichten versenden und beispielsweise neue Teilnehmer einladen. Die Ausführung der Maßnahmen ist auf einen Zeitpunkt nach Beendigung der entsprechenden Operation verschoben.

Partielle Sichten Die dauerhafte Nutzung von partiellen Sichten innerhalb wiedervereinigter Gruppen wird in [73] vorgeschlagen (siehe Abbildung 9.3). In diesem Ansatz wird das globale Wissen der Teilgruppen nicht synchronisiert. Wie in der Abbildung zu erkennen, werden die unterschiedlichen Sichten stattdessen über Peer P_3 verknüpft. Die Verbindung wird während der weiteren Gruppensitzung aufrechterhalten. Dies ist vergleichbar mit unabhängigen IP-Netzen, welche über Router verbunden werden. Die entsprechende Funktion wird hier von P_3 erbracht: die Umsetzung von Verwaltungsinformationen zwischen den unabhängigen Teilgruppen. Damit stellt P_3 jedoch einen in kollaborativen Anwendungen wenig erwünschten Single-Point-of-Failure dar, dessen Ausfall die Gruppe in die partiellen Sichten zerbrechen lassen würde. Dieser Ansatz würde die Verfügbarkeit nur kurzzeitig einschränken, aber die Partitionierung der Gruppen nie vollständig beseitigen.

Im Ergebnis des Vergleichs dieser Ansätze werden zwei Synchronisationsarten als Lösung vorgeschlagen [98]: die *aktive* und die *reaktive Synchronisation*. Die *aktive Synchronisation* bezieht sich auf die Behandlung von Nachrichten in den Warteschlangen, bevor ein *SPLIT* oder *MERGE* durchgeführt wird. Damit werden nach den ausgeführten Operationen nur Nachrichten versandt, welche in der neu eingerichteten Sicht gültig sind. Umgekehrt synchronisiert die *reaktive Synchronisation* die Warteschlangen erst nach der Operationsausführung. Die aktive Synchronisation wird dabei in *vollständig* (full) und *leeren* (flush) eingeteilt, die reaktive in *verwerfen* (drop) und *semantisch* (semantic).

Volle Synchronisation Bei der vollen Synchronisation werden im Zuge der Vereinigung zweier Gruppen die Warteschlangen der Transferdienste beider Gruppen G' und G'' vereinigt. Bevor die Gruppensichten zusammengeführt werden, muss der Inhalt der Warteschlange zwischen den beiden Teilgruppen ausgetauscht werden. Anschließend wird eine totale Ordnung über die vereinigte Warteschlange gebildet und verarbeitet. Die Nachrichten werden an die Gruppenmitglieder zugestellt. Hiernach kann die gemeinsame Sicht der neuen Gruppe eingerichtet werden. Prinzipiell ist dieser Ansatz für die Transferdienste von *Moversight* umsetzbar. Dies erscheint jedoch nicht vorteilhaft, weil der Abgleich der Nachrichten einen hohen Kommunikationsaufwand erzeugt. So ist die Dauer der Operation abhängig von den Größen beider Gruppen und der Anzahl von Nachrichten, welche die Gruppen in der aktuellen Sicht ausgetauscht haben. Zusätzlich müssen auch zugestellte Nachrichten der aktuellen Sicht gespeichert werden, um im Falle einer vollen Synchronisation für den Abgleich abrufbar zu sein.

Flushsynchronisation Die Flushsynchronisation liefert zunächst die Nachrichten in der Warteschlange des Peers aus, bevor die *SPLIT*- oder *MERGE*-Operation durchgeführt wird. Neu eintreffende Nachrichten werden zwischengespeichert und später verarbeitet, gegebenenfalls angepasst. Damit existieren nach der durchgeführten Operation nur noch Nachrichten aus der aktuellen Sicht. Ein wesentlicher Vorteil dieses Ansatzes ist, dass keine umfangreiche Kommunikation dafür notwendig ist. Darüber hinaus ist er kompatibel mit dem Next-View-Buffer-Konzept aus Kapitel 8. Die unbestimmte Ausführungszeit der Synchronisation und der erhöhte Speicheraufwand für die Nachrichtenzwischenspeicherung sind die Nachteile dieses Ansatzes.

Dropsynchronisation Nach der durchgeführten *SPLIT*- oder *MERGE*-Operation richtet das Gruppenkommunikationssystem eine neue Sicht ein. Reaktiv werden über die Dropsynchronisation automatisch alle die Nachrichten durch den Peer verworfen, welche nicht der aktuellen Sicht entsprechen. Diese leichtgewichtige Synchronisation kann mit geringem Aufwand realisiert werden. Nachteilig ist, dass es potentiell zu einer Vielzahl von wiederholten Nachrichtenübertragungen kommen kann. Das führt zu erhöhten Nachrichtenlatenzen und größeren Nachrichtenverlusten.

Semantische Synchronisation Die semantische Synchronisation ähnelt der Dropsynchronisation und ist ebenfalls ein reaktives Verfahren. Adaptiert wurde dieser Ansatz aus dem Verfahren von Enokido und Takizawa [73], indem jeder Peer auf die aktuelle und die vorangehende Sicht zugreifen kann. Dadurch könnten auch Nachrichten aus der vorhergehenden Sicht zugestellt werden, was einer abgewandelten Sendesichtzustellung entsprechen würde. Um Inkonsistenzen zu vermeiden, dürfen aus der vorhergehenden Sicht jedoch keine sichtändernden Nachrichten zugestellt werden. Wird ein weiterer Sichtwechsel nach der *SPLIT*- oder *MERGE*-Operation vollzogen, wechselt das Protokoll wieder in den normalen Modus zurück und nutzt nur die aktuelle Sicht bei der Eingangsprüfung. Bei einem *MERGE* kommt als weiteres Problem hinzu, dass die IDs der Sichten identisch sein können. Der Aufwand für die Umsetzung der semantischen Synchronisation entspricht dem der Dropsynchronisation. Gleiches gilt für die übrigen Vor- und Nachteile. Zusätzlich wird die Eingangsprüfung der Nachrichten im TOM-Transfer umfangreicher.

Im Rahmen der Umsetzung der Split- und Mergedienste in *Moversight* wurde nur die Flushsynchronisation realisiert, da dieses Konzept am besten mit den Prinzipien der übrigen Dienste, insbesondere des Wartungsdienstes, vereinbar ist. Die Verfügbarkeit der Gruppe wird nur kurzzeitig auf Protokollebene eingeschränkt. Aus Anwendungssicht ist es weiter möglich, Gruppennachrichten zu versenden. Deren Transfer wird zeitlich hinter den Abschluss der *SPLIT*- bzw. *MERGE*-Operation verschoben. Nach dem Leeren der jeweiligen Warteschlangen gibt es folglich einen definierten Zeitpunkt, an dem die Konsistenz der Gruppe wiederhergestellt ist.

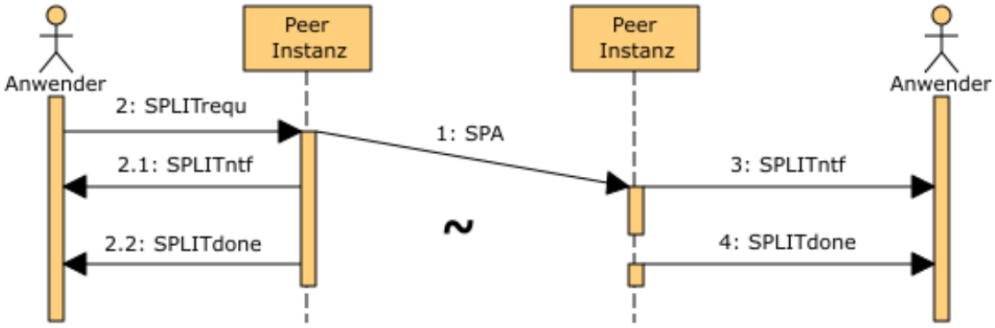


Abbildung 9.4: Zeitablaufdiagramm für einen erfolgreichen *SPLIT*

9.3 Die Operation *SPLIT*

Der Splitdienst realisiert die Aufspaltung einer Gruppe unter Verwendung der Flush-synchronisation auf Basis des Next-View-Buffers aus Kapitel 8. In Abbildung 9.4 ist der erfolgreiche Ablauf einer *SPLIT*-Operation dargestellt. Sender und Empfänger der Dienstprimitive sind jeweils die Splitdienste der einzelnen Gruppenmitglieder. Die Übertragung der Protokolldaten zwischen den Dienstnutzern erfolgt für Gruppennachrichten durch den TOM-Transfer, sonst über den Basistransferdienst.

Durch einen SPLITrequ wird die *SPLIT*-Operation gestartet und per *Split-Announce-Nachricht* (SPA) in der Gruppe über den TOM-Transfer verteilt. Dadurch wird zum einen in der Gruppe die Liste der Peer-IDs verteilt, die sich von der Gruppe abspalten. Zum anderen wird über den SPLITrequ definiert, wie der Dienst auf eine fehlgeschlagene Nachrichtenzustellung bei einigen Gruppenteilnehmern zu reagieren hat (im Sinne von „*missed Peers*“, vgl. dazu Abschnitt 6.3.1). Hierbei kann der Dienstnutzer wählen zwischen:

- alle Peers müssen auf die Anfrage reagieren,
- mindestens die nicht abzuspaltenden Peers müssen reagieren oder
- mindestens die abzuspaltenden Peers müssen reagieren.

Variante eins ist die Standardeinstellung. Wird die SPA-Nachricht erfolgreich in der Gruppe zugestellt und ist das durch den SPLITrequ definierte Antwortverhalten gegeben, so wird dem Nutzer der Beginn der *SPLIT*-Operation über SPLITntf angezeigt. Da *SPLIT* eine sichtändernde Operation ist, kann immer nur ein *SPLIT* ausgeführt werden. Sollten mehrere *SPLIT*-Operationen gestartet worden sein, so wird die Operation ausgeführt, deren SPA-Nachricht als erstes in der Gruppe erfolgreich zugestellt wird (siehe Kapitel 5.4). Alle anderen werden verworfen. Nach Beendigung der *FLUSH*-Operation wird die Gruppenteilung ausgeführt. Der erfolgreiche Abschluss von *SPLIT* wird allen Gruppenteilnehmern über SPLITdone angezeigt. War die Zustellung der SPA-Nachricht nicht erfolgreich oder trat das geforderte Antwortverhalten nicht ein, wird dem Auslöser der *SPLIT*-Operation das Fehlschlagen über SPLITabort angezeigt.

Algorithmus 7 zeigt, wie *SPLIT* innerhalb Moversight umgesetzt wird. Über die Funktion *createMemberRegister* wird ein neues Gruppenregister erzeugt. Dabei wird automatisch bestimmt, welcher der beiden Gruppen der lokale Peer zugeordnet ist. Anschließend wird die Peer-ID des lokalen Peers ermittelt. Das ist notwendig, da die abgespaltene Gruppe bei eins beginnend nummeriert. Anschließend wird der Gruppenverwaltungsdienst mit dem erstellten Gruppenregister erneut initialisiert und damit die Abspaltung der Peers durchgeführt. Abschließend wird dem Nutzer das Ende der *SPLIT*-Operation angezeigt und der lokale Peer über seine neue Peer-ID informiert.

Die Funktion *createMemberRegister* erzeugt eine lokale Kopie des Gruppenregisters und entfernt anschließend alle Peers, welche nicht mehr zur neuen Gruppe gehören. Welche Peers zu entfernen sind, hängt davon ab, zu welcher Gruppe der lokale Peer gehört. Details zu den einzelnen Arbeitsschritten sind in [98] ausgeführt.

Algorithmus 7 *SPLIT* Ausführung

```
1: function doSplit(splitPeers)
2:     ▷ Erzeuge neues Gruppenregister
3:     mr ←createMemberRegister(splitPeers)

4:     ▷ Bestimme den lokalen Peer im neuen Register und schalte auf dieses um
5:     myTa ← getLocalAddress()
6:     localPeerID ← mr.getPeer(myTa)
7:     ms ← getMembershipService()
8:     ms.setupGroupFromMemberRegister(newMR, localPeerID)

9:     signal(SplitDoneEvent)
10:    signal(LocalPeerChangedEvent, getLocalPeer())
11: end function
```

9.4 Die Operation *MERGE*

Beim *MERGE* wird im Gegensatz zu einer Wiedervereinigung von Netzpartitionen angenommen, dass die Gruppen eigenständig sind, also keinen gemeinsamen Ursprung haben. Die ausführliche Dienst- und Protokollspezifikation ist in [98] gegeben. Die Operation besteht aus drei Phasen:

1. **Einladungsphase:** In dieser Phase tauschen sich zwei Stellvertreter der Gruppen – die Mergedirektoren – über die Absicht zur Vereinigung aus und bereiten die Warteschlangen der Gruppenteilnehmer auf die Vereinigung vor

2. **Austauschphase:** In Phase dieser werden die Informationen über die beiden Gruppen wechselseitig ausgetauscht und innerhalb der Gruppen verteilt

3. **Vereinigungsphase:** In der letzten Phase wird die Vereinigung der Gruppen vollzogen. Insbesondere wird die Gruppenverwaltungsinformation für die vereinigte Gruppe aufgebaut und deren Funktionsfähigkeit bzw. der Zusammenhang der Topologie überprüft.

Abbildung 9.5 zeigt als Sequenzdiagramm beispielhaft den Ablauf aller drei Phasen für einen erfolgreichen *MERGE*. Ausgehend von zwei getrennten Gruppen mit zwei bzw. drei Peers wird eine gemeinsame Gruppe mit fünf Peers aufgebaut. Die Peers 2 bzw. 1 aus den ursprünglichen Gruppen agieren als Mergedirektoren. Welcher Roboter bzw. Peer in dieser Rolle agiert, hängt von der konkreten Anwendung ab. Die einzelnen Mergephasen sind über die Markierungen Phase $1-3$ gekennzeichnet. Je nach aktueller Verkehrssituation und Füllstand der Warteschlangen kann der zeitliche Ablauf insbesondere der Phasen eins bis drei stark variieren.

In der (1) Einladungsphase schickt der Mergedirektor der einladenden Gruppe eine entsprechende Anfrage an den Mergedirektor der einzuladenden Gruppe. Der Ablauf dieser Phase ist in Abbildung 9.6 für den erfolgreichen Fall gezeigt. Ausgelöst wird der *MERGE* durch ein MERGErequ, welcher als *Merge-Request-Nachricht* (MR) an den einzuladenden Mergedirektor übermittelt und dem Nutzer per MERGEind angezeigt wird. Die Anfrage wird mittels eines MERGEresp beantwortet und die Antwort über eine *Merge-Confirm-Nachricht* (MC) an den einladenden Direktor übermittelt und per MERGEconf dem Nutzer angezeigt.

In der Abbildung 9.7 sind die möglichen Fehlersituationen der ersten Phase dargestellt: (a) Ablehnung der Mergeanfrage (per MERGErej und *Merge-Reject-Nachricht* (MRJ)) und (b) Verlust einer Nachricht. Die Nachrichten werden zwischen den Mergedirektoren mittels des Basistransfers zuverlässig ausgetauscht. Im Fehlerfall wird die Übertragung mehrfach wiederholt, bevor ein Fehler an den Mergedienst signalisiert wird. In beide Situationen wird der Abbruch des *MERGE* dem Nutzer per MERGEabort signalisiert und der Gegenseite über eine MRJ-Nachricht mitgeteilt. Danach wird *MERGE* abgebrochen.

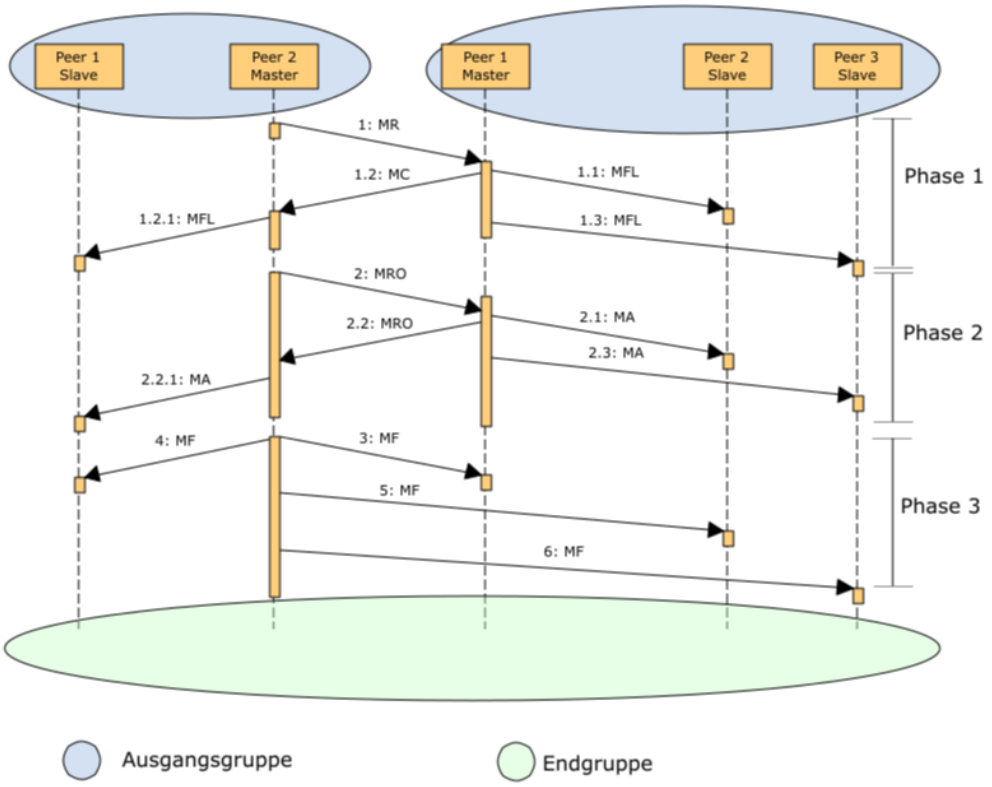


Abbildung 9.5: Nachrichtenversand bei einem erfolgreichen *MERGE*

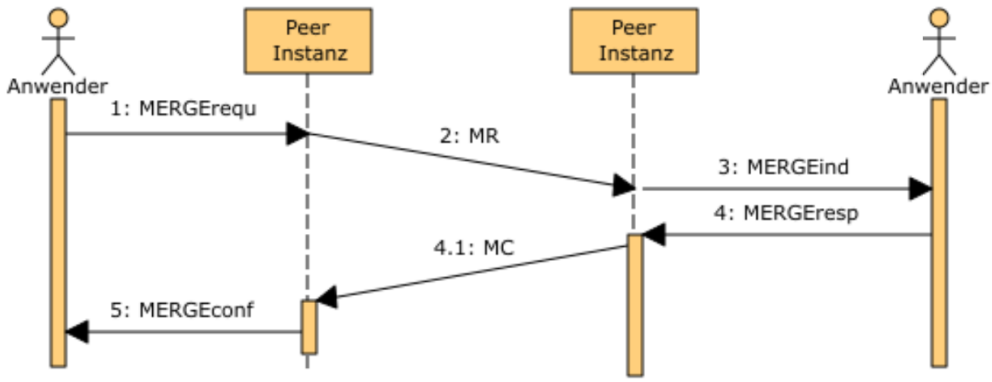
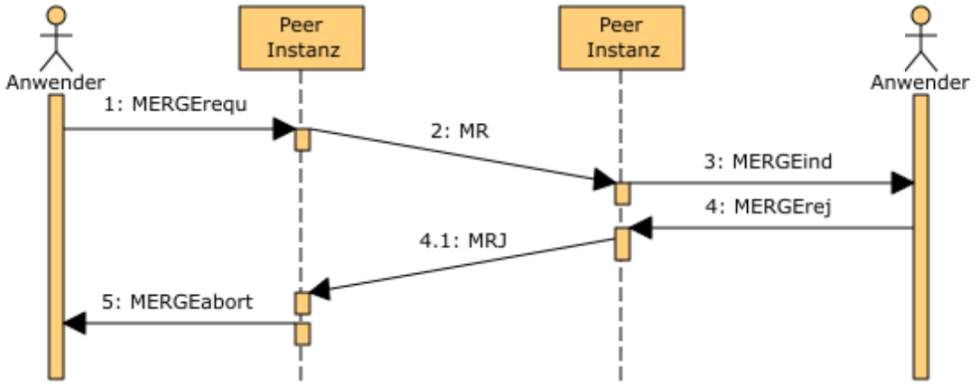
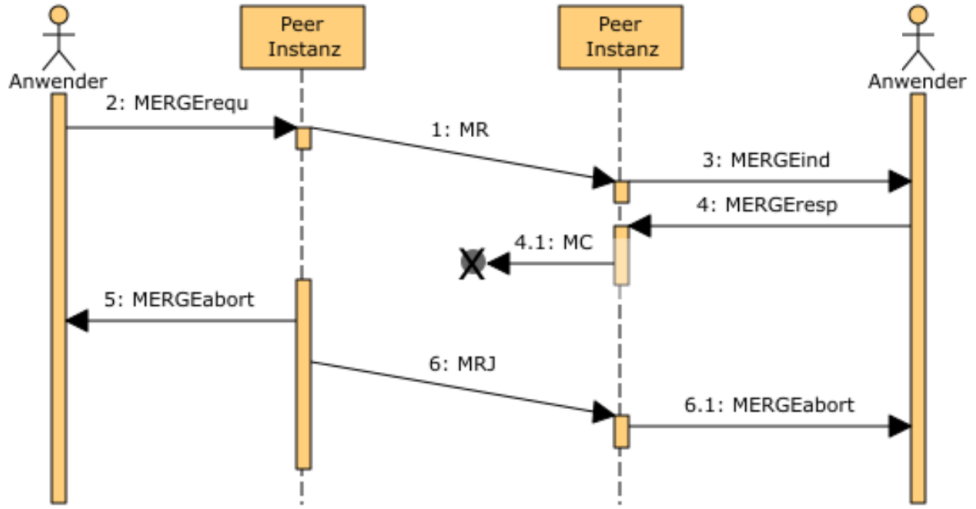


Abbildung 9.6: Ablauf *MERGE* Phase 1 im Erfolgsfall



(a) Ablehnung durch den Nutzer



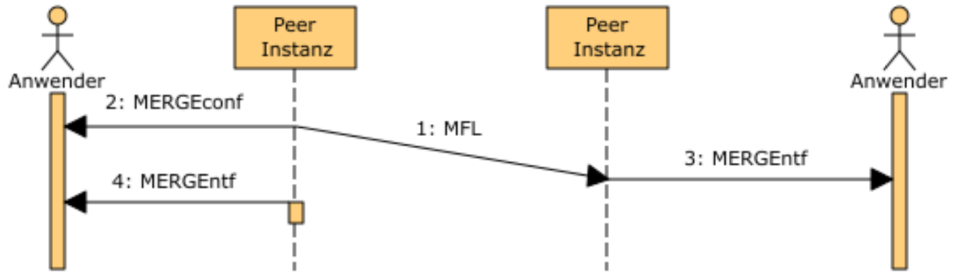
(b) Ablehnung aufgrund von Nachrichtenverlust

Abbildung 9.7: Ablauf *MERGE* Phase 1 im Fehlerfall

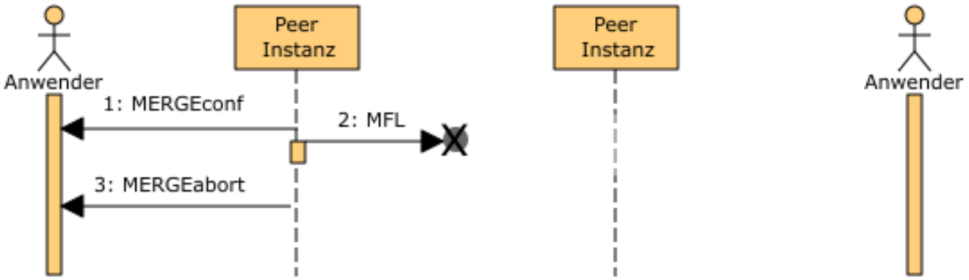
Abbildung 9.8a zeigt den Abschluss der ersten Mergephase. In ihr werden seitens der jeweiligen Mergedirektoren die Mitglieder ihrer Gruppe durch das Aussenden einer *Merge-Flush-Nachricht* (MFL) in den Zustand *FLUSHING* versetzt. Damit beginnt die eigentliche Gruppenvereinigung. Dieser Zustandwechsel wird dem Nutzer über das MERGEntf-Primitiv angezeigt. Im Zustand *FLUSHING* wird die TOM-Warteschlange der Peers geleert. Dadurch wird zur Sicherung der Konsistenz ein neuer Ursprungspunkt für das gemeinsame Wissen der Gruppe definiert bzw. ermittelt. Während des Leerens der Warteschlange ist die Verfügbarkeit der Gruppe eingeschränkt. Der Empfang einer MFL-Nachricht stellt für die übrigen Gruppenmitglieder des jeweiligen Mergedirektors den Beginn der eigentlichen *MERGE*-Operation dar.

Die MFL-Nachricht ist eine Gruppennachricht. Es ist daher festzulegen, wie mit Peers verfahren wird, welche den Empfang nicht bestätigen. In der Grundeinstellung werden Pending-Peers mit dem gleichen Zustand in die vereinigte Gruppe übernommen. Es kann jedoch über die Konfiguration auch definiert werden, dass ein *MERGE* nur dann erfolgreich durchgeführt werden kann, wenn jedes Gruppenmitglied die MFL-Nachricht korrekt behandelt. In diesem Fall führt die fehlende Antwort eines Peers zum Abbruch der *MERGE*-Operation, wie in Abbildung 9.8b dargestellt.

Wie bei der *SPLIT*-Operation kann immer nur eine *MERGE*-Operation gleichzeitig ausgeführt werden. Zur Vereinfachung der Koordination wird die Operation ausgeführt, welche als erste erfolgreich eine MFL-Gruppennachricht zustellt. Weitere parallele Operationen werden mittels einer MRJ-Nachricht an die dritten Mergedirektoren abgebrochen.



(a) Erfolgsfall



(b) Abbruch aufgrund von Nachrichtenverlust

Abbildung 9.8: Abschluss *MERGE* Phase 1

Mit der erfolgreichen Zustellung der MFL-Nachricht beginnt die zweite Mergephase. Darin werden die jeweils aktuellen Gruppenregister zwischen den beiden Mergedirektoren über eine *Merge-Roster-Nachricht* (MRO)[1] zuverlässig über den Basistransfer ausgetauscht. Bei einer fehlerhaften Übertragung der MRO-Nachricht muss die jeweilige Gruppe durch den Mergedirektor über den Abbruch der Operation per *Merge-Abort-Nachricht* (MAB) explizit informiert werden. Durch den Abbruch des *MERGE* wird die Verfügbarkeit der Gruppe wiederhergestellt und der Partitionsmodus verlassen.

Nach dem erfolgreichen Empfang der MRO-Nachricht werden die Gruppendaten der anderen Gruppe über eine *Merge-Announce-Nachricht* (MA) in der jeweiligen Gruppe verteilt. Dazu ist es notwendig, dass alle fehlerfreien Peers der einzelnen Gruppe diese Nachricht korrekt empfangen, da anderenfalls keine gemeinsame Sicht installiert werden kann. War die Übertragung erfolgreich, so wird der *MERGE* abgeschlossen, indem durch den Initiator des *MERGE* eine *Merge-Finish-Nachricht* (MF) an die neue, vereinigte Gruppe versendet wird. Die MF-Nachricht muss von allen Mitgliedern der neuen Gruppe empfangen werden. Damit wird der Gruppenzusammenhang wieder hergestellt und die Korrektheit der Gruppenverwaltungsinformation gewährleistet.

Kommt es in der zweiten und dritten Phase des *MERGE* zu Übertragungsfehlern, wird der *MERGE* abgebrochen. In dieser Phase wurde jedoch bereits das Gruppenregister verändert. Diese Änderungen werden über ein so genanntes Rollback rückgängig gemacht. Dazu werden mit Hilfe von Kopien der Gruppenregister des *MERGE* die ursprünglichen getrennten Gruppensichten in den Peers wiederhergestellt. In Abbildung 9.9 ist eine fehlgeschlagene *MERGE*-Operation für das bereits bekannte Beispiel dargestellt. Hier können in Phase 2 die Gruppendaten nicht ausgetauscht werden, wodurch es zum Abbruch der Operation mittels einer MAB-Nachricht kommt. In diesen Fall ist ein Rollback nicht notwendig, da noch keine Änderungen an den Gruppenregistern vorgenommen wurden.

Details zur Behandlung der einzelnen Nachrichten, zu den verwendeten Timern und internen Zuständen des Mergedienstes sind in [98] ausführlicher dargestellt.

Ein spezielles Problem bei der Vereinigung von Gruppen ist die eindeutige Identifikation der Peers. Bei einer *MERGE*-Operation besteht die Möglichkeit, dass Peer-IDs

[1] Roster steht für Gruppenteilnehmerdaten (vergleiche Kapitel 5)

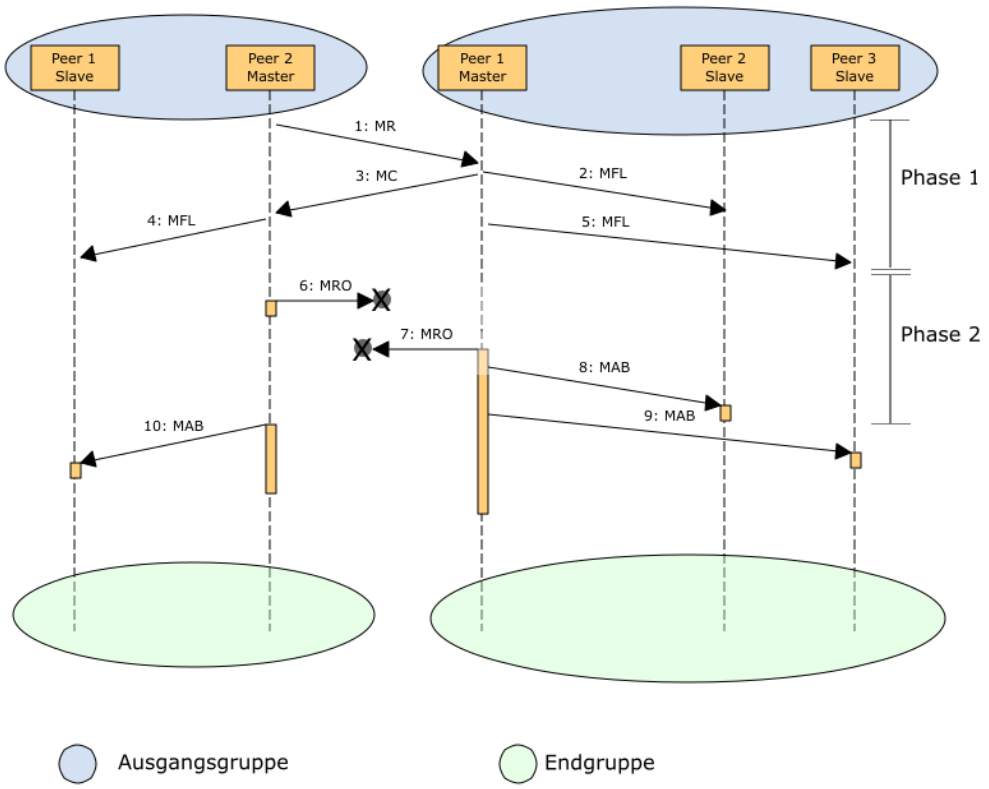


Abbildung 9.9: Nachrichtenversand bei einem fehlerhaften *MERGE*

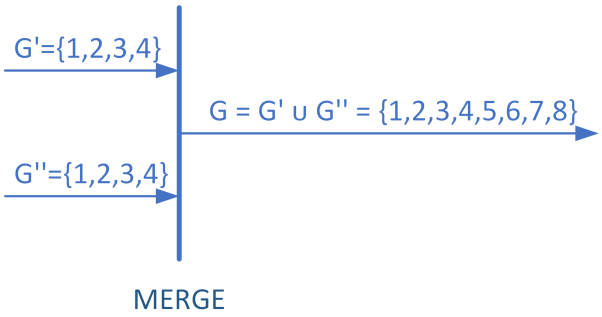


Abbildung 9.10: Auflösung doppelter Peer-ID bei einem *MERGE*

in beiden Gruppen, d.h. doppelt, vergeben werden, wie in Abbildung 9.10 dargestellt. Die beiden Gruppen G' und G'' sollen zu einer Gruppe G vereinigt werden. Die Mengen der vergebenen Peer-IDs sind jedoch in beiden Gruppen identisch. Um zukünftige TOM-Übertragungen korrekt ausführen zu können, müssen die Identifikationskonflikte aufgelöst werden. Die Anzahl der konfliktbehafteten Peers ist dabei unerheblich, da eindeutige Bezeichner in Gruppenkommunikationssystemen notwendig sind [10, 54, 114, 184]. Auf der rechten Seite der Abbildung ist das Ergebnis der Konfliktauflösung dargestellt.

Die *MERGE*-Operation in *Moversight* soll jedoch zwei eigenständige Gruppen unabhängig davon, ob sie einen gemeinsamen Ursprung haben, vereinigen. Daher kann hier eine beliebige Anzahl von Peer-IDs in beiden Gruppen konfliktbehaftet sein. Auf eine Umbenennung könnte verzichtet werden, wenn die Peer-ID durch einen eindeutigen Hashwert über dessen Transportadresse und einen Zeitwert, wie beispielsweise in [132], bestimmt wird. Dazu ist es jedoch notwendig, die Peer-ID von aktuell 32 bit auf 256 bit zu vergrößern, um Kollisionen bei der Hashwertbestimmung (weitgehend) zu vermeiden.

Da die Peer-ID jedoch als Absenderadresse, als Last-Hop oder in diversen anderen Listen verwendet wird (z. B. die Liste der „*missed Peers*“ für eine Übertragung oder die Liste der abzuspaltenden Peers bei einem *SPLIT*, Listen von Anwendungsdaten usw.) würde dies die durchschnittliche Nachrichtengröße deutlich erhöhen und den Aufwand für eine Nachrichtenübertragung erheblich vergrößern.

Um die Gruppen G' und G'' zu vereinigen und dabei die konfliktbehaften Peer-IDs zu entfernen, wird in Moversight ein vereinfachtes Verfahren verwendet. Es fügt die Mitglieder einer Teilgruppe der anderen hinzu, wodurch automatisch konfliktfreie Peer-IDs für die neuen Gruppenmitglieder gebildet werden. Zunächst wird eine der beiden Gruppen als Beitrittsgruppe ausgewählt. Dies erfolgt über die Initiierung der *MERGE*-Operation (hier angenommen G'). Als Konvention wird definiert: Der einladenden Gruppe wird beigetreten. Anschließend werden G' die Mitglieder von G'' hinzugefügt. Die Peers der Ergebnisgruppe G sind wie in der Abbildung 9.10 benannt. Der Nachteil dieser Lösung besteht darin, dass jedes Gruppenmitglied nach der Vereinigung in der neuen Gruppe seine möglicherweise neue Bezeichnung identifizieren muss, da seine Peer-ID in der Ursprungsgruppe von der Peer-ID in der neuen Gruppe abweicht. Beispielsweise ändert sich in Abbildung 9.10 für Peer 1 aus G'' die Bezeichnung in G. Um seine neue Peer-ID zu ermitteln, muss jeder Peer die Gruppenverwaltungsinformationen nach seiner aktuellen Transportadresse durchsuchen und so die ihm zugeordnete Peer-ID feststellen. In großen Gruppen ist dieser „*Lookup*“ ressourcenaufwendig und damit teuer, muss jedoch nur einmal pro *MERGE* durchgeführt werden.

9.5 Evaluierung

Die Funktionsfähigkeit der *SPLIT*- und *MERGE*-Operationen wurde mittels verschiedener Gruppengrößen (mit 6, 12, 24, 48, 72 und 96 Peers) getestet. Für die Tests wurde das OMNeT++ genutzt, das ein idealisiertes Kommunikationsnetz mit einer Bandbreite von 11 Mbit/s und 13 ms Verzögerung pro Kommunikationslink simulierte. Um Einflüsse der Netzebene auf die Messergebnisse zu vermeiden, wurden alle Peers direkt miteinander verbunden.

9.5.1 Analyse der *SPLIT*-Operation

Die *SPLIT*-Operation wurde in vier Testszenarien untersucht, wobei jedes dieser Szenarien auf die genannten sechs Gruppengrößen angewendet wurde:

- **Testszenario 1 – Abspaltung eines Clusters** In diesem Testszenario wird ein kompletter Cluster von Robotern abgespalten.
- **Testszenario 2 – Abspaltung aller Master** Hier werden alle Master der bestehenden Gruppe in eine neue Gruppe abgespalten. In der ursprünglichen Gruppe müssen jeweils neue Master in den Clustern gewählt werden. Die Master können hier beispielsweise für ressourcenmäßig belastete Roboter stehen, deren Ressourcen erschöpft sind bzw. zur Neige gehen und die für einen Ladevorgang die kollaborative Gruppe verlassen.
- **Testszenario 3 – Abspaltung von Slaves** Dieser Test wird in zwei Untertestfälle gegliedert. Im Szenario 3*a* werden weniger als 50 % der Slaves einer Gruppe abgespalten, im Szenario 3*b* mehr als 50 %.
- **Testszenario 4 – Abspaltung beliebiger Peers** Auch hier wurden wieder zwei Varianten getestet. Im Szenario 4*a* werden weniger als 50 % aller Gruppenteilnehmer abgespalten, im Szenario 4*b* mehr als 50 %.

Die Szenarien drei und vier entsprechen der anwendungsgetriebenen Aufteilung der Robotergruppen für die Untersuchung unterschiedlicher Zielgebiete. In allen Testszenarios müssen die Peerplatzierung (vgl. Abschnitt 8.4.1) und die Gruppentopologie wieder an die neue Situation angepasst werden, was zu einer kompletten Neuordnung der Clustertopologie führen kann.

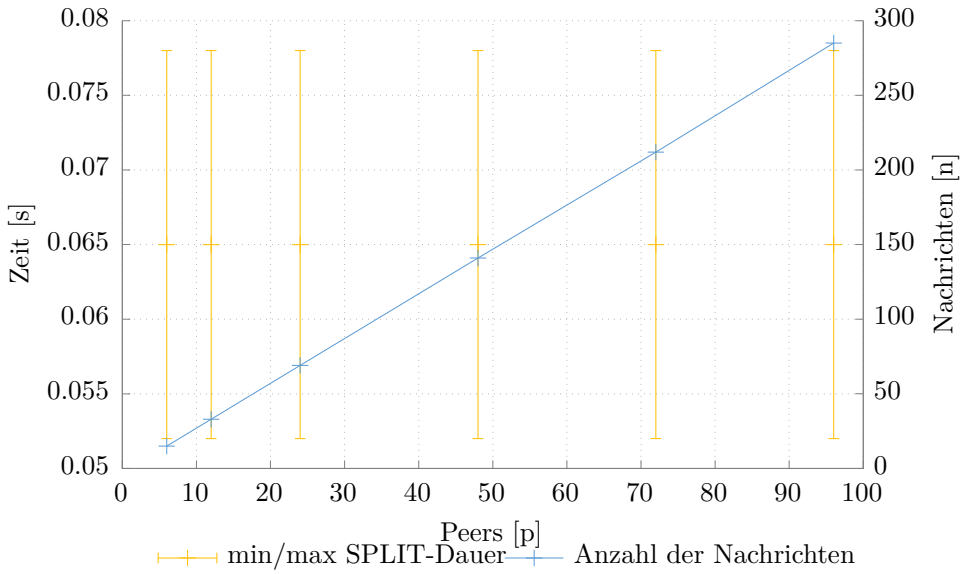


Abbildung 9.11: Dauer der *SPLIT*-Operation und Anzahl der versandten Nachrichten für Testfall 1

Tabelle 9.1: Vergleich der *SPLIT*-Testergebnisse je Szenario

Szenario	**Gruppe**					
in s	**6**	**12**	**24**	**48**	**72**	**96**
1						
2						
3a	0.052	0.052	0.052	0.052	0.052	0.052
3b	–	–	–	–	–	–
4a	0.078	0.078	0.078	0.078	0.078	0.078
4b						

Abbildung 9.11 zeigt den zeitlichen Verlauf einer *SPLIT*-Operation für den ersten Testfall. Es wurde hier die Differenz zwischen der Startzeit (Initiierung von *SPLIT*) und dem Ende des Vorgangs bestimmt und deren minimale und maximale Werte innerhalb der Gruppe für jede Gruppengröße dargestellt. Diese Differenz wird *SPLIT*-Dauer genannt. Durch die direkte Verbindung der Gruppenteilnehmer untereinander sind die vorhandenen Schwankungen in erster Linie auf die Änderungen der Gruppentopologie (speziell Clusteranzahl und -größe) zurückzuführen. Zusätzlich wurden die für die Durchführung der *SPLIT*-Operation notwendigen Nachrichten bestimmt und in der Abbildung 9.11 auf der rechten Achse dargestellt. Für die Anzahl der Nachrichten wurde vereinfachend davon ausgegangen, dass *SPLIT* jeweils durch einen Master initiiert wird.

Die minimalen und maximalen Werte für die *SPLIT*-Dauer je Testszenario und Gruppengröße zeigt Tabelle 9.1. Durch die Verwendung des idealisierten Testnetzes und die Eliminierung möglicher Engpässe, Paketverluste und sonstiger Kommunikationsstörungen innerhalb des Netzes zeigt sich der konstante Aufwand der *SPLIT*-Operation.

Um die Leistungsfähigkeit der *SPLIT*-Operation zu bewerten, kann man diese einer kombinierten *LEAVE*/*JOIN*-Operation gegenüberstellen. So soll beispielsweise eine Gruppe von sechs Peers in zwei gleich große Gruppen geteilt werden. Das kann zum einen über eine *SPLIT*-Operation und zum anderen über den Austritt von drei Peers

Tabelle 9.2: Vergleich der *MERGE*-Testergebnisse je Szenario

Szenario	**Gruppe**					
in s	**6**	**12**	**24**	**48**	**72**	**96**
1	0.156	0.156	0.209	0.209	0.262	0.262
	–	–	–	–	–	–
	0.182	0.182	0.235	0.235	0.288	0.288

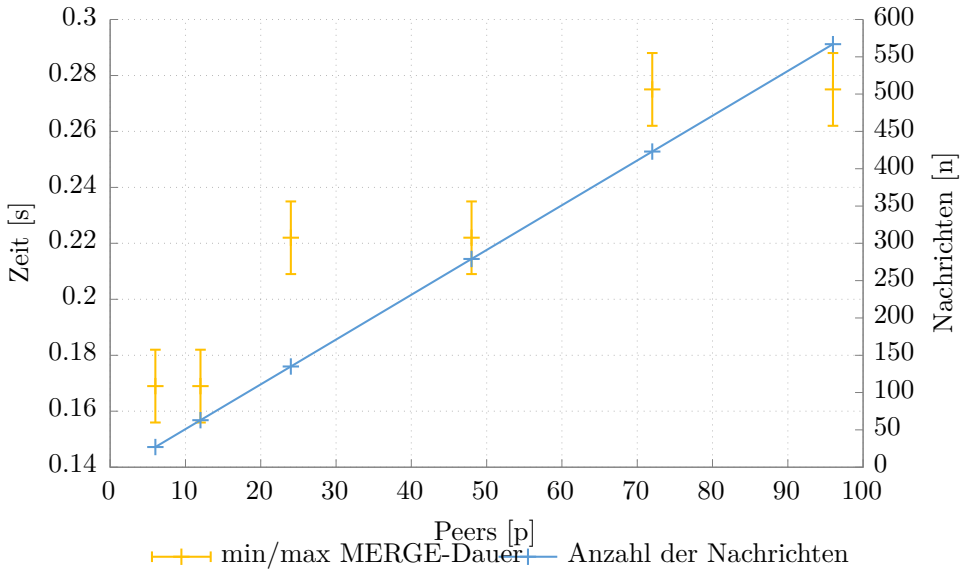


Abbildung 9.12: Dauer der *MERGE*-Operation und Anzahl der versandten Nachrichten für Testfall 1

und die anschließende Bildung einer neuen Gruppe erreicht werden. Die *LEAVE*/*JOIN*-Operation würde also aus insgesamt fünf Operationen bestehen (drei *LEAVE*- und zwei *JOIN*-Operationen). Im hier verwendeten idealisierten Testnetz dauert sowohl eine *LEAVE*- als auch eine *JOIN*-Operation 0,026 s. Somit benötigen die fünf Operationen idealisiert 0,13 s. Die *SPLIT*-Operation mit maximal 0,078 s Ausführungszeit ist also deutlich schneller.

9.5.2 Analyse der *MERGE*-Operation

Die *MERGE*-Operation wurde in einem Szenario mit den zuvor genutzten Gruppengrößen (6, 12, 24, 48, 72 und 96 Peers) getestet.

Die Auswertung schließt mit einer zeitlichen Betrachtung des zeitlichen Aufwandes einer *MERGE*-Operation (vergleichbar zum Splitdienst) gegenüber einem n-fachen *LEAVE*/*JOIN*. Wie beim Splitdienst soll hier wiederum eine 6er-Gruppe untersucht werden. Entsprechend dem vorgestellten Testszenario bestehen somit die Teilgruppen aus jeweils drei Peers, verwaltet in je einem Cluster. Damit ist die Dauer des Merges mit der Dauer von zwei *LEAVE*- und anschließenden drei *JOIN*-Operationen zu vergleichen. Wie zuvor wird im idealisierten Testnetz sowohl für eine *LEAVE* als auch eine *JOIN*-Operation eine Dauer von 0,026 s angenommen. Dies ergibt für die fünf Operationen eine Gesamtdauer von 0,13 s unter der Annahme, dass die *JOIN*s nacheinander erfolgen. Die *MERGE*-Operation dauert bei gleicher Gruppengröße 0,182 s und damit länger als die Gesamtdauer des *LEAVE*/*JOIN*-Ansatzes.

Jedoch bereits bei einer Gruppengröße von 12 Peers ist die *MERGE*-Operation mit ebenfalls 0,182 s im Vergleich zu 11 *LEAVE*/*JOIN*-Operationen mit insgesamt 0,286 s Dauer deutlich schneller. Selbst für Gruppen mit 24 und 48 Peers ist die *MERGE*-

Operation schneller als der *LEAVE/JOIN*-Ansatz für 12 Peers. Für Gruppen mit 72 und mehr Peers kann man den Vergleich auf diese Weise nicht durchführen, da hier Latenz, Timereinstellungen und die konkrete Warteschlangenbelegung einen größeren Einfluss auf die Operationsdauer haben und somit eine Abschätzung über die Anzahl der Operationen nicht zielführend ist.

Zusammenfassend kann festgehalten werden, dass mit Split und Merge zwei kollaborative Dienste für das Teilen und Vereinigen von Gruppen definiert wurden, wie sie bisher in keinem der existierenden Gruppenkommunikationssysteme existieren (vgl. Kapitel 3). Beide Operationen ermöglichen die anwendungsgetriebene Neustrukturierung der kollaborativen Gruppe. Die für die Sicherung des globalen Wissens notwendigen Maßnahmen wurden aus thematisch verwandten Konzepten abgeleitet und weiterentwickelt. Die Leistungsanalyse zeigt, dass beide Operationen bereits bei kleinen Gruppengrößen effizienter sind als der alternative *LEAVE/JOIN*-Ansatz. Darüber hinaus bleiben bei beiden Operationen der Anwendungskontext sowie das globale Wissen der Gruppe(n) erhalten.

Kapitel 10

Zusammenfassung

Die Vision des Ubiquitous Computing ist für die mobile Nutzung des Internets weitgehend verwirklicht. Damit einher geht eine stetig zunehmende Zahl von Anwendungen auch in mobilen Umgebungen. Diese Anwendungen sind durch spezifische Anforderungen und eine enge Kooperation zwischen deren Nutzern gekennzeichnet. Sie modellieren zumeist mobile verteilte Entscheidungsprozesse, welche die Bewertung von Sachverhalten bzw. die Lösung von Aufgaben zum Ziel haben. Diese neue Klasse von Anwendungen heißt *mobile kollaborative Anwendungen*.

In dieser Arbeit wurde diese Klasse von Anwendungen eingeführt und ihre Eigenschaften und Besonderheiten diskutiert. Charakteristisch für solche Anwendungen ist eine enge Kooperation zwischen den Nutzern bzw. Teilnehmern, als deren Ergebnis ein gemeinsames Wissen aufgebaut wird. Auf dieser Basis treffen die Anwendungen lokale Entscheidungen, über die Konsens in der Gruppe besteht. Die instabile Kommunikation in mobilen Umgebungen steht jedoch der Nutzung herkömmlicher Verfahren zur Konsistenzsicherung aus gruppenorientierten Systemen drahtgebundener Netze entgegen. Deshalb wurde die Sicherung der Konsistenz des globalen Wissens in mobilen Umgebungen zum Leitthema dieser Arbeit. Anhand des Beispielszenarios kooperierender Suchroboter wurden die besonderen Problemstellungen in mobile kollaborative Anwendungen diskutiert. Aufgrund der großen Spanne und Heterogenität der potentiellen Einsatzszenarien mobiler kollaborativer Anwendungen ist es zweckmäßig, derartige Anwendungen unter Nutzung einer Anwendungsplattform zu realisieren. Ausgewählte Vertreter solcher Plattformen wurden vorgestellt und deren grundlegende Funktionen diskutiert. Allen Plattformen ist das Fehlen eines gruppenorientierten Kommunikationsdienstes gemein, welcher in mobilen Umgebungen ein konsistentes globales Wissen innerhalb der Gruppe bereitstellt.

In mobilen Umgebungen muss ein gruppenorientierter Kommunikationsdienst insbesondere robust gegenüber Verbindungsabbrüchen sein und diese bei Schonung der begrenzten Ressourcen der mobilen Endgeräte zeitnah beheben. Die kontinuierliche Adaption der Gruppe an sich ändernde Bedingungen ist dabei ebenso eine Herausforderung. Mögliche Ansätze für die Realisierung solch eines Dienstes wurden in dieser Arbeit untersucht. Es zeigt sich, dass Gruppenkommunikationssysteme am besten für die Realisierung eines gruppenorientierten kollaborativen Kommunikationsdienstes geeignet sind.

Existierenden Gruppenkommunikationssystemen fehlen bisher Funktionen zur Behandlung der Nutzermobilität, Verfahren zur Adaption an sich kontinuierlich ändernde Kommunikationsumgebungen und notwendige Mechanismen zur Sicherung der Konsistenz des gemeinsamen Wissens in einem instabilen Kommunikationsumfeld. Die Auswirkungen des Fehlens solcher Funktionen und Mechanismen auf die Konsistenz des globalen Wissens wurde in einem Experiment unter Nutzung des Gruppenkommunikationssystems *JGroups* im Rahmen dieser Arbeit illustriert. Häufige Verbindungsunterbrechungen können das gemeinsame Wissen der Gruppe zerstören und zu einer Aufspaltung in logische Partitionen führen. Folgen dieser Partitionierung können unter anderem unterschiedliche bzw. fehlerhafte Annahmen über die Gruppenzusammensetzung sein, die eine erfolgreiche Datenverteilung an alle Gruppenteilnehmer verhindern oder die

Gruppe bzw. einzelne Teilnehmer durch Fehlermaßnahmen blockieren. Nicht zuletzt können Teilnehmer aufgrund von Fehleinschätzungen ausgeschlossen werden.

Zur Lösung der genannten Probleme wurde in dieser Arbeit mit dem Gruppenkommunikationsprotokoll *Moversight* ein Ansatz für einen gruppenorientierten Kommunikationsdienst für mobile kollaborative Anwendungen vorgeschlagen, der sich speziell auf die Sicherung der Konsistenz des gemeinsamen Wissens innerhalb einer mobilen Gruppe konzentriert. Grundlage dieses Dienstes ist ein neues semantisches Modell – *Mobile Optimistic Virtual Synchronity* (MOVS) – welches das klassische Modell der Wissenssicherung – die virtuelle Synchronität – um Aspekte erweitert, die die Nutzermobilität und insbesondere die variierende Konnektivität der Teilnehmer berücksichtigen. Im Unterschied zu anderen Varianten der virtuellen Synchronität sieht MOVS eine Rückkehr der Teilnehmer in die Gruppe innerhalb eines definierten Zeitraums und ihre komplette Resynchronisation vor. Anders als bei der schwachen und der optimistischen virtuellen Synchronität kommt es bei MOVS nie zu einer Rücknahme von Nachrichten, die bereits an die Anwendung ausgeliefert wurden. Stattdessen entscheidet die Anwendung, ob die Gruppennachricht auch bei temporär nicht erreichbaren Gruppenteilnehmern versendet werden soll.

Moversight setzt das MOVS-Paradigma prototypisch mit Hilfe von drei Diensten um: mittels eines total geordneten, mobilen Multicastdienstes (TOM), eines Fehlerdetektors und der Mobiltätsunterstützung. Im Gegensatz zu anderen Gruppenkommunikationssystemen wurden diese Dienste zusammen als kooperative Einheit entworfen. Sie erkennen gemeinsam frühzeitig Verbindungsabbrüche zu anderen Gruppenteilnehmern und leiten deren Behandlung ein. Das Protokoll reagiert nicht wie vergleichbare Systeme mit dem Ausschluss nichterreichbarer Teilnehmer. Der *Moversight*-Ansatz zur Erkennung von Fehlersituationen mit anschließenden *REJOIN* ist deutlich leistungsfähiger als der zumeist verwendete Gruppenausschluss und -wiedereintritt.

Das Erkennen nichterreichbarer Teilnehmer und die Installation der optimistischen Sicht reduzieren die Verfügbarkeit des Gruppenkommunikationssystems. Die Gruppe wird dadurch in zwei oder mehrere unabhängige Partitionen unterteilt. Während dieser Trennung können die Partitionen vollständig unabhängig voneinander weiterarbeiten und sind nur teilweise in ihrem Funktionsumfang eingeschränkt. Vergleichbare Systeme blockieren zumeist vollständig bzw. beenden den Dienst in den abgetrennten Partitionen. Damit definiert *Moversight* einen Kompromiss für die durch das *CAP*-Theorem beschriebenen zeitweiligen Einschränkungen mobiler kollaborativer Anwendungen, wonach im Falle einer Partitionierung immer nur zwei der drei Eigenschaften Verfügbarkeit, Partitionstoleranz und Konsistenz in einem verteilten System gelten können. Die Umsetzung dieses Kompromisses in *Moversight* basiert auf dem erweiterten semantischen Modell für die virtuelle Synchronisation MOVS.

Neu ist neben der prototypischen Umsetzung der mobilen optimistischen virtuellen Synchronität mit dem *Moversight*-Protokoll der eingeführte Ansatz für eine verteilte globale Latenzschätzung in verteilten mobilen Umgebungen. Korrekte Latenzschätzungen sind für die Fehlerdetektion sowie die Fehlerbehandlung in Transferdiensten notwendig. Existierende Gruppenkommunikationssysteme verwenden Verfahren, welche nur eine teilnehmerlokale Schätzung mit einer lokalen Uhr ermöglichen bzw. verwenden konstante Intervalle für ihre Dienste. Die angenommenen Verzögerungen spiegeln also nicht die aktuelle Kommunikationssituation wider, wodurch es zu einer fehlerhaften bzw. verzögerten Erkennung von nichterreichbaren Teilnehmern kommt, Übertragungen verzögert oder fälschlicherweise wiederholt werden. Dadurch werden die Leistungsfähigkeit des Systems und die Ressourcen der mobilen Teilnehmer (deutlich) reduziert. Der neu eingeführte verteilte Schätzansatz kommt vollständig ohne zentrale Instanzen oder lokale Uhren aus. Stattdessen werden Elemente des globalen Wissens genutzt, wodurch individuelle Schätzungen innerhalb der Gruppe konsistent bleiben. Das Grundprinzip der Schätzung kann mit verschiedenen Verfahren kombiniert werden. Dies können die in dieser Arbeit entwickelten oder aus der Literatur bekannte Verfahren sein. Die hier vorgeschlagenen Verfahren berücksichtigen die typischen Gammaverteilungseigenschaften von Latenzen in drahtlosen Netzen mit Hintergrundverkehr. Die so gewonnenen Abschätzungen der aktuellen Verzögerungen innerhalb der Gruppe werden als ein Baustein bei der kontinuierlichen Adaption des Gruppenkommunikationssystems an die sich

permanent ändernden Kommunikationsbedingungen verwendet.

Ein weiterer Baustein ist die kontinuierliche Optimierung der verwendeten clusterbasierten Kommunikationstopologie. Obwohl für mobile kollaborative Anwendungen konzipiert, eignet sie sich auch für andere clusterbasierte Anwendungsszenarien, beispielsweise in mobilen Ad-hoc Netzen und drahtlosen Sensornetzen. Die Optimierung erfolgt mittels einer dynamischen Anordnung der Gruppenteilnehmer in Clustern, wodurch die Topologie auf Basis statistischer Kenngrößen sowie der Ressourcen der einzelnen Gruppenteilnehmer kontinuierlich an die Kommunikationsbedingungen angepasst wird. Die erreichbare Anwendungslebensdauer des Ansatzes übertrifft vergleichbare Optimierungsansätze und ist unabhängig von der Clusteranzahl. Dabei müssen nicht – wie in anderen ähnlichen Verfahren – Annahmen über die Gruppenstruktur während der Entwicklung getroffen werden.

Mit *Split* und *Merge* besitzt *Moversight* zwei kollaborative Dienste für das Teilen und Vereinigen von Gruppen, wie sie bisher in keinem der existierenden Gruppenkommunikationssysteme vorhanden sind. Beide Dienste ermöglichen eine anwendungsgetriebene Neustrukturierung der kollaborativen Gruppe und nutzen zur Sicherung des globalen Wissens Maßnahmen, welche thematisch mit dem Konzept der virtuellen Synchronität bzw. den Clusteralgorithmen verwandt sind. Bereits bei kleinen Gruppengrößen ist die Nutzung dieser Dienste effizienter als der zumeist verwendete Ausschluss mit expliziter Wiedereinladung. Darüber hinaus bleiben der Anwendungskontext und das globale Wissen der Gruppe(n) bei der Ausführung beider Dienste erhalten.

10.1 Erfüllung der Anforderungen durch das *Moversight*-Protokoll

Zusammengefasst erfüllt das *Moversight*-Protokoll die formulierten Anforderungen an mobile kollaborative Anwendungen wie folgt:

1) Kleinteilige geschlossene Gruppe für spontane Kollaboration Das Protokoll unterstützt geschlossene kollaborative Gruppen bis zu 100 Teilnehmern, die über ein P2P-Overlay miteinander kommunizieren. Die gewählte Clusterstruktur bietet die notwendige Robustheit gegenüber Verbindungsabbrüchen mobiler Teilnehmer und verwaltet die Verbindungen zwischen den Gruppenmitgliedern ressourceneffizient.

2) Gemeinsames Wissen gleichberechtigter Gruppenmitglieder Mit Hilfe der mobilen optimistischen virtuellen Synchronität wird das gemeinsame Wissen der Gruppe in mobilen Umgebungen gesichert. Durch den Multicastdienst TOM wird eine zuverlässige und geordnete Nachrichtenauslieferung an alle Gruppenmitglieder gewährleistet. Die *REJOIN*-Operation erlaubt die Wissenswiederherstellung bei kurzzeitigen Verbindungsabbrüchen. Der Aufwand für diese Sicherung bleibt durch die Clusterstruktur des Overlays begrenzt.

3) Robustheit gegenüber Verbindungsabbrüchen und deren zeitnahe transparente Behandlung Das *Moversight*-Kommunikationsmodell sichert die Kommunikation zwischen den Peers, wenn in mobilen Umgebungen die Verbindung zu einzelnen Teilnehmern ausfällt. Durch die kontinuierliche Fehlerdetektion erkennt und behandelt es nicht protokollkonformes Verhalten von Teilnehmern aufgrund von Ausfällen der Endgeräte, Verbindungsunterbrechungen oder Netzpartitionierungen zeitnah und transparent. Der Wiedereintrittsmechanismus verhindert unnötige Abbrüche der Kollaboration.

4) Optimierung für Endgeräte mit begrenzten Ressourcen *Moversight* berücksichtigt die Ressourcenbeschränkung der Endsysteme, indem es die Rollenzuweisungen der Peers im Overlays unter dem Gesichtspunkt der Ressourcenauslastung vornimmt. Zusätzlich wird das Overlay kontinuierlich optimiert, um eine Überbeanspruchung einzelner Peers zu verhindern. Außerdem werden überflüssige Sendewiederholungen und unnötiges Warten auf Antworten von Kommunikationspartnern infolge falsch eingestellter Überwachungsfunktionen vermieden. Um diese zumeist auf falschen Annahmen

bezüglich der aktuellen Übertragungsverzögerung beruhenden Einstellungen zu verhindern, bietet *Moversight* ein Adaptionsverfahren für eine gruppenweite Abschätzung der Übertragungsverzögerung.

5) Integration in existierende Systeme mittels eines dezentralen Ansatzes *Moversight* arbeitet ohne zusätzliche Infrastrukturelemente nach einem vollständigen P2P-Ansatz[1]. Die gruppenorientierte Kommunikation realisiert ein Anwendungsoverlay, das auf dem UDP/IP-Stack[2]aufbaut und mit dem die Anwendung über die Protokollschnittstellen interagiert.

6) Kontinuierliche Anpassung der Gruppe an die sich ändernden Netz- und Gruppenbedingungen Die Fehlerbehandlungsroutinen von *Moversight* realisieren eine kontinuierliche Anpassung der Topologie an die Kommunikationsbedingungen und passen die Gruppe proaktiv an geänderte Netz- und Gruppenbedingungen[3] an. Über das neuartige verteilte Latenzschätzverfahren werden die Überwachungsfunktionen der Transfer- und Fehlerdetektionsdienste an die aktuelle Kommunikationssituation adaptiert, um unnötige Sendewiederholungen und Perioden ohne Protokollfortschritt zu vermeiden.

7) Adaptierende verteilte gruppenweite Koordination Mithilfe der verschiedenen Dienste ermöglicht *Moversight* eine gruppenweite Koordination zwischen gleichberechtigten Teilnehmern. Durch den P2P-Ansatz kann auf zentrale Instanzen verzichtet werden. Innerhalb des Overlays werden den einzelnen Mitgliedern Rollen zugewiesen, welche beispielsweise zeitlich oder nach Ressourcensituation wechseln. Koordiniert wird der Rollenwechsel auf Basis von Gruppenereignissen, welche aus der Nachrichtenordnung des TOM-Dienstes abgeleitet werden.

8) Anwendungsgetriebene Gruppenteilung bzw. Gruppenvereinigung Auf semantisch höherer Ebene kann die Gruppe durch die Dienste *Split* und *Merge* an sich ändernde Anforderungen der Kollaboration dynamisch angepasst werden. Es ist möglich, eine Gruppe in Untergruppen aufzuteilen, die getrennt voneinander Teilprobleme/-aspekte bearbeiten. Diese Teilgruppen können sich wiedervereinigen. Auch eine Vereinigung unabhängiger Gruppen zu einer gemeinsamen Gruppe ist möglich. Dabei wird das gemeinsame Wissen gesichert, d. h. beide Teilgruppen behalten ihr bis dahin erworbenes Gruppenwissen. Die Wiedervereinigung erfolgt in einem Schritt unter Wahrung der Konsistenz. Die Neubildung einer Gruppe wird ebenfalls in einem Schritt bei Beibehaltung des Gruppenwissens beider Teilgruppen ausgeführt.

10.2 Ausblick auf mögliche fortführende Arbeiten

Der *Moversight*-Ansatz biete eine Reihe von Möglichkeiten für weiterführende Untersuchungen zur Vervollkommnung bzw. Optimierung.

In den für die Wiederherstellung der Gruppenkonsistenz verwendeten Abgleichstrategien ruft ein zurückkehrender Peer die verpassten Nachrichten zumeist bei seinem Master ab. Die Definition von weiteren Abfrageschemen ist denkbar. So können unterschiedliche Peereigenschaften für die Selektion der abzufragenden Peers verwendet werden, wie beispielsweise die Rolle der Peers innerhalb der Kommunikationstopologie. Der Verbindungsstatus der Peers, die Dauer ihrer Gruppenmitgliedschaft, ihr Ressourcenwert, ihr Mobilitätstyp oder die Anzahl der durch den Peer versendeten Gruppennachrichten eines Peers etc. sind weitere mögliche Selektionseigenschaften. Potentiell könne so die

[1] Nur im Rahmen der Benutzerlokalisation, die nicht Bestandteil von*Moversight* ist, kann es unter Umständen zu externen Abhängigkeiten kommen.

[2] Alternativ kann auch TCP/IP verwendet werden.

[3] Beispielsweise: Anzahl der Gruppenmitglieder, individuelle Ressourcenausstattung einzelner Teilnehmer, Übertragungsverzögerung zwischen den Teilnehmern und Anzahl von Paketverlusten bzw. notwendige Sendewiederholungen.

Aufwände pro Peer für die Resynchronisation sowie die Dauer der Operation reduziert werden.

Bei diesen Strategien werden entweder einzelne Peers, eine Teilmenge oder alle Peers aus der Gruppe als Synchronisationsquelle genutzt. Für die Verteilung der Anfrage der Peers wäre es hilfreich, die Menge der vermissten Nachrichten zu kennen. Dazu müsste der wiedereintretende Peer die Referenz seiner letzten lokal stabilen Nachricht an den Kontaktpeer übermitteln. Dieser kann mit dessen Hilfe die Anzahl der verpassten Nachrichten bestimmen und sie an den wiedereintretenden Peer übermitteln. Daraufhin kann der wiedereintretende Peer die Menge der verpassten Nachrichten auf die Menge der Synchronisationsquellen verteilen und ihnen die zu übermittelnde Teilmenge bekanntgeben. Damit kann der Aufwand für die Bereitstellung der verpassten Nachrichten auf mehrere Peers verteilt werden.

Die Abgleichstrategien bewerten die Peers über die Anzahl von Hops innerhalb des Overlays. Dieses Verfahren könnte durch die Definition eines „*nahen*“ Peers ersetzt werden, welcher auf der Basis virtueller Koordinaten [40] oder seiner *GPS*-Position die Synchronisationsquellen bestimmt. Dabei wird unterstellt, dass der Austausch mit „*nahen*“ Peers einen geringeren Ressourcenverbrauch als beim Austausch mit „*fernen*“ Peers hat. Gründe für diese Annahme sind die theoretisch geringere Nachrichtenverzögerung, die geringere Anzahl von Nachrichtenverlusten und der daraus resultierenden geringeren Resynchronisationsdauer im Vergleich zu den verwendeten Abgleichstrategien.

Die Leistungsbeurteilung des in dieser Arbeit vorgestellten Konzeptes hat gezeigt, dass der *LEAVE*/*JOIN*-Ansatz bei kleinen Gruppen effizienter ist als der hier entwickelte Rejoin-Ansatz. Daher wäre eine Kombination beider Verfahren zu prüfen. So könnte die Gruppe, sofern weniger als drei Peers die Verbindung zu dieser verlieren, diese per *LEAVE* ausschließen. Die betreffenden Peers behalten lokal jedoch ihre Gruppenmitgliedschaft bei. Kann ein Peer die Gruppe wieder erreichen, wird der Wiedereintritt gestartet. Der Wiedereintrittswunsch wird mit einer Einladung aus der Gruppe beantwortet, welche der betroffene Peer annimmt. Der einladende Peer kennzeichnet bei der Bekanntmachung den Peer als wieder eintretenden. Nach dem erfolgreichen Eintritt in die Gruppe wird der Wiedereintritt mit der *RESYCNHRONIZE* Operation abgeschlossen. Nachteil dieses Ansatzes wäre jedoch die unterbrochene Gruppenmitgliedschaft der betroffenen Peers. Aufgrund dessen stellt sich die Frage, welche Daten an den wieder eintretenden Peer übermittelt werden dürfen, die während seiner Abwesenheit in der Gruppe ausgetauscht wurden.

Der vorgestellte Ansatz zur Behandlung der Nutzermobilität zielt auf mobile kollaborative Anwendungen. Denkbar wäre auch die Nutzung des Ansatzes in Cloudumgebungen. So könnten einzelne Cloudinstanzen (z. B. Berechnungs- oder Speicherinstanzen) als Peers verstanden werden. Die durch das dynamische Zu- und Abschalten notwendige Resynchronisation könnte über den Mobilitätsansatz diese Arbeit gelöst werden. Dazu bilden alle Instanzen eine kollaborative Gruppe. Deren Abschaltung wird als Verbindungsverlust modelliert, deren erneute Zuschaltung als Wiedereintritt in die Gruppe. Dadurch könnte sich die Gruppe selbst verwalten und wäre nicht auf Loadbalancer und zentrale Koordinationsserver etc. angewiesen.

Der in dieser Arbeit entwickelte Ansatz der mobilen optimistischen Synchronität kann für eine private mobile Cloudlösung verwendet werden. Dabei werden Nutzerdaten von verschiedenen stationären und mobilen Endgeräten innerhalb der Gruppe abgeglichen. Nach dem Eintritt in die „*Cloudgruppe*“ werden sämtliche geteilten Daten über das MOVS-Konzept konsistent zwischen den Gruppenmitgliedern geteilt. Die zu teilenden Dokumente werden fragmentiert und über den TOM an die Gruppe gesendet. Die mobile Cloudanwendung setzt die Dokumente im Endgerät wieder zusammen und stellte diese dem Nutzer zur Verfügung. Für die Umsetzung muss der Zeitraum, die ein Peer maximal von der Gruppe getrennt sein kann, deutlich erweitert und eine Strategie für eventuelle Konflikte bei konkurrierendem Dokumentenzugriff definiert werden. Ebenso sollte die Kommunikation von Moversight vollständig verschlüsselt werden. Der Vorteil dieser mobilen Cloudlösung ist der vollständige dezentrale Ansatz. Durch das Fehlen einer zentralen Instanz und dem ausschließlichen Einsatz von Endgeräten des oder der Endanwender haben dritten Parteien keinen Zugriff auf die Daten. Dadurch wird ein optimalen Schutz der Privatsphäre sicherstellt.

Teil III

Anhang

Anhang A

Ergänzungen zum Kapitel Sicherung der Konsistenz des globalen Wissens

A.1 Allgemeine Definitionen

- $\mathbb{P}$: Die Menge der möglichen Nutzer (engl. *Peers*) einer mobilen, kollaborativen Anwendung.
- Peer p: Ein aktives Mitglied einer kollaborativen Gruppen (auch Knoten, Prozess oder Mitglied genannt).
- $\mathbb{M}$: Die Menge der Nachrichten, welche durch die Anwendung an die Gruppe versandt werden.
- $\mathbb{VID}$: Die Menge der Sichtenbezeichner (engl. *View* IDs), welche partiell über den Operator „<“ geordnet ist (auch Ordnung der Sichten bzw. View Order genannt).
- V: Die *Sicht* oder *View* umfasst eine Menge von Peers, welche zu einem definierten Zeitpunkt innerhalb der Gruppe als aktive Mitglieder bekannt sind. Diese ist definiert als:

  $$V = (id, members) \tag{A.1}$$

  mit $id \in \mathbb{VID}$ als Sichtkennung und $members \in 2^{\mathbb{P}}$ als Menge der aktiven Mitglieder der aktuellen Sicht.

  Ändert sich die Sicht des Gruppenverwaltungsdienstes, so wird ein Sichtänderungsereignis (engl. *view change event*) durch den Dienst ausgeliefert, welches die neue Sicht enthält. Somit ist die Sicht ein 2er Tupel, auf dessen Elemente mit $V.id$ bzw. $V.members$ zugriffen werden kann.
- $\mathbb{V}$: Die Menge der durch ein Sichtänderungsereignis ausgelieferten Sichten ist definiert als: $\mathbb{VID} \times 2^{\mathbb{P}}$.

A.2 Schnittstellen des Gruppenkommunikationssystems

Das Gruppenkommunikationssystem interagiert mit seiner Umwelt über eine Reihe von Schnittstellen, wobei nach Chockler *et al.* [55] die minimale Schnittstelle definiert ist als:

- *send*: Für den Versand einer Nachricht über das Gruppenkommunikationssystem durch die Anwendung, formal:

  $$send(p, m), p \in \mathbb{P}, m \in \mathbb{M} \tag{A.2}$$
- *recv*: Für den Empfang von Nachrichten aus der Gruppe, welche über das Netz versandt wurden, formal:

  $$recv(p, m), p \in \mathbb{P}, m \in \mathbb{M} \tag{A.3}$$

- *view_chng*: Zum Empfang von Sichtänderungsereignissen, welche alle Empfänger informieren, dass eine neue Sicht V im Gruppenkommunikationssystem beim Peer p eingerichtet wurde. Aus Darstellungsgründen formal vereinfacht:

$$view\_chng(p, V), p \in \mathbb{P}, V \in \mathbb{VID} \tag{A.4}$$

- *receives*: Peer p empfängt Nachricht m

$$receives(p, m) == \exists i : t_i = recv(p, m) \tag{A.5}$$

- *receives_in*: Peer p empfängt Nachricht m in Sicht V.

$$receives\_in(p, m, V) == \exists i : (t_i = recv(p, m) \wedge viewof(t_i) = V) \tag{A.6}$$

- *sends*: Peer p sendet die Nachricht m.

$$sends(p, m) == \exists i : t_i = send(p, m) \tag{A.7}$$

- *sends_in*: Peer p sendet Nachricht m in Sicht V.

$$sends\_in(p, m, V) == \exists i : (t_i = send(p, m) \wedge viewof(t_i) = V) \tag{A.8}$$

- *installs*: Peer p installiert Sicht V.

$$installs(p, V) = \exists i : t_i = view\_chng(p, V) \tag{A.9}$$

- *installs_in*: Peer p installiert Sicht V in V'.

$$installs\_in(p, V, V') = \exists i : (t_i = view\_chng(p, V) \wedge viewof(t_i) = V') \tag{A.10}$$

  Die Operation $viewof(t_i) : e \Rightarrow V$ gibt für ein Ereignis t_i bei Peer p an, in welcher Sicht V dieses Ereignis stattfand. Die detaillierte formale Definition der Operation *viewof()* ist in Chockler *et al.* zu finden.

Eine umfangreiche Einführung zum Thema Nachrichtenordnung in Prozessgruppen enthält [136].

A.3 Eigenschaften von Gruppenkommunikationssystemen

Die Definition der nachfolgenden Eigenschaften erfolgt in Anlehnung bzw. entsprechend der Notation von Chockler *et al.* [55].

A.3.1 Grundeigenschaften

- *Selbstinklusion*:

$$installs(p, V) \Rightarrow p \in V.members \tag{A.11}$$

- *lokaleMonotonie*:

$$t_i = view\_change(p, V) \wedge t_j = view\_change(p, V') \wedge i > j \Rightarrow V.id > V'.id \tag{A.12}$$

- *Sichtsequenz*: Die Sichtsequenz f (engl. *trace*) ist definiert als eine Abbildung von der Menge der installierten Sichten auf die natürlichen Zahlen.

$$f : \{V \mid \exists p : installs(p, V)\} \Rightarrow \mathbb{N} \text{ mit:} \tag{A.13}$$

$$(f(V) = f(V') \Rightarrow V = V') \wedge$$
$$\forall V(f(V) > 1 \Rightarrow \exists V' : (f(V) = f(V') + 1 \wedge$$
$$\exists p \in V.members : installs\_in(p, V, V'))). \tag{A.14}$$

- *Zustellungsintegrität*:

$$t_i = recv(p, m) \Rightarrow \exists q, \exists j : (j < i \wedge t_j = send(q, m)) \tag{A.15}$$

- *Duplikationsfreiheit*:

$$t_i = recv(p, m) \wedge t_j = recv(p, m) \Rightarrow i = j \tag{A.16}$$

A.3.2 Zustellungseigenschaften

- *FIFO-Zustellung*:

$$t_i = send(p,m) \wedge t_j = send(p,m') \wedge i < j \wedge \\ t_k = recv(q,m) \wedge t_l = recv(q,m') \wedge k < l \tag{A.17}$$

- *zuverlässige FIFO-Übertragung*:

$$t_i = send(p,m) \wedge t_j = send(p,m') \wedge i < j \wedge \\ viewof(t_i) = viewof(t_j) \wedge \\ receives(q,m') \Rightarrow recv\_before(q,m,m') \tag{A.18}$$

- *kausale Ordnung zweier Ereignisse*:

$$t_i \rightarrow t_j \Leftrightarrow j >= i \vee \\ (t_i = send(p,m) \wedge t_j = recv(q,m)) \vee \\ \exists k : (t_i \Rightarrow t_k \wedge t_k \Rightarrow t_j) \tag{A.19}$$

- *kausale Zustellung*:

$$t_i = send(p,m) \wedge t_j = send(p',m') \wedge \\ t_i \rightarrow t_j \wedge t_k = recv(q,m) \wedge \\ t_l = recv(q,m') \Rightarrow k < l \tag{A.20}$$

- *zuverlässige kausale Übertragung*:

$$t_i = send(p,m) \wedge t_j = send(p',m') \wedge \\ t_i \rightarrow t_j \wedge viewof(t_i) = viewof(t_j) \wedge \\ receives(q,m') : recv\_before(q,m,m') \tag{A.21}$$

- *totale Ordnung*:
  - *strenge totale Ordnung*:

$$ts(f) \Leftrightarrow f : \mathbb{M} \Rightarrow \mathbb{N} \wedge \\ f(m) = f(m') \Rightarrow m = m' \tag{A.22}$$

$$\exists f : ts(f) \wedge \\ \forall p, \forall m, \forall m' : recv\_before(p,m,m') \\ \Rightarrow f(m) < f(m') \tag{A.23}$$

  - *schwache totale Ordnung*:

$$\forall V, \forall V', \exists f : ts(f) \wedge \\ \forall p, \forall m, \forall m' : (installs\_in(p,V,V') \wedge \\ recv\_before\_in(p,m,m',V') \Rightarrow f(m) < f(m')) \tag{A.24}$$

A.3.3 Eigenschaften der Gruppenverwaltung

- *stabile Komponente*: Solche Peers, die fehlerfrei symmetrisch miteinander kommunizieren können (im Sinne von: nicht abgestürzt, mit dem Netz verbunden und kommunikationsfähig) werden als stabile Komponenten bezeichnet. Sie bilden den transitiven Abschluss über die Menge aller fehlerfreien, bidirektional verbundenen Peers. Eine formale Herleitung der stabilen Komponente ist in [55] zu finden.

- *letzte Sicht*:

$$last\_view(p, V) \Leftrightarrow \exists i : (t_i = view\_chng(p, V) \wedge \nexists j > i \; \exists V' : t_j = view\_chng(p, V')) \tag{A.25}$$

- *Gruppenmitgliedschaftsexaktheit* (engl. *Membership Precision*): Die Gruppenmitgliedschaftsexaktheit ist eine Lebendigkeitseigenschaft der Gruppenkommunikationssysteme, welche für die erfolgreiche Zustellung von Multicastnachrichten in der Gruppe notwendig ist. Eine ausführliche Einführung zum Thema Lebendigkeitseigenschaften ist wiederum bei Chockler *et al.* [55] gegeben.

  Für die exakte Bestimmung der Gruppenmitgliedschaft (engl. *Membership*) wird ein Fehlerdetektor FD der Klasse *eventually perfect* ($\diamond P$) benötigt. Für diesen gilt, dass der Fehlerdetektor nach einem Ausfall eines Peers der Gruppe ab einem bestimmten Zeitpunkt den Zustand aller Peers der Gruppe korrekt einschätzt. Mit Hilfe dieses Detektors kann dann die Gruppenmitgliedschaft exakt bestimmt werden über:

$$\diamond P \text{ – like FD} \wedge stable\_component(S) \Rightarrow \exists V : (V.members = S \wedge \forall p \in S : last\_view(p, V)) \tag{A.26}$$

- *Empfangsselbstinklusion* (engl. *Self Delivery*): Die Empfangsselbstinklusion besagt, dass jeder Sender einer Nachricht diese auch als Empfänger zugestellt bekommt, solange er nicht aufgrund eines Fehlers ausfällt. Das Scheitern der Sendeoperation ist entweder über einen Timer zu überwachen oder wird dem Sender über ein Ereignis angezeigt. Die Funktion $crash(p)$ mit $p \in \mathbb{P}$ gibt den Zeitpunkt des Ausfalls für Peer p zurück, oder ∞ für den fehlerfreien Fall:

$$\diamond P \text{ – like FD} \wedge stable\_component(S) \Rightarrow \exists V : (V.members = S \wedge \forall p \in S : t_i = send(p, m) \wedge \nexists j > i : t_j = crash(p) \Rightarrow (receives(p, m) \vee send\_failed\_event(p, m)) \tag{A.27}$$

- *Zustellungslebendigkeit* (engl. *Safe Indication Liveness*): Für die erfolgreiche Zustellung von Nachrichten muss das Gruppenkommunikationssystem für jede einzelne Nachricht, welche durch einen Peer p in Sicht V versandt wurde, sicherstellen, dass der Empfang durch alle Peers in S bestätigt wird.

$$\diamond P \text{ – like FD} \wedge stable\_component(S) \Rightarrow \exists V : (V.members = S \wedge \forall p \in S : sends\_in(p, m, V) \Rightarrow \forall p \in S : indicated\_safe(q, m, V) \tag{A.28}$$

  Die Funktion $indicated\_safe$ gibt dabei an, dass ein Peer q dem Sender den Empfang der Nachricht m in der Sicht V bestätigt.

Anhang B

Ergänzungen zum Kapitel Moversight

B.1 Auswahl einer geeigneten Topologie

Die Overlaytopologie bestimmt das Interaktionsschema der einzelnen Gruppenmitglieder, deren Rollen sowie Verantwortlichkeiten und beeinflusst damit wesentlich die Eigenschaften des Kommunikationssystems. Die Auswahl einer geeigneten Topologie soll nach den Kriterien Ressourceneffizienz, Robustheit bezüglich Kommunikationsausfällen sowie Übertragungsverzögerung erfolgen.

B.1.1 Grundannahmen und Einordnung in die Netzarchitektur

Mobile kollaborative Anwendungen setzen eine Netzschicht sowie eine UDP/IP-basierte Transportschicht voraus, auf welchen sie aufbauen. Weiter wird unterstellt, dass die Netzverbindungen transitiv und bidirektional sind. Angenommen, es existiert eine „*erreicht*“ Kommunikationsfunktion $\rightarrow$, dann gilt für die Knoten Gleichung B.1.

$$A, B, C : A \rightarrow B \wedge B \rightarrow C \Rightarrow A \rightarrow C \tag{B.1}$$

Aus der Annahme der Bidirektionalität eines Kommunikationskanals folgt:

$$C \rightarrow A$$

Die Netz- und die Transportschicht sind verantwortlich für das Routing sowie für die Bereitstellung eines virtuellen Transportkanales zwischen beliebigen Knoten im Netz (evtl. unter Nutzung von Multi-Hop-Verbindungen bzw. der virtuellen Kopplung mehrerer $1-1$-Verbindungen). Um verschiedene Netzarten zu koppeln, wird eine Netzschicht, vergleichbar zum *Unified Networking* (siehe Abschnitt 2.3) vorausgesetzt, welche verantwortlich für deren Kopplung zu einem gemeinsamen virtuellen Netz ist. Dadurch können beispielsweise die Ad-hoc-Netze im Einsatzgebiet der Roboter sowie dort bestehende Infrastruktur- bzw. zelluläre Netze gekoppelt werden. Entsprechend des OSI-Schichtenprinzips [118] soll der gruppenorientierte Kommunikationsdienst so entwickelt werden, dass er unabhängig von den zugrundeliegenden Schichten ist und nur über definierte Dienstzugangspunkte mit diesen interagiert. Daher sollen bei der Auswahl der Kommunikationstopologie nur die logischen Verbindungen zwischen den einzelnen Gruppenmitgliedern betrachtet werden, nicht aber die Anzahl tatsächlichen Verbindungen zwischen ihnen auf Netzebene.

Abbildung B.1 stellt diesen Unterschied beispielhaft für eine virtuelle Verbindung zwischen zwei Gruppenmitgliedern A und B dar. Gruppenmitglied A befindet sich in einem Ad-hoc-Netz, Gruppenmitglied B in einem kabelgebundenen Netz. Der virtuelle Transportkanal wird über einen zwischenliegenden Knoten aufgebaut, welcher Mitglied in beiden Netzen ist. Ob dieser Zwischenknoten ebenfalls ein Gruppenmitglied ist oder – wie beispielsweise bei SpoVNet – ein gruppenfremdes Mitglied des Underlays, soll nicht Gegenstand der Betrachtung sein.

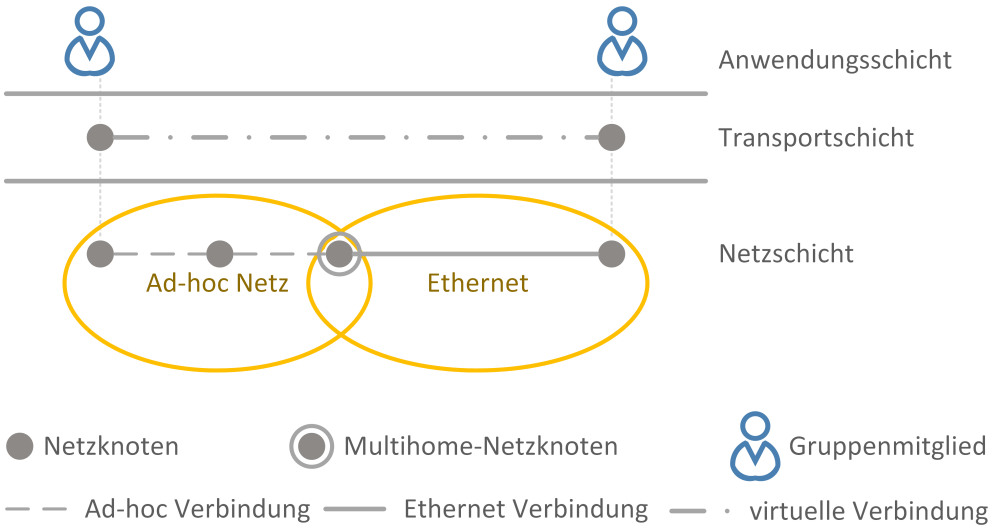


Abbildung B.1: Unterschied zwischen virtueller und tatsächlicher Verbindung

Wie im Kapitel 2 ausgeführt, sind mobile kollaborativen Anwendung auf Anwendungsebene dezentral und unabhängig von einer spezifischen Anwendungsinfrastruktur zu realisieren. Daher sind für das Overlay Topologien zu prüfen, welche ohne eine feste Infrastruktur umgesetzt werden können. Folgenden Topologien sind zu untersuchen:

- Ringtopologie,
- voll vermaschte Topologie,
- virtueller Pfadtopologie,
- (einstufige) Clustertopologie,
- Binärbaumtopologie und
- k-Baumtopologie.

Beispielsweise wird die Ringtopologie beim Gruppenkommunikationsprotokoll GCP [231] verwendet. Das Gruppenkommunikationsprotokoll *ISIS* [32] nutzt eine voll vermaschte Topologie. Dressler und Gerla präsentieren in [70] eine Lösung für das Datenmanagement in mobilen Ad-hoc-Netzen unter Nutzung einer virtuellen Pfadtopologie, welche über ein vermaschtes Ad-hoc-Netz aufgespannt wird. Der Middlewareansatz SpoVNet [34] und das Gruppenkommunikationssystem Transis [69] nutzen jeweils eine clusterbasierte Topologie für die Kommunikation unter den Gruppenteilnehmern. Verschiedene baumbasierte Topologien werden beispielsweise in den Application-Level Multicastsystemen Narada [57] und NICE [19] genutzt.

B.1.2 Aspekt der Ressourceneffizienz und der Robustheit bei Kommunikationsausfällen

Tabelle B.1 gibt einen Überblick über die Anzahl der zu verwaltenden Verbindungen je Gruppenmitglied sowie die Anzahl der in der Gruppe von n Mitgliedern insgesamt zu verwaltenden Verbindungen.

Abbildung B.1.2 stellt die Gesamtanzahl der Verbindungen in den einzelnen Topologien für Gruppen bis zu 100 Mitgliedern graphisch dar. Zu erkennen ist, dass eine voll vermaschte Topologie die meisten Verbindungen aufweist. Die geringste Anzahl von Verbindungen zwischen den Gruppenteilnehmern entsteht bei der Verwendung von Ring-, Binärbaum-, k-Baum- oder Pfadtopologien. Diese sind jedoch wegen der geringen Anzahl von Verbindungen anfällig für Verbindungsunterbrechungen, weil das Fehlen

Tabelle B.1: Verbindungszahlen verschiedener Topologien

Topologie	**Verbindungen je Mitglied**	**Gesamtanzahl der Verbindungen**
Ring	2	n
Virtueller Pfad	1, für Start und Endknoten 2, sonst	$n-1$
Vollvermascht	$n-1$	$(n^2-n)/2$
Cluster	1, für Slaves $(\#M-1)+(\#S_x)$, für Master x $\#M$ die Anzahl der Master in der Gruppe, $\#S_x$ die Anzahl von Slaves im Cluster x, welcher durch den Master x verwaltet wird	$(n-\#M)+(\#M^2-\#M)/2$
Binärbaum	1, für Blattknoten 2, für Halbblätterknoten 3, sonst (für innere Knoten)	$n-1$
k-Baum	Zwischen 1 und $(k+1)$ Verbindungen (je nach Anzahl der Kinder pro Knoten)	$n-1$

redundanter Übertragungspfade zu einem vollständigen oder teilweisen Kommunikationsausfall innerhalb der Gruppe führen kann. Die Verwendung einer Clustertopologie stellt demnach einen Kompromiss zwischen Ausfallsicherheit und Begrenzung des Aufwandes für die Verbindungsverwaltung dar.

Die Anzahl der zu verwaltenden Verbindungen der einzelnen Topologien pro Gruppenmitglied wurde in Kapitel 5 Abbildung 5.3 grafisch dargestellt.

B.1.3 Aspekt der Übertragungsverzögerung

Neben den Aspekten Ressourceneffizienz und Robustheit ist die bei einer Multicastübertragung innerhalb der Gruppe entstehende Übertragungsverzögerung für die Auswahl der Topologie relevant. In mobilen kollaborativen Anwendungen ist eine gruppenweite Multicastübertragung das häufigste Kommunikationsschema, begründet durch die gemeinsame Kollaboration (im Sinne einer *one-to-many*-Semantik).

Tabelle B.2 zeigt die entstehenden Übertragungsverzögerungen für die einzelnen Topologien und verschiedene Gruppengrößen. Die Abschätzung erfolgt unter Berücksichtigung folgender Vereinfachungen und Annahmen:

- Die Übertragungsverzögerung ist auf allen Verbindungen gleich.
- Die Übertragung einer Nachricht über eine Verbindung erzeugt eine Verzögerung von 1 s, unabhängig von der tatsächlichen Nachrichtengröße oder der auf der Verbindung zur Verfügung stehenden Übertragungskapazität.

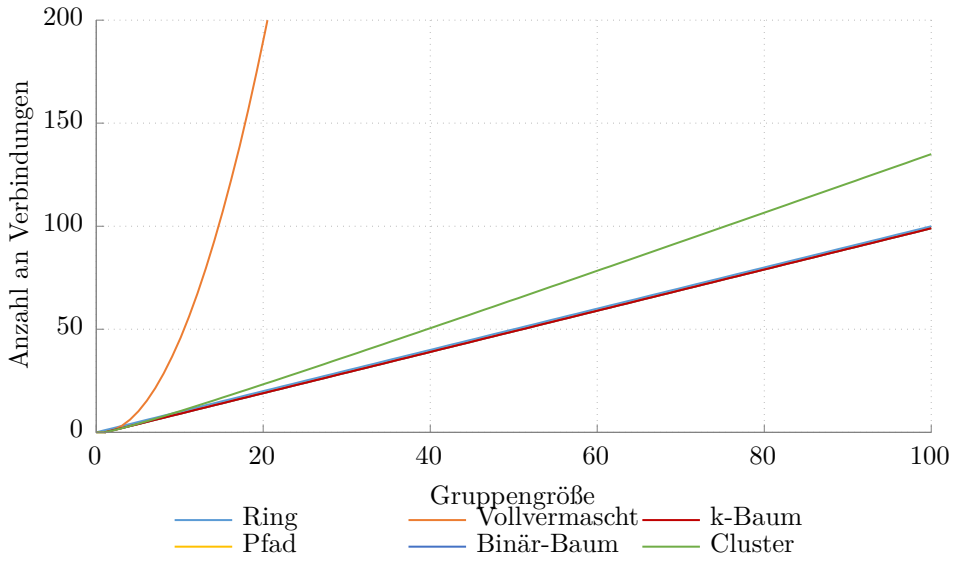


Abbildung B.2: Anzahl von Verbindungen in der Gruppe

Tabelle B.2: Abschätzung der Übertragungsverzögerungen für verschieden Topologien und Gruppengrößen

Topologie	**Gruppe**			
	10er	**40er**	**70er**	**100er**
Ring	10 s	40 s	70 s	100 s
Pfad	18 s	78 s	138 s	198 s
Binärbaum	8 s	12 s	14 s	16 s
k-Baum	4 s	6 s	6 s	6 s
Cluster	4 s/6 s	4 s/6 s	4 s/6 s	4 s/6 s
Vollvermascht	2 s	2 s	2 s	2 s

- Die Verzögerung für die Serialisierung und Verarbeitung wird mit 0 s angenommen.
- Jede Nachrichtenübertragung ist durch den Empfänger zu bestätigen.
- Hat ein Knoten mehrere Kinder, so wird vereinfachend angenommen, dass die Übertragung an alle Gruppenmitglieder „*quasi*“-parallel erfolgt.
- Mögliche Einflüsse wie Paketverluste, Verzögerungen oder sich gegenseitig beeinflussende Übertragungen werden ignoriert.

Folglich entspricht das Ergebnis (die zu erwartende Übertragungsverzögerung) einer theoretischen unteren Schranke (im Sinne einer „*best case*“ Abschätzung) und ist in einer realen Umsetzung nie erreichbar. Dennoch kann sie für eine verhältnis- und größenordnungsmäßige Einordnung der zu erwartenden Verzögerungen innerhalb der mobilen kollaborativen Anwendungen dienen.

Bei der Ringtopologie ist keine explizite Bestätigung notwendig, da der Sender einer Nachricht als erster Ringknoten diese Nachricht vom letzten Ringknoten übermittelt bekommt. Der rechtzeitige Empfang der versendeten Nachricht vom Vorgängerknoten des Senders kann als implizite Gruppenbestätigung genutzt werden und beendet die Weiterleitung der Nachricht. Aus den n-Verbindungen innerhalb des Ringes (vergleiche

Tabelle B.1) ergeben sich somit n-Einzelübertragungen mit je 1 s Dauer. Wie in Tabelle B.1 zu erkennen, ist für größere Gruppen die durch die Ringtopologie entstehende Übertragungsverzögerung ungünstig.

Bei der Pfadtopologie wird die erfolgreiche Übermittlung der Nachricht an den Sender durch den letzten Knoten des Pfades bestätigt. Die Bestätigung durchläuft den Pfad invers und erzeugt somit eine Verdoppelung der Übertragungsverzögerung, wodurch diese Topologie noch ungünstiger für die Nutzung in mobilen kollaborativen Anwendungen ist. Für eine Binärbaumtopologie wird o.B.d.A. angenommen, dass diese einen balancierten vollständigen bzw. fast-vollständigen Binärbaum ausbildet. Jeder Knoten des Baumes entspricht einem Gruppenmitglied. Die maximale Übertragungsverzögerung wird daher durch die Übertragung an den Knoten in der größten Tiefe bestimmt.

Für einen Baum der Höhe h ist die Anzahl der Knoten n gleich $2^h - 1$. Damit entspricht die maximale Pfadlänge p eines balancierten Baumes $p = log_2(n) + 1$ (n gleich Gruppengröße). Auch im Binärbaum müssen die empfangen Nachrichten von den Blättern bestätigt werden, um die Übertragung abzusichern, wodurch die Gesamtübertragungsverzögerung verdoppelt wird. Es wird weiter angenommen, dass die inneren Knoten Bestätigungen aggregieren, wodurch keine weiteren Verzögerungen auftreten. Für die angegebenen Gruppengrößen in der Tabelle B.1 ergeben sich Pfadlängen von 4, 6, 7 und 8 Verbindungen, was zu einer deutlich reduzierten Übertragungsverzögerung im Vergleich zu der Ring-oder Pfadtopologie führt. Die Pfadlänge für den k-Baum wird analog zum Binärbaum bestimmt $p = log_k(n) + 1$. Für $k = 7$ ergeben sich folglich die gerundeten Pfadlängen von 2, 3, 3 und 3 Verbindungen.

Bei der einstufigen Clustertopologie agieren die Knoten entweder als Slave oder als Master. Wird die Nachricht durch einen Master versandt, ist diese innerhalb seines Clusters zu verteilen. Sie muss auch an die übrigen Master weitergeleitet werden, welche wiederrum die Nachricht an die Slaves ihres Clusters verteilen. Auch hier sind alle Nachrichten zu bestätigen. Dies ergibt im ungünstigsten Fall eine Pfadlänge von 2. Diese ist unabhängig von der gewählten Gruppengröße.

Eine Ausnahme stellen kleine Gruppen dar, welche nur aus einem einzelnen Cluster bestehen, bzw. Cluster, welche nur aus einem Masterknoten bestehen. Für diese ist die Pfadlänge gleich eins. Ist der Sender der Nachricht ein Slave, so erfolgt der Versand über dessen Masterknoten, was die Pfadlänge jeweils um eins erhöht. In der Tabelle B.1 ist daher für jede Gruppengröße bei der Clustertopologie sowohl eine Master- als auch eine Slave-Übertragung für den allgemeinen Fall angegeben (im Sinne von: M/S).

Die Clustertopologie hat im besten Fall eine geringere Übertragungsverzögerung gegenüber der Ring-, der Pfad- und der baumbasierten Topologie. Die geringste Übertragungsverzögerung aller Topologien weist die voll vermaschte Topologie auf, in welcher ein Sender theoretisch parallel an alle Empfänger der Gruppe senden kann und von diesen im Anschluss eine Bestätigung erhält.

B.1.4 Schlussfolgerung

Zusammenfassend lässt sich feststellen, dass eine clusterbasierte Topologie den besten Kompromiss aus Robustheit, Verwaltungsaufwand, Ressourcenverbrauch und Übertragungsverzögerung bietet und daher deren Nutzung für die Umsetzung eines gruppenorientierten Kommunikationssystems naheliegt.

B.2 Peer Eigenschaften

In Tabelle B.3 sind die wesentlichen Merkmale eines *Moversight*-Peers aufgelistet.

Die *Transportadresse* abstrahiert die Netzadresse eines Peers entsprechend des zugrundeliegenden Netz- und Transportprotokollstapels. Bei der empfohlenen Verwendung eines IP/UDP-Protokollstapels ist dies ein Tupel aus IP*v4*/IP*v6*-Adresse sowie entsprechenden UDP-Port.

Die für jeden Peer verwaltete Peerbeschreibung (engl. *Peer Description*) ist ein anwendungsspezifisches Merkmal des Gruppenteilnehmers. Diese Eigenschaft wird nicht vom Gruppenkommunikationsprotokoll selbst verwendet, sondern durch die Anwendung bestimmt. Als Beispiel sei hier die Nutzerkennung innerhalb der kollaborativen

Tabelle B.3: Eigenschaften eines *Moversight*-Peers

Eigenschaft	**Beschreibung**
Peer ID	Eindeutige Identifikationsnummer innerhalb der Gruppe (siehe Anhang A)
Transportadresse	Abstraktion der Netzadresse des Peers.
Zustand (*Peer State*)	Der Zustand des Peers
Cluster Referrer	Eine Referenz auf den Cluster innerhalb des Overlays, zu dem der Peer zugeordnet wurde.
Rolle	Rolle des Peers innerhalb der Clustertopologie, im Sinne von Slave oder Master.
Peerbeschreibung	Anwendungsspezifisches Merkmal des Gruppenteilnehmers.
Peerressourcen	Abstraktion der verfügbaren Teilnehmer-ressourcen und dessen Mobilitätstyp.

Anwendung genannt (wie: *nutzer_123@beispiel.anwendung.com*). Diese kann bei der Ermittlung einer Endgerätetransportadresse verwendet werden, welche für die Einladung von neuen Teilnehmern notwendig ist. Ebenso kann sie für das NAT-/Firewall-Traversal genutzt werden (siehe Abschnitt 2.3).

Die Peerressourcen (engl. *Peer Resources*) abstrahieren die verfügbaren Ressourcen sowie den Mobilitätstyp des Teilnehmers als 3er Tupel. Die Funktion dieses Tupels wird im Kapitel 8 erläutert. Verkürzt dargestellt, gibt er unter anderem Auskunft darüber, ob der Peer aktuell sich eher mobil oder stationär verhält und wieviel Ressourcen diesem zur Verfügung stehen (normiert in einem Bereich von 0 bis 100%).

B.3 Ereignisse des TOM-Transferdienstes

Folgende Ereignisse werden durch den TOM-Transfer ausgesendet:

Mulitcast-Nachricht-Ausgeliefert – engl. *Multicast-Message-Delivered* – zeigt an, dass eine Multicastnachricht an die Dienste bzw. die Anwendung ausgeliefert wurde.

Mulitcast-Nachricht-Verworfen – engl. *Multicast-Message-Dropped* – zeigt an, dass die Verarbeitung einer empfangenen Multicastnachricht beendet wurde, da in der Zwischenzeit ein Sichtwechsel stattgefunden hat bzw. diese Nachricht schon zu einem früheren Zeitpunkt durch das System verarbeitet wurde (im Sinne von bereits gesehen).

Multicast-Nachricht-Empfangen – engl. *Multicast-Message-Enqueued* – zeigt an, dass eine Multicastnachricht empfangen wurde und zur Verarbeitung in die Warteschlange des Transferdienstes eingeordnet wurde.

Multicast-Nachricht-im-Cluster-Verteilt – engl. *Multicast-Message-Pushed-To--Cluster* – zeigt an, dass eine Multicastnachricht erfolgreich durch einen Master an alle Slaves seines Clusters verteilt wurde.

Multicast-Nachricht-Zurückgewiesen – engl. *Multicast-Message-Rejected* – zeigt an, dass eine Multicastnachricht aufgrund einer Verletzung einer Zustellungseigenschaft innerhalb der Gruppe endgültig verworfen wurde. Details zu Zurückweisungsmechanismus werden in Abschnitt 6.5.2 behandelt.

Multicast-Nachricht-Versand-Fehlgeschlagen – engl. *Multicast-Message-Send-Failed* – zeigt an, dass der Versand einer Multicastnachricht an die Gruppe fehlgeschlagen ist.

Vermisste-Peers – engl. *Missed-Peers* – zeigt an, welche Gruppenteilnehmer durch den TOM-Transfer als fehlerhaft eingestuft werden, da sie nicht korrekt auf einen Übertragungsversuch reagiert haben (Details zum Thema Fehlerdetektion sind in Kapitel 6 ausgeführt).

B.4 Beispielhafte Algorithmen der Gruppenverwaltung

Im Folgenden wird schematisch gezeigt, wie die Verarbeitung der Gruppenbeitrittsankündigung in der Gruppenverwaltung erfolgt.

Algorithmus 8 Behandlung der Gruppenbeitrittsankündigung

```
1:  function HANDLEJOINANNOUNCE(msg)

2:      ▷ Prüfe Sendesichtzustellung erneut, da der Ereignisdienst mehrere sichtändern-
        den Nachrichten auf einmal ausgeliefert haben kann.
3:      if VIOLATESSENDINGVIEWPROPERTY(msg) then return
4:      end if

5:      mr ← getMemberRegister()
6:      ta ← msg.getJoinedPeerTA()
7:      des ← msg.getPeerDescription()
8:      res ← msg.getPeerResources()
9:      pId ← mr.CREATEPEER(ta, des, res)

10:     ▷ Lokaler Peer ist einladender Peer?
11:     if msg.getSourceID() == getLocalID() then
12:         ▷ Gültige Einladung?
13:         if NOT ta ∈ invitationList then return
14:         end if

15:         ▷ Sende Roster an neuen Peer
16:         SENDROSTERMESSAGE(ta, pId)
17:     end if

18:     SIGNAL(PeerJoinedEvent, pId, ta, des)
19: end function
```

B.5 Beispielhafte Algorithmen des TOM-Transferdienstes

Im Folgenden wird die Wirkungsweise einzelner Funktionen des TOM-Transfers schematisch dargestellt. Die Verteilung einer Nachricht durch den TOM-Transfer zeigt Algorithmus 9.

Algorithmus 9 Verteilung einer Gruppennachricht

```
1:  function sendMessage(m)

2:      if localPeer.role == SLAVE then
3:          b ← cluster(localPeer)
4:          a ← { p | p.role == MASTER, p ∈ b}
5:      else
6:          c ← cluster(localPeer) ∩ {localPeer}
7:          b ← { p | p.role == MASTER, p ∈ cluster, ∀ cluster ∈ group} ∩
    {localPeer}
8:          a ← b ∪ c
9:      end if

10:     SEND(m, a)
11: end function

12: function receiveMessage(m)

13:     if localPeer.role == MASTER then
14:         if m.sender ∈ cluster(localPeer) then          ▷ primärer Master
15:             SENDMESSAGE(m)
16:         else                                           ▷ sekundärer Master
17:             SEND(m, cluster(localPeer) ∩ {localPeer})
18:         end if
19:     end if

20:     STORE(m)
21: end function
```

In Algorithmus 10 wird die Herstellung der globalen Ordnung über die Verarbeitung der GT-Nachrichten illustriert.

Algorithmus 10 Die Aktualisierungs- und Auslieferungsoperation.

```
1: function updateAndDeliver(GT)

2:     queue.UPDATE(GT)
3:     queue.SORT()

4:     while queue.TOP().deliverable == TRUE do
5:         m ← queue.POP()
6:         DELIVER(m, m.pendingPeers)
7:     end while
8: end function
```

Die Nachrichteneingangsprüfung im TOM-Transferdienst ist komplex und spezifisch für jeden Zustand des Peers. Daher wird sie in der Folge für die einzelnen Zuständen separat vorgestellt. Algorithmus 11 zeigt dies für die Zustände *WAITING_FOR_ROSTER* bzw. *DISCONNECT*, Algorithmus 12 für den Zustand *LEAVING*, Algorithmus 13 für den Zustand *JOINED*. Der Zustand *FLUSHING* wird gesondert geprüft, wie im Algorithmus 14 und 15 dargestellt.

Algorithmus 11 Nachrichteneingangsprüfung: Fragment Gruppenprüfung

```
1: if getLocalState() < WAITING_FOR_ROSTER then return FALSE
2: end if
```

Algorithmus 12 Nachrichteneingangsprüfung: Fragment Prüfung Zustand $LEAVING$

```
1: ▷ Akzeptiere keine neuen Nachrichten im Zustand LEAVING
2: if getLocalState() == LEAVING then
3:     rejectMessage(mref, lastHop)
4:     return FALSE
5: end if
```

Algorithmus 13 Nachrichteneingangsprüfung: Fragment Prüfung Zustand $JOINED$

```
1: a ← (NOT fromGroup AND s == WAITING_FOR_ROSTER)
2: if fromGroup == TRUE OR a == TRUE then
3:     ▷ Nachricht bereits gesehen oder in Behandlung?
4:     if last_seen.contains(mref) then
5:         primaryMaster ← isPrimaryMaster(lastHop)

6:         ▷ Master und in Behandlung
7:         if localPeer.role == MASTER then
8:             if queue.contains(mref) AND (NOT primaryMaster) then
9:                 ▷ Ist die Nachricht bereits zustellbar?
10:                if isMessageDeliverable(mref) then
11:                    return FALSE                    ▷ Nachricht verwerfen
12:                else if AND isClusterDistributionInProgress(mref) then

13:                    ▷ Die empfangene Nachricht ist ein Duplikat.
14:                    ▷ Der primäre Master erwartet die LT-Nachricht.
15:                    resendLTMessageToPeer(mref, lastHop)
16:                end if
17:            end if

18:            ▷ else - Ignoriere doppelte Nachrichten
19:        else
20:            if NOT isMessageHandlingFinished(mref) then
21:                resendLTMessageToPeer(mref, lastHop)
22:            end if
23:        end if
24:        return FALSE
25:    end if

26:    if mVID == lVID then
27:        return TRUE

28:        ▷ Ist die Nachricht aus der Zukunft?
29:    else if mVID > lVID then
30:        ▷ Nachrichten aus der Zukunft werden im Next-View-Buffer zwischengespeichert
31:        nvb.pushReceivedMessage(pdu)
32:    else
33:        ▷ Nachrichten aus der Vergangenheit werden abgewiesen
34:        ▷ Über den Nachrichtenrückweisungsmechanismus
35:        [...]
36:    end if
37:    ▷ Die Nachricht ist nicht gültig
38:    return FALSE
39: end if
```

Algorithmus 14 Nachrichteneingangsprüfung: Fragment Prüfung Zustand *FLUSHING*

```
1:  mVID ← pdu.viewID
2:  lVID ← getViewID()
3:  mref ← pdu.messageReference
4:  source ← pdu.sourceID
5:  lastHop ← pdu.lastHop
6:  ms ← getMembershipService()
7:  fromGroup ← ms.isMember(lastHop)
8:  wasResent ← pdu.hasBeenResent

9:  s ← getLocalState()
10: known ← queue.contains(mref)

11: if s == FLUSHING AND (NOT known) AND mVID == lVID then

12:     ▷ Empfang einer sichtändernden Nachricht während FLUSH?
13:     if pdu.isViewChangeable then
14:         isAcceptable = true
15:         ▷ Wurde bereits eine sichtändernde Nachricht behandelt?
16:         i ← 0
17:         while i < queue.size() do
18:             ▷ Sichtändernde Nachricht
19:             m = queue.get(i)
20:             if m.viewChangeable == TRUE then

21:                 if m.messageReference > mref then
22:                     ▷ Akzeptiere neue Nachricht
23:                     ▷ Breche gespeicherten Transfer ab
24:                     cancelTransfer(m.messageReference)
25:                 else
26:                     rejectMessage(mref, lastHop)
27:                     isAcceptable ← false
28:                     break
29:                 end if
30:             end if
31:             i ← i + 1
32:         end while

33:         ... Fortsetzung im nachfolgenden Algorithmus 8 ...

34:     end if
35: end if
```

Algorithmus 15 Fortsetzung Nachrichteneingangsprüfung Zustand $FLUSHING$

if $s == FLUSHING$ **AND** (**NOT** $known$) **AND** $mVID == lVID$ **then**

▷ Empfang einer sichtändernden Nachricht während FLUSH?
if $pdu.isViewChangeable$ **then**

... Fortsetzung des Algorithmus 7 ...

▷ Prüfe auf Ende FLUSHING
if $queue$.isEmpty() **then**
signal(FlushDoneEvent)
end if

▷ Wenn keine sichtändernde Nachricht in der Warteschlange gespeichert ist, kann die neue Nachricht akzeptieren werden. Anderenfalls bestimmt die obige Schleife das Ergebnis.
return $isAcceptable$
else
▷ Sendewiederholungen sind zu akzeptieren
if $wasResent$ **then**
return $TRUE$
end if

▷ Jede weitere nicht behandelte Nachricht sollte während des FLUSHING im Next-View-Buffer gespeichert werde
nvb.pushReceivedMessage(pdu)
return $FALSE$
end if
end if

Anhang C

Ergänzungen zum Kapitel Nutzermobilität

Das Konzept zur Behandlung der Nutzermobilität von Moversight erweitert die Eigenschaften von Gruppenkommunikationssystemen wie folgt:

- *pendingPeer*: Für die Definition vom MOVS wird eine Funktion $pp(V)$ angenommen, welche die Menge $\mathbb{PP}$ derjenigen Peers ermittelt, die in der Sicht V den Zustand *PENDING* haben.

$$pp(V) : \mathbb{V} \Rightarrow \mathbb{PP} \quad \text{mit: , } \mathbb{PP} \subset V.members \tag{C.1}$$

- *lokal stabil*:

$$local\_stable(p, m) \Rightarrow p \notin pp(V) \wedge p \notin m.pendingPeers \tag{C.2}$$

- (*global*) *stabil*

$$\begin{aligned} stable(m) \Rightarrow & pp(V) == \varnothing \Leftrightarrow \\ & m.pendingPeers == \varnothing \Leftrightarrow \\ & \forall p \in V.members(stable(p, m)) \Leftrightarrow \\ & \forall p \in V.members(receives(p, m)) \end{aligned} \tag{C.3}$$

- *pendingSicht* V:

$$V = (id, members, pendingPeers) \tag{C.4}$$

  Mit: $id \in VID$ als Sichtkennung, $members \in 2^{\mathbb{P}}$ als Menge der aktiven Mitglieder der aktuellen Sicht und $pendingPeers \subset members$

- *Sichtübergang reguläre Sicht → pending Sicht*: Die lokale Nachrichtenauslieferung im Fall eines pending Peers bewirkt einen Sichtwechsel, ohne eine weitere Kommunikation innerhalb der Gruppe. Kann eine Nachricht m in einer regulären Sicht V einem Peer q nicht zugestellt werden, so wird eine pending Sicht V' installiert:

$$\begin{aligned} \exists q \in \mathbb{P} \wedge \forall p \in V.members \wedge p \neq q : & \\ & receives\_in(p, m, V) \wedge \\ & q \in m.pendingPeers \rightarrow installs(p, V') \wedge \\ & V'.members = V.members \wedge \\ & q \in V'.pendingPeers \end{aligned} \tag{C.5}$$

- *pendingZustellungslebendigkeit*: Für jede Nachricht, welche durch einen Peer p in der pending Sicht V versandt wurde, wird der Empfang durch alle Peers in der

stabilen Komponente S bestätigt.

$$\begin{aligned}\diamond P - \text{like FD} \wedge stable\_component(S) \Rightarrow \exists\ pending\_view(V) : \\ V.members\ V.pendingPeers == S \wedge \\ \forall p \in S : sends\_in(p, m, V) \Rightarrow \\ \forall p \in S : indicated\_safe(q, m, V)\end{aligned} \quad \text{(C.6)}$$

Die pending Zustellungslebendigkeit gilt gleichfalls im MOVS-Paradigma, da auch in einer pending Sicht die Eigenschaft der Multicastlebendigkeit ebenso wie die Gruppenmitgliedschaftsexaktheit (siehe Anhang A.26) unverändert erhalten bleiben.

- *pendingPeriode*: Für jeden pending Peer wird eine individuelle pending Periode definiert. Diese umfasst den Zeitraum, in welchem der pending Peer p von der Gruppe aufgrund eines Verbindungsproblems oder allgemein eines Fehlers getrennt ist. Innerhalb dieser Periode wird der Peer semantisch als passives Mitglied der Gruppe behandelt. Um die pending Sicht wieder in eine reguläre Sicht zu überführen, muss der pending Peer entweder über die *REJON* Operation in die Gruppe zurückkehren oder aus dieser entfernt werden. Die maximale Zeitspanne, welche dem pending Peer zur Rückkehr in die Gruppe zugestanden wird, ist als maximale pending Periode $PENDING\_TIMEOUT$ definiert. Somit gilt für die pending Periode tp eines Peers p:

$$p \in V.pendingPeers \wedge tp(p) < PENDING\_TIMEOUT : p \in V.members \quad \text{(C.7)}$$

Anhang D

Ergänzungen zum Kapitel Dynamische Optimierung der Clustertopologie

D.1 Ergänzungen Aufrechterhaltung der MOVS-Eigenschaften

Gruppenkommunikationssysteme definieren eine Vielzahl von Eigenschaften, welche auch im Wartungsfall aufrechterhalten werden müssen. Diese Eigenschaften werden üblicherweise in die Kategorien Sicherheits- und Lebendigkeitseigenschaften unterteilt (vgl. dazu Chockler *et al.* [55]). Im Kapitel 6 und Anhang C wurden die für die Mobile Optimistic Virtual Synchronity relevanten Eigenschaften eingeführt und formal definiert. Die folgenden ergänzenden Ausführungen zur Aufrechterhaltung der MOVS-Eigenschaften nehmen die in diesen Abschnitten eingeführte Syntax auf und ergänzen sie bzw. beziehen sich auf dort getroffene Definitionen.

D.1.1 Ergänzungen zur globalen Ordnung der Sichten

Die Operationen des Wartungsdienstes sind formal in einer Funktion *optimize()* zusammengefasst, welche in der aktuellen Sicht eine optimale Topologie für die Gruppe berechnet und diese lokal in einer neuen Sicht einrichtet. Formal ist sie definiert als:

$$\forall p, q \in V.members : V'_p = optimize(p, V) \wedge V'_q = optimize(q, V) \Rightarrow V'_p = V'_q \quad \text{(D.1)}$$

Wie bereits ausgeführt, ist die partielle Ordnung der Sichten (engl. *View Order*) definiert über den Operator $<$, welche auch im Falle einer Wartungsoperation gelten muss. Das bedeutet, dass die Ordnung der Sichten zwischen allen fehlerfreien Gruppenmitgliedern innerhalb der Gruppe gleich ist und eine eventuell ausgeführte Wartungsoperation sich bei allen Peers auf die gleiche Sicht auswirkt. Desweitern muss der der Wartungsdienst während des Protokollablaufs seiteneffektfrei aktiviert beziehungsweise deaktiviert werden können.

D.1.2 Ergänzungen zur Sichtübereinstimmung

Die Übereinstimmung der Sichten (engl. *View Agreement*) ist im Anhang A implizit als Sichtsequenz f definiert (vgl. Formel A.13). Die Konsequenzen aus den Konsistenzforderungen Fischers *et al.* [83] (keine zusätzlichen Sichten bzw. Teilfunktion einer sichtändernden Funktion, gleiche Operationsreihenfolge bei allen fehlerfreien Peers) führen verallgemeinert wiederum zur Formel D.1.

D.1.3 Ergänzungen zur Selbstinklusion

Aufgrund der Selbstinklusion (siehe Kapitel 4, Sichtintegrität (engl. *View Integrity*) in [15] genannt) darf im Kontext des Wartungsdienstes die Funktion *optimize()* nur

dann in einer Sicht V ausgeführt werden, wenn der ausführende Peer p Bestandteil der Ergebnissicht V' ist oder formal:

$$\forall p : V' = optimize(p, V) \Rightarrow p \in V'.members \tag{D.2}$$

Es ist nicht zwingend notwendig, dass die Sichten V und V' verschieden sind (im Sinne von: $V.id \mathrel{!}= V'.id$).

D.1.4 Ergänzungen zur Multicastlebendigkeit

Bei der Betrachtung der Multicastlebendigkeit (engl. *Multicast Liveness*) muss auch die Empfangseelbstinklusion (engl. *Self Delivery*) berücksichtigt werden, welche direkt aus der Multicastlebendigkeit folgt (vergleiche dazu wiederum Kapitel 6 und Anhang C).

Die formale Definition der Anforderung zur Aufrechterhaltung der Multicastlebendigkeit im Wartungsfall erfolgt wiederum unter der Annahme eines Fehlerdetektors der Klasse $\diamond P$ sowie einer stabilen Komponente S:

$$\begin{aligned} \diamond P - \text{like} \wedge stable\_component(S) \Rightarrow \exists t_j \exists V, V' : V.members = S \wedge \\ \forall p \in S : V' = optimize(p, V) \wedge \\ t_i = view\_chng(p, V) \wedge viewof(t_i) = V' \wedge \\ \exists j > i, t_j = send(p, m) \Rightarrow \forall q \in S : receives(q, m) \end{aligned} \tag{D.3}$$

Die Empfangsselbstinklusion im Wartungsfall ist analog definiert als:

$$\begin{aligned} \diamond P - \text{like} \wedge stable\_component(S) \Rightarrow \exists t_j \exists V : V.members = S \wedge \\ \forall p \in S : V' = optimize(p, V) \wedge \\ t_i = view\_chng(p, V) \wedge viewof(t_i) = V' \wedge \\ t_i = send(p, m) \wedge \nexists j > i : t_j = crash(p) \\ \Rightarrow (receives(p, m) \vee send\_failed\_event(p, m)) \end{aligned} \tag{D.4}$$

D.2 Ergänzung Synchronisationsprobleme

D.2.1 Nachrichten aus zukünftigen Sichten

Übertragungsprobleme wie unterschiedliche Latenz oder selektives Verwerfen von Nachrichten können dazu führen, dass Gruppennachrichten zu unterschiedlichen Zeitpunkten bei den einzelnen Peers lokal zugestellt werden. Ein solches Szenario ist in Abbildung D.1 dargestellt.

Bei *Peer* 2 wird der Sichtwechsel im Vergleich zur Restgruppe deutlich verzögert ausgeführt. Begründet sein könnte solch ein Verhalten beispielsweise in einer stark verzögerten GT-Nachricht. Vor dem Sichtwechsel $V \rightarrow V'$ erhält *Peer* 2 eine Nachricht von *Peer* 3, welche bereits zur Sicht V' gehört. Nachrichten, welche aus Sichten mit einer VID größer als der aktuellen VID stammen (hier die Nachrichten aus Sicht V') werden als Nachrichten aus zukünftigen Sichten bezeichnet.

Die Nachricht aus der zukünftigen Sicht würde ohne zusätzliche Maßnahmen durch *Peer* 2 verworfen, da sie nicht die Gruppensichtzustellungseigenschaft erfüllt (siehe Abschnitt 6.4). Für eine erfolgreiche Zustellung muss *Peer* 2 zunächst den Sichtwechsel $V \rightarrow V'$ vollziehen. Anschließend muss *Peer* 3 die Nachricht erneut an *Peer* 2 versenden. Dies ist für die Synchronisation innerhalb der Gruppe kritisch. Aufgrund der Zuverlässigkeitseigenschaften des Multicastdienstes werden nicht bestätigte Nachrichten nach einem gewissen Zeitraum erneut übertragen. Daher ist es prinzipiell möglich, diesen Synchronisationsfehler durch „*abwarten*“ zu beheben. Dieser Ansatz ist in Abbildung D.2 dargestellt, in dem die abgelehnte Nachricht durch den Sender erneut an *Peer* 2 versendet wird. Die erneute Übertragung ist erfolgreich, da *Peer* 2 in der Zwischenzeit den Sichtwechsel vollzogen hat und somit die Gruppensichtzustellungseigenschaft erfüllt.

Die übertragungsbezogenen Aufwände des sendenden Peers sowie die Latenz der betreffenden Nachricht werden dabei erhöht. Beides kann vermieden werden kann,

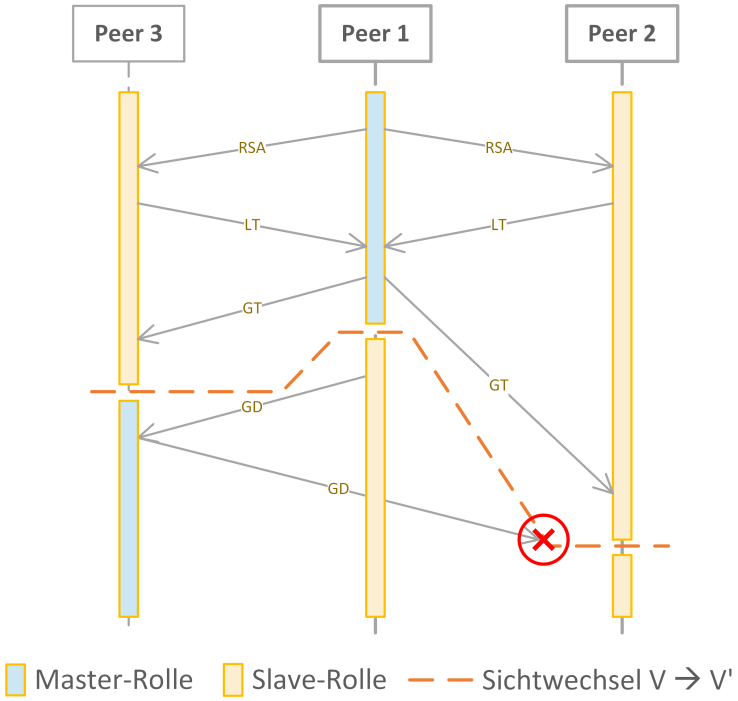


Abbildung D.1: Beispiel für ein Synchronisationsproblem bei Nachrichten aus zukünftigen Sichten

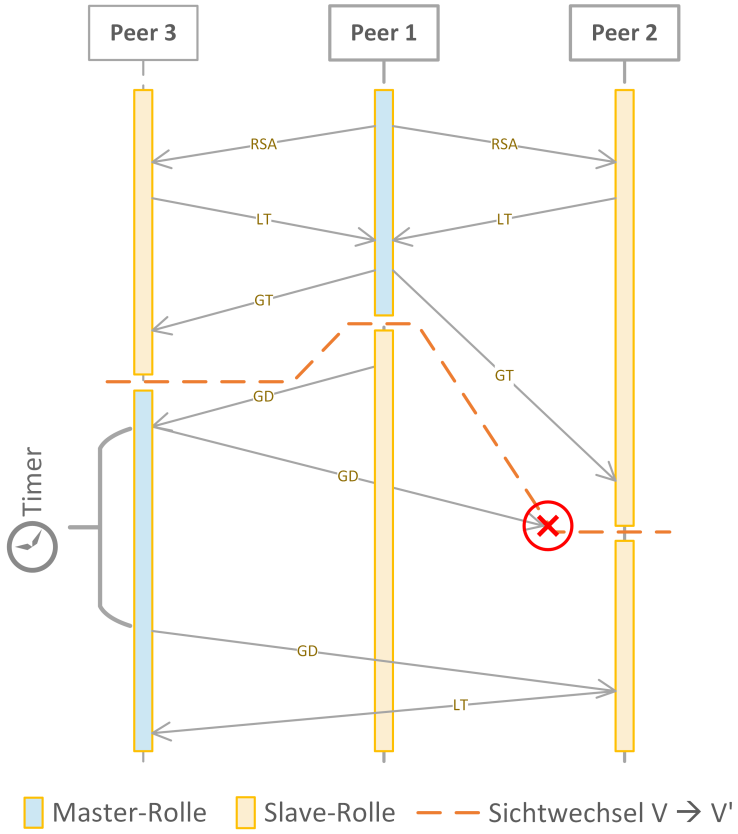


Abbildung D.2: Beispiellösung für das Synchronisationsproblem bei Nachrichten aus zukünftigen Sichten

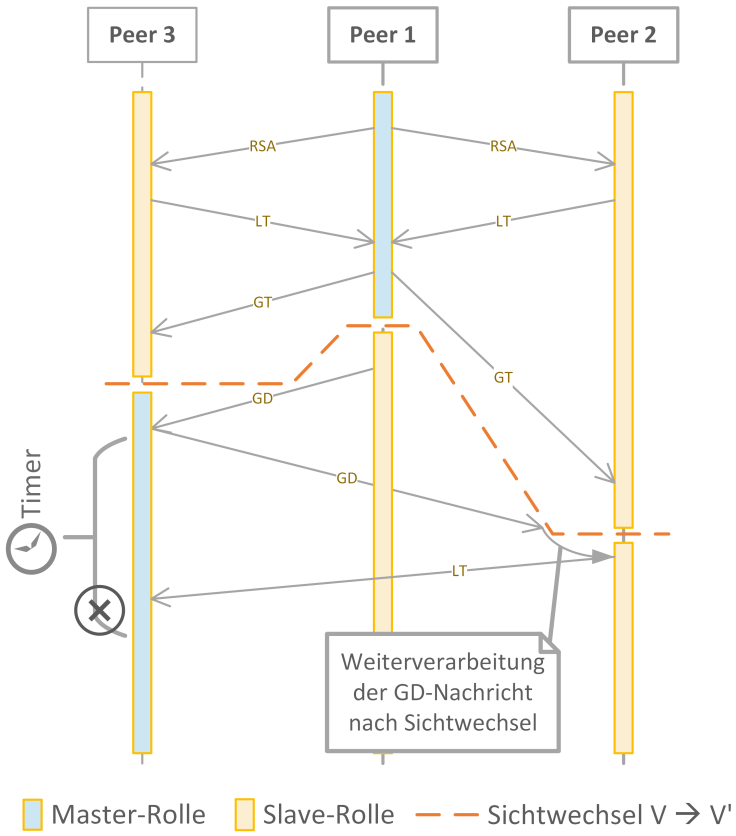


Abbildung D.3: Beispiellösung mit NVB für das Synchronisationsproblem der Nachrichten aus zukünftigen Sichten

da die Nachricht bereits erfolgreich durch *Peer* 2 empfangen und nur aufgrund der Sichteigenschaften verworfen wurde. Ebenso steigt das Risiko, dass durch die fehlende Bestätigung *Peer* 2 als fehlerhaft eingeschätzt wird. Dies würde fälschlicherweise Fehlerbehandlungsmaßnahmen auslösen.

Roman *et al.* [184] schlägt vor, empfangen Nachrichten aus zukünftigen Sichten zu speichern, statt diese zu verwerfen. Dadurch können diese Nachrichten nach einem Sichtwechsel lokal verzögert zugestellt werden. Dieses Prinzip ist in Abbildung D.3 dargestellt. Die *Group-Data-Nachricht* (GD) aus der Sicht V' wird beim Empfang durch *Peer* 2 zwischengespeichert. Unmittelbar nach dem Sichtwechsel wird die gespeicherte Nachricht weiterverarbeitet. Im Beispiel bestätigt *Peer* 2 den Empfang der GD-Nachricht mittels der LT direkt an den neuen Master *Peer* 3. Damit ist eine wiederholte Übertragung der GD-Nachricht nicht notwendig.

Für die Implementierung des Zwischenspeichers wird der NVB genutzt. Die empfangenen Nachrichten werden geprüft und gegebenenfalls gespeichert. Die Prüfbedingungen, welche zu einer Speicherung führen, wurden im Abschnitt B.5 dargestellt. Die Speicherung innerhalb des NVB erfolgt sichtweise und je Sicht in FIFO-Ordnung.

Über dieses Konzept kann der Nutzer ohne Unterbrechung Nachrichten an die Gruppe senden und von dieser empfangen. Deren Auslieferung wird jedoch verzögert, bis der letzte Peer der Gruppe in die aktuelle Sicht gewechselt ist.

D.2.2 Nachrichten aus vergangenen Sichten

Auch Nachrichten aus vergangenen Sichten können Synchronisationsprobleme erzeugen. Vergangene Sichten sind alle Sichten vor der aktuellen Sicht. Eine Verarbeitung von Nachrichten aus vergangenen Sichten würde die Sendesichtzustellung verletzen. Solche Situationen können entstehen, wenn beispielsweise *Peer* 2 eine Nachricht in der Sicht

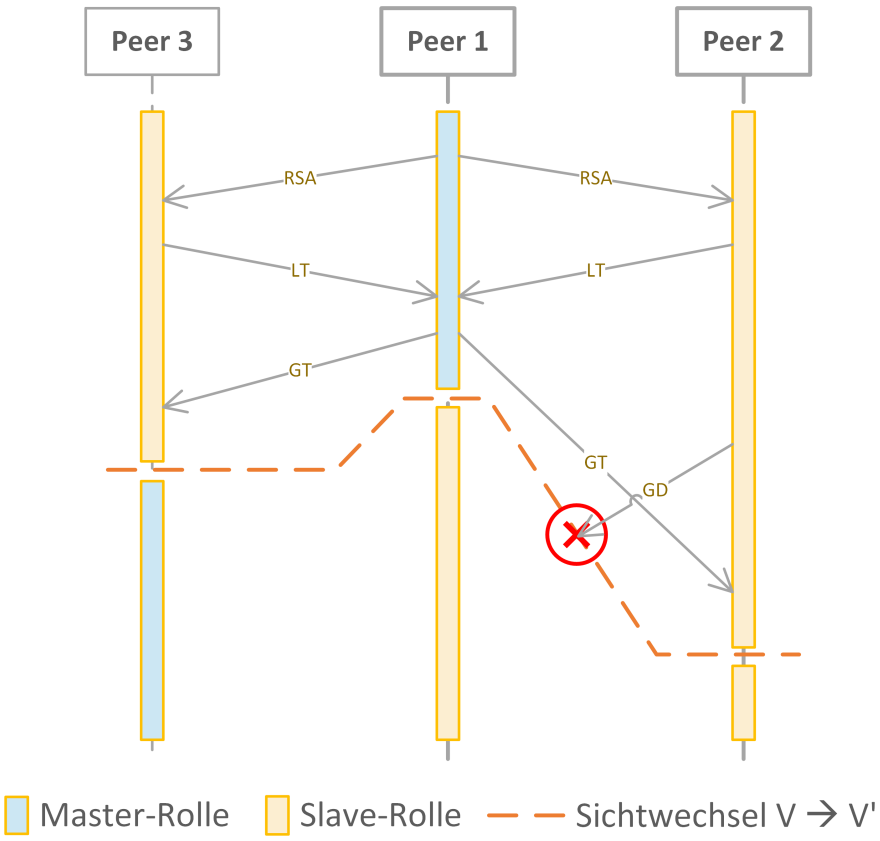


Abbildung D.4: Beispiel für ein Synchronisationsproblem bei Nachrichten aus vergangenen Sichten

V sendet, aber *Peer* 1 und *Peer* 3 bereits Sicht V' installiert haben, wie in Abbildung D.4 dargestellt.

Da Nachrichten aus vergangenen Sichten durch die Peers nicht verarbeitet werden, können die gleichen Probleme wie bei den Nachrichten aus zukünftigen Sichten auftreten. Die Wahrscheinlichkeit des Auftretens von Problemen erhöht sich, je mehr sichtbezogene Nachrichten quasi-parallel versendet werden. Daher ist es wünschenswert, den Sichtwechsel zwischen den *Peers* 1, 2 und 3 zu synchronisieren.

Auch durch die Neuordnung der Topologie ändert sich die Sicht der Gruppe. Erfolgt ein Sichtwechsel aufgrund einer Wartung, würden sämtliche verschickten, in der Gruppe aber noch nicht zugestellten Nachrichten verworfen werden müssen. In Abbildung D.5 ist dies verdeutlicht durch die lokale Zustellung einer *Leave-Announce-Nachricht* (LA), ausgesendet durch *Peer* 1. Mit der Zustellung der LA-Nachricht tritt *Peer* 1 aus der Gruppe aus. Da die GD-Nachricht von *Peer* 2 nicht von *Peer* 1 verarbeitet wird, blockiert sie die Warteschlangen des TOM-Transfers von *Peer* 2 und *Peer* 3 und somit deren Sichtwechsel. Zusätzlich versucht *Peer* 3, eine Bestätigung an das das ehemalige Gruppenmitglied *Peer* 1 zu senden, was zu weiteren Verzögerungen führt. Die Nachrichten blockieren die Warteschlangen der Peers solange bis sie durch eine timergesteuerte Fehlerbehandlung bei den Peers entfernt werden bzw. eine erneute Übertragung ausgelöst wird.

Somit können die Sichten der einzelnen Gruppenteilnehmer auseinanderlaufen und die Konsistenz der Gruppe mit den oben beschriebenen Konsequenzen zerstört werden. Daher ist es nicht optimal, für Nachrichten aus zukünftigen Sichten auf Synchronisationsmaßnahmen zu verzichten. Ein alternativer Lösungsansatz wäre die Erweiterung der mobilen optimistischen virtuellen Synchronität zu einem Ansatz mit zwei Sichtarten, vergleichbar dem EVS-Paradigma von Moser *et al.* [163] (siehe Abschnitt 6.4). Die reguläre Sicht würde um eine Transitionssicht ergänzt, welche ausschließlich für die

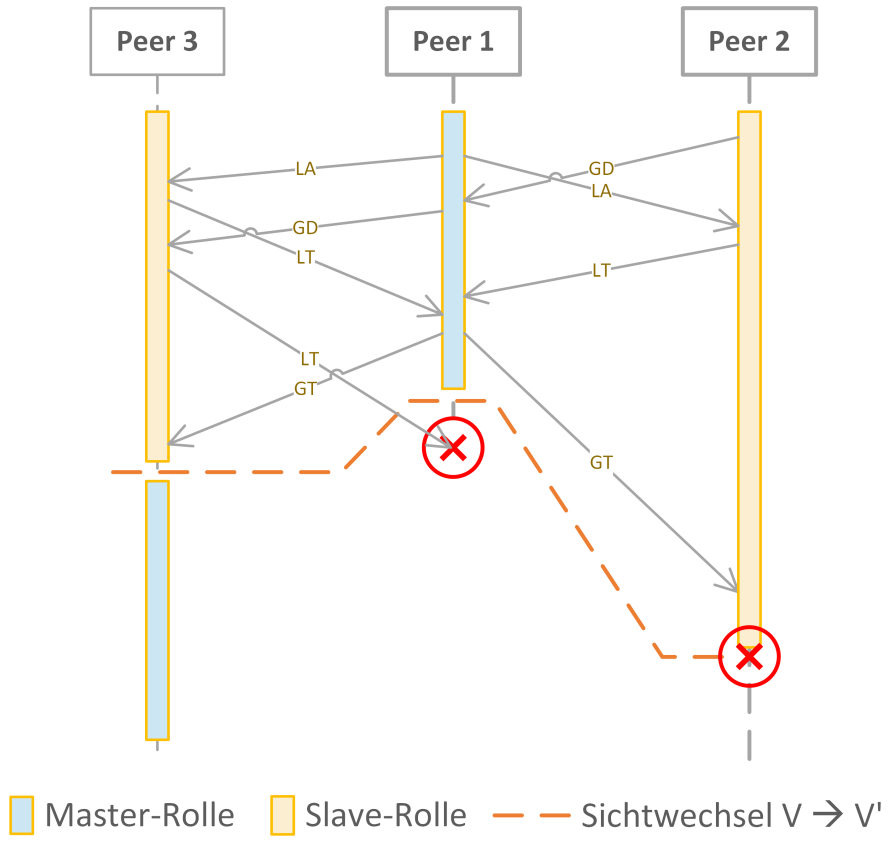


Abbildung D.5: Beispiel für ein Synchronisationsproblem bei parallelen Nachrichten während eines Sichtwechsels

Zwischenspeicherung von Nachrichten genutzt würde, um die Gruppensichtzustellung zu garantieren. Wurden alle Nachrichten aus der alten Sicht ausgeliefert, erfolgt ein Wechsel in eine neue reguläre Sicht. Die neu zu versendenden Nachrichten werden verarbeitet. Abbildung D.6 stellt dieses veränderte Verhalten für das vorangegangene Szenario dar. Der Sichtwechsel erfolgt hier von der regulären Sicht V über die Transitionssicht V' zur regulären Sicht V'', welche nur noch *Peer* 2 und *Peer* 3 umfasst. Die Änderung des Sichtkonzepts würde das Synchronisationsproblem lösen, jedoch die Komplexität der Sichtverwaltung deutlich erhöhen.

Vergleichbar mit der Einführung neuer Sichtarten ist die Verschiebung des Sichtwechsels, bis alle anstehenden Nachrichten ausgeliefert wurden. Um dies zu ermöglichen, müssen wie bei der erweiterten virtuellen Synchronisation neue Nachrichten in dieser Übergangsphase zwischengespeichert werden.

Ist die neue Sicht eingerichtet, können die gespeicherten Nachrichten an die Gruppe gesendet werden. Im Gegensatz zur erweiterten virtuellen Synchronisation wird aber keine eigenständige Sicht eingerichtet. Stattdessen wird bei jeder durch den TOM-Transfer zu versendenden Nachricht geprüft, ob diese potentiell sichtändernd ist. Ist dies der Fall, erfolgt ein Wechsel in den *FLUSHING*-Zustand.

Alle neu zu versendenden Nachrichten werden zwischengespeichert, bis die TOM-Warteschlange geleert wurde. Algorithmus 16 zeigt, wie der *FLUSHING*-Zustand in *Moversight* durch ein Ereignis ausgelöst wird. Das Ereignis wird durch den TOM-Transfer beim Einreihen einer empfangenen Nachricht in die Warteschlange ausgelöst, wenn diese potentiell sichtändernd ist. Die Prüfung erfolgt über die Eigenschaft *viewChangeable* der Multicastnachricht.

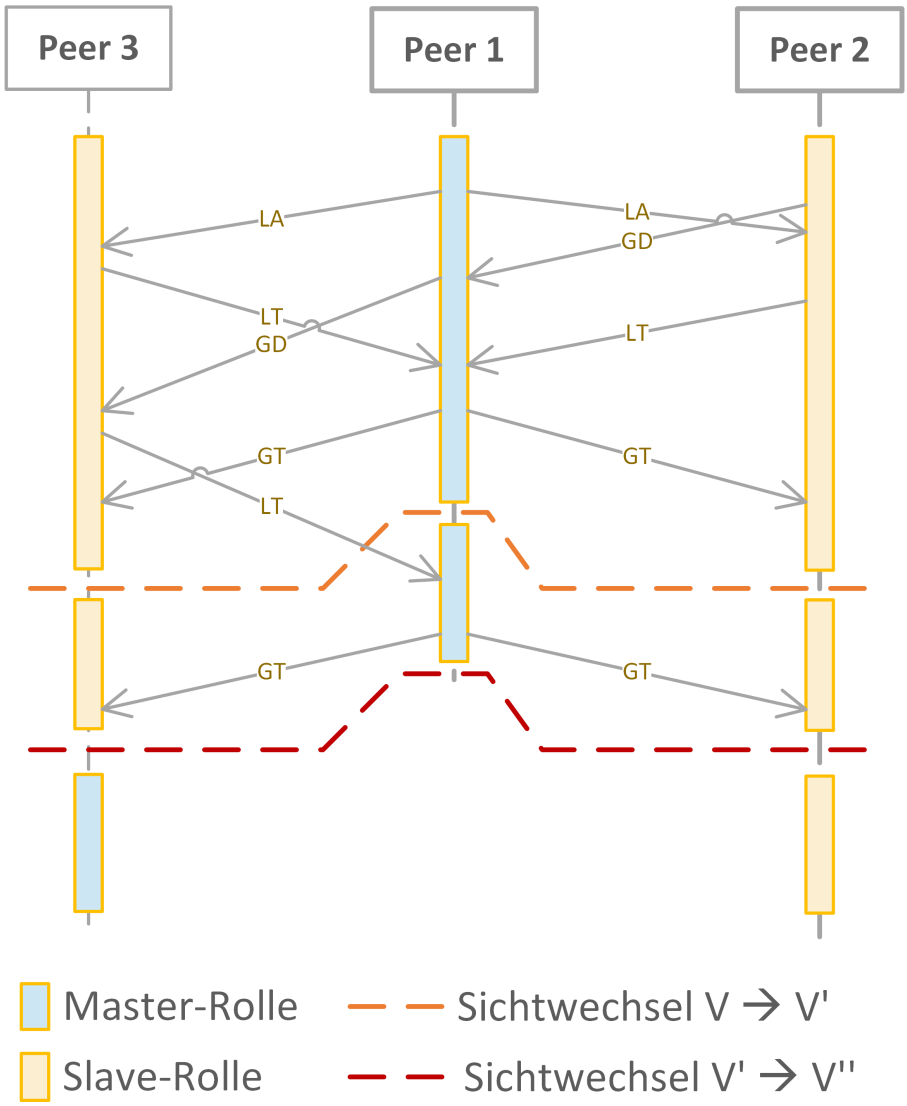


Abbildung D.6: Veränderter Ablauf des Beispielszenarios durch Einführung der erweiterten virtuellen Synchronität

Algorithmus 16 Prüfung auf eine potentielle Sichtänderung durch den RMS bei Einlagerung einer Nachricht in die TOM-Warteschlange

```
1: function handleEvent(MulticastMessageEnqueuedEvent e)

2:     ▷ Sichtändernde Nachrichten können die Konsistenz gefährden
3:     m ← e.getMessage()

4:     if m.viewChangeable == TRUE then
5:         ▷ Wechsel in den FLUSHING-Zustand
6:         startFlushing()
7:     end if
8:     evaluateMaintenanceMomentMetric(m, false)
9: end function
```

Literatur

[1] 3rd Generation Partnership Project (3GPP). *Overview of 3GPP Release 4, Summary of all Release 4 Features v. TSG number 26*. ETSI Mobile Competence Centre. Okt. 2016. URL: http://www.3gpp.org/specifications/releases (siehe S. 15).

[2] E. H. L. Aarts und J. K. Lenstra, Hrsg. *Local search in combinatorial optimization*. 2. Aufl. 41 William Street, Princeton, New Jersey 08540, USA: Princeton University Press, 2003. URL: https://books.google.de/books?id=NWghN9G7q9MC&lpg=PR7&ots=XrSkoFEwNo&dq=Local%20search%20(optimization)%20book&lr&hl=de&pg=PR7#v=onepage&q=Local%20search%20(optimization)%20book&f=false (siehe S. 125).

[3] A. A. Abbasi und M. Younis. „A survey on clustering algorithms for wireless sensor networks“. In: *Computer Communications - Special issue: Network Coverage and Routing Schemes for Wireless Sensor Networks* 30.14–15 (Okt. 2007), S. 2826–2841. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2007.05.024. URL: http://dx.doi.org/10.1016/j.comcom.2007.05.024 (siehe S. 123, 127, 128, 139).

[4] T. Abdelkader u. a. „A Performance Comparison of Delay-Tolerant Network Routing Protocols“. In: *IEEE Network* 30.2 (März 2016), S. 46–53. ISSN: 0890-8044. DOI: 10.1109/MNET.2016.7437024. URL: https://ieeexplore.ieee.org/abstract/document/7437024/ (siehe S. 35).

[5] P. K. Agarwal und C. M. Procopiuc. „Exact and Approximation Algorithms for Clustering“. In: *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*. Bd. 33. SODA '98 2. Philadelphia, PA, USA: Society for Industrial und Applied Mathematics, 2002, S. 201–226. URL: http://dl.acm.org/citation.cfm?id=315040&CFID=609711108&CFTOKEN=63527777 (siehe S. 125).

[6] P. Ahlbrecht. „Impact of Mobility on Information Systems and Information System Design“. PhD Thesis. Technical University Braunschweig, Sep. 2004. URL: http://rzbl68.biblio.etc.tu-bs.de:8080/docportal/servlets/MCRFileNodeServlet/DocPortal_derivate_00001640/Document.pdf (siehe S. 4, 25).

[7] M. Aigner. „A Characterization of the bell numbers“. In: *Discrete Mathematics* 205.1-3 (Juli 1999), S. 207–210. URL: http://www.sciencedirect.com/science/article/pii/S0012365X99001089 (siehe S. 125).

[8] R. Albert und A.-L. Barabási. „Statistical mechanics of complex networks“. In: *Reviews of Modern Physics* 74 (1 Jan. 2002), S. 47–97. DOI: 10.1103/RevModPhys.74.47. URL: https://link.aps.org/doi/10.1103/RevModPhys.74.47 (siehe S. 76).

[9] Y. Amir u. a. „Transis: A Communication Sub-System for High Availability“. In: *Proceedings of the 22th International Symposium on Fault-Tolerant Computing, 1992. FTCS-22. Digest of Papers*. FTCS. Bosten, MA, USA: IEEE, Juli 1992, S. 76–84. DOI: 10.1109/FTCS.1992.243613. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=243613&tag=1 (siehe S. 144).

[10] Y. Amir und J. Stanton. *The Spread Wide Area Group Communication System*. Technical Report CNDS-98-4. `http://www.spread.org/SpreadResearch.html`. The Center for Networking und Distributed Systems, The Johns Hopkins University, Apr. 1998. URL: `http://www.cnds.jhu.edu/pub/papers/spread.ps` (siehe S. 24, 44, 87, 144, 154).

[11] A. Antonić u. a. „A mobile Crowd Sensing Ecosystem enabled by CUPUS: Cloud-based Publish/Subscribe Middleware for the Internet of Things“. In: *Future Generation Computer Systems* 56 (März 2016), S. 607–622. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2015.08.005`. URL: `http://www.sciencedirect.com/science/article/pii/S0167739X15002575` (siehe S. 31).

[12] R. Asati u. a. *RFC 7527: Enhanced Duplicate Address Detection*. RFC 7527 (Proposed Standard). Internet Engineering Task Force, Apr. 2015. URL: `http://www.ietf.org/rfc/rfc7527.txt` (siehe S. 18, 31).

[13] D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. 1. Aufl. New York, USA: Springer-Verlag New York, Inc., 2006. DOI: `10.1007/0-387-31909-3`. URL: `http://www.springer.com/fr/book/9780387221960` (siehe S. 125).

[14] Ö. Babaoğlu, A. Bartoli und G. Dini. „Enriched View Synchrony: A Programming Paradigm for Partitionable Asynchronous Distributed Systems“. In: *IEEE Transactions on Computers* 46.6 (Juni 1997), S. 642–658. ISSN: 0018-9340. DOI: `10.1109/12.600823` (siehe S. 41).

[15] Ö. Babaoğlu, R. Davoliand und A. Montresor. „Group Membership and View Synchrony in Partitionable Asynchronous Distributed Systems: Specifications“. In: *ACM SIGOPS Operating Systems Review* 31.2 (Apr. 1997), S. 11–22. ISSN: 0163-5980. DOI: `10.1145/250007.250010`. URL: `http://doi.acm.org/10.1145/250007.250010` (siehe S. 41, 185).

[16] R. M. Babu, P. Mabel und K. B. Murthy. „Group Communication Scheme for Mobile Networks with Mobile Router“. In: *Proceedings of the 2010 3rd International Conference on Emerging Trends in Engineering and Technology*. ICETET '10. Los Alamitos, CA, USA: IEEE Computer Society, 2010, S. 304–307. ISBN: 978-0-7695-4246-1. DOI: `10.1109/ICETET.2010.67`. URL: `http://www.computer.org/portal/web/csdl/doi/10.1109/ICETET.2010.67` (siehe S. 3, 31).

[17] J. Bacon u. a. „Generic Support for Distributed Applications“. In: *Computer* 33.3 (März 2000), S. 68–76. ISSN: 0018-9162. DOI: `10.1109/2.825698` (siehe S. 31).

[18] M. Balakrishnan und K. Birman. „Reliable Multicast for Time-Critical Systems“. In: *Proceedings of the 1st Workshop on Apllied Software Reliability*. WASR 2006. IEEE, Juni 2006, S. 1–6. URL: `https://www.cs.cornell.edu/projects/quicksilver/public_pdfs/timecriticalfinal.pdf` (siehe S. 131).

[19] S. Banerjee, B. Bhattacharjee und C. Kommareddy. „Scalable Application Layer Multicast“. In: *ACM SIGCOMM Computer Communication Review - Proceedings of the 2002 SIGCOMM Conference* 32 (4 Aug. 2002), S. 205–217. ISSN: 0146-4833. DOI: `10.1145/964725.633045`. URL: `http://doi.acm.org/10.1145/964725.633045` (siehe S. 14, 18, 25, 67, 68, 71, 73, 172).

[20] X. Bao u. a. „SyncViews: Toward Consistent User Views in Cloud-Based File Synchronization Services“. In: *2011 6th Annual Chinagrid Conference*. Aug. 2011, S. 89–96. DOI: `10.1109/ChinaGrid.2011.35`. URL: `http://ieeexplore.ieee.org/abstract/document/6051738/` (siehe S. 93).

[21] A.-L. Barabási und R. Albert. „Emergence of Scaling in Random Networks“. In: *Science* 286.5439 (1999), S. 509–512. ISSN: 0036-8075. DOI: `10.1126/science.286.5439.509`. eprint: `http://science.sciencemag.org/content/286/5439/509.full.pdf`. URL: `http://science.sciencemag.org/content/286/5439/509` (siehe S. 76).

[22] L. Barolli und F. Xhafa. „JXTA-Overlay: A P2P Platform for Distributed, Collaborative, and Ubiquitous Computing“. In: *IEEE Transactions on Industrial Electronics* 58.6 (Juni 2011), S. 2163–2172. ISSN: 0278-0046. DOI: 10.1109/TIE.2010.2050751. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5475248 (siehe S. 9, 27).

[23] S. Basagni. „Distributed Clustering for Ad Hoc Networks“. In: *Proceedings of the 4th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN '99)*. Perth/Fremantle, WA, Australia: IEEE, Juni 1999, S. 310–315. DOI: 10.1109/ISPAN.1999.778957 (siehe S. 124, 129).

[24] P. Baumung. „P2P-basierte Gruppenkommunikation in drahtlosen Ad-hoc Netzen“. Dissertation. Karlsruhe, Germany: Faculty of Computer Science, University of Karlsruhe (TH), Sep. 2008. URL: http://www.ubka.uni-karlsruhe.de/dbkit/uv/getUvkaDocument.php?vv_id=1000009060 (siehe S. 121).

[25] J. Behrens u. a. *Derecho: Group Communication at the Speed of Light*. Techn. Ber. Cornell University Ithaca, NY: Cornel University, 2016. URL: http://www.cs.cornell.edu/projects/Quicksilver/public_pdfs/Derecho-Protocols.pdf (siehe S. 33).

[26] I. Beier und H. König. „A Protocol Supporting Distributed Group and QoS Management“. In: *Proceedings of the IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking*. Santiago de Chile, Chile: IEEE, Nov. 1997, S. 213–222. DOI: 10.1109/PRMNET.1997.638898 (siehe S. 14).

[27] N. Benamar u. a. „Challenges of the Internet of Things: IPv6 and Network Management“. In: *Proceedings of the 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2014)*. IMIS. Birmingham, United Kingdom: IEEE Computer Society, Juli 2014, S. 328–333. DOI: doi:10.1109/IMIS.2014.43. URL: http://www.computer.org/csdl/proceedings/imis/2014/4331/00/4331a328-abs.html (siehe S. 121).

[28] M. Bertier, O. Marin und P. Sens. „Implementation and Performance Evaluation of an Adaptable Failure Detector“. In: *Proceedings of the 2002 International Conference on Dependable Systems and Networks*. DSN '02. Los Alamitos, CA, USA: IEEE Computer Society, 2002, S. 354–363. ISBN: 0-7695-1597-5. DOI: 10.1109/DSN.2002.1028920 (siehe S. 114, 115, 117).

[29] D. P. Bertsekas. „Auction Algorithms for Network Flow Problems: A Tutorial Introduction“. In: *Computational Optimization and Applications* 1.1 (Okt. 1992), S. 7–66. ISSN: 0926-6003. DOI: 10.1007/BF00247653. URL: http://link.springer.com/article/10.1007%2FBF00247653 (siehe S. 129).

[30] S. Bhowmik u. a. „High Performance Publish/Subscribe Middleware in Software-Defined Networks“. In: *IEEE/ACM Transactions on Networking* 25.3 (Juni 2017), S. 1501–1516. ISSN: 1063-6692. DOI: 10.1109/TNET.2016.2632970. URL: https://doi.org/10.1109/TNET.2016.2632970 (siehe S. 31).

[31] K. P. Birman u. a. „Implementing Fault-Tolerant Distributed Objects“. In: *IEEE Transactions on Software Engineering* 11.6 (Juni 1985), S. 502–508. ISSN: 0098-5589. DOI: 10.1109/TSE.1985.232242 (siehe S. 24, 29, 47, 81, 87, 88).

[32] K. P. Birman und T. A. Joseph. „Reliable Communication in the Presence of Failures“. In: *ACM Transactions on Computer Systems (TOCS)* 5.1 (Jan. 1987), S. 47–76. ISSN: 0734-2071. DOI: 10.1145/7351.7478. URL: http://doi.acm.org/10.1145/7351.7478 (siehe S. 44, 172).

[33] K. P. Birman und R. V. Renesse. *Reliable Distributed Computing with the Isis Toolkit*. Hrsg. von R. V. Renesse. Los Alamitos, CA, USA: IEEE Computer Society Press, 1994. URL: http://dl.acm.org/citation.cfm?id=561724 (siehe S. 23, 41, 45, 48, 73, 107).

[34] R. Bless u. a. „The Underlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture“. In: *Proceedings of the 4th EuroNGI Conferemce on Next Generation Internet Networks (NGI 2008)*. Krakow, Poland: IEEE, Apr. 2008, S. 115–122. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4510794 (siehe S. 15–18, 27, 28, 36, 67, 172).

[35] Bluetooth Special Interest Group. *Bluetooth Service Discovery Application Profil (SDAP)*. Bluetooth Specification Version 1.1. Depricated. Feb. 2001. URL: https://developer.bluetooth.org/TechnologyOverview/Pages/SDAP.aspx (siehe S. 16).

[36] Bluetooth Special Interest Group. *Bluetooth Core Specification Version 2.1 - Basic Rate / Enhanced Data Rate (BR / EDR)*. online. 2016. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification (siehe S. 15).

[37] R. Boichat und L. Duchien. „Network Membership: A Partition Model for Reliable Mobile Communication“. In: *Proceedings of the 8th International Conference on Parallel and Distributed Systems (ICPADS 2001)*. Aug. 2001, S. 45–52. DOI: 10.1109/ICPADS.2001.934800. URL: http://ieeexplore.ieee.org/document/934800/ (siehe S. 146, 147).

[38] E. Brewer. „CAP Twelve Years Later: How the „Rules“ Have Changed“. In: *Computer* 45.2 (Feb. 2012), S. 23–29. ISSN: 0018-9162. DOI: http://doi.ieeecomputersociety.org/10.1109/MC.2012.37 (siehe S. 4, 82–84).

[39] E. A. Brewer. „Towards Robust Distributed Systems (Invited Talk)“. In: *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*. PODC '00. Portland, Oregon, USA: ACM, 2000, S. 7. ISBN: 1-58113-183-6. DOI: 10.1145/343477.343502. URL: http://www.cs.berkeley.edu/~brewer/PODC2000.pdf (siehe S. 4, 82).

[40] A. F. Buoud und A. P. Jayasumana. „Topology Preserving Map to Physical Map – A Thin-Plate Spline Based Transform“. In: *Proceedings of the 41st IEEE Conference on Local Computer Networks (LCN)*. Nov. 2016, S. 262–270. DOI: 10.1109/LCN.2016.54. URL: http://ieeexplore.ieee.org/document/7796798/media (siehe S. 71, 163).

[41] N. Cacho u. a. „Handling Exceptional Conditions in Mobile Collaborative Applications: An Exploratory Case Study“. In: *Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '06)*. Manchester, UK: IEEE, Juni 2006, S. 137–142. DOI: 10.1109/WETICE.2006.45. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4092196 (siehe S. 9).

[42] C. Caini u. a. „Delay- and Disruption-Tolerant Networking (DTN): An Alternative Solution for Future Satellite Networking Applications“. In: *Proceedings of the IEEE* 99.11 (Nov. 2011), S. 1980–1997. ISSN: 0018-9219. DOI: 10.1109/JPROC.2011.2158378. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5959951 (siehe S. 34).

[43] J.-C. Cano und P. Manzoni. „A Performance Comparison of Energy Consumption for Mobile Ad Hoc Network Routing Protocols“. In: *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. San Francisco, CA, USA: IEEE, Aug. 2000, S. 57–64. DOI: 10.1109/MASCOT.2000.876429 (siehe S. 122).

[44] A. Carroll und G. Heiser. „An Analysis of Power Consumption in a Smartphone.“ In: *Proceedings of the 2010 USENIX annual technical conference (USENIXATC '10)*. Bd. 14. USENIX Association Berkeley, CA, USA, Juni 2010, S. 21–21. URL: https://www.usenix.org/event/usenix10/tech/full_papers/Carroll.pdf (siehe S. 122).

[45] V. Cerf u. a. *RFC 4838: Delay-Tolerant Networking Architecture*. RFC 4838 (Informational). Internet Engineering Task Force, Apr. 2007. URL: http://www.ietf.org/rfc/rfc4838.txt (siehe S. 34).

[46] E. Chan-Tin und N. Hopper. „Accurate and Provably Secure Latency Estimation with Treeple“. In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, USA: The Internet Society, Feb. 2011. URL: http://www.isoc.org/isoc/conferences/ndss/11/pdf/7_1.pdf (siehe S. 114).

[47] T. D. Chandra und S. Toueg. „Unreliable Failure Detectors for Reliable Distributed Systems“. In: *Journal of the ACM (JACM)* 43.2 (2 März 1996), S. 225–267. ISSN: 0004-5411. DOI: 10.1145/226643.226647. URL: http://portal.acm.org/citation.cfm?doid=226643.226647# (siehe S. 109).

[48] J.-M. Chang und N. F. Maxemchuk. „Reliable Broadcast Protocols“. In: *ACM Transactions on Computer Systems (TOCS)* 2.3 (Aug. 1984), S. 251–273. ISSN: 0734-2071. DOI: 10.1145/989.357400. URL: http://doi.acm.org/10.1145/989.357400 (siehe S. 44).

[49] Y. Chawathe. „Scattercast: an adaptable broadcast distribution framework“. In: *Multimedia Systems* 9.1 (Juli 2003), S. 104–118. ISSN: 0942-4962. DOI: 10.1007/s00530-002-0082-z. URL: http://link.springer.com/article/10.1007/s00530-002-0082-z (siehe S. 25, 26, 68).

[50] W. Chen, S. Toueg und M. Aguilera. „On the Quality of Service of Failure Detectors“. In: *IEEE Transactions on Computers* 51.5 (Mai 2002), S. 561–580. ISSN: 0018-9340. DOI: 10.1109/TC.2002.1004595. URL: http://www.computer.org/portal/web/csdl/doi/10.1109/TC.2002.1004595 (siehe S. 109, 113).

[51] S. Cheshire, B. Aboba und E. Guttman. *RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses*. RFC 3927 (Proposed Standard). Internet Engineering Task Force, Mai 2005. URL: http://www.ietf.org/rfc/rfc3927.txt (siehe S. 18, 31).

[52] S. Cheshire und M. Krochmal. *RFC 6772: Multicast DNS*. RFC 6772 (Proposed Standard). Internet Engineering Task Force, Feb. 2013. URL: http://www.ietf.org/rfc/rfc6762.txt (siehe S. 16).

[53] K. Cheverst u. a. „Services to Support Consistency in Mobile Collaborative Applications“. In: *Proceedings of 3rd International Workshop on Services in Distributed and Networked Environments*. IEEE, Juni 1996, S. 27–34. DOI: 10.1109/SDNE.1996.502444. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=502444 (siehe S. 9).

[54] M. Chitra und R. W. Banu. „Towards Robust and Reliable Group Communication in Ad hoc Mobile Environment“. In: *Journal of Theoretical and Applied Information Technology* 3.1 (2007), S. 55–67. URL: http://www.jatit.org/volumes/research-papers/AD_HOC_MOBILE_COMMUNICATION_3_1.pdf (siehe S. 154).

[55] G. V. Chockler, I. Keidar und R. Vitenberg. „Group Communication Specifications: A Comprehensive Study“. In: *ACM Computing Surveys (CSUR)* 33.4 (Dez. 2001), S. 427–469. ISSN: 0360-0300. DOI: 10.1145/503112.503113. URL: http://portal.acm.org/citation.cfm?id=503113 (siehe S. 11, 41, 44, 52, 53, 87, 88, 91, 167–170, 185).

[56] H.-L. Choi, L. Brunet und J. P. How. „Consensus-Based Decentralized Auctions for Robust Task Allocation“. In: *IEEE Transactions on Robotics* 25.4 (Aug. 2009), S. 912–926. ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2022423 (siehe S. 129).

[57] Y.-h. Chu, S. G. Rao und H. Zhang. „A Case for End System Multicast (Keynote Address)“. In: *ACM SIGMETRICS Performance Evaluation Review - Special issue on proceedings of ACM SIGMETRICS 2000* 28.1 (1 Juni 2000), S. 1–12. ISSN: 0163-5999. DOI: 10.1145/345063.339337 (siehe S. 23, 25, 68, 71, 172).

[58] Y. Chu u. a. „Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture“. In: *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*. ACM SIGCOMM. San Diego, California, United States: ACM, Aug. 2001, S. 55–67. ISBN: 1-58113-411-8. DOI: 10.1145/383059.383064 (siehe S. 26, 122).

[59] B.-G. Chun u. a. „CloneCloud: Elastic Execution between Mobile Device and Cloud“. In: *Proceedings of the 6th Conference on Computer Systems (EuroSys '11)*. EuroSys. Salzburg, Austria: ACM, Apr. 2011, S. 301–314. ISBN: 978-1-4503-0634-8. DOI: 10.1145/1966445.1966473 (siehe S. 33).

[60] T. Clausen und P. Jacquet. *RFC 3626: Optimized Link State Routing Protocol (OLSR)*. RFC 3626 (Experimental). Internet Engineering Task Force, Okt. 2003. URL: http://www.ietf.org/rfc/rfc3626.txt (siehe S. 29).

[61] D. Conan u. a. „Failure, Disconnection and Partition Detection in Mobile Environment“. In: *Proceedings of the 7th IEEE International Symposium on Network Computing and Applications (NCA '08)*. Los Alamitos, CA, USA: IEEE, Juli 2008, S. 119–127. ISBN: 978-0-7695-3192-2. DOI: 10.1109/NCA.2008.18. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4579647 (siehe S. 88).

[62] N. Costagliola u. a. „Energy- and Delay-Efficient Routing in Mobile Ad Hoc Networks“. English. In: *Mobile Networks and Applications* 17.2 (Apr. 2012), S. 281–297. ISSN: 1383-469X. DOI: 10.1007/s11036-011-0335-1 (siehe S. 29).

[63] M. Cotton, L. Vegoda und D. Meyer. *RFC 5771: IANA Guidelines for IPv4 Multicast Address Assignments*. RFC 5771 (Best Current Practice). Internet Engineering Task Force, März 2010. URL: http://www.ietf.org/rfc/rfc5771.txt (siehe S. 16).

[64] G. Cugola, E. Di Nitto und A. Fuggetta. „The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS“. In: *IEEE Transactions on Software Engineering* 27.9 (Sep. 2001), S. 827–850. ISSN: 0098-5589. DOI: 10.1109/32.950318. URL: http://dx.doi.org/10.1109/32.950318 (siehe S. 23, 31).

[65] F. Dabek u. a. „Vivaldi: A Decentralized Network Coordinate System“. In: *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '04. Portland, Oregon, USA: ACM, 2004, S. 15–26. ISBN: 1-58113-862-8. DOI: 10.1145/1015467.1015471. URL: http://doi.acm.org/10.1145/1015467.1015471 (siehe S. 114).

[66] S. E. Deering. *RFC 1112: Host extensions for IP multicasting*. Obsoletes RFC 0988, RFC 1054. See also STD 0005. Updated by RFC 2236. Status: STANDARD. Internet Engineering Task Force, Aug. 1989. URL: http://www.faqs.org/rfcs/rfc1112.html (siehe S. 16).

[67] X. Défago, A. Schiper und P. Urbán. „Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey“. In: *ACM Computing Surveys (CSUR)* 36.4 (Dez. 2004), S. 372–421. ISSN: 0360-0300. DOI: 10.1145/1041680.1041682. URL: http://doi.acm.org/10.1145/1041680.1041682 (siehe S. 41, 44, 45).

[68] T. Dierks und E. Rescorla. *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685. Internet Engineering Task Force, Aug. 2008. URL: http://www.ietf.org/rfc/rfc5246.txt (siehe S. 17).

[69] D. Dolev und D. Malki. „The Transis Approach to High Availability Cluster Communication“. In: *Communications of the ACM* 39.4 (4 Apr. 1996), S. 64–70. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/227210.227227 (siehe S. 23, 41, 172).

[70] F. Dressler und M. Gerla. „A Framework for Inter-Domain Routing in Virtual Coordinate based Mobile Networks“. English. In: *Wireless Networks* 19.7 (Okt. 2013), S. 1611–1626. ISSN: 1022-0038. DOI: 10.1007/s11276-013-0554-4. URL: http://dx.doi.org/10.1007/s11276-013-0554-4 (siehe S. 172).

[71] R. Droms. *RFC 2131: Dynamic Host Configuration Protocol*. RFC 2131 (Draft Standard). Updated by RFCs 3396, 4361, 5494, 6842. Internet Engineering Task Force, März 1997. URL: http://www.ietf.org/rfc/rfc2131.txt (siehe S. 140).

[72] B. Elbhiri u. a. „A New Spectral Classification for Robust Clustering in Wireless Sensor Networks“. In: *Proceedings of 2013 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*. Apr. 2013, S. 1–10. DOI: 10.1109/WMNC.2013.6548982 (siehe S. 124, 126).

[73] T. Enokido und M. Takizawa. „Flexible Group Communication Protocol“. In: *Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2003)*. IEEE, Mai 2003, S. 272–278. DOI: 10.1109/FTDCS.2003.1204347. URL: http://ieeexplore.ieee.org/document/1204347/ (siehe S. 147, 148).

[74] T. Ernst. *RFC 4886: Network Mobility Support Goals and Requirements*. RFC 4886 (Informational). Internet Engineering Task Force, Juli 2007. URL: http://www.ietf.org/rfc/rfc4886.txt (siehe S. 29).

[75] Eurescom und the European Commision. *European Future Internet Portal*. 2008. URL: http://www.future-internet.eu (siehe S. 35).

[76] European Telecommunications Standards Institute (ETSI). *Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Overall description of the GPRS radio interface; Stage 2 (3GPP TS 43.064 Version 12.2.0 Release 12)*. 3GPP Specification ETSI TS 143 064 V12.2.0 (2014-10). Okt. 2014. URL: http://www.3gpp.org/DynaReport/43064.htm (siehe S. 15).

[77] K. Fall. „A Delay-Tolerant Network Architecture for Challenged Internets“. In: *Proceedings of the 2003 Conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM'03. Karlsruhe, Germany: ACM, 2003, S. 27–34. ISBN: 1-58113-735-4. DOI: 10.1145/863955.863960. URL: http://doi.acm.org/10.1145/863955.863960 (siehe S. 34, 35).

[78] K. Fall, S. Farrel und J. Ott. *Delay-Tolerant-Networking Research Group (DTNRG)*. 2002. URL: http://www.dtnrg.org (siehe S. 34).

[79] S. Farrell und V. Cahill. „Evaluating LTP-T: A DTN-Friendly Transport Protocol“. In: *Proceedings of the International Workshop on Satellite and Space Communications (IWSSC '07)*. Sep. 2007, S. 178–181. DOI: 10.1109/IWSSC.2007.4409412. URL: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=4409412 (siehe S. 34).

[80] A. Fekete, N. Lynch und A. Shvartsman. „Specifying and Using a Partitionable Group Communication Service“. In: *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*. PODC '97. Santa Barbara, California, USA: ACM, Aug. 1997, S. 53–62. ISBN: 0-89791-952-1. DOI: 10.1145/259380.259422. URL: http://doi.acm.org/10.1145/259380.259422 (siehe S. 50).

[81] C. Fetzer, M. Raynal und F. Tronel. „An Adaptive Failure Detection Protocol“. In: *Proceedings of the 2001 IEEE Pacific Rim International Symposium on Dependable Computing*. Seoul, South Korea: IEEE, Dez. 2001, S. 146–153. ISBN: 0-7695-1414-6. DOI: http://doi.ieeecomputersociety.org/10.1109/PRDC.2001.992691. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=992691 (siehe S. 87, 88).

[82] L. Fiege u. a. „Supporting Mobility in Content-Based Publish/Subscribe Middleware“. In: *Middleware*. Hrsg. von M. Endler. Bd. 2672. Lecture Notes in Computer Science. USENIX The Advanced Computing System Association. New York, NY, USA: Springer Verlag New York, Inc., Juni 2003, S. 103–122. ISBN: 3-540-40317-5. URL: http://dl.acm.org/citation.cfm?id=1515923 (siehe S. 31).

[83] M. J. Fischer, N. A. Lynch und M. S. Paterson. „Impossibility of Distributed Consensus with One Faulty Process“. In: *Journal of the ACM (JACM)* 32.2 (Apr. 1985), S. 374–382. ISSN: 0004-5411. DOI: 10.1145/3149.214121. URL: http://doi.acm.org/10.1145/3149.214121 (siehe S. 82, 122, 185).

[84] H. Fotouhi u. a. „mRPL+: A mobility management framework in RPL / 6LoWPAN“. In: *Computer Communications* 104 (Jan. 2017), S. 34–54. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2017.01.020. URL: http://www.sciencedirect.com/science/article/pii/S0140366417301512 (siehe S. 34).

[85] P. Francis. *Yoid: Extending the Internet Multicast Architecture*. Technical Report. Berkeley, CA, USA: The ICSI Networking und Security Group, Apr. 2000. URL: http://www.icir.org/yoid/docs/ycHtmlL/htmlRoot.html (siehe S. 3, 25, 26).

[86] R. Friedman und K. Birman. *Trading Consistency for Availability in Distributed Systems*. Techn. Ber. Ithaca, NY, USA: Cornell University, Apr. 1996. URL: https://ecommons.cornell.edu/handle/1813/7235 (siehe S. 82).

[87] R. Friedman und R. van Renesse. *Strong and Weak Virtual Synchrony in Horus*. Technical Report TR95-1537. Cornell University Computer Science Technical Reports. Ithaca, NY, USA: Department of Computer Science, Cornell University, Aug. 1995. URL: http://hdl.handle.net/1813/7194 (siehe S. 45, 46, 48, 50).

[88] M. Frincu u. a. „Self-Healing Distributed Scheduling Platform“. In: *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011)*. CCGrid. Newport Beach, CA, USA: IEEE, Mai 2011, S. 225–234. DOI: 10.1109/CCGrid.2011.23. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5948613 (siehe S. 124).

[89] C. Fu, R. H. Glitho und F. Khendek. „Signaling for Multimedia Conferencing in Stand-Alone Mobile Ad Hoc Networks“. In: *IEEE Transactions on Mobile Computing* 8.7 (2009), S. 991–1005. ISSN: 1536-1233. DOI: http://dx.doi.org/10.1109/TMC.2008.177. URL: http://portal.acm.org/citation.cfm?id=1550632# (siehe S. 29, 67, 124).

[90] J. Gäbler und H. König. „Enhancing group communication systems with mobility support“. In: *Proceedings of the 19th IEEE International Conference on Networks (ICON 2013)*. Dez. 2013, S. 1–6. DOI: 10.1109/ICON.2013.6781995 (siehe S. 23).

[91] J. Gäbler und H. König. „Moversight: A Group Communication Protocol for Mobile Collaborative Applications“. In: *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*. Dubai, United Arab Emirates: IEEE, Apr. 2013, S. 1–8. DOI: 10.1109/WMNC.2013.6548965. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6548965 (siehe S. 85, 87, 95).

[92] J. Gäbler und H. König. „Moversight: An Approach to Support Mobility in Collaborative Applications“. In: *Proceedings of the 10th Annual Conference on Wireless On-Demand Network Systems and Services (WONS 2013)*. Banff, AB, Canada: IEEE Communication Society, März 2013, S. 110–112. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6578331 (siehe S. 63).

[93] J. Gäbler und H. König. „Distributed Latency Estimation Using Global Knowledge for Mobile Collaborative Applications“. In: *Proceedings of the 11th International Wireless Communications and Mobile Computing Conference (IWCMC 2015)*. Dubrovnik, Croatia: IEEE, Aug. 2015, S. 474–479. DOI: 10.1109/IWCMC.2015.7289130. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7289130&tag=1 (siehe S. 110, 111, 116, 117, 119).

[94] J. Gäbler und H. König. „*Moversight*: a group communication protocol for mobile scenarios“. English. In: *Telecommunication Systems* 61.4 (Mai 2015), S. 1–22. ISSN: 1572-9451. DOI: 10.1007/s11235-015-0062-1. URL: http://dx.doi.org/10.1007/s11235-015-0062-1 (siehe S. 95).

[95] J. Gäbler und H. König. „Distributed Cluster-Topology Maintenance for Mobile Collaborative Applications“. In: *Proceedings of the 41st Annual IEEE Conference on Local Computer Networks (LCN 2016)*. Dubai, United Arab Emirates (UAE): IEEE, Nov. 2016, S. 271–279. DOI: 10.1109/LCN.2016.55. URL: http://ieeexplore.ieee.org/document/7796799/ (siehe S. 121, 126, 127, 130, 133, 137, 139, 140).

[96] J. Gäbler u. a. „Mobile kollaborative Apps“. In: *Smart Mobile Apps*. Hrsg. von S. Verclas und C. Linnhoff-Popien. Xpert.press. Springer Berlin Heidelberg, 2012, S. 447–463. ISBN: 978-3-642-22259-7 (siehe S. 15, 16, 18).

[97] J. Gäbler u. a. „uBeeMe; A Platform to Enable Mobile Collaborative Applications“. In: *Proceedings of the 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2013)*. ICST, the Institute for Computer Sciences, Social Informatics und Telecommunications Engineering. Austin, TX, USA: IEEE, Okt. 2013, S. 188–196. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6679984 (siehe S. 16).

[98] S. Gäbler. „Entwurf und Auswertung von Diensten zum Aufteilen und Vereinigen mobiler Gruppenkommunikationssitzungen.“ Magisterarb. Lehrstuhl Rechnernetze und Kommunikationssysteme, BTU Cottbus-Senftenberg, Konrad-Wachsmann-Allee 5, 03046 Cottbus: Brandenburgische Technische Universität Cottbus-Senftenberg, Aug. 2011 (siehe S. 143, 146, 147, 149, 150, 153).

[99] D. Gelernter. „Generative Communication in Linda“. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7.1 (Jan. 1985), S. 80–112. ISSN: 0164-0925. DOI: 10.1145/2363.2433. URL: http://doi.acm.org/10.1145/2363.2433 (siehe S. 28).

[100] I. GigaSpaces Technologies. *Cloudify*. online. 2015. URL: http://www.gigaspaces.com/ (siehe S. 29).

[101] S. Gilbert und N. Lynch. „Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services“. In: *ACM SIGACT News* 33.2 (Juni 2002), S. 51–59. ISSN: 0163-5700. DOI: 10.1145/564585.564601. URL: http://doi.acm.org/10.1145/564585.564601 (siehe S. 82).

[102] F. Glover und E. Taillard. „Tabu Search: An introduction“. English. In: *Annals of Operations Research* 41.1 (März 1993), S. 1–28. ISSN: 0254-5330. DOI: 10.1007/BF02078647. URL: http://dx.doi.org/10.1007/BF02078647 (siehe S. 125).

[103] L. Gong. „JXTA: A Network Programming Environment“. In: *IEEE Internet Computing* 5.3 (Mai 2001), S. 88–95. ISSN: 1089-7801. DOI: 10.1109/4236.935182. URL: https://www.computer.org/csdl/mags/ic/2001/03/w3088-abs.html (siehe S. 9).

[104] V. Gramoli u. a. „SONDe, a Self-Organizing Object Deployment Algorithm in Large-Scale Dynamic Systems“. In: *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*. Kaunas, Lithuania: IEEE, Mai 2008, S. 157–166. DOI: 10.1109/EDCC-7.2008.17. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4556000&tag=1 (siehe S. 124).

[105] R. Guerraoui u. a. „Atum: Scalable Group Communication Using Volatile Groups“. In: *Proceedings of the 17th International Middleware Conference*. Middleware '16. Trento, Italy: ACM, 2016, 19:1–19:14. ISBN: 978-1-4503-4300-8. DOI: 10.1145/2988336.2988356. URL: http://doi.acm.org/10.1145/2988336.2988356 (siehe S. 25).

[106] D. Gunawardena, P. Key und L. Massoulié. „Network Characteristics: Modelling, Measurements, and Admission Control“. English. In: *Quality of Service – IWQoS 2003*. Hrsg. von K. Jeffay, I. Stoica und K. Wehrle. Bd. 2707. LNCS. Springer, 2003, S. 3–20. ISBN: 978-3-540-40281-7. DOI: 10.1007/3-540-44884-5_1. URL: http://dx.doi.org/10.1007/3-540-44884-5_1 (siehe S. 111, 112, 116).

[107] S. K. S. Gupta und P. K. Srimani. „An Adaptive Protocol for Reliable Multicast in Mobile Multi-hop Radio Networks“. In: *Proceedings of the 2th IEEE Workshop on Mobile Computer Systems and Applications*. WMCSA '99. New Orleans, LA, USA: IEEE, Feb. 1999, S. 111–122. ISBN: 0-7695-0025-0. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=749283 (siehe S. 37).

[108] A. Hakiri u. a. „Publish/Subscribe-enabled Software Defined Networking for efficient and scalable IoT Communications“. In: *IEEE Communications Magazine* 53.9 (Sep. 2015), S. 48–54. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7263372. URL: http://ieeexplore.ieee.org/abstract/document/7263372/ (siehe S. 31).

[109] N. Hariri, B. Hariri und S. Shirmohammadi. „A Distributed Measurement Scheme for Internet Latency Estimation“. In: *Instrumentation and Measurement, IEEE Transactions on* 60.5 (Mai 2011), S. 1594–1603. ISSN: 0018-9456. DOI: 10.1109/TIM.2010.2092871 (siehe S. 114).

[110] N. Hayashibara u. a. „The ϕ Accrual Failure Detector“. In: *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems*. Los Alamitos, CA, USA: IEEE, Okt. 2004, S. 66–78. DOI: 10.1109/RELDIS.2004.1353004 (siehe S. 109, 114).

[111] Y. He und D. Perkins. „Achieving seamless handoffs via backhaul support in Wireless Mesh Networks“. English. In: *Telecommunication Systems* 52.4 (Apr. 2013), S. 1917–1930. ISSN: 1018-4864. DOI: 10.1007/s11235-011-9474-8. URL: http://dx.doi.org/10.1007/s11235-011-9474-8 (siehe S. 34).

[112] W. B. Heinzelman, A. P. Chandrakasan und H. Balakrishnan. „An Application-Specific Protocol Architecture for Wireless Microsensor Networks“. In: *IEEE Transactions on Wireless Communications* 1.4 (Dez. 2002), S. 660–670. DOI: 10.1109/TWC.2002.804190. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1045297&tag=1 (siehe S. 124, 128, 140).

[113] M. Hiltunen und R. Schlichting. „Properties of Membership Services“. In: *Proceedings of the 2th International Symposium on Autonomous Decentralized Systems (ISADS)*. Phoenix, AZ, USA: IEEE, Apr. 1995, S. 200–207. DOI: 10.1109/ISADS.1995.398973. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=398973 (siehe S. 44, 49).

[114] Q. Huang, C. Julien und G. C. Roman. „Relying on Safe Distance to Achieve Strong Partitionable Group Membership in Ad hoc Networks“. In: *IEEE Transactions on Mobile Computing* 3.2 (Apr. 2004), S. 192–205. ISSN: 1536-1233. DOI: 10.1109/TMC.2004.14. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1316056 (siehe S. 154).

[115] C. Hübsch u. a. *Reconnecting the Internet with ariba: Self-Organizing Provisioning of End-to-End Connectivity in Heterogeneous Networks*. Poster at the SIGCOMM 2009 Demo session. 2009. URL: http://www.spovnet.de/about/files/sigcomm-aribademo-poster.pdf (siehe S. 16).

[116] P. Hunt u. a. „ZooKeeper: Wait-free Coordination for Internet-scale Systems“. In: *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*. USENIXATC'10. Berkeley, CA, USA: USENIX Association, Juni 2010, S. 11–11. URL: http://dl.acm.org/citation.cfm?id=1855840.1855851 (siehe S. 30, 88, 109).

[117] IEEE 802 LAN/MAN Standards Committee. *IEEE 802.11 - 2012: IEEE Standard for Information Technology – Telecommunications and information exchange between systems, Local and metropolitan area networks – Specific requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society Standards. März 2012. URL: http://standards.ieee.org/about/get/802/802.11.html (siehe S. 15).

[118] International Telecommunication Union (ITU). *SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATION - Open Systems Interconnection - Model and Notation; Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. ITU-T Recommendation X.200. Juli 1994. URL: http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=2820&lang=en (siehe S. 16, 171).

[119] International Telecommunication Union (ITU). *SERIES X: DATA NET- AND OPEN SYSTEM COMMUNICATION - OSI management – Systems Management framework and architecture; Information technology – Open Systems Interconnection – Systems management overview*. ITU-T Recommendation X.701. Aug. 1997. URL: http://www.itu.int/rec/T-REC-X.701-199708-I (siehe S. 121).

[120] International Telecommunication Union (ITU). *SERIES M: TMN AND NETWORK MAINTENANCE: INTERNATIONAL TRANSMISSION SYSTEMS, TELEPHONE CIRCUITS, TELEGRAPHY, FACSIMILE AND LEASED CIRCUITS - Telecommunications management network; Overview of TMN Recommendations*. ITU-T Recommendation M.3000. Feb. 2000. URL: https://www.itu.int/rec/T-REC-M.3000-200002-I/en (siehe S. 121).

[121] V. Jacobson. „Congestion Avoidance and Control“. In: *SIGCOMM Computer Communication Review* 18.4 (Aug. 1988). see RFC 6298, S. 314–329. ISSN: 0146-4833. DOI: 10.1145/52325.52356. URL: http://doi.acm.org/10.1145/52325.52356 (siehe S. 109–111, 113).

[122] V. Jacobson. *A New Way to look at Networking*. Google Tech Talks. Aug. 2006. URL: https://www.youtube.com/watch?v=oCZMoY3q2uM (siehe S. 35).

[123] V. Jacobson u. a. „VoCCN: Voice-over Content-centric Networks“. In: *Proceedings of the 2009 Workshop on Re-architecting the Internet*. ReArch '09. Rome, Italy: ACM, 2009, S. 1–6. ISBN: 978-1-60558-749-3. DOI: 10.1145/1658978.1658980. URL: http://doi.acm.org/10.1145/1658978.1658980 (siehe S. 35).

[124] H. Jasani. „Quality of Service Evaluations of On Demand Mobile Ad-Hoc Routing Protocols“. In: *Proceedings of the 5th International Conference on Next Generation Mobile Applications, Services and Technologies*. Cardiff, Wales, United Kingdom: IEEE Computer Society, Sep. 2011, S. 123–128. ISBN: 978-0-7695-4496-0. DOI: 10.1109/NGMAST.2011.31. URL: http://www.computer.org/portal/web/csdl/doi/10.1109/NGMAST.2011.31 (siehe S. 29).

[125] J.-R. Jiang, C.-W. Hung und J.-W. Wu. „Bandwidth- and Latency-Aware Peer-to-Peer Instant Friendcast for Online Social Networks“. In: *Proceedings of the 16th International Conference on Parallel and Distributed Systems*. ICPADS '10. Washington, DC, USA: IEEE Computer Society, Dez. 2010, S. 829–834. ISBN: 978-0-7695-4307-9. DOI: 10.1109/ICPADS.2010.101 (siehe S. 114).

[126] G. T. Karetsos, E. Z.Tragos und G. I. Tsiropoulos. „A holistic approach to minimizing handover latency in heterogeneous wireless networking environments“. English. In: *Telecommunication Systems* 52.4 (Apr. 2013), S. 1845–1858. ISSN: 1018-4864. DOI: 10.1007/s11235-011-9498-0. URL: http://dx.doi.org/10.1007/s11235-011-9498-0 (siehe S. 34).

[127] C. Kaufman u. a. *RFC 7321: Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296 (INTERNET STANDARD). Updated by RFCs 7427, 7670. Internet Engineering Task Force, Okt. 2014. URL: http://www.ietf.org/rfc/rfc7296.txt (siehe S. 17).

[128] V. Kawadia und P. R. Kumar. „Power Control and Clustering in Ad Hoc Networks“. In: *Proceedings of the 22th Annual Joint Conference of the IEEE Computer and Communications Societies Conference on Computer Communications*. Bd. 1. IEEE Societies. San Francisco, CA, USA: IEEE, März 2003, S. 459–469. DOI: 10.1109/INFCOM.2003.1208697. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1208697 (siehe S. 121).

[129] S. Kent. *RFC 4303: IP Encapsulating Security Payload (ESP)*. RFC 4303 (Proposed Standard). Internet Engineering Task Force, Dez. 2005. URL: http://www.ietf.org/rfc/rfc4303.txt (siehe S. 17).

[130] B. Kim, D. Lee und D. Nam. „Scalable Group Membership Service for Mobile Internet“. In: *Proceedings of the 7th International Workshop on Object-Oriented Real-Time Dependable Systems, 2002. (WORDS 2002)*. San Diego, CA, USA: IEEE, Jan. 2002, S. 295–298. ISBN: 0-7695-1576-2. DOI: http://doi.ieeecomputersociety.org/10.1109/WORDS.2002.1000065. URL: http://www.computer.org/portal/web/csdl/doi/10.1109/WORDS.2002.1000065 (siehe S. 67).

[131] J. Kim u. a. „Group communication over LTE: A Radio Access Perspective“. In: *IEEE Communications Magazine* 54.4 (Apr. 2016), S. 16–23. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7452261 (siehe S. 25).

[132] Y. Kim, A. Perrig und G. Tsudik. „Group Key Agreement Efficient in Communication“. In: *IEEE Transactions on Computers* 53.7 (Juli 2004), S. 905–921. ISSN: 0018-9340. DOI: 10.1109/TC.2004.31. URL: http://dx.doi.org/10.1109/TC.2004.31 (siehe S. 154).

[133] R. Klauck u. a. „Mobile XMPP and Cloud Service Collaboration: An Alliance for Flexible Disaster Management“. In: *Proceedings of the 7th International ICST Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2011)*. Orlando, FL, USA: IEEE, Okt. 2011, S. 201–210. DOI: http://eudl.eu/doi/10.4108/icst.collaboratecom.2011.247108. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6144805 (siehe S. 18).

[134] U. Kubach u. a. „Business Web: Cloud-basierte Flexibilisierung und Mobilisierung von Geschäftsprozessen“. German. In: *Smart Mobile Apps - Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse*. Hrsg. von S. Verclas und C. Linnhoff-Popien. Xpert.press. Springer Berlin Heidelberg, 2012, S. 385–399. ISBN: 978-3-642-22258-0. DOI: 10.1007/978-3-642-22259-7_25. URL: http://dx.doi.org/10.1007/978-3-642-22259-7_25 (siehe S. 10).

[135] P. J. van Laarhoven und E. H. Aarts. *Simulated Annealing: Theory and Applications*. Bd. 37. Mathematics and Its Applications. Kluwer Academic Publishers, 1987. URL: https://books.google.de/books?id=-IgUab6Dp_IC&lpg=PR4&hl=de&pg=PR3#v=onepage&q&f=false (siehe S. 125).

[136] L. Lamport. „Time, Clocks, and the Ordering of Events in a Distributed System“. In: *Communications of the ACM* 21.7 (7 Juli 1978), S. 558–565. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/359545.359563 (siehe S. 45, 74, 109, 168).

[137] L. Lamport. „The Part-Time Parliament“. In: *ACM Transactions on Computer Systems (TOCS)* 16.2 (Mai 1998), S. 133–169. ISSN: 0734-2071. DOI: 10.1145/279227.279229. URL: http://doi.acm.org/10.1145/279227.279229 (siehe S. 23, 48).

[138] K.-H. Lee u. a. „A scalable network-based mobility management framework in heterogeneous IP-based networks“. English. In: *Telecommunication Systems* 52.4 (Apr. 2013), S. 1989–2002. ISSN: 1018-4864. DOI: 10.1007/s11235-011-9479-3. URL: http://dx.doi.org/10.1007/s11235-011-9479-3 (siehe S. 29, 34).

[139] B. Li und K. H. Wang. „NonStop: Continuous Multimedia Streaming in Wireless Ad Hoc Networks With Node Mobility“. In: *IEEE Journal on Selected Areas in Communications* 21.10 (Dez. 2003), S. 1627–1641. DOI: 10.1109/JSAC.2003.815964. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1254579 (siehe S. 29, 124).

[140] X. Li u. a. „DTN Routing in Vehicular Sensor Networks“. In: *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2008)*. New Orleans, LO, USA: IEEE, Nov. 2008, S. 1–5. DOI: 10.1109/GLOCOM.2008.ECP.150 (siehe S. 34).

[141] C. Liang u. a. „SyncCS: A Cloud Storage Based File Synchronization Approach“. In: *Journal of Software* 9.7 (2014), S. 1679–1686. DOI: 10.4304/jsw.9.7.1679-1686. URL: http://www.jsoftware.us/show-82-981-1.html (siehe S. 93).

[142] Y.-N. Lien, H.-C. Jang und T.-C. Tsai. „Design of P2Pnet: An Autonomous P2P Ad-Hoc Group Communication System“. In: *Proceedings of the 2009 10th International Conference on Mobile Data Management: Systems, Services and Middleware*. MDM '09. Taipei, Taiwan: IEEE, Mai 2009, S. 649–654. ISBN: 978-0-7695-3650-7. DOI: 10.1109/MDM.2009.111. URL: http://dx.doi.org/10.1109/MDM.2009.111 (siehe S. 29).

[143] D. Lin, M. Sherr und B. T. Loo. „Scalable and Anonymous Group Communication with MTor“. In: *PoPETs – Privacy Enhancing Technologies*. Bd. 2016. PoPETs 2. Berlin, Germany: de Gruyte, Dez. 2016, S. 22–39. DOI: https://doi.org/10.1515/popets-2016-0003. URL: https://www.degruyter.com/view/j/popets.2015.2016.issue-2/popets-2016-0003/popets-2016-0003.xml (siehe S. 25).

[144] M. Little, S. Shrivastava und S. Wheater. „Another look at the middleware for dependable distributed computing“. English. In: *Journal of Internet Services and Applications* 3.1 (Mai 2012), S. 95–105. ISSN: 1867-4828. DOI: 10.1007/s13174-011-0055-6. URL: http://link.springer.com/article/10.1007/s13174-011-0055-6 (siehe S. 32).

[145] F. Liu und H. König. „Security Considerations on Pervasive Realtime Collaboration“. In: *MASS*. IEEE, 2009, S. 722–727. URL: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5336928 (siehe S. 17).

[146] J. Liu u. a. „Group Management for Mobile Ad Hoc Networks: Design, Implementation and Experiment“. In: *Proceedings of the 6th international conference on Mobile Data Management*. MDM '05. Ayia Napa, Cyprus: ACM, 2005, S. 192–199. ISBN: 1-59593-041-8. DOI: http://doi.acm.org/10.1145/1071246.1071276 (siehe S. 37).

[147] P. G. López, R. G. Tinedo und J. M. B. Alsin. „Moving routing protocols to the user space in MANET middleware“. In: *Journal of Network and Computer Applications - Special issue: Middleware Trends for Network Applications* 33.5 (Sep. 2010), S. 588–602. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2010.03.018. URL: http://dx.doi.org/10.1016/j.jnca.2010.03.018 (siehe S. 29).

[148] P. G. Lopez u. a. „StackSync: Bringing Elasticity to Dropbox-like File Synchronization“. In: *Proceedings of the 15th International Middleware Conference*. Middleware '14. Bordeaux, France: ACM, 2014, S. 49–60. ISBN: 978-1-4503-2785-5. DOI: 10.1145/2663165.2663332. URL: http://doi.acm.org/10.1145/2663165.2663332 (siehe S. 93).

[149] J. Luo, P. T. Eugster und J.-P. Hubaux. „Pilot: Probabilistic Lightweight Group Communication System for Ad Hoc Networks“. In: *IEEE Transactions on Mobile Computing* 3.2 (Apr. 2004), S. 164–179. DOI: 10.1109/TMC.2004.12. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1316054 (siehe S. 23, 29).

[150] C. P. Malloth. „Conception and Implementation of a Toolkit for building Fault-Tolerant Distributed Applications in Large Scale Networks“. Diss. Lausanne: IIF, 1996, S. 179. URL: https://infoscience.epfl.ch/record/32034 (siehe S. 88).

[151] R. M. Mateo, B. Gerardo und J. Lee. „Scalable Group Communication for Efficient Knowledge Sharing in Cloud-Enabled Robots“. In: *Knowledge-Based Information Systems in Practice*. Hrsg. von J. W. Tweedale u. a. Bd. 30. Smart Innovation, Systems and Technologies (SIST). Cham: Springer International Publishing, 2015, S. 201–216. ISBN: 978-3-319-13545-8. DOI: 10.1007/978-3-319-13545-8_12. URL: https://doi.org/10.1007/978-3-319-13545-8_12 (siehe S. 33).

[152] K. Mayes u. a. „Reliable Group Communication for Dynamic and Resource-Constrained Environments“. In: *2009 20th International Workshop on Database and Expert Systems Application*. IEEE, Aug. 2009, S. 14–18. DOI: 10.1109/DEXA.2009.9. URL: http://ieeexplore.ieee.org/document/5337573/ (siehe S. 147).

[153] D. McGrew und P. Hoffman. *RFC 7321: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. RFC 7321 (Proposed Standard). Internet Engineering Task Force, Aug. 2014. URL: http://www.ietf.org/rfc/rfc7321.txt (siehe S. 17).

[154] H. Meling u. a. „Jgroup/ARM: a distributed object group platform with autonomous replication management“. In: *Software: Practice and Experience* 38.9 (Juli 2008), S. 885–923. DOI: 10.1002/spe.853 (siehe S. 144).

[155] P. Mell und T. Grance. *The NIST Definition of Cloud Computing*. Technical Report. Gaithersburg, MY, USA: National Institute of Standards und Technology, Information Technology Laboratory, Juli 2009. URL: http://www.csrc.nist.gov/groups/SNS/cloud-computing/ (siehe S. 32).

[156] P. M. Melliar-Smith und L. E. Moser. „Surviving Network Partitioning“. In: *IEEE Computer* 31.3 (März 1998), S. 62–68. DOI: 10.1109/2.660191 (siehe S. 146).

[157] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard, Version 3.1*. Juni 2015. URL: http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf (siehe S. 9).

[158] H. Miranda, A. Pinto und L. Rodrigues. „*Appia*, a flexible protocol kernel supporting multiple coordinated channels“. In: *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001)*. Los Alamitos, CA, USA: IEEE, Apr. 2001, S. 707–710. ISBN: 0-7695-1077-9. DOI: 10.1109/ICDSC.2001.919005 (siehe S. 24).

[159] N. Mohammed u. a. „Mechanism Design-Based Secure Leader Election Model for Intrusion Detection in MANET“. In: *IEEE Transactions Dependable and Secure Computing* 8.1 (Jan. 2011), S. 89–103. ISSN: 1545-5971. DOI: 10.1109/TDSC.2009.22 (siehe S. 129).

[160] T. Montgomery. „Design, Implementation, and Verification of the Reliable Multicast Protocol“. Magisterarb. West Virginia University, Dez. 1994. URL: https://ntrs.nasa.gov/search.jsp?R=19960011362 (siehe S. 88).

[161] A. Montresor. „The JGroup Reliable Distributed Object Model“. In: *Proceedings of the 2th IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS'99)*. Department of Computer Science, University of Helsinki, Finland. Helsinki, Finland: Springer US., Juli 1999, S. 389–402. URL: https://www.cs.helsinki.fi/events/DAIS99/index.html (siehe S. 23, 24, 29, 30, 33, 44, 53, 54, 83, 87, 131).

[162] L. E. Moser u. a. „Extended Virtual Synchrony“. In: *Proceedings of the 14th International Conference on Distributed Computing Systems*. Poznan, Poland: IEEE, Juni 1994, S. 56–65. DOI: 10.1109/ICDCS.1994.302392. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=302392 (siehe S. 49, 146).

[163] L. E. Moser u. a. „Totem: A Fault-Tolerant Multicast Group Communication System“. In: *Communications of the ACM* 39.4 (4 Apr. 1996), S. 54–63. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/227210.227226 (siehe S. 75, 189).

[164] S. Navaratnam, S. Chanson und G. Neufeld. „Reliable Group Communication in Distributed Systems“. In: *Proceedings of the 8th International Conference on Distributed Computing Systems*. San Jose, CA, USA: IEEE, Juni 1988, S. 439–446. DOI: 10.1109/DCS.1988.12546 (siehe S. 44, 75).

[165] B. Neto u. a. „A Coordination Framework for Dynamic Adaptation in Ubiquitous Systems Based on Distributed Tuple Space“. In: *Proceedings of the 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC 2013)*. Sardina, Italy: IEEE, Juli 2013, S. 1430–1435. DOI: 10.1109/IWCMC.2013.6583766. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6583766 (siehe S. 29).

[166] A. Olteanu u. a. „An Adaptive Scheduling Approach in Distributed Systems“. In: *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*. ICCP '10. Cluj-Napoca, Cluj, Romania: IEEE, Aug. 2010, S. 435–442. ISBN: 978-1-4244-8228-3. DOI: 10.1109/ICCP.2010.5606400. URL: http://dx.doi.org/10.1109/ICCP.2010.5606400 (siehe S. 123, 125).

[167] S. J. Opitz, N. Todtenberg und H. König. „Mobile Bandwidth Prediction in the Context of Emergency Medical Service“. In: *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*. PETRA '14. Rhodes, Greece: ACM, 2014, S. 1–7. ISBN: 978-1-4503-2746-6. DOI: 10.1145/2674396.2674411. URL: http://doi.acm.org/10.1145/2674396.2674411 (siehe S. 17).

[168] K. Ostrowski, K. Birman und D. Dolev. „Extensible Architecture for High-Performance, Scalable, Reliable Publish-Subscribe Eventing and Notification“. In: *International Journal of Web Services Research (IJWSR)* 4.4 (2007), S. 18–58. DOI: 10.4018/jwsr.2007100102. URL: http://www.igi-global.com/gateway/article/3108 (siehe S. 29, 36, 77).

[169] Z. Ou u. a. „Performance evaluation of a Kademlia-based communication oriented P2P system under churn“. In: *Computer Networks* 54.5 (Apr. 2010), S. 689–705. ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/j.comnet.2009.09.022. URL: http://www.sciencedirect.com/science/article/pii/S1389128609002990 (siehe S. 10, 122).

[170] P. Patil u. a. „Scalable and Adaptive Software Defined Network Management for Cloud-hosted Group Communication Applications“. In: *Proceedings of the10th International Conference on Utility and Cloud Computing*. UCC '17. Austin, Texas, USA: ACM, Dez. 2017, S. 111–120. ISBN: 978-1-4503-5149-2. DOI: 10.1145/3147213.3147220. URL: http://doi.acm.org/10.1145/3147213.3147220 (siehe S. 33).

[171] V. Paxson u. a. *RFC 6298: Computing TCP's Retransmission Timer*. RFC 6298 (Proposed Standard). Internet Engineering Task Force, Juni 2011. URL: http://www.ietf.org/rfc/rfc6298.txt (siehe S. 113).

[172] S. G. Pease, I. W. Phillips und L. Guan. „Adaptive Intelligent Middleware Architecture for Mobile Real-Time Communications“. In: *IEEE Transactions on Mobile Computing* 15.3 (März 2016), S. 572–585. ISSN: 1536-1233. DOI: 10.1109/TMC.2015.2412932. URL: http://ieeexplore.ieee.org/abstract/document/7095600/ (siehe S. 27).

[173] C. Perkins. *RFC 3220: IP Mobility Support for IPv4*. RFC 3220 (Proposed Standard). Obsoleted by RFC 3344. Internet Engineering Task Force, Jan. 2002. URL: http://www.ietf.org/rfc/rfc3220.txt (siehe S. 29).

[174] C. Perkins, D. Johnson und J. Arkko. *RFC 6275: Mobility Support in IPv6*. RFC 6275 (Proposed Standard). Internet Engineering Task Force, Juli 2011. URL: http://www.ietf.org/rfc/rfc6275.txt (siehe S. 29).

[175] I. Petri u. a. „Exploring Models and Mechanisms for Exchanging Resources in a Federated Cloud“. In: *Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E)*. Bosten, MA, USA: IEEE, März 2014, S. 215–224. DOI: 10.1109/IC2E.2014.9. URL: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6903476 (siehe S. 29).

[176] G. Picco, A. Murphy und G. Roman. „LIME: Linda Meets Mobility“. In: *Proceedings of the 1999 International Conference on Software Engineering*. Mai 1999, S. 368–377. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=841027 (siehe S. 28, 36).

[177] C. Pinciroli, A. Lee-Brown und G. Beltrame. „A Tuple Space for Data Sharing in Robot Swarms“. In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*. BICT'15. New York City, United States: ICST (Institute for Computer Sciences, Social-Informatics und Telecommunications Engineering), 2016, S. 287–294. ISBN: 978-1-63190-100-3. DOI: 10.4108/eai.3-12-2015.2262503. URL: http://dx.doi.org/10.4108/eai.3-12-2015.2262503 (siehe S. 29).

[178] M. Pink, T. Pietsch und H. König. „Towards a Seamless Mobility Solution for the Real World: Handover Decision“. In: *Proceedings of the International Symposium on Wireless Communication Systems (ISWCS 2012)*. IEEE. Paris, FRA, Aug. 2012, S. 651–655. DOI: 10.1109/ISWCS.2012.6328448. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6328448 (siehe S. 17).

[179] G. J. Popek und R. P. Goldberg. „Formal Requirements for Virtualizable Third Generation Architectures“. In: *Communications of the ACM* 17.7 (Juli 1974), S. 412–421. ISSN: 0001-0782. DOI: 10.1145/361011.361073. URL: http://doi.acm.org/10.1145/361011.361073 (siehe S. 33).

[180] J. Postel. *RFC 0793: Transmission Control Protocol*. RFC 0793 (Standard). See also STD0007. Status: STANDARD. Sep. 1981. URL: https://www.ietf.org/rfc/rfc793.txt (siehe S. 113).

[181] J. Protic, M. Tomasevic und V. Milutinovic. „Distributed Shared Memory: Concepts and Systems“. In: *IEEE Parallel Distributed Technology: Systems Applications* 4.2 (Aug. 1996), S. 63–71. ISSN: 1063-6552. DOI: 10.1109/88.494605. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=494605 (siehe S. 28).

[182] A. Ricci u. a. „Spatial Tuples: Augmenting Physical Reality with Tuple Spaces“. In: *Intelligent Distributed Computing X*. Hrsg. von C. Badica u. a. Bd. 678. Studies in Computational Intelligence. Cham: Springer International Publishing, Okt. 2017, S. 121–130. ISBN: 978-3-319-48829-5. URL: https://link.springer.com/chapter/10.1007%2F978-3-319-48829-5_12 (siehe S. 29).

[183] L. Rodrigues und K. Guo. „Partitionable Light-Weight Groups“. In: *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000)*. ICDCS '00. Washington, DC, USA: IEEE Computer Society, 2000, S. 38–46. ISBN: 0-7695-0601-1. URL: http://dl.acm.org/citation.cfm?id=850927.851775 (siehe S. 147).

[184] G.-C. Roman, Q. Huang und A. Hazemi. „Consistent Group Membership in Ad Hoc Networks“. In: *Proceedings of the 23rd International Conference on Software Engineering (ICSE'01)*. Los Alamitos, CA, USA: IEEE Computer Society, Mai 2001, S. 381–388. ISBN: 0-7695-1050-7. DOI: http://doi.ieeecomputersociety.org/10.1109/ICSE.2001.919111. URL: http://www2.computer.org/portal/web/csdl/doi/10.1109/ICSE.2001.919111 (siehe S. 138, 147, 154, 188).

[185] A. V. Rudolf und Hornig. „An Overview of the OMNeT++ Simulation Environment“. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08 60. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics und Telecommunications Engineering), 2008, S. 1–10. ISBN: 978-963-9799-20-2. URL: http://dl.acm.org/citation.cfm?id=1416222.1416290 (siehe S. 139).

[186] V. Sacramento u. a. „MoCA: A Middleware for Developing Collaborative Applications for Mobile Users“. In: *IEEE Distributed Systems Online* 5.10 (Okt. 2004), S. 1–14. ISSN: 1541-4922. DOI: 10.1109/MDSO.2004.26. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1355907 (siehe S. 9, 15–18).

[187] P. Saint-Andre. *XEP-0174: Serverless Messaging*. Nov. 2008. URL: http://www.xmpp.org/extensions/xep-0174.html (siehe S. 18, 31, 37).

[188] P. Saint-Andre. *RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core*. RFC 6120 (Proposed Standard). Internet Engineering Task Force, März 2011. URL: http://www.ietf.org/rfc/rfc6120.txt (siehe S. 18, 31, 144).

[189] P. Saint-Andre. *XEP-0045: Multi-User Chat*. Feb. 2012. URL: http://xmpp.org/extensions/xep-0045.html (siehe S. 31).

[190] M. Satyanarayanan u. a. „The Case for VM-Based Cloudlets in Mobile Computing“. In: *IEEE Pervasive Computing* 8.4 (Okt. 2009), S. 14–23. ISSN: 1536-1268. DOI: 10.1109/MPRV.2009.82. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5280678&tag=1 (siehe S. 33).

[191] N. Schiper und F. Pedone. „Fast, Flexible, and Highly Resilient Genuine Fifo and Causal Multicast Algorithms“. In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. Sierre, Switzerland: ACM, 2010, S. 418–422. ISBN: 978-1-60558-639-7. DOI: 10.1145/1774088.1774178. URL: http://doi.acm.org/10.1145/1774088.1774178 (siehe S. 44).

[192] G. Schneider. „Entwurf und Simulation eines Dienstes zur Erkennung von Netzpartitionierung in einer mobilen P2P-Gruppenkommunikationssitzung“. Magisterarb. Lehrstuhl Rechnernetze und Kommunikationssysteme, BTU Cottbus-Senftenberg, Konrad-Wachsmann-Allee 5, 03046 Cottbus: Brandenburgische Technische Universität Cottbus-Senftenberg, Jan. 2013 (siehe S. 87, 93, 100, 101, 104–107).

[193] K. Scott und S. Burleigh. *RFC 5050: Bundle Protocol Specification*. RFC 5050 (Experimental). Internet Engineering Task Force, Nov. 2007. URL: http://www.ietf.org/rfc/rfc5050.txt (siehe S. 34).

[194] W. Segall und D. Arnold. „Elvin Has Left the Building: A Publish/Subscribe Notification Service with Quenching“. In: *Proceedings of the 1997 Australian UNIX Users Group*. Brisbane, Australia, 1997. URL: http://www.elvin.org/papers/auug97/auug97.html (siehe S. 31).

[195] A. Sendonaris, E. Erkip und B. Aazhang. „User Cooperation Diversity. Part I. System Description“. In: *IEEE Transactions on Communications* 51.11 (Nov. 2003), S. 1927–1938. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2003.818096. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1246003&tag=1 (siehe S. 109).

[196] W. Shang u. a. „Named Data Networking of Things (Invited Paper)“. In: *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE Computer Society, Apr. 2016, S. 117–128. DOI: 10.1109/IoTDI.2015.44. URL: http://ieeexplore.ieee.org/abstract/document/7471356/ (siehe S. 35).

[197] G. Shao, F. Berman und R. Wolski. „Master/Slave Computing on the Grid“. In: *Proceedings of the 9th Heterogeneous Computing Workshop (HCW 2000)*. Cancún, Mexiko: IEEE Communication Society, 2000, S. 3–16. DOI: 10.1109/HCW.2000.843728 (siehe S. 128).

[198] W.-L. Shen u. a. „Autonomous Mobile Mesh Networks“. In: *IEEE Transactions on Mobile Computing* 13.2 (Dez. 2012), S. 364–376. ISSN: 1536-1233. DOI: http://doi.ieeecomputersociety.org/10.1109/TMC.2012.259 (siehe S. 144).

[199] M. Skorepa und R. Klugl. „Enhanced analytical method for IP mobility handover schemes cost evaluation“. English. In: *Telecommunication Systems* 52.3 (März 2013), S. 1573–1582. ISSN: 1018-4864. DOI: 10.1007/s11235-011-9524-2. URL: http://dx.doi.org/10.1007/s11235-011-9524-2 (siehe S. 29, 34).

[200] J. Solomon und S. Glass. *RFC 2290: Mobile-IPv4 Configuration Option for PPP IPCP*. Updates RFC2002. Status: PROPOSED STANDARD. Internet Engineering Task Force, Feb. 1998. URL: http://www.faqs.org/rfcs/rfc2290.html (siehe S. 17).

[201] C. Spiker u. a. „Experiment and Field Demonstration of Serverless Group Communication“. In: *Proceedings of the 2015 IEEE Military Communications Conference*. MILCOM 2015. Okt. 2015, S. 127–132. DOI: 10.1109/MILCOM.2015.7357430 (siehe S. 25, 58).

[202] J. Stanik. „A Conversation with Van Jacobson“. In: *Queue* 7.1 (Jan. 2009), S. 8–16. ISSN: 1542-7730. DOI: 10.1145/1508211.1508215. URL: http://doi.acm.org/10.1145/1508211.1508215 (siehe S. 35).

[203] F. Stubbe. „Entwurf und Umsetzung eines verteilten Topologie-Wartungsdienstes für das mobile Gruppenkommunikationsprotokoll Moversight.“ Magisterarb. Lehrstuhl Rechnernetze und Kommunikationssysteme, BTU Cottbus-Senftenberg, Konrad-Wachsmann-Allee 5, 03046 Cottbus: Brandenburgische Technische Universität Cottbus-Senftenberg, Apr. 2014 (siehe S. 67, 121, 125–127, 137).

[204] J. Sussman, I. Keidar und K. Marzullo. „Optimistic Virtual Synchrony“. In: *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS-2000)*. Nürnberg, Germany: IEEE Computer Society, Okt. 2000, S. 42–51. DOI: 10.1109/RELDI.2000.885391. URL: http://www.computer.org/portal/web/csdl/doi/10.1109/RELDI.2000.885391 (siehe S. 51, 52, 89).

[205] S. Thomson, T. Narten und T. Jinmei. *RFC 4862: IPv6 Stateless Address Autoconfiguration*. RFC 4862 (Draft Standard). Updated by RFC 7527. Internet Engineering Task Force, Sep. 2007. URL: http://www.ietf.org/rfc/rfc4862.txt (siehe S. 18, 31).

[206] C. Toh. „Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks“. In: *IEEE Communications Magazine* 39.6 (Juni 2001), S. 138–147. ISSN: 0163-6804. DOI: 10.1109/35.925682. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=925682 (siehe S. 122).

[207] S. Tornell u. a. „DTN Protocols for Vehicular Networks: An Application Oriented Overview“. In: *IEEE Communications Surveys & Tutorials* 17.2 (Mai 2015), S. 868–887. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2375340. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6977878 (siehe S. 34).

[208] J. Totzke und K. Klug. „Agile Geschäftsprozesse durch integrierte mobile Kommunikation“. German. In: *Smart Mobile Apps - Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse*. Hrsg. von S. Verclas und C. Linnhoff-Popien. Xpert.press. Springer Berlin Heidelberg, 2012, S. 333–349. ISBN: 978-3-642-22258-0. DOI: 10.1007/978-3-642-22259-7_22. URL: http://dx.doi.org/10.1007/978-3-642-22259-7_22 (siehe S. 10).

[209] R. Van Renesse, T. M. Hickey und K. P. Birman. *Design and Performance of Horus: A Lightweight Group Communications System*. Technical Report TR94-1442. Cornell University Computer Science Technical Reports. Ithaca, NY, USA: Department of Computer Science, Cornell University, Aug. 1994. URL: https://ecommons.cornell.edu/handle/1813/6228 (siehe S. 75, 87).

[210] P. Veríssimo und L. Rodrigues. „Distributed System Paradigms“. English. In: *Distributed Systems for System Architects*. Bd. 1. Advances in Distributed Computing and Middleware. Springer US, 2001, S. 21–88. ISBN: 978-0-7923-7266-0. DOI: 10.1007/978-1-4615-1663-7_2. URL: http://dx.doi.org/10.1007/978-1-4615-1663-7_2 (siehe S. 45).

[211] D. Waitzman, C. Partridge und S. Deering. *RFC 1075: Distance Vector Multicast Routing Protocol*. RFC 1075 (Experimental). Internet Engineering Task Force, Nov. 1988. URL: http://www.ietf.org/rfc/rfc1075.txt (siehe S. 25).

[212] O. Waldhorst u. a. „Spontaneous Virtual Networks: On the Road towards the Internet's Next Generation". In: *it - Information Technology Special Issue on Next Generation Internet* 50.6 (Dez. 2008), S. 367–375. URL: http://telematics.tm.kit.edu/article.php?publication_id=310 (siehe S. 27, 30, 109).

[213] Q. Wang u. a. „File Updates Under Random/Arbitrary Insertions and Deletions". In: *IEEE Transactions on Information Theory* 63.10 (Okt. 2017), S. 6487–6513. ISSN: 0018-9448. DOI: 10.1109/TIT.2017.2705100 (siehe S. 93).

[214] Y. Wang und J. Wu. „A dynamic multicast tree based routing scheme without replication in delay tolerant networks ". In: *Journal of Parallel and Distributed Computing* 72.3 (März 2012), S. 424–436. ISSN: 0743-7315. DOI: http://dx.doi.org/10.1016/j.jpdc.2011.11.007. URL: http://www.sciencedirect.com/science/article/pii/S0743731511002231 (siehe S. 35).

[215] M. Weiser. „The Computer for the 21st Century". In: *ACM SIGMOBILE Mobile Computing and Communications Review - Special issue dedicated to Mark Weiser* 3.3 (Juli 1999), S. 3–11. ISSN: 1559-1662. DOI: 10.1145/329124.329126. URL: http://doi.acm.org/10.1145/329124.329126 (siehe S. 9).

[216] J. E. White. *RFC 0707: High-level framework for network-based resource sharing*. Status: UNKNOWN. Internet Engineering Task Force, Dez. 1975. URL: https://www.rfc-editor.org/rfc/rfc707.txt (siehe S. 33).

[217] U. Wilhelm und A. Schiper. „A Hierarchy of Totally Ordered Multicasts". In: *Proceedings of the 14th Symposium on Reliable Distributed Systems*. Bad Neuenahr, Germany: IEEE Communication Society, Sep. 1995, S. 106–115. DOI: 10.1109/RELDIS.1995.526218 (siehe S. 45).

[218] J. J. Wilke. „Coordination Languages and MPI Perturbation Theory: The FOX Tuple Space Framework for Resilience". In: *Proceedings of the 2014 IEEE International Parallel Distributed Processing Symposium Workshops (IPDPSW)*. IEEE. Phoenix, AZ, USA, Mai 2014, S. 1208–1217. DOI: 10.1109/IPDPSW.2014.136. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6969518 (siehe S. 29).

[219] L. Wood u. a. „Saratoga: a Delay-Tolerant Networking convergence layer with efficient link utilization". In: *International Workshop on Satellite and Space Communicationsn, 2007. IWSSC '07*. Salzburg, AT: IEEE, Sep. 2007, S. 168–172. DOI: 10.1109/IWSSC.2007.4409410. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4409410 (siehe S. 34).

[220] J. Wu und Y. Wang. „A non-replication multicasting scheme in delay tolerant networks". In: *Proceedings of the IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS 2010)*. San Francisco, CA, USA: IEEE, Nov. 2010, S. 89–98. DOI: 10.1109/MASS.2010.5663963. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5663963 (siehe S. 35).

[221] J. Wu und Y. Wang, Hrsg. *Opportunistic Mobile Social Networks*. Technology & Engineering. CRC Press, Taylor & Francis Group, Aug. 2014. ISBN: 9781466594944. URL: https://books.google.de/books?id=3SAbBAAAQBAJ (siehe S. 35).

[222] X. Xiang, X. Wang und Y. Yang. „Supporting Efficient and Scalable Multicasting over Mobile Ad Hoc Networks". In: *IEEE Transactions on Mobile Computing* 10.4 (Sep. 2011), S. 544–559. ISSN: 1536-1233. DOI: http://doi.ieeecomputersociety.org/10.1109/TMC.2010.176 (siehe S. 144).

[223] K. Xu u. a. „Latency Estimation for Time-Sensitive Applications under Wireless Network Coding Scheme". In: *WRI International Conference on Communications and Mobile Computing, 2009 (CMC '09)*. Bd. 2. Jan. 2009, S. 263–266. DOI: 10.1109/CMC.2009.42. URL: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=4797129#sec6 (siehe S. 116).

[224] Y.-S. Yen u. a. „A novel predictive scheduling handover on mobile IPv6“. In: *Telecommunication Systems* 52.2 (Feb. 2013), S. 461–473. ISSN: 1018-4864. DOI: 10.1007/s11235-011-9448-x. URL: http://dx.doi.org/10.1007/s11235-011-9448-x (siehe S. 34).

[225] M. M. Zavlanos, L. Spesivtsev und G. J. Pappas. „A Distributed Auction Algorithm for the Assignment Problem“. In: *Proccedings of the 47th IEEE Conference on Decision and Control (CDC 2008)*. Cancún, Mexiko: IEEE Communications Society, Dez. 2008, S. 1212–1217. DOI: 10.1109/CDC.2008.4739098 (siehe S. 129).

[226] L. Zhang u. a. *Named Data Networking (NDN) Project*. Technical Report NDN-0001. Los Angeles, CA, USA: University of California, Okt. 2010. URL: http://named-data.net/techreport/TR001ndn-proj.pdf (siehe S. 35).

[227] L. Zhang u. a. „Named Data Networking“. In: *ACM SIGCOMM Computer Communication Review* 44.3 (Juli 2014), S. 66–73. ISSN: 0146-4833. DOI: 10.1145/2656877.2656887. URL: http://doi.acm.org/10.1145/2656877.2656887 (siehe S. 35).

[228] Z. Zhu, R. Wakikawa und L. Zhang. *RFC 6301: A Survey of Mobility Support in the Internet*. RFC 6301 (Informational). Internet Engineering Task Force, Juli 2011. URL: http://www.ietf.org/rfc/rfc6301.txt (siehe S. 29).

[229] S. Q. Zhuang u. a. „Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination“. In: *Proceedings of the 11th International Workshop on Network and Operating Systems Support for digital Audio and Video*. NOSSDAV '01. Port Jefferson, New York, United States: ACM, 2001, S. 11–20. ISBN: 1-58113-370-7. DOI: 10.1145/378344.378347 (siehe S. 3).

[230] ZigBee Alliance. *ZigBee 3.0: The Foundation for the Internet of Things*. online. 2015. URL: http://www.zigbee.org/zigbee-for-developers/zigbee3-0/ (siehe S. 15).

[231] M. Zühlke und H. König. „GCP - A Group Communication Protocol for Supporting Closed Groups in the Internet“. In: *Proceedings of IFIP TC6 WG 6.7 7th International Conference on Smart Networks 2002 (SMARTNET 2002)*. Hrsg. von O. Martikainen, K. Raatikainen und J. Hyvärinen. Bd. 84. IFIP Advances in Information and Communication Technology. Saariselkä, Lapland, Finland: Springer US, Apr. 2002, S. 211–227. ISBN: 1-4020-7008-X. DOI: 10.1007/978-0-387-35584-9_13. URL: http://link.springer.com/chapter/10.1007%2F978-0-387-35584-9_13 (siehe S. 23, 24, 41, 44, 48, 88, 144, 172).