University of Passau

Faculty of Computer Science and Mathematics
Chair of Computer Networks and Computer Communications

PhD thesis

Towards High Performability in Advanced Metering Infrastructures

Michael Niedermeier

1. Reviewer Hermann de Meer
Chair of Computer Networks and Computer Communications
University of Passau

2. Reviewer Anne Remke
Group of Safety-Critical Systems
University of Münster

11/06/2020

Michael Niedermeier
Towards High Performability in Advanced Metering Infrastructures
PhD thesis, 11/06/2020
Reviewers: Hermann de Meer and Anne Remke

University of Passau

Chair of Computer Networks and Computer Communications
Faculty of Computer Science and Mathematics
Innstrasse 43
94032 Passau

Abstract

The current movement towards a smart grid serves as a solution to present power grid challenges by introducing numerous monitoring and communication technologies. A dependable, yet timely exchange of data is on the one hand an existential prerequisite to enable Advanced Metering Infrastructure (AMI) services, yet on the other a challenging endeavor, because the increasing complexity of the grid fostered by the combination of Information and Communications Technology (ICT) and utility networks inherently leads to dependability challenges.

To be able to counter this dependability degradation, current approaches based on high-reliability hardware or physical redundancy are no longer feasible, as they lead to increased hardware costs or maintenance, if not both. The flexibility of these approaches regarding vendor and regulatory interoperability is also limited. However, a suitable solution to the AMI dependability challenges is also required to maintain certain regulatory-set performance and Quality of Service (QoS) levels.

While a part of the challenge is the introduction of ICT into the power grid, it also serves as part of the solution. In this thesis a Network Functions Virtualization (NFV) based approach is proposed, which employs virtualized ICT components serving as a replacement for physical devices. By using virtualization techniques, it is possible to enhance the performability in contrast to hardware based solutions through the usage of virtual replacements of processes that would otherwise require dedicated hardware. This approach offers higher flexibility compared to hardware redundancy, as a broad variety of virtual components can be spawned, adapted and replaced in a short time. Also, as no additional hardware is necessary, the incurred costs decrease significantly. In addition to that, most of the virtualized components are deployed on Commercial-Off-The-Shelf (COTS) hardware solutions, further increasing the monetary benefit.

The approach is developed by first reviewing currently suggested solutions for AMIs and related services. Using this information, virtualization technologies are investigated for their performance influences, before a virtualized service infrastructure is devised, which replaces selected components by virtualized counterparts. Next, a novel model, which allows the separation of services and hosting substrates is developed, allowing the introduction of virtualization technologies to abstract from the underlying architecture. Third, the performability as well as monetary savings are investigated by evaluating the developed approach in several scenarios using analytical and simulative model analysis as well as proof-of-concept approaches. Last, the practical applicability and possible regulatory challenges of the approach are identified and discussed.

Results confirm that—under certain assumptions—the developed virtualized AMI is superior to the currently suggested architecture. The availability of services can be severely increased and network delays can be minimized through centralized hosting. The availability can be increased from 96.82 % to 98.66 % in the given scenarios, while decreasing the costs by over 60 % in comparison to the currently suggested AMI architecture. Lastly, the performability analysis of a virtualized service prototype employing performance analysis and a Musa-Okumoto approach reveals that the AMI requirements are fulfilled.

Acknowledgement

Throughout the writing of this dissertation I have received a great deal of support and assistance, both from a professional academic as well as personal level.

I would first like to express my sincere gratitude to my primary supervisor, Prof. Hermann de Meer. Over the years of my career, he brought me closer to multiple important aspects of academic research, among others the importance of correct assumptions, methodological impact, and formalization. Moreover, he also provided me with numerous opportunities to gain insights into a wide range of research fields through national and international projects, which in turn enabled me to cooperate with interesting partners from industry and academia.

Also, I want to cordially thank Prof. Anne Remke for kindly agreeing to serve as second reviewer.

I feel sincerely indebted to my colleagues at the Chair of Computer Networks and Computer Communications, who made my time at the at the University of Passau the most valuable and pleasant experience I could hope for. In particular, I want to thank my former colleagues Dr. Patrick Wüchner, Prof. Andreas Fischer, Prof. Andreas Berl, Ralph Herkenhöhner, and Gergő Lovász, as well as my current research fellows for their support, inspiration, and fruitful discussions on research-related and non-research-related topics.

Apart from the colleges in my group, I would like to give special credit to two academic co-workers and friends, namely Matthias Schmid and Stefan Brand. While Matthias is topic-wise located in the database domain, he never refrained from listening to questions regarding current research challenges and offered his critique. Moreover, he provided a more than welcome distraction by being the most ambitious gym buddy possible, which allowed me to keep a level head in times of need. Special thanks also go to Stefan Brand, who provided invaluable programming work in the development of the *AMIgo* simulation framework. Even after finishing his master thesis he selflessly provided support and even extended his work, for which I can only provide my severe gratitude.

I am permanently thankful to my parents and brothers for their loving support, no matter if professional or non-academic questions were regarded.

Last but definitely not least, I want to express my sincere gratefulness to my beloved girlfriend Sabine for guarding my back and enduring my (for any person except myself) ridiculous thesis writing sessions throughout uncounted nights.

Table of Contents

List of Acronyms **xi**

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**
1.1 Dependable Smart Grids . . . 2
1.2 Challenges and Solution Approach . . . 3
1.3 Contributions . . . 5
1.4 Thesis Structure . . . 7

2 Background and Related Work **11**
2.1 Background . . . 12
2.1.1 Dependability, Performance and Performability . . . 13
2.1.1.1 Dependability . . . 13
2.1.1.2 Performance . . . 16
2.1.1.3 Performability . . . 18
2.1.2 Smart Grid and Advanced Metering Infrastructure . . . 21
2.1.2.1 Smart Grid . . . 21
2.1.2.2 Advanced Metering Infrastructure . . . 22
2.1.3 Virtualization . . . 29
2.1.3.1 Host Virtualization . . . 29
2.1.3.2 Network Virtualization . . . 30
2.1.3.3 Network Function Virtualization . . . 30
2.2 Related Work . . . 32
2.2.1 Performability/Dependability in Smart Grids . . . 32
2.2.2 Virtualization for Performability/Dependability . . . 34
2.2.3 Combining Approaches & Own Contributing Work . . . 35
2.3 Summary . . . 36

3 Improvement of AMI Systems using Virtualization **39**
3.1 Overview of Current AMI Architecture . . . 40
3.1.1 Architecture Overview . . . 40
3.1.2 Preliminary Evaluation . . . 40
3.2 Performability Impact of Virtualization . . . 41
3.2.1 Reliability . . . 42
3.2.2 Maintainability . . . 45
3.2.2.1 Corrective Maintenance . . . 45
3.2.2.2 Preventive Maintenance . . . 49

3.2.3 Availability . . . 51
3.2.4 Performance/Performability . . . 52
3.2.4.1 Virtualization Overheads . . . 52
3.2.4.2 Influence of “Bare” Virtualization . . . 53
3.2.4.3 Influence of “Applied” Virtualization . . . 57
3.3 Summary . . . 58

4 Creating a Network Function Virtualized AMI 59
4.1 General Idea . . . 60
4.2 Hardware Abstraction and Centralization . . . 61
4.2.1 Hardware Requirements and Abstraction . . . 61
4.2.2 Relocation of Hardware . . . 63
4.2.2.1 Dependability . . . 64
4.2.2.2 Performance . . . 64
4.2.2.3 Comparison and Server Location Distribution Selection . . . 66
4.3 Softwarized Service Generation and Location . . . 66
4.3.1 Softwarization of AMI Services . . . 66
4.3.2 Virtual Network Function Component Composition . . . 67
4.3.3 Embedding of Virtual Network Function Forwarding Graphs . . . 69
4.4 Introducing Performability-Enhancing Methods . . . 74
4.4.1 Substrate Enhancements . . . 74
4.4.1.1 Circle Backup Network Layout . . . 74
4.4.1.2 All-for-All Backup Network Layout . . . 76
4.4.2 Software Enhancements . . . 76
4.4.2.1 Software Rejuvenation . . . 76
4.4.2.2 Service Replication . . . 79
4.5 Summary . . . 80

5 Modeling a Virtualized AMI 81
5.1 Current AMI Models and their Shortcomings . . . 82
5.2 Model Requirements . . . 83
5.3 Development of an AMI model . . . 86
5.3.1 Failure Behavior . . . 86
5.3.1.1 Root Cause Breakdown . . . 86
5.3.1.2 Hardware and Software Failures . . . 87
5.3.1.3 Failure Modes . . . 90
5.3.2 Recovery Behavior . . . 92
5.3.2.1 Possibility . . . 92
5.3.2.2 Quality . . . 92
5.3.2.3 Structure . . . 93
5.3.3 Macro Model (Inter-System) . . . 94
5.3.3.1 VNFA Macro Model . . . 94
5.3.3.2 SSMA Macro Model . . . 98
5.3.4 Micro Model (Intra-System) . . . 98
5.3.4.1 Nomenclature . . . 99
5.3.4.2 Assumptions . . . 99
5.3.4.3 NFV Failure Behavior . . . 100
5.3.4.4 Recovery . . . 102

5.3.4.5 Modeling Performability-Enhancing Methods . . . 103
5.3.4.6 Micro Model Generation . . . 107
5.4 Overall Model . . . 110
5.4.1 Idea and Prerequisites . . . 110
5.4.2 Generation of Overall Queuing Network Model . . . 112
5.4.2.1 General Properties . . . 112
5.4.2.2 Converting Non-Virtualized Networks . . . 112
5.4.2.3 Converting Virtualized Networks . . . 113
5.5 Summary . . . 116

6 Analysis **117**
6.1 Description of Assumptions and Scenarios . . . 118
6.1.1 Assumptions . . . 118
6.1.1.1 Failure and Recovery Properties of Entities . . . 118
6.1.1.2 Communication Networks Properties . . . 119
6.1.1.3 Location Distributions . . . 120
6.1.1.4 Further Assumptions . . . 121
6.1.2 Single Service Scenario . . . 122
6.1.3 Single User Scenario . . . 122
6.1.4 Passau City Scenario . . . 123
6.2 Performability Assessment . . . 124
6.2.1 Evaluation: Single Service Scenario . . . 125
6.2.1.1 Test Environment . . . 125
6.2.1.2 Single Service Scenario: Evaluation of SSMA Model . . . 127
6.2.1.3 Single Service Scenario: Evaluation of VNFA Model . . . 130
6.2.2 Evaluation: Single User Scenario . . . 136
6.2.2.1 Single User Scenario: Test Environment . . . 136
6.2.2.2 Single User Scenario: Evaluation of SSMA Model . . . 137
6.2.2.3 Single User Scenario: Evaluation of VNFA Model . . . 139
6.2.3 Evaluation: Passau City Scenario . . . 142
6.2.3.1 Passau City Scenario: Test Environment . . . 142
6.2.3.2 Passau City Scenario: *AMIgo* Performance Evaluation . . . 143
6.2.3.3 Simulation Parameters . . . 145
6.2.3.4 Passau City Scenario: Evaluation of SSMA Model . . . 147
6.2.3.5 Passau City Scenario: Evaluation of VNFA Model . . . 150
6.2.4 Result Comparison . . . 154
6.3 Proof-of-Concept: Virtualized Smart Meter Gateway . . . 156
6.3.1 Gateway Task Description . . . 157
6.3.2 Proof-of-Concept Implementation . . . 158
6.3.3 vSMGW Performability Analysis . . . 158
6.3.3.1 Test Environment . . . 158
6.3.3.2 Performance Evaluation . . . 159
6.3.3.3 Dependability Evaluation . . . 161
6.3.4 Results . . . 164
6.4 Cost Analysis . . . 165
6.4.1 Deriving a Cost Function . . . 165
6.4.1.1 Example Scenario . . . 167

6.4.2 Evaluation and Results . . . 167
6.5 Summary . . . 169

7 Applicability, Conclusions and Future Work **171**
7.1 Practical Applicability . . . 172
7.1.1 Usability Challenges . . . 172
7.1.1.1 AMI Hardware Architecture . . . 172
7.1.1.2 IT Security Considerations . . . 173
7.1.2 Regulatory Challenges . . . 175
7.1.2.1 Specific, Realization-Centered AMI Regulations in Germany . . . 175
7.1.2.2 Solution Approaches . . . 176
7.2 Conclusion and Outlook . . . 177
7.3 Future Work . . . 179

A Appendix **181**
A.1 *AMIgo* Configuration File . . . 181
A.2 Virtualized Smart Meter Gateway Evaluation Data . . . 185

Bibliography **187**

List of Acronyms

ALEVIN	Algorithms for Embedding of Virtual Networks	142
AMI	Advanced Metering Infrastructure	iii
AMR	Automated Meter Reading	23
AM	Aging Related Mandelbug	78
ANSI	American National Standards Institute	178
BPL	Broadband over Power Lines	123
BSI	Bundesamt für Sicherheit in der Informationstechnik	24
BMWi	Bundesministerium für Wirtschaft und Energie	175
CI	Confidence Interval	144
CLS	Controllable Local Systems	25
COSEM	Companian Specification for Energy Metering	178
COTS	Commercial-Off-The-Shelf	iii
CPU	Central Processing Unit	52
CSPL	C based SPN Language	125
CTMC	Continuous Time Markov Chain	34
DCE	Data Communication Equipment	30
DES	Discrete Event Simulation	9
DKE	Deutsche Kommission Elektrotechnik Elektronik Informationstechnik	179
DoS	Denial of Service	174
DSL	Digital Subscriber Line	25
DSPN	Deterministic and Stochastic Petri Net	85
DSRN	Deterministic and Stochastic Reward Net	4
DTE	Data Terminal Equipment	30
EIKE	European Institute for Climate and Energy	21
ETSI	European Telecommunications Standards Institute	26
EU	European Union	173
FERC	Federal Energy Regulatory Commission	22
FIFO	First In – First Out	121
GPRS	General Packet Radio Service	27
GSPN	Generalized Stochastic Petri Net	86
HAN	Home Area Network	24
HDD	Hard Disk Drive	100
HES	Head End System	24
I/O	Input/Output	52
ICT	Information and Communications Technology	iii
IEC	International Electrotechnical Commission	13
IHD	In-Home Display	61
IRET	Interrupt Return	53

IRQ	Interrupt Request	53
JMT	Java Modelling Tools	136
KKT	Karush-Kuhn-Tucker	128
LMN	Local Meteorological Network	24
LTE	Long Term Evolution	123
MDMS	Meter Data Management System	24
MIS	Markov Imbeddable Structure	40
MMU	Memory Management Unit	53
MRGP	Markov Regenerative Process	107
MRM	Markov Reward Model	20
MSE	Mean Squared Error	163
MTTF	Mean Time To Failure	32
MTTR	Mean Time To Repair	41
NAM	Non-Aging Related Mandelbug	101
NAN	Neighborhood Area Network	24
NETL	National Energy Technology Laboratory	167
NFV-MANO	Network Functions Virtualization Management/Orchestration	30
NFVI	Network Functions Virtualization Infrastructure	30
NFVO	Network Functions Virtualization Orchestrator	31
NFV	Network Functions Virtualization	iii
NHPP	Non-Homogeneous Poisson Process	162
NIC	Network Interface Card	53
NIST	National Institute of Standards and Technology	22
OSGP	Open Smart Grid Protocol	26
OS	Operating System	29
PC	Personal Computer	61
PLC	Power Line Communication	25
PNF	Physical Network Function	30
PV	Photovoltaic	178
QoS	Quality of Service	iii
RAID	Redundant Array of Independent Disks	100
RAM	Random Access Memory	53
RBD	Reliability Block Diagram	16
RTP	Real-Time Pricing	27
RTT	Round Trip Time	160
SCP	Secure Copy	160
SDN	Software-Defined Networking	34
SGAM	Smart Grid Architecture Model	82
SMGW	Smart Meter Gateway	23
SPNP	Stochastic Petri Net Package	125
SPN	Stochastic Petri Net	20
SRN	Stochastic Reward Net	20
SSMA	Standard Smart Metering Architecture	9
TLS	Transport Layer Security	25
TPM	Trusted Platform Module	62
TPS	Third Party Service	24
UMTS	Universal Mobile Telecommunications System	123

vCPU	Virtual Central Processing Unit	53
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik	179
VEIN	Virtualized Energy Information Network	35
VIM	Virtualized Infrastructure Manager	31
VL	Virtual Link	30
VMM	Virtual Machine Manager	4
VM	Virtual Machine	4
VNEP	Virtual Network Embedding Problem	69
VNE	Virtual Network Embedding	142
VNF-FG	Virtualized Network Function Forwarding Graph	67
VNFA	Virtualized Network Function Architecture	5
VNFC	Virtualized Network Function Component	4
VNFM	Virtualized Network Function Manager	31
VNFR	Virtualized Network Function Request	67
VNF	Virtualized Network Function	4
vNIC	Virtual Network Interface Card	53
VN	Virtual Network	30
VPN	Virtual Private Network	30
vSMGW	Virtual Smart Metering Gateway	9
vTPM	Virtual Trusted Platform Module	176
WAM	Wide Area Monitoring	27
WAN	Wide Area Network	24
WiMAX	Worldwide Interoperability for Microwave Access	33
XML	eXtensible Markup Language	142

List of Figures

1.1 Energy demand and world population: history and projections 2
1.2 Chapter structure, dependencies and contributions 8

2.1 Overview of thesis' research areas and their interrelation 12
2.2 Dependability tree 14
2.3 Overview of AMI architecture, networks, and technologies 28
2.4 Scheme of the VNF architecture, including VNFs, VNFI, and NFV-MANO 31
2.5 Localization of related work in the research domains covered by this thesis 36

3.1 Standard Smart Metering Architecture scheme 40
3.2 Reliability and availability of SSMA 41
3.3 RBD models of non-virtualized and virtualized scenarios 42
3.4 Number of VMs required to surpass non-virtualized system reliability 44
3.5 Maintenance time relations 45
3.6 Influence of λ_v^{vmm} and $M_v^{ct_{vmm}}$ on $\overline{M}_v^{ct}$ compared to $\overline{M}_{nv}^{ct}$ 47
3.7 Influence of virtual machine and service failure rate on $\overline{M}_v^{ct}$ compared to $\overline{M}_{nv}^{ct}$ 48
3.8 Comparison of virtualized and non-virtualized systems' maintainability 49
3.9 Comparison of CPU performance in substrate and virtualized environment 55
3.10 Comparison of RAM performance in substrate and virtualized environment 55
3.11 Comparison of substrate and virtual I/O read/write performance 56
3.12 Comparison of network performance in substrate and virtualized environment 57

4.1 Rough scheme of VNFA 60
4.2 VNFR featuring required and optional services and corresponding VNF-FG 68
4.3 Backup strategy in circle backup networks 74
4.4 Backup strategy in all-for-all backup networks 76

5.1 Classification of currently used AMI models 83
5.2 Quantitative evaluation approaches 85
5.3 Root cause breakdown chart 87
5.4 Development of hardware failure rate over time, based on IEC 61508 [66] 88
5.5 Development of software failure rate over time 89
5.6 Failure detection based on function ϕ and specification ρ 91
5.7 Failure detection based on non-functional requirements 92
5.8 Service, VM, VMM, and substrate plane of the VNFA macro model 97
5.9 Service, and substrate plane of the SSMA macro model 98
5.10 Modeling on substrate plane 100
5.11 Modeling on virtualization plane 101
5.12 Modeling on service plane 102
5.13 Deterministic and dynamic rejuvenation models 104

5.14 Cold (deterministic) service rejuvenation models . . . 105
5.15 Warm (deterministic) service rejuvenation models . . . 106
5.16 Migration (deterministic) service rejuvenation models . . . 107
5.17 Merged VNFA performability stochastic reward net model . . . 108
5.18 SSMA performability stochastic reward net model (cold, deterministic rejuvenation) . . . 110
5.19 Overall VNFA AMI model . . . 111
5.20 Challenges during VNFA AMI conversion . . . 114

6.1 Influence of δ^s_{nv} in deterministic, cold rejuvenation SSMA . . . 127
6.2 Influence of ∇^s_{nv} in dynamic, cold rejuvenation SSMA . . . 130
6.3 Failure sensitivity analysis of SSMA's performance and availability . . . 130
6.4 Failure sensitivity analysis of VNFA's performance and availability . . . 131
6.5 Influence of rejuvenation parameters on performability in deterministic, cold VNFA . . . 132
6.6 Influence of rejuvenation parameters on performability in dynamic, cold VNFA . . . 133
6.7 ΔA of warm and migration compared to cold rejuvenation in deterministic VNFA . . . 134
6.8 $\Delta Perf$ of warm and migration compared to cold rejuvenation in deterministic VNFA . . . 134
6.9 ΔA of warm and migration compared to cold rejuvenation in dynamic VNFA . . . 135
6.10 $\Delta Perf$ of warm and migration compared to cold rejuvenation in dynamic VNFA . . . 135
6.11 Comparison of deterministic and dynamic rejuvenation timing in SSMA . . . 137
6.12 *AMIgo* program flow . . . 143
6.13 *AMIgo* runtime distributions . . . 144
6.14 *AMIgo* embedding time requirements . . . 144
6.15 Investigated AMI topologies . . . 145
6.16 Influence of sm, r, and gw failures on SSMA dependability . . . 147
6.17 Influence of dc failures on SSMA dependability . . . 148
6.18 Influence of tps_{cm} failures on SSMA dependability . . . 148
6.19 Influence of tps_{eo} failures on SSMA dependability . . . 149
6.20 No redundancy performability test . . . 151
6.21 Circle backup performability test . . . 151
6.22 All-for-all backup performability test . . . 152
6.23 Example of a minimal gateway design . . . 157
6.24 vSMGW test environment . . . 159
6.25 Results of the startup time test (sequential starting) . . . 160
6.26 Results of the startup time test (parallel starting) . . . 160
6.27 Results of the throughput test . . . 161
6.28 Results of the latency test . . . 162
6.29 MSE-based prediction of Musa-Okumoto Θ and λ_0 parameters . . . 163
6.30 Development of vSMGW failure rate and reliability . . . 163
6.31 Availability of vSMGWs in relation to expected MTTR . . . 164
6.32 Cost development of SSMA compared to VNFA . . . 167
6.33 VNFA $\Delta\mathcal{C}$ using replica(s) for dc . . . 168
6.34 $\Delta\mathcal{C}$ in VNFA using replica(s) for dc and gw . . . 169

7.1 Comparison of encryption behavior in SSMA and VNFA . . . 174

List of Tables

2.1 Mutual influences between reliability, maintainability and availability 15
2.3 Properties of AMI communication technologies 26
2.5 AMI communication requirements 28

3.1 Inherent availability of non-virtualized and virtualized systems 52

4.2 Virtualization possibilities of AMI services 62

5.2 Requirements of the novel AMI model 84
5.3 Micro model nomenclature 99
5.5 Variables used for calculating the link probabilities 115

6.1 Failure and repair rate information stated in related work 119
6.3 Single service scenario VNFA and SSMA parameters 123
6.5 Single user scenario VNFA parameters 124
6.7 Single user scenario SSMA parameters 125
6.9 Passau city scenario VNFA and SSMA parameters 126
6.10 $\overline{Perf}_i$ and $\overline{A}_i$ calculation 129
6.11 Results of S_{il}^2 calculations 129
6.12 Influence of replicas on service availability in VNFA 135
6.13 VNFA availability *[%]* in single user scenario using 0, 1, and 2 replicas for (either dc or tps) 141
6.14 VNFA availability *[%]* in single user scenario using 0, 1, and 2 replicas each (gw and either dc or tps) 141
6.15 Offered and demanded CPU resources of nodes in *AMIgo* 146
6.16 Service times and rates of queuing stations 147
6.18 Results of *AMIgo* simulation for SSMA (mean values) 149
6.20 Results of JMT calculations for VNFA (mean values) 154

A.1 Input and result data for vSMGW test ($\Theta = 0.58$, $\lambda_0 = 0.2$) 186

1 Introduction

Electricity is needed in nearly every part of people's lives [121]. From heating and air conditioning, over transportation, to powering electrical devices, electricity is omnipresent. Especially a modern society, heavily relying on powered devices, such as computers and the Internet, to fulfill even its most basic needs, cannot be imagined without a dependable power supply anymore. To overcome the power grid's currently present challenges, an updated form of utility infrastructure, namely the smart grid, is suggested as a solution to most issues at hand [100]. However, as smart grids greatly rely on the information relayed through the newly created networking technology on top of the existing utility infrastructure, known as the AMI, its flawless operation is built upon dependable and timely data originating from this complex network [36]. Because of this development, the AMI's—and in turn the smart grid's—vulnerability against failures on hard- and software level also rises, which can lead to major disturbances [51]. In this thesis, a solution to this challenge is presented. In Section 1.1, an overview of the current status of the power grid, its properties and development are reviewed. Thereafter, in Section 1.2, the scientific challenges are deduced and the approaches taken to solve them are explained. Section 1.3 delineates the contributions of this thesis to the research area. Finally, Section 1.4 gives a structural overview of the remaining parts of this thesis.

Contents

1.1 Dependable Smart Grids 2
1.2 Challenges and Solution Approach 3
1.3 Contributions 5
1.4 Thesis Structure 7

1.1 Dependable Smart Grids

The smart grid (as in Definition 2.6 in Section 2.1.2.1) is a current development to counter the failing modernizations that have accumulated in almost every country's power grid systems; as a matter of fact, the vertical structure (production solely at power plant levels, demand mostly at household, business, and industry level) of the grid and its operation methods have basically remained the same since the time of their creation. For example, the core infrastructure of the US power grid was created over a century ago[1], while the European system has already about 50 years of age [11]. Along with missing innovations, the power grid faces severe challenges in recent years, such as a growing tension between reliability and cost, an increasing strain of the old infrastructure, missing surveillance and situational awareness in wide parts of the grid, and long response times in case of disruptions. Apart from that, the overall requirement for dependable power delivery has grown and will continue to grow continuously, as illustrated in Figure 1.1 [67].

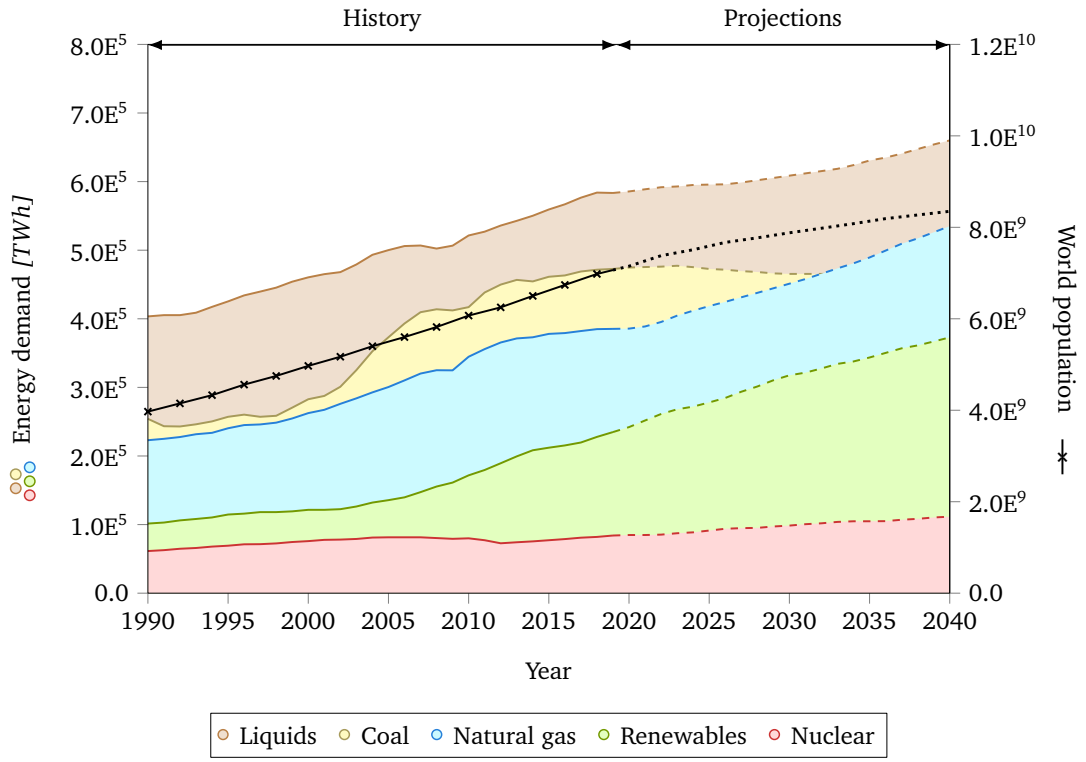


Figure 1.1.: Energy demand and world population: history and projections up till 2040 based on data of the International Energy Agency [67]

To overcome these issues, the smart grid, which combines two network types into a *network of networks*, namely communication and power infrastructure, was found as a solution. The resulting grid offers several benefits, such as automatic monitoring and reporting functions, more active inclusion of consumers, usage of storage capacities (e. g. by electric car accumulators), and more decentralized generation, among others. To be able to offer these improvements, it becomes apparent that there needs to be a massive increase in information exchange, especially between service providers and their consumers, as the currently still used ferrite phase meters only allow to measure the summarized energy demand over at least monthly billing periods. In order to enable such a

[1] American Electric Power, History of AEP, URL: `http://www.aep.com/about/history`, last accessed: 08/05/2015

growing information exchange, it is important to have a dependable infrastructure that enables two-way communication, which meets the necessary requirements to facilitate an exchange of real-time information between the parties mentioned. In smart grids, it is planned to realize this by the AMI (as in Definition 2.7 in Section 2.1.2.2), an infrastructure that automatically sends fine-granular demand and billing data from consumers to service providers, as well as price information and signaling messages to consumers [106].

A more detailed description of the issues present in the power grid and the features smart grids are going to provide are given in Section 2.1.2.1. While AMIs promise to solve the challenges at hand in the current power grid, the inherent complexity of the smart grid originates from exactly the aforementioned combination of ICT and power infrastructure AMIs are based on. This, in turn, leads to severe challenges in the areas of dependability and financial feasibility in AMIs. While there have already been approaches to increase the dependability and performance (forming the concept of performability, being the topic of Section 2.1.1) of AMIs and smart grid communication infrastructures in general, which are discussed in Sections 2.1.2.2 and 2.2.1, solutions for AMI improvements were generally focused on the provider's side of the network. This leaves room for significant improvements, especially on the consumer's side and third party services, which benefit the whole AMI and its stakeholders.

1.2 Challenges and Solution Approach

Section 1.1 introduced the current state of the power supply infrastructures in modern societies and gave an outlook towards the future. Also, implications of the development and implementation of smart grid technologies into the power grid were discussed. While smart grids without a doubt have numerous advantages, e. g. decreased reaction time to incidents and better predictability of power demand/production, the downside of this development is an inevitably increasing complexity of the infrastructure, which leads to a rising number of possible failures in ICT that can also impact the performance of AMIs and the smart grid as a whole [44]. To give an example, missing alarm signals or the inability to relay information due to component malfunctions might cause cascading failures throughout the system. This in turn can result in an instability of the grid and might provoke large-scale outages.

Such challenges are only further fostered by the current trend to focus strategies to strengthen dependability at the energy provider side, while other parties, such as consumers and third party service providers, do not receive the required attention and are mainly discussed in the context of security and privacy questions. However, while these questions have been discussed and mostly already been solved properly, the lack to consider system-wide dependability, including consumers and third parties, leads to several issues that potentially have a severe impact on the overall operation of the smart grid. To understand the far-reaching implications of lacking service dependability, it is important to see that nearly all of the smart grid's goals are based and rely on the underlying ICT infrastructure. However, while a dependable ICT infrastructure may be a key enabler to provide beneficial new services to consumers as well as service providers, an apparent lack of this property might lead to the exactly opposite behaviors.

One ostensibly obvious solution to these problems at hand is the introduction of multiple redundant devices and network links, which has been suggested, e. g. by Niyato *et al.* [114] and Shi *et al.* [146]. While the details of these approaches are discussed further in Section 2.2.1, the main findings are that though the introduction of redundant hardware may improve the reliability and availability

of AMIs, the downsides of such an approach are also numerous. The system suffers from even more complexity apart from the additional monetary overhead that is inherently present in a highly redundant infrastructure. This is neither favorable nor feasible, as not only the direct hardware costs increase massively, but also maintenance, repair and operational costs are rising.

To handle the issue of increasing complexity and the likelihood of failures within the infrastructure without crippling its cost-effectiveness, a novel approach is needed, which can simultaneously offer highly dependable services to satisfy the requirements posed by the communication infrastructure of the smart grid and consolidate the needed functions on as few devices as possible. This way, the necessary financial expenses as well as the hardware complexity of the approach could be decreased to a feasible level.

To achieve the required dependability, as well as match the various performance requirements stated for smart grid communication networks (cf. Table 2.1.2.2), the components comprising this infrastructure currently realized as hardware entities are transferred into Virtual Machines (VMs) and combined into softwarized Virtualized Network Functions (VNFs). While such a solution can offer great benefits, its concrete realization poses numerous questions itself. Hence, to extend and improve previous solutions in this area, which are discussed in detail in Sections 2.2.1 and 2.2.2, the challenges posed and solution approaches pursued in this thesis are listed in the following.

1. The first part deals with the general **applicability of virtualization** to improve the dependability characteristics of ICT equipment in the context of AMIs. The question here is if virtualization is an appropriate technology to increase the dependability of critical infrastructures such as AMIs; this also includes the absence of any insuperable obstacles (e. g. in the form of requirements strongly contradicting the usage of virtualization). Such requirements can originate from both the reliability/availability part of dependability as well as performance requirements.

2. Second, the **virtualization of AMI services**, including the core of the infrastructure and third party services needs further investigation. This includes the question *which* services can be virtualized (including possible requirements originating from either the cyber-physical interaction of these service or their location demands), and the possible benefits/downsides coming along with such a virtualization.

3. In the third part, the **integration of virtualized AMI services** into a given smart grid infrastructure is investigated. It needs to be questioned if and how VNFs can be integrated into the current or an extended form of the smart grid infrastructure. To fully benefit from the novel AMI approach based on NFV, it is required to analyze how services need to be structured (meaning the sequence and interaction of Virtualized Network Function Components (VNFCs) realizing VNFs) and distributed (meaning the distribution of VNFs onto their substrate) within the infrastructure to achieve the desired improved properties.

4. Fourth, the ideas investigated in the previous third part are used to **establish a novel, hierarchical AMI model**, which is able to represent NFV AMIs, their service and cyber-physical demands, virtualization infrastructure (services, VMs, and Virtual Machine Managers (VMMs)), as well as substrate and logical network links. In addition, it is required that the model supports the calculation of various performability metrics of a dynamically changing infrastructure. To do so, a combined performability model of virtualized smart grid communication infrastructures based on Deterministic and Stochastic Reward Nets (DSRNs) is developed. This model is on the one hand used to compare the novel designs to currently suggested infrastructures and evaluate them, and on the other to quantitatively assess them independently from each other.

5. The fifth part uses three different scenarios to **evaluate and compare the models** of the currently suggested AMI to the developed virtualized AMI. The analyses investigate both the models' dependabilities as well as their performances, to ensure their suitability for the required tasks. Also, the evaluation of the scenarios is used to optimize various parameters of the virtualized infrastructure, such as rejuvenation settings and substrate backup strategies. As a result, it is possible to identify the sensitivity of both architecture approaches towards rising or falling failure/impairment rates, the influences of rejuvenation timings and strategies as well as the impact of substrate and service backup choices.

6. Sixth, the **applicability of the novel NFV AMI approach** is investigated, including the usability challenges, such as necessary updates to the hardware architecture and potential security implications of such a change. Also, the costs of both the traditional AMI approach and the NFV AMI are compared in a cost analysis. Last, the regulatory challenges that may intervene with the realization of a virtualized AMI are analyzed and possible solutions are suggested.

After the challenges as well as potential solution approaches were introduced, the concrete scientific contributions of this thesis are further elaborated in the upcoming Section 1.3.

1.3 Contributions

In this thesis, a novel solution to the current challenges of smart grid services is proposed focusing on AMIs as a use case. The solutions are based on current research results, mainly from the areas of network and service virtualization as well as dependability models. Also, novel solutions are proposed to counter the rising complexity of utility services and communication networks, which leads to increased dependability issues that in turn threaten the service availability leading to a decreased security of supply for end users. In the following, the contributions of this thesis are briefly listed.

c_1) **Virtualization performability analysis:** In this thesis, the influence of virtualization on the achievable performability of a service is compared to a substrate-based solution. The results are used to evaluate if virtualization benefits performability and if so, to derive a virtualization approach suited for AMIs.

c_2) **Virtualized AMI services:** The developed Virtualized Network Function Architecture (VNFA) reduces the introduction of new hardware components into the smart grid, which in turn leads to a less complex infrastructure. As a result, a main source of failures can be significantly brought down, offering increased service performability. In contrast to approaches requiring additional, specialized hardware, VNFA instead uses chained VNFCs emulating the functions of their respective hardware counterparts.

c_3) **Multi-level performability:** While current AMIs rely solely on the durability of components arranged in serial way trying to achieve performability, VNFA uses a multi-level approach realizing an architecture offering performability in multiple dimensions.

- In contrast to the currently suggested AMI, VNFA offers substrate-sided redundancy to be able to mitigate hardware failures without significant monetary drawbacks. This is enabled by a widely hardware-agnostic virtualized realization of AMI functions hosted on COTS servers, which backup one another.

- Virtualization is used to discard most specialized AMI hardware components, significantly reducing the overall hardware count. This also leads to significantly lowered restoration times, as software can be restored much faster than hardware components providing similar services.
- To further increase the AMI performability, VNFA offers a concept of dynamic service distribution and migration of services. This enables a dynamic failure mitigation in two different ways, depending on the failure type present.
  - In case of software failures, in most occurrences, a service restart can restore its functionalities, thereby offering increased service availability. Further details regarding the failure behavior and restoration of software are given in Section 5.3.1.
  - In case of a hardware failure, the affected softwarized services can be migrated onto a nearby, healthy substrate to be able to offer continued availability while the primary hosting site is reestablished. This is further elaborated in Section 4.4.1.

c_4) **Classification of AMI models:** To provide an overview of currently used AMI models as well as their respective advantages and downsides, the currently used models to represent AMIs in literature are identified and classified. This serves as a prerequisite for the subsequent development of a novel AMI model enabling the evaluation of virtualized AMIs.

c_5) **Novel AMI and service model:** In this thesis, the research fields of smart grids are combined with virtualization approaches leading to virtualized smart grid AMIs. To represent and analyze a virtualized AMI, a novel model has to be established, which is capable to handle not only single virtual services, but whole AMIs comprising vast amounts of interacting services, their VMs, as well as the hosting substrate and their networking infrastructure.

c_6) **Scalable architecture:** In comparison to current AMI approaches relying on numerous specialized hardware components, the usage of softwarized services offers not only a less costly realization of AMI functions, but is also faster and easier to develop, set up and distribute. This enables both a faster integration of new users into an existing AMI as well as rapidly deployable upgrades to either satisfy the substrate demands of future services or to adapt software features to new requirements of stakeholders.

c_7) **Cost-effective realization:** Current AMI architectures are complex infrastructures realized as a combination of power and communication networks. To enable an infrastructure spanning from consumers to power distributors, a large variety and number of hardware components are used. This leads to increased development, maintenance and operational costs inside the smart grid. VNFA avoids the usage of most currently required hardware by relying on softwarized entities emulating the functions of their respective hardware counterparts. Apart from the previously described benefits, there are also monetary advantages:

- The hardware costs can be significantly reduced, as software replacing these hardware components do not need to be produced on a per consumer basis, but rather only once.
- Even in the case of potential incompatibilities, software can be adapted to new requirements much easier than hardware.
- By simultaneously reducing the number of required hardware devices and increasing the utilization of the hosting substrate, the energy demand can be reduced [93].

c_8) **Adaptable, future-proof architecture:** The softwarization of AMI services enables an adaptable service provision that can be adjusted to fit various requirements, e. g. regulatory obligations in other countries, functional service updates or security enhancements. Also, employing virtualized services is a significant step towards a future-proof AMI, as substrate and virtual services are independent, offering the benefit of easy to realize upgrades of both hard- and software.

1.4 Thesis Structure

Chapter 2 consists of two parts. First, it discusses background information related to the thesis' topics, namely performability, the power grid and virtualization. Regarding performability, first, its more well-known parts dependability and performance are defined, before performability is introduced. For the power grid, the current situation and arising challenges are identified. After that, the evolution from the momentarily used power grid to an emerging smart grid is explained, stressing the AMI, its goals, benefits, as well as its challenges. Next, some background information concerning virtualization technologies is given, including system, network and network function virtualization. Second, the related work that features approaches similar to the one in this thesis are reviewed. Here, different publications that use dependability/performability in a smart grid context or virtualization to increase dependability/performability are discussed first. To be precise, several approaches that either investigate AMI reliability or availability issues, use virtualization to increase dependability, and/or discuss the usage of NFV are reviewed. At last, the previous work of the author as well as current concepts and approaches that combine the aforementioned research areas directly related to the suggested solution in this thesis are discussed and classified.

In Chapter 3, it is evaluated whether virtualization offers a suitable approach to enhance the performability of computer systems and AMIs in particular. To do so, first, the current AMI and its architecture are briefly assessed and a preliminary analysis is performed. After that, the performability impact of virtualization is estimated. This includes its impact on reliability, maintainability, availability, and performance. To answer the research question of this chapter, two different scenarios for virtualization are evaluated: First, the possible overheads induced by virtualization itself, i. e. "bare" virtualization, are discussed, highlighting its negative consequences. Second, "applied" virtualization, which also estimates the beneficial influences of virtualization, such as increased availability through redundant service provision and load balancing, is evaluated.

Chapter 4 focuses on the development of VNFA. Before the new infrastructure is explained in detail, its general ideas and foundation are presented. Afterwards, these general concepts are expanded, including the hardware abstraction and relocation, the service virtualization, function chain generation and subsequent embedding. After that, several possible performability-enhancing methods on both substrate and software layer are introduced: On substrate layer, redundancy mechanisms are employed to ensure that occasional hardware failures do not lead to severe performability deterioration; on software layer, VM backups are used to guarantee that important services do not suffer from long-lasting outages.

The modeling of the developed VNFA approach is detailed in Chapter 5. Before the novel model is designed, the currently used AMI models are classified and their shortcomings are identified. After that, the novel model's requirements are deducted based on the necessities inherited from the virtualization employed and the metrics to be calculated. This results in a model that is hierarchically structured into three tiers: micro, macro and overall model. Before these three models are extensively

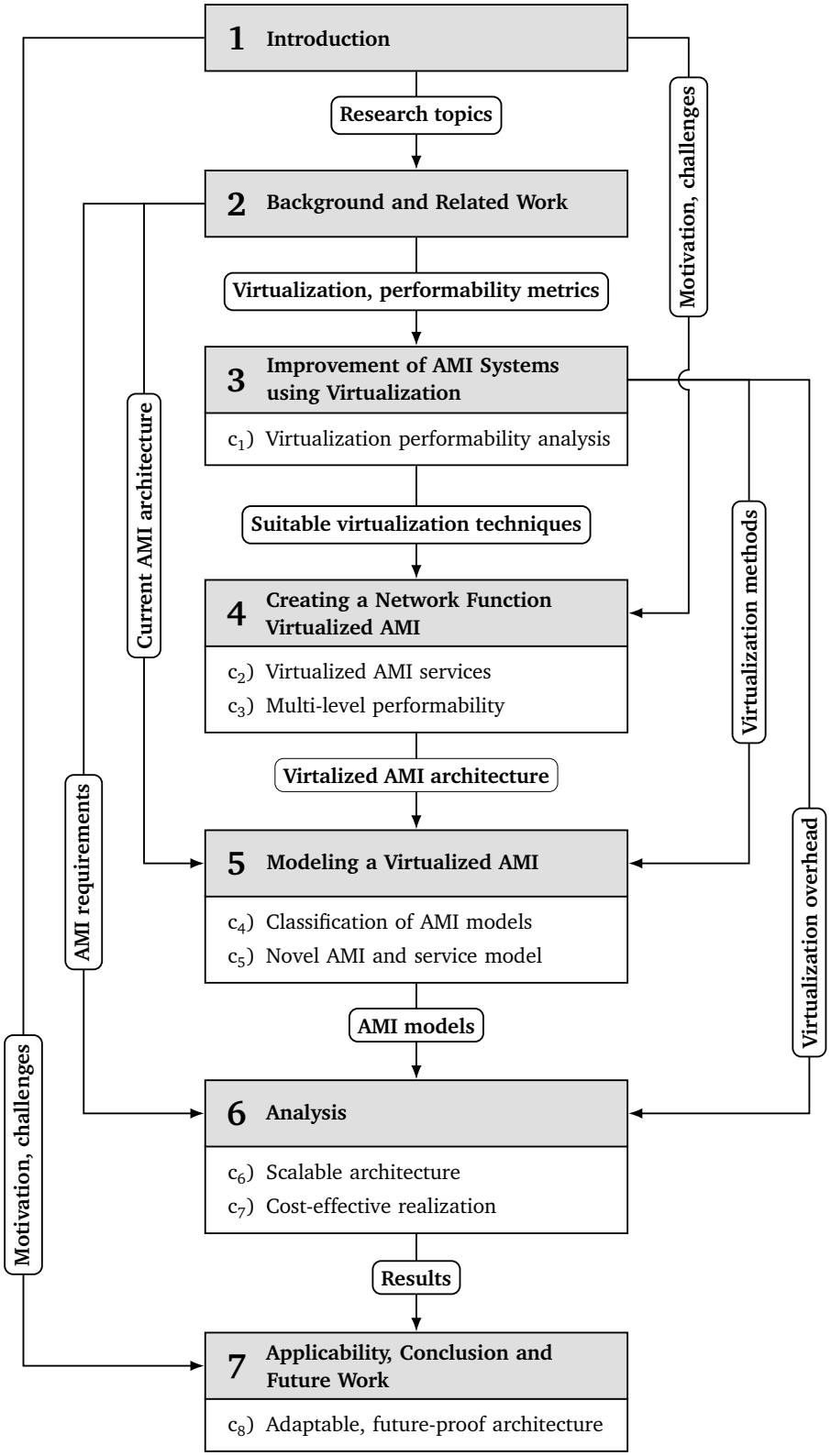


Figure 1.2.: Chapter structure, dependencies and contributions

discussed, the failure and restoration behaviors of the AMI architectures are covered. After the model generation, the properties of the developed infrastructure are compared to the current AMI design during a qualitative analysis.

Chapter 6 evaluates the previously developed AMI architecture based on NFV in multiple dimensions using different methodologies. First, a dependability analysis is performed using network graphs and DSRNs. Second, the computational and network performance of the developed architectures is evaluated using Discrete Event Simulation (DES) on queuing networks and the previously developed DSRNs. Also, the costs of VNFA are compared to Standard Smart Metering Architecture (SSMA) the investigate possible financial benefits. Next, a prototype implementation of a Virtual Smart Metering Gateway (vSMGW) service is assessed regarding its dependability and performance. At last, the results of all analytical analysis are analyzed and conclusions are drawn.

In Chapter 7 possible limitations of the approach are evaluated and the practical applicability of the proposed VNFA is discussed further. In detail, the currently suggested AMI architecture and the re-usability of its infrastructure for VNFA as well as possible security influences are investigated. Finally, a conclusion is given and possible future research directions and extensions of the approach are discussed.

The interplay of the aforementioned chapters is visualized in Figure 1.2, where the thesis' chapters structure is depicted along with the present dependencies and the contributions given in Section 1.3.

2 Background and Related Work

The research areas spanned by the challenges addressed in this thesis are diverse in nature. While the basic topic of dependability in critical infrastructures is already around for several decades, the combination with smart grid and virtualization adds recent technological innovations to the mix. In this chapter, both the background information and the state-of-the-art related to the three main topics of this thesis (performability, smart grids and virtualization), are presented.

First, the scientific backgrounds of dependability, performance and performability are presented along with a brief overview of current analysis methods in Section 2.1.1. After that, in Section 2.1.2, the history and development of power grids towards the currently emerging smart grid are elaborated. In this section, a major focus is the technical background of AMIs. Thereafter, the history and development of virtualization—including host and network virtualization techniques—are covered in Section 2.1.3. After that, NFV, which is later on used as a cornerstone of the novel AMI architecture, is discussed in detail. After the background of all three research areas of this thesis are covered in the previous sections, the related work is presented in Section 2.2. More specifically, recent approaches that combine two of the research fields, i. e. performability-enhancement in smart grids (Section 2.2.1) and virtualization approaches to increase performability (Section 2.2.2) are presented. In Section 2.2.3, both related work combining all three research areas and the author's own contributions to this research field are elaborated. Finally, in Section 2.3, a summary of the chapter is given.

Contents

2.1 Background **12**
2.1.1 Dependability, Performance and Performability 13
2.1.2 Smart Grid and Advanced Metering Infrastructure 21
2.1.3 Virtualization 29
2.2 Related Work **32**
2.2.1 Performability/Dependability in Smart Grids 32
2.2.2 Virtualization for Performability/Dependability 34
2.2.3 Combining Approaches & Own Contributing Work 35
2.3 Summary **36**

To put the three different major research areas of this thesis (smart grids, dependability/performance/performability and virtualization) into perspective, Figure 2.1 illustrates their interrelations and overlaps. The middle of the figure represents the thesis' research focus. Searching for the number

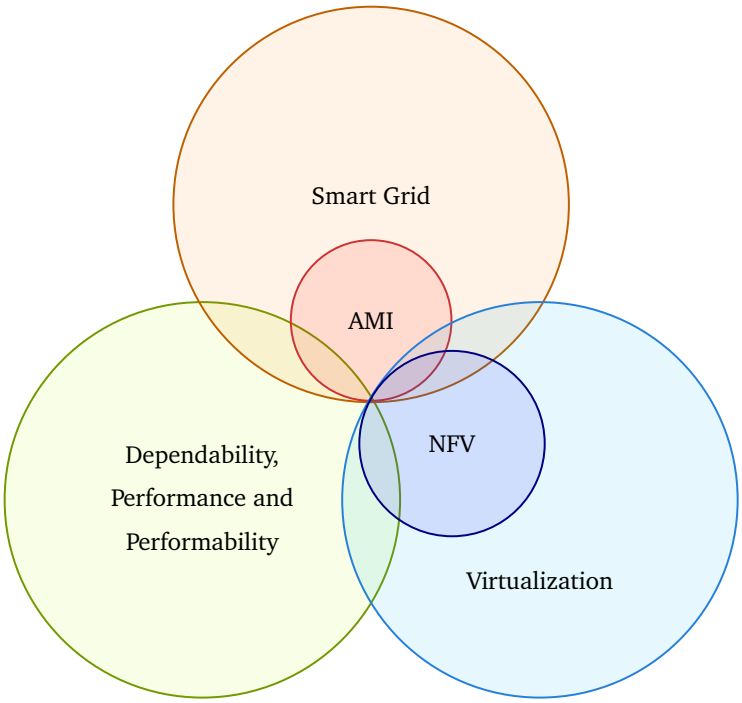


Figure 2.1.: Overview of thesis' research areas and their interrelation

of publications in each of the research topics using Google Scholar[2], it becomes apparent that the three basic topics covered in this thesis are broadly researched areas, while in contrast, especially the usage of NFV in an AMIs context is a relatively new topic featuring only a few publications. It needs to be noted however that the results of a Google Scholar search may only be used as a rough indicator, due to possible false negatives using slightly different keywords or false positives mentioning the research areas indicated by the keywords only as examples, instead of being the main research focus. The following Chapter 2 is first split in two main sections. First, Section 2.1 encompasses background information regarding the thesis' three research areas. After that, in Section 2.2, the related work relevant in the context of the aforementioned research areas is presented in Section 2.2.3.

2.1 Background

The backgrounds of this thesis' research topics are discussed in the following, to allow a deeper insight regarding the motivational and technical drivers behind the scientific approach taken to realize an AMI based on NFV. To limit the length of this section, the general background of each topic is only briefly described, while a stronger focus is placed on areas which are further utilized in the main parts of this thesis.

[2] Google Scholar, URL: `https://scholar.google.com/`, last accessed: 03/12/2019

2.1.1 Dependability, Performance and Performability

While already having some twenty years of age, performability is—in relation to both concepts it comprises, namely **dependability** and **performance**—a relatively recent concept that was developed due to the need to model systems experiencing degradable performance in the presence of failures. Before that, the general approach was to deal with both dependability and performance separately. Basically, this led to the idea that first, the dependability of a system was analyzed before optimizing its performance. While such systems perform well under fault-free conditions, as soon as impairments are present, their performance may face severe degradation. Performability is therefore typically defined as a composite measure of both dependability and performance of systems.

Examples of systems that face degraded performance are, e. g., distributed or redundant and fault-tolerant systems, such as complex utility infrastructures employing high levels of redundancy. Because such systems may be able to deliver proper service, at least up to a certain extent, even in the presence of failures, neither dependability nor performance analysis alone covers their behavior properly.

To be able to better grasp the concept of performability, however, the core aspects of dependability as well as performance analyses are briefly summarized in the next two paragraphs, before a definition of performability in the context of this thesis is given.

2.1.1.1 Dependability

Dependability is a major concern in nearly all types of systems. While the striving for high dependability of software and hardware entities that only fulfill functional requirements is basically due to the pursuit of high quality systems in general and lower costs due to maintenance in particular, especially in safety-relevant systems, such as critical infrastructures, high dependability levels are of deeper importance. In the following, the goals and metrics associated with dependability are discussed, before modeling approaches are explained. Finally, the formal analysis methods available for dependability are reviewed.

Definition. The term dependability was first defined by Laprie [79]. After that, several refinements of the definition were given, e. g., by Trivedi *et al.* [164], the International Electrotechnical Commission (IEC) [65] and Parhami [118]. A recent definition of the term is given by Avižienis *et al.* [9] and is cited in Definition 2.1.

Definition 2.1: Dependability

The dependability of a computer system is the ability to deliver a service that can justifiably be trusted. The service delivered by a system is its behavior as it is perceived by its user(s); a user is another system (physical, human) that interacts with the former at the service interfaces.

If thought about further, this leads to a number of different, but complementary properties as well as several metrics, depending on the application area of the system under investigation [78]. The attributes, means and threats to dependability are given by Avižienis & Laprie [7] and were subsequently extended [9]. They are depicted in Figure 2.2.

The **attributes** of dependability are availability, reliability, maintainability, safety, and security. In the context of this thesis, not all attributes of dependability are analyzed in a similar depth, however. As the main goal is to offer a service that is free of interruptions in a sense of offering mainly high

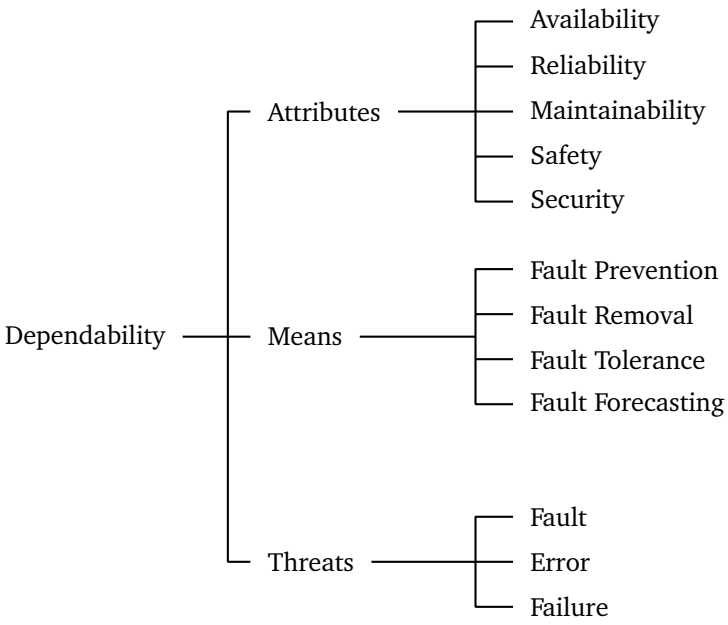


Figure 2.2.: Dependability tree according to Avižienis *et al.* [9]

availability and performance, the attributes of safety and security are not broadly discussed further. While both attributes are without a doubt of significant importance for a critical infrastructure, there are already standards available covering both aspects. For safety in the context of smart grids, the functional safety standard IEC 61508 is applicable. Similarly, security may be achieved by measures defined in, among others, ISO 2700x, IEC 62351, IEC 62443, and ASAP-SG Security Profiles [37]. Therefore, in the rest of this thesis, only the three attributes reliability, availability, and maintainability are considered to a larger extent, while safety and security are merely covered very briefly.

The **means** of achieving dependability include fault prevention, which is achieved by careful system design (i. e. structured, unambiguous requirements documentation and intersecting requirement views) to minimize the number of faults present [183], and fault removal, which consists of both finding and fixing bugs, during testing as well as operation [141]. Fault tolerance means that different techniques are applied that allow a system to continue its operation despite of component failures. Fault forecasting deals with the prediction of faults, errors and failures in components [59, 130].

The **threats** for dependability can be characterized as faults, errors and failures. A fault is the *adjudged or hypothesized cause of an error*. A fault is active when it produces an error, otherwise it is dormant. An error is present if there is *any discrepancy between a computed, observed, or measured quantity and the true, specified, or theoretically correct value or condition*. An error is that part of the system state that may cause a subsequent failure. A failure occurs when an error reaches the service interface and *renders the system unable to perform its required functions within specified performance requirements* [8].

Metrics. Next, the selected attributes of dependability, namely reliability, maintainability and availability are briefly introduced in a somewhat informal manner; a more elaborate discussion of the attributes follows in Chapter 3.

- **Reliability:** Reliability is the probability of a system or system component to perform a required function under stated conditions for a specified period of time. The reliability function $R(t)$ is mathematically defined as:

  $R(t) = P[X > t] = 1 - F(t)$,

  where $F(t)$ represents the unreliability, meaning the probability that a component has failed before time t. $F(t) = P[X \leq t]$, where $X \in \mathbb{R}_0^+$ is a random variable representing the time a component performs its required function under stated conditions.

- **Maintainability:** Maintainability is the ability of a system or system element to be retained in, or restored to, a specified condition when maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance and repair. Mathematically, maintainability is expressed as:

  $M(t) = P[T \leq t]$,

  where $T \in \mathbb{R}^+$ is a random variable representing the restoration time of a repairable component.

- **Availability:** Availability is the probability that a repairable system or system element is in operable state and can be used when it is requested at a random point in time. Depending on the definition of availability, it includes various downtimes, among others repair times, logistics, or organizational delays. As further elaborated in Section 3.2.3, within this thesis, the "inherent availability" (A_i) is used, mathematically defined as:

  $A_i(t) = \frac{\text{MTBF}}{\text{MTBF+MTTR}}$,

  where MTBF = MTTF + MTTR.

In Table 2.1, the mutual influences between reliability, maintainability and availability are shown.

Reliability	Maintainability	Availability
higher	constant	higher
lower	constant	lower
constant	higher	higher
constant	lower	lower

Table 2.1.: Mutual influences between reliability, maintainability and availability

Evaluation. Dependability evaluation deals with the analysis of changes in a system, which are generally due to disruptions, and how such changes influence the behavior of the system measured by the metrics associated with dependability, introduced in the previous paragraph. Generally, two kind of evaluation methods need to be distinguished:

- **Measurements:** Measurements acquire results by observing an existing system. In case of a dependability evaluation, the main focus of observation is the occurrence of faults, errors, and failures as well as their influence on the system's behavior. After the data gathering phase is finished, the respective measures and metrics of interest are calculated. The two options regarding measurements are on the one hand field [4, 58], on the other test measurements [171, 185].

- **Models:** In contrast, models are constructed to represent an existing or non-existing system in an idealized manner. The measures and metrics of interest are gathered through subsequent evaluation of the finished model. Modeling approaches can be classified in several dimensions, e. g. being qualitative or quantitative in nature, their modeling and decision power, their modeling formalisms, and their solution method (analytical (closed form/numerical) or simulative) [163].

While many approaches do exist to model dependability and its associated metrics, only a brief overview of various dependability modeling formalisms is given next. A model selection in the context of this thesis and an in-depth description thereof is provided in Section 5.3. Modeling formalisms can coarsely be separated into three classes:

- **Non-state-space models** provide a high analytical tractability, yet at the cost of modeling power, due to the assumption that all analyzed components are independent. Example formalisms in this class are Reliability Block Diagrams (RBDs), reliability graphs, and fault trees.
- **State-space models** provide a high modeling power, yet suffer from low analytical tractability, because of the need for a state space generation. This easily leads to state space explosions, especially in complex systems. Example formalisms are Markov chains and Petri nets.
- **Multi-level models** combine both previously discussed classes. There are two main forms of such multi-level models, namely hierarchical and fixed-point iterative models.

2.1.1.2 Performance

Performance is one of the most well-known indicators for a system, encompassing many major fields, e. g. economics, risk management, and computer sciences. In the following, a focus is set on the performance in networked computing systems, which is defined in the following.

Definition. As performance is a very generic term that may refer to various different properties of a system, a formal definition of the term is hardly possible. Informally, performance is defined by Allen [2] as:

> *"The word performance in computer performance means the same thing that performance means in other contexts, that is, it means 'how well is the computer doing the work it is supposed to do?'"*
>
> — Arnold O. Allen, 1994

The "work it is supposed to do" can thereby vary greatly, depending on the goal a computer system is trying to achieve. Several examples of performance metrics relevant in the context of this thesis are discussed in the following.

Metrics. A performance metric is a measurable quantity that precisely captures a value of interest, which—in the case of performance—can take many forms. There is no generally defined performance metric fitting every system, instead, it is system dependent and requires a deep understanding of the respective system and its users. However, there are several common metrics available for well-known performance evaluation scenarios that are briefly introduced next.

- Latency (L) is the time required for a request to traverse a system. Latency is most often used as a metric in computer networks and depends on the speed of the transmission medium (e. g., copper wire, optical fiber or radio waves) and the delays in the transmission by devices along the way (e. g., routers and modems). It is therefore calculated as:

  $$L = d_q + d_p + d_t,$$

  where d_q is the queuing delay of the user's request, and d_p is the propagation delay (i. e. the time a signal change requires to travel over a physical medium from the sender to the receiver), and d_t is the transmission delay of a request.

- Response time (R) is defined as the interval between a user's request and the system's response. While response time can be generically assessed for almost all types of systems, in a computer network, it is calculated as:

  $$R = d_s + d_q + 2(d_p + d_t),$$

  where d_s is the service delay of the request and d_q, d_p, and d_t are defined as previously.

- Throughput (T), in general terms, is the rate of production or the rate at which something can be processed. The associated unit varies based on the system type, including jobs/requests per seconds, MIPS/MFLOPS, or bbs/bps. Generally speaking, the throughput of a system increases along with its load until the usable capacity is exceeded, leading to a drop in throughput again. T is calculated as:

  $$T = \frac{n}{\tau},$$

  where n is the number of jobs in the system and τ is the time a job requires to be serviced.

- Utilization (U) of a service is measured by the fraction of time the resource is busy servicing requests, i. e. the ratio of busy time and total time over a predefined time interval. It is calculated as:

  $$U = \frac{n}{\mu t},$$

  where n is defined as previously, μ is the service rate ($= \tau^{-1}$), and t is the time interval.

Evaluation. To evaluate performance, the same basic methodologies as discussed in Section 2.1.1.1 for dependability are used, i. e. measurements and models [61]. Moreover, the specific types of analyses for both measurements and models are similar, too, i. e. field and test measurements and analytical (closed form/numerical) or simulative approaches respectively. Regarding the modeling formalisms available, there are the shared options of Markov Chain- and Petri net-based solutions available, as well as the additional option of queuing systems/networks. Petri nets and queuing system/networks, which are the modeling options used later in this thesis, are described next.

Petri nets are a modeling formalism employed in various fields of computer science, combining a well defined mathematical theory with a graphical representation allowing the representation of dynamic systems. Petri nets are bipartite, directed graphs populated by three types of objects (places, transitions, and directed arcs). Definition 2.2 gives a formalization of Petri nets based on Cassandras & Lafortune [28].

Definition 2.2: Petri net graph

A Petri net graph (or Petri net structure) is a weighted bipartite graph represented as a 5-tuple (P, T, A, w, m_0), where P is a finite set of places, T is a finite set of transitions, $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs from places to transitions and from transitions to places in the graph, $w : A \to \mathbb{N}$ is a weight function on the arcs, and $m_0 : P \to \mathbb{N}_0$ the initial marking.

A queuing system (see Definition 2.3) is a station that consists of a queue and a server. When a package arrives at a station, it is queued. Eventually it is processed which takes a certain amount of time (= service time $\frac{1}{\mu}$). The service time follows a probabilistic distribution which can be defined.

Definition 2.3: Queuing system

A queuing system is defined by Kendall's notation as $A|S|m|B|P|Q$, where A is the probability distribution of inter-arrival times, S is the probability distribution of service times, m is the number of servers, B is the buffer size, P is the population size, and Q is the queuing discipline. An abbreviated form of Kendall's notation is available as $A|S|m-Q$.

After the processing is done at the queuing station and more stations do exist (forming a queuing network, see Definition 2.4), the package is sent to that next connected station. If multiple such connections exist, a probability is assigned to each link which determines how likely the package is sent through this link. Generically speaking, in queuing networks packages are sent from one queuing system to another until they reach the end of the network [22].

Definition 2.4: Queuing network

A queuing network is a directed, weighted graph with $QN = (V, E, w)$ with $V = \{QS\}$, $E = \{R\}$, and $w(E) \to \mathbb{R}$, where QS is a set of queuing systems and R is a set of directed edges $\big(E \in (V, V)\big)$ with an associated weight function $w(E)$ representing either routing probabilities or arrival rates.

Depending on the exact nature of both the queuing systems (e. g. $M|M|1-LCFS$ or $M|G|1-FCFS$) and the type of queuing network (e. g. open or closed queuing networks) itself, different solution strategies are available, which are not further discussed at this point; instead, it is referred to Haverkort [60] and Bolch *et al.* [22] for a detailed discussion of the subject.

2.1.1.3 Performability

After a brief introduction to dependability and performance measures, it becomes apparent—as already shortly announced in Section 2.1.1—that despite sharing several methodological techniques in their modeling and evaluation, mutual influences of dependability and performance are not considered in either of them. In today's complex and distributed systems, however, the individual assessments of system performance and dependability cannot be easily combined, particularly if performance in the presence of faults is degradable, i. e., errors or failures do not necessarily cause a system failure, but may as well only reduce the quality/performance of a delivered service, while the service itself remains proper [61].

Definition. The most well-known formal description of performability is given by Meyer *et al.* [98]. The performability ($Perf$) of a system is characterized by two subsystems comprising the total system (S), namely:

- the *object subsystem* (C), i. e., the technical system being evaluated, and
- the *environment subsystem* (E), i. e., external systems (i. e., environment or human) interacting with the object system.

S can thereby be denoted as a stochastic process $X = \big(X(t), t \geq 0\big)$, $t \in I$, describing the structure of the system at time t; the time base I may be discrete or continuous. The state space of X is denoted Q, which is a product space of the object and environment subsystems, i. e. $Q = Q_C \times Q_E$. To evaluate the performability of a system S, a performance variable Y is introduced, specified by

- a utilization period T, where T is an interval of the time base over which object subsystem performance is being observed, and
- an accomplishment set A in which Y takes its values.

This leads to the following definition of performability:

Definition 2.5: Performability

For a system S with performance Y taking values in accomplishment set A, the performability of S is the probability measure $Perf$ induced by Y where, for any measurable set B of accomplishment levels ($B \subseteq A$),
$Perf(B) = P[Y \in B]$, i. e. the probability that S performs at a level in B.

Metrics. The metrics associated with performability are required to reflect both the dependability and its impact on the performance. Therefore, the metrics are based on two interdependent components as suggested by Haverkort *et al.* [61], namely:

- the system structure Q, denoting the set of all possible configurations in which the system can operate. Also, the associated, previously defined stochastic process X describing which system components are up and which are down at time t, and
- the associated performance of the system in structure state $q \in Q$ is given by $r(q)$, where $r : Q \to \mathbb{R}$ is a reward function on the state space Q. $r(q)$ has to be defined by multiple performance analyses, i. e. for every $q \in Q$, the value $r(q)$ has to be obtained using a performance analysis.

For $x \in X$, let the row vector $\boldsymbol{\pi} = [\ldots, \pi_x, \ldots]$ denote the initial probability vector on X, the row vector $\boldsymbol{p} = [\ldots, p_x, \ldots]$ the steady-state probability of residing in state x, and the row vector $\boldsymbol{p}(t) = [\ldots, p_x(t), \ldots]$ the (transient) probability of residing in x at time t. The following metrics are defined:

1. Steady-state performability: $Perf(S) = \sum_{q \in Q} p_q r(q)$
2. Transient performability: $Perf(S, t) = \sum_{q \in Q} p_q(t) r(q)$
3. Cumulative performability: $Y(t) = \int_0^t r\big(X(s)\big) ds$
4. Performability distribution: $F(t, y) = P[Y(t) \leq y]$

The basic model discussed so far is a so-called rate-based reward model which means that when residing in a particular structure state $q \in Q$ at time t, the system performs with rate $r(q)$. The rates, however, may also depend on the global time t, thus having a reward rate function $r(q,t)$ for every state $q \in Q$. In contrast to rate-based models, it is also possible to define impulse-based reward models. With these models, a reward impulse function $r : Q \times Q \to \mathbb{R}$ has to be defined which associates a reward $r(q,q^*)$ with every transition from state $q \in Q$ to state $q^* \in Q$. Every time a transition from state q to state q^* takes place, an instantaneous reward of $r(q,q^*)$ is gained. These rewards may again depend on the global time t, leading to transition reward functions $r(q,q^*,t)$.

Evaluation. While being similar in nature again to both the evaluation approaches discussed for dependability and performance before, the performability adds an additional layer to all evaluation techniques: a reward analysis. In the case of measurements, this can easily be achieved by logging the system state (faults, errors, failures) and its respective influence on the system's performance.

If modeling is used for performability evaluation, especially state-space based approaches are often used to represent both the system state/behavior and its associated rewards; non-state-space models are mostly not suited due to their inherent assumption of system components being mutually independent. In comparison to the modeling formalisms proposed in Sections 2.1.1.1 and 2.1.1.2, a performability model is generally build up of two distinct models: a *behavior model* and its associated *reward structure* [98]. The two most well-known formalisms are Markov Reward Models (MRMs) and Stochastic Reward Nets (SRNs), which are extended versions of Markov models and Stochastic Petri Nets (SPNs), respectively. Further details regarding their structure, usage, as well as techniques to solve them are given, e. g., by Haverkort *et al.* [61], Muppala *et al.* [103], and Ciardo *et al.* [34].

2.1.2 Smart Grid and Advanced Metering Infrastructure

As briefly described in Chapter 1, a modernization of the existing grid is needed in order to meet the challenges it faces and the available information technologies could play an important role to achieve this endeavor. As the smart grid itself is the umbrella technology housing the AMI, it is only briefly described in Section 2.1.2.1, while the main part of the background in this regard is dealing with AMIs themselves in Section 2.1.2.2.

2.1.2.1 Smart Grid

While the power grid of Germany is still among the most reliable in the world[3], the increasingly out-of-date and overburdened infrastructure leads to a rising number of challenges, which has e. g. been confirmed by the *Bundesnetzagentur*:

> "*Due to the shutdown of the nuclear power plants and the installation of renewable energy generation, the existing power grid is however under considerably more stress. The transmission network operators, who are responsible for the functioning grid operation, must intervene in the use of the grid far more often in order to ensure the stable operation of the grid.*[4]"
>
> — Jochen Hoffmann, President of the *Bundesnetzagentur*, 2014

This trend is also affirmed by the European Institute for Climate and Energy (EIKE) and data from the German *Netztransparenz.de*[5], stating that the number of emergencies in the power grid, which needed to be corrected by redispatch measures has risen from 500 in 2011 to approximately 5500 in 2018. The current state of the power grid and its resulting issues are manifold, but can coarsely be classified into four main categories, according to *Alstom*[6] [92, 143]:

1. **Distributed generation:** The ongoing rise of small-scale, distributed renewable power generators, which are usually connected to the distribution grid, do not only offer benefits, but are also a potential reason for grid disturbances. At low levels of penetration, distributed generation simply reduces the load at individual substations. Higher levels of penetration in the system however further increase the uncertainty in supply, and at the same time add additional stress to the existing infrastructure.
2. **Aging power equipment:** While the environment the power grid operates in changed dramatically over time, the grid, which was not originally designed to reduce emissions or environmental impacts, be energy efficient, ensure cyber security or integrate renewable energy sources, stayed nearly the same. Many parts of today's power grid are already over 40 years old [135]. This causes two types of negative influences on the overall infrastructure: First, older equipment has higher failure rates, leading to customer interruption rates affecting

---

[3] According to data released by the *Bundesnetzagentur*, the country's power grid's total unplanned outage time was 15:32 minutes in 2013. Overall, these outage levels put Germany in the top five most reliable power grids in the world.

[4] As quoted by German national daily FAZ on 06/25/2014.

[5] Netztransparenz > EnWG > Redispatch, URL: `http://www.netztransparenz.de/de/redispatch.htm`, last accessed: 03/01/2019

[6] Alstorm | Challenges of electrical grids, URL: `http://www.alstom.com/grid/about-us/understanding-electrical-grids/challenges-of-electrical-grids/`, last accessed: 03/01/2019

the economy and society. Second, older systems need to be inspected and maintained more carefully, which increases maintenance and repair costs.

3. **Missing situational awareness:** As already shortly described in the previous paragraph, the current power grid is missing the technological features to offer a wide situational awareness to its operators. To prevent—or contain—failures in the power grid, it is essential to have real-time information from all levels of the grid, which is currently not available, especially in the low voltage level.

4. **Increasing demand and quality:** Our digital society more and more depends on and demands power supply of high quality and reliability. Even small fluctuations in the power grid can lead to massive challenges, especially in the area of industrial production. Therefore, the current power system faces challenges that are constantly growing over time. Examples are a growing worldwide energy demand, increasing difficulties to access fossil fuels in a feasible manner [142], as well as criminal and terrorist threats [80].

To cope with the presented challenges and situations the creation of a new electricity infrastructure is necessary, which is able to improve management, monitoring and use of electricity, allowing utilities to analyze the status and behavior of the grid efficiently. This new infrastructure also needs to effectively integrate new sources of energy without negatively affecting the power grid, and to manage and regulate their volatile power output. This will allow the system to efficiently provide energy, protect the climate, besides meeting future power demand and quality requirements. This so-called smart grid (see Definition 2.6) is further explained in the following.

Definition 2.6: Smart Grid

The term "smart grid" refers to a modernization of the electricity delivery system so it monitors, protects and automatically optimizes the operation of its interconnected elements—from the central and distributed generator through the high-voltage transmission network and the distribution system, to industrial users and building automation systems, to energy storage installations and to end-use consumers and their thermostats, electric vehicles, appliances and other household devices. The smart grid will be characterized by a two-way flow of electricity and information to create an automated, widely distributed energy delivery network. It incorporates into the grid the benefits of distributed computing and communications to deliver real-time information and enable the near-instantaneous balance of supply and demand at the device level [42].

While this definition does not give a detailed insight into all areas and technologies employed in the smart grid, no further clarification is given at this point for the sake of briefness. However, for further information, it is referred to the National Institute of Standards and Technology (NIST) Framework and Roadmap for Smart Grid Interoperability Standards [108]. In the following, a short introduction on AMIs, a major enabler of the smart grid, is given.

2.1.2.2 Advanced Metering Infrastructure

AMI is one of the major technologies employed in smart grids required to enhance the distribution domain by continuous information acquisition and transmission from and to the customer domain, as stated in Definition 2.7 from the Federal Energy Regulatory Commission (FERC) [53].

Definition 2.7: Advanced Metering Infrastructure

A metering system that records customer consumption hourly or more frequently and that provides for daily or more frequent transmittal of measurements over a communication network to a central collection point. It includes communication hardware and software, associated system and data management software that creates a network between advanced meters and utility business systems and which allows collection and distribution of information to customers and other parties such as competitive retail providers, in addition to providing it to the utility itself.

AMI can therefore be seen as an umbrella term covering multiple systems. In the definition, these are however only mentioned on an abstract, logical plane, while a detailed realization is not further explained. Because of that, in the following paragraphs, first, the hardware components are described; after that, the interconnecting networks are analyzed (based on Mohassel *et al.* [100]).

AMI Substrate Entities. The AMI's substrate entities provide the services which—in their interplay—enable the AMI's core functionalities as well as optional additions enabled through third parties.

- **Smart meters:** Smart meters are a significant building block of the AMI. These devices are a combination of hard- and software that are capable to measure and collect consumption information in small time intervals. The stored data can then be used both internally and externally via the Smart Meter Gateway (SMGW). Unlike Automated Meter Reading (AMR), to communicate with external parties, smart meters are equipped with two-way communications capabilities, enabling them to both send and receive data and act accordingly. The network technologies and communication protocols employed by smart meters are further explained later on. While the internal technical realization of smart meters varies based on vendor, location and model, there are several essential functionalities which smart meters have to fulfill regardless of their differences [149]. These functionalities include, among others:
  - quantitative measurement of energy consumption,
  - two-way communication,
  - real-time display of consumption,
  - time-based pricing,
  - providing consumption data for consumer and utility,
  - failure and outage notification,
  - remote command (turn on/off) operations,
  - load limiting for demand-response purposes,
  - power quality monitoring (e. g., phase, voltage and current), and
  - energy theft detection.

- **AMI smart meter gateway:** In difference to most other countries, the usage of SMGWs was first demanded by the German Bundesamt für Sicherheit in der Informationstechnik (BSI) in 2013 [25]. The first essential objective of the SMGW results from the property of the SMGW as an interface between Local Meteorological Network (LMN), Home Area Network (HAN), Neighborhood Area Network (NAN) and Wide Area Network (WAN). For the security of the overall system, a partitioning of these networks isolating them from one another is beneficial, which can be realized by firewall mechanisms. Furthermore, every network connection established by the SMGW generally has to be encrypted and integrity protected [25, 26]. Additionally, as stated by the BSI in 2012, SMGWs may differ in their architecture, depending on the features they implement (e. g., communication capabilities can either be provided by the SMGW or by another device, such as a router) [25].

- **AMI data concentrator:** Another component of an AMI system is the data concentrator. This element aims to collect and store periodical meter readings, often from multiple meters located in the LMNs, before forwarding them to the utility's AMI Head End System (HES). Data concentrators are usually positioned in transformer centers, and collect communications from several different homes. They are especially crucial and heavily used in densely-populated areas. Data concentrators also have capabilities that allow them to establish bidirectional communication links with the HES, enabling at the same time to monitor and maintain the status of both network paths. A data concentrator manages 50 to 100 households in more rural areas, while this number can decrease to 20 to 50 in more urban areas, depending on the architecture and communication medium [53, 129].

- **AMI head end system:** The HES is hosted at utility's site. Among its responsibilities are the execution of commands and management of metering data and other components in order to abstract details of the AMI system and communication network [47]. An important part of the HES is the Meter Data Management Systems (MDMSs), which provides the required database to store and manage the analysis of data collected from meters. This system allows keeping track of all meters, their status and communication paths within the network. This facilitates troubleshooting device's failures more efficiently, besides enabling remote managing of firmware updates, reconfigurations, and diagnostics, among others.

- **Third party services:** In addition to the necessary AMI services—which are key elements of the infrastructure used to provide the most crucial functionalities of AMIs, such as automated remote meter reading and bidirectional communication altogether—Third Party Services (TPSs) offer enhanced functionalities that are not inherently required by AMIs to work, but may yet provide more convenience for users. An example for such a service is an energy optimization service that helps to conserve energy by educating end users after analyzing their power consumption. Such services are not offered by the service provider and run on hardware which is under the supervision of a third party service provider; however, most TPSs require data generated by devices such as smart meters, which may either be controlled by the third party itself or other AMI participants.

AMI Communication Networks. The backbone of future smart grid applications depends on its communications infrastructure. Without a dependable communication infrastructure, neither the gathering of information, nor the planned messaging to and control of actuators in the smart grid is possible.

- **Network Topology:** Trick [161] states that the LMN is used for communication between the smart meter and the respective SMGW (other meters, such as water and gas meters, are also connected to the SMGW via the LMN). The transmitted data within the LMN contains energy consumption and production information of a household as well as locally measured parameters such as voltage, frequency or phase angle. The BSI [26] specifies the allowed protocols for this interface as: OMS Part 2 (Open Metering System) based on M-Bus DIN EN 13757-3, DLMS/COSEM DIN EN 62056-61/62/53 or SML prEN 62056-5-8 (Smart Message Language). Beyond that, also IEC 61850, IEC 61968/61970 CIM, ZigBee SEP2, KNX, OASIS Energy Interoperation, ANSI C12.22 or FNN SyM2 are allowed. A secure data transfer is ensured by means of Transport Layer Security (TLS). For data transmission, both a wireless interface (M-Bus DIN EN 13757-4) as well as a wired Ethernet interface are provided. For wired interfaces, the standards M-Bus DIN EN 13757-2, DIN EN 13757-6 Lo-Bus and RS 485 are also permitted.

  The HANs connects smart meters, smart devices within the home premises, possible energy storage and generation (solar, wind, etc.), electric vehicles as well as Controllable Local Systems (CLS) inside of households. As the data flow in HANs is instantaneous/intermittent rather than continuous, the required bandwidth may vary from 10 Kbps to 100 Kbps for each device, depending on its task. The network needs be expandable, too, as the number of devices or data rate may increase in the future. The calculated reliability and accepted delay are also based on the consideration that the loads and usage are not critical. Given the above requirements and considering the short distances among nodes that enable low power transmission, wireless technologies are the dominant solutions for HANs. These technologies include WiFi, 802.11 wireless networking, ZigBee and HomePlug [100].

  The NAN is a network interconnecting data concentrator devices and neighboring gateways in a specific region. As data from each smart meter is transferred to the concentration points, they require a high bandwidth, as well as a decent QoS to support a near real-time delivery of data. Currently, there are several technologies suggested to realize these requirements in NAN, each having advantages and downsides. Power Line Communication (PLC) addresses the communication needs between in-home gateways and concentration points in sparsely populated areas. If communication at the aggregation point is meant to be distributed to each, or most of the smart devices inside the home rather than the meter, then more bandwidth is needed which narrowband PLC is unable to provide. Here, existing Internet connections, such as Digital Subscriber Line (DSL), offer higher bandwidths. However, Internet connections often suffer from bad QoS due to their best-effort type of service.

  The WAN connects the SMGW to external market participants. These include all potential communication partners, such as the SMGW administrator, meter operators, energy suppliers, telecommunication providers or third party services. Especially, it also serves as a connection between SMGWs, data concentrators, and the AMI HES present at the service provider. Since millions of metering data are transferred in the WAN, the requirements of both bandwidth and dependability of the network are considerably high, which is detailed further in Table 2.1.2.2. Currently, several technologies are suggested to realize these requirements in WANs, each having advantages and downsides. PLC addresses the communication needs between SMGWs and other participants in sparsely populated areas. However, in densely populated areas, more bandwidth is needed which PLC is not able to provide. Here, dedicated fiber networks or existing Internet connections are able to satisfy these requirements. However, dedicated fiber

networks are expensive, while Internet connections often suffer from unpredictable QoS due to their best-effort type of service.

- **AMI Communication Protocols:** The communication protocols employed in AMIs are numerous and vary depending on the concrete realization and location of an AMI. In the following, several well-established standards are discussed, before possible architectures are presented. The ANSI created a protocol suite (C12.18, C12.19, and C12.21) for the communication with smart meters which is mainly used in the US. ANSI C12.18 is a protocol used for two-way communication with smart meters via an ANSI Type 2 optical port. ANSI C12.19 specifies the data tables to be used. ANSI C12.21 is an extension of C12.18 written for modem instead of optical communications. In the European Union, the IEC 62056 specifies communication protocols for sending ASCII data using a serial port.

Technology	**Protocol**	**Bandwidth**	**Range**	**Networks**		
				HAN	NAN	WAN
Wired						
Optical fiber	PON	0.15 – 2.5 Gbps	≤ 60 km			✓
	WDM	30 – 40 Gbps	≤ 100 km			✓
	SONET	8 – 10 Gbps	≤ 100 km			✓
Cable line	DOCSIS	100 – 500 Mbps	≤ 30 km		✓	
DSL	ADSL	1 – 8 Mbps	≤ 5 km		✓	
	HDSL	1 – 2 Mbps	≤ 4 km		✓	
	VDSL	15 – 100 Mbps	≤ 2 km		✓	
PLC	HomePlug	14 – 200 Mbps	≤ 200 m	✓		
	Broadband	0.5 – 500 Mbps	≤ 3 km		✓	✓
	Narrowband	10 – 500 kbps	≤ 3 km		✓	
Wireless						
Wi-Fi	802.11x	2 – 600 Mbps	≤ 100 m	✓		
Bluetooth	802.15.1	500 – 750 kbps	≤ 100 m	✓		
ZigBee	ZigBee	150 – 250 kbps	≤ 100 m	✓		
Cellular	2G	10 – 15 kbps	≤ 50 km		✓	✓
	3G	1 – 2 Mbps	≤ 50 km			✓
	4G	50 – 100 Mbps	≤ 50 km			✓

Table 2.3.: Properties of AMI communication technologies, based on Kuzlu *et al.* [77]

Other options include:

  - Open Smart Grid Protocol (OSGP), developed by the European Telecommunications Standards Institute (ETSI). It is used in conjunction with the ISO/IEC 14908 standard for smart metering and smart grid applications.

- M-Bus (Meter-Bus), which is a European standard (EN 13757-2 physical and link layer, EN 13757-3 application layer) for the remote reading of smart meters. The M-Bus interface communicates—just as IEC 62056—via a pair of wires, making it very cost effective.
- ZigBee Smart Energy V2.0, which is a special branch of ZigBee defined as a protocol to monitor and control energy (also smart meters) as an enhancement of the ZigBee Smart Energy V1.0 specification.

In addition, it is noteworthy that there is a trend toward the usage of TCP/IP technology as a common communication platform for smart metering, which, e. g., is noticeable in the employment of ZigBee IP for ZigBee solutions, a network layer that routes standard IPv6 traffic over IEEE 802.15.4. The usage of TCP/IP offers numerous benefits, i. e. utilities can deploy multiple communication systems, while using IP technology as a common management platform. This would allow a single product to be used globally even if regional communication standards vary. An overview of the communication technologies used in AMIs is provided in Table 2.3.

- **AMI Architecture:** In the following, several possible architecture options are showcased in Figure 2.3, based on the approach of NIST [108] and Foreman & Gurugubelli [55]. As shown, the infrastructure especially differs in the possible communication mediums available at the NAN and WAN part. At the NAN part, the connection from the user to the data concentrator, can be established via: radio point-to-point network, radio mesh network, PLC, and dedicated or public copper/fiber networks. In the WAN part of the infrastructure, the connections from the data concentrators to the back-haul routers are either realized via cellular networks, public/private RF, or public/private copper/fiber networks. After that, only wired connections (based on high-bandwidth copper or fiber) link the back-haul routers to the providers infrastructure including the AMI HES [97].

  It has to be noted however that these different technologies are facing various challenges, such as reliability, transmission rates, and real-time capability. Apart from the apparent limitations of the technologies, it is additionally required to assess the advantages and downsides of either employing a provider-owned, dedicated AMI communication network or relying on public network infrastructures, such as General Packet Radio Service (GPRS) or DSL. As an example, public networks may face the disadvantage that—in case of power disturbances—parts of the AMI are not reachable anymore, which may lead to issues during the restoration. In contrast, a utility's dedicated communication infrastructure can be constructed in a way to be operable even during power outages. However, the construction of a wide area dedicated network induces increased initial and maintenance costs to the utility.

- **AMI Services and Requirements:** The service that is the closest to the customers is the AMR. This service gathers consumption and event data from smart meters and sends it to the utility for further processing and analysis. It is the main service studied in this thesis. In a future smart grid realization, every household is expected to participate in AMR. The Real-Time Pricing (RTP) is part of the demand-response approach that is one of the novel ideas in smart grids. It gives incentives to users to shift the usage of appliances that demand large quantities of energy to a specific time when it is abundantly available. RTP requires bidirectional communication between the utility and the households. Utilities are particularly interested in Wide Area Monitoring (WAM) services because they are used to continuously monitor the grid's health through performance indicators such as frequency, real and reactive power as well as phase angle.

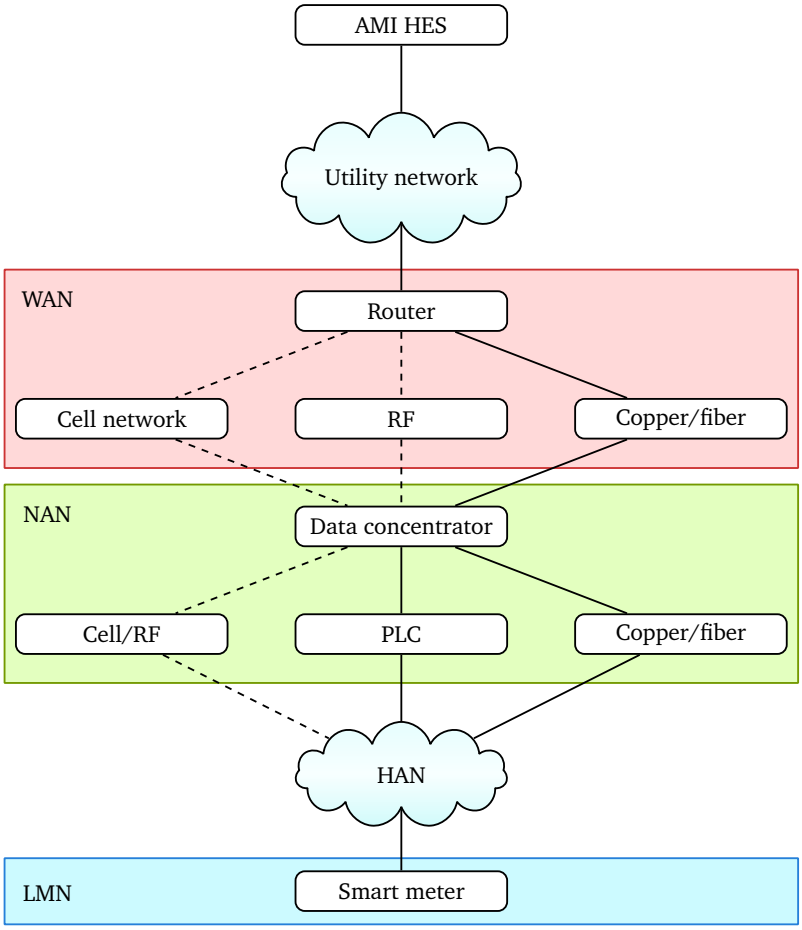


Figure 2.3.: Overview of AMI architecture, networks, and technologies

All services have different attributes like message size and sending intervals as well as different requirements such as a maximal allowed latency. From this, the necessary data rate can be calculated for each service, which is an important constraint for the selection of an appropriate communication technology and network topology. The requirements are of major importance later within this thesis (mainly throughout Chapter 6) to evaluate whether the suggested solution developed in Chapter 4 can fulfill the subsequently stated requirements (see Table 2.1.2.2).

Application	Data rate	Intervals	Max. latency	Protocols
AMR	10 – 128 kbps	5, 10, 15, 30, 60 min	100 ms	HTTP, TCP
RTP	10 – 100 kbps	15, 60 min	200 ms	HTTP, TCP
WAM	6 – 24 kbps	40 – 100 ms	10 ms	UDP, IP

Table 2.5.: AMI communication requirements based on Ramírez *et al.* [129] and Shiobara *et al.* [147]

2.1.3 Virtualization

Virtualization is a technology that enables the usage of a common hardware substrate by multiple virtual appliances in parallel. It was first developed by the company IBM in the 1960s as the Control Program/Cambridge Monitor System (CP/CMS) which offered a partition of their mainframe computer, separating it into several logical instances each usable by a separate user. Originally, this technology was needed, as mainframes were too large and expensive to be used by only one person at a time. Also, the ability to run multiple processes and applications at the same time increased the utilization [124, 64].

Since then, the term virtualization has been broadened up and covers several approaches to run software by isolating and abstracting from lower layer functions and the actual substrate hardware. By doing so, it is possible to achieve a portability of higher level functions. Also, the resources on hardware level can either be shared or aggregated into virtual environments. In the following, several technological branches of virtualization are mentioned.

2.1.3.1 Host Virtualization

In the following, possibilities to perform host virtualization are covered based on Walters *et al.* [169].

- **Full Virtualization:** Using this approach, a VMM is running on top of a host operating system, usually from user space. The VMs that are hosted on top of the VMM are each provided with their own set of virtual hardware, which is—in turn—used by a guest Operating System (OS) that is able to run applications. One important advantage is that guest OSs running inside of their VMs do not need to be modified in any way to be compatible to full virtualization, as the required hardware is emulated by the VMM.

- **OS-Layer Virtualization:** Also known as container-based virtualization, this concept implements virtualization by running more instances of the same OS in parallel. This means that not the hardware, but the host OS is the one being virtualized. The resulting VMs all use the same virtualized OS image. Due to that, this approach has a major drawback: since the VMs use the same kernel as the host OS, the guest OS must be the same as the host OS (and such, it is not possible to run, e. g., Windows on top of Linux).

- **Hardware-Layer Virtualization:** This approach is commonly used on the server market due to its high virtual machine isolation and performance. Here, the VMM runs directly on hardware, controlling and synchronizing the access of the guest OSs to the hardware resources. Paravirtualization is the technique used by Xen which provides a virtual machine interface representing a slightly modified copy of the underlying hardware, where the non-virtualizable portions of the x86 original instruction set are replaced with their virtualized equivalents.

- **Paravirtualization:** Unlike full virtualization, in paravirtualization the running guest OS needs to be modified in order to be operated in the virtual environment. An interesting fact in this technology is that the guest machines are aware of the fact that they are running in a virtualized environment. One of the main characteristics of paravirtualization technology is that the VMM is simple which allows paravirtualization to achieve performance closer to non-virtualized hardware [95].

2.1.3.2 Network Virtualization

Network virtualization is the approach of setting up various virtualized networks on top of a substrate network. A virtual network thereby exists of virtual nodes, interconnected by virtual links. Both are hosted on a substrate network, which is also composed of nodes and links connecting them. By applying virtualization to nodes and links, it is possible to create and co-host multiple, logically isolated virtual networks, each having different properties and requirements (e. g., QoS or security), on a substrate network [54].

Node Virtualization is thereby done by offering a slice of a substrate node and allow the implementation of separate customized control protocols on them. Virtual nodes thereby are similar in functionality as in a physical networks, i. e., a node may offer the services of Data Communication Equipment (DCE) such as a hub, bridge or switch, or Data Terminal Equipment (DTE), such as a host computer. In addition, virtual nodes offer a clear mutual isolation as well as dynamic instantiation and capability. To establish a substrate and protocol agnostic way to connection between virtual nodes, link virtualization is required. A well-known example employing virtual links exists in the form of Virtual Private Networks (VPNs), creating isolated tunnels over multiple physical links. Similar tunneling mechanisms can also be used in case of Virtual Networks (VNs). As virtual nodes, they allow dynamic instantiation as well as capability changes during runtime. In addition, virtual links can be aggregated to offer more resources (e. g., bandwidth) [33].

2.1.3.3 Network Function Virtualization

NFV was recently introduced by the NFV Industry Specification Group at the ETSI [50]. It introduces a network architecture concept employing virtualization technologies to softwarize network functions previously executed by proprietary (vendor-specific), dedicated hardware. These VNFs replacing physical network node functions comprise of VNFCs, which realize single functions within the VNF.

NFV is built upon the virtualization technologies introduced in Sections 2.1.3.1 and 2.1.3.2. To realize a VNF comprised of several VNFCs, a combination of host and network virtualization is required to generate both the virtual machines running different software and processes (on the Network Functions Virtualization Infrastructure (NFVI)) and their interconnections, assuming that the VNFCs are hosted in a distributed way. To realize a NFV architecture, a framework of three main components (VNF, NFVI, and Network Functions Virtualization Management/Orchestration (NFV-MANO)) is required. These are covered in the following.

- **VNF:** VNFs are a softwarized implementation of a network function previously realized in hardware, capable of running on top of the provided NFVI. The VNF is the entity corresponding to today's network nodes, which are now expected to be delivered as pure software free from hardware dependency. Internally, a VNF comprises several VNFCs connected among each other to provide a service realized by the VNF in their interplay.

- **NFVI:** The NFVI provides the virtual resources required to support the execution of VNFs. It includes COTS hardware as well as a virtualization layer which abstracts from the underlying substrate hardware and logically partitions it. Above the virtualization layer, VNFCs are typically mapped to the VMs available through the NFVI. The deployment, execution and operation of VNFCs on the NFVI are managed by the NFV-MANO.

- **NFV-MANO:** Since Virtual Links (VLs), Physical Network Functions (PNFs), VNFs, NFVI and the mutual relationships did not exist before the emergence of NFV, their handling requires a

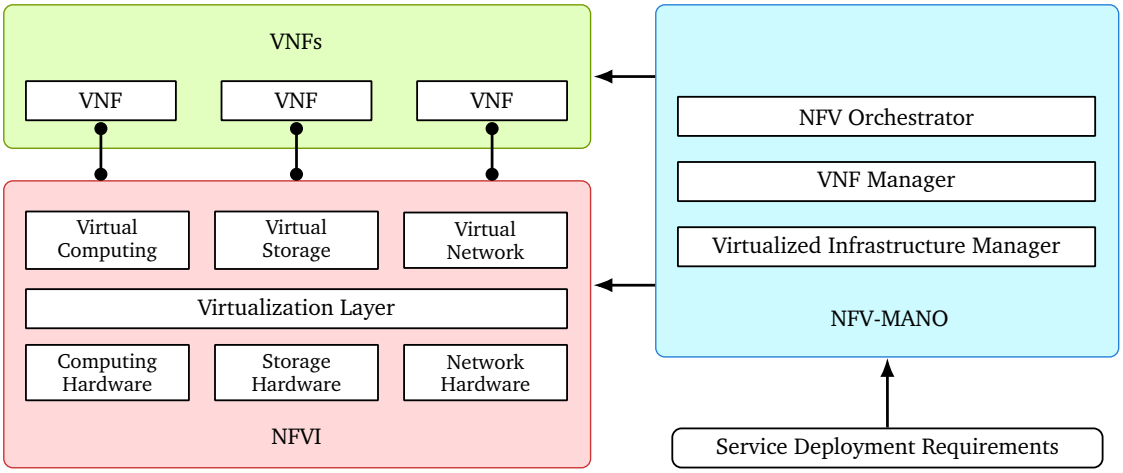


Figure 2.4.: Scheme of the VNF architecture, including VNFs, VNFI, and NFV-MANO

new set of management and orchestration functions. The NFV-MANO architectural framework has the role to manage the NFVI and to orchestrate the allocation of resources needed by the VNFs [49]. The NFV-MANO is itself comprised of three interconnected functional blocks:

- Virtualized Infrastructure Manager (VIM): The VIM controls and manages the resources (compute, storage and network) of the NFVI within an operator's infrastructure by keeping an inventory of the allocation of virtual resources to physical resources. This allows the VIM to orchestrate the allocation, upgrade, release, and reclamation of NFVI resources and optimize their use. Also, it is responsible for the collection and forwarding of performance measurements and events.
- Virtualized Network Function Manager (VNFM): The VNFM acts as a lifecycle manager of VNFs under the control of the Network Functions Virtualization Orchestrator (NFVO), which it achieves by instructing the VIM. This includes the instantiation, scaling, updating, and termination of VNFs. All VNFs have an associated VNFM, which may manage a single or several VNFs at once that may be of the same or different types.
- NFVO: The NFVO provides VNF lifecycle management (including, e. g., instantiation, performance measurements, event correlation, and termination). To do so, the NFVO ensures that—if possible—a VNF receives an adequate amount of compute, storage, and network resources from the NFVI. The NFVO supports two operating modes: in coordination with the VIM; or directly with NFVI resources, depending on the requirements. Thereby, the NFVO can coordinate, authorize, release, and engage NFVI resources independently of any specific VIM. To provide service orchestration, the NFVO creates end-to-end service among different VNFs—that may be managed by different VNFMs with which the NFVO coordinates.

2.2 Related Work

As the three fields themselves offer extensive related work, in this section, only articles that deal with a combination of at least two of the scientific areas present in this thesis are covered. In Section 2.2.1, performability concepts for the smart grid are discussed; in Section 2.2.2, virtualization approaches to improve smart grids are reviewed; finally, in Section 2.2.3, related work combining all three scientific fields are presented, including the author's own previous work in the field.

2.2.1 Performability/Dependability in Smart Grids

Regarding both the dependability and performance of smart grids, numerous works exist encompassing the networking infrastructure as well as the hardware and storage facilities involved in the communication and computation processes in the grid. Both the analysis and improvement of these properties of the smart grid have been investigated using various approaches in the past. In contrast, performability in smart grid networks is a widely disregarded topic till today.

For the analysis of smart grids, several schemes have been proposed, e. g., Kaitovic *et al.* [70] discuss a unified dependability approach for combined ICT and electric power infrastructures. In addition, a taxonomy of faults for smart grids is presented and a outlook on the present dependability state of smart grids is given. A combination of dependability analysis techniques and a novel AMI architecture proposal is given by Xiang *et al.* [174], where the dependability modeling and resource allocation in redundant communication networks using RBDs and SPNs is investigated. Also, several network redundancy mechanisms under the influence of varying Mean Time To Failure (MTTF) are compared. Metrics for the analysis of smart grid reliability and resilience are the focus of Albasrawi *et al.* [1], providing a reliability and resilience analysis of the ICT components of the smart grid. Special focus is laid on the metrics available for assessment of two phases of smart grid operation: the duration before a failure occurs, and the recovery phase after an inevitable failure. The former is characterized by reliability, which is determined based on information about cascading failures. The latter is quantified using resilience, which can in turn facilitate comparison of recovery strategies. Huang proposes a k-to-n interdependence model for smart grids to represent cascading failures between physical and cyber infrastructure [63]. Through calculating the fraction of functioning parts (survival ratio) using percolation theory and generating functions, it is revealed that there is a non-linear relation between controlling cost and system robustness. It is proven that there is a threshold for the proportion of faulty nodes beyond which the system collapses. Similarly, Chopade & Bikdash provide a novel approach to analyze vulnerabilities of interdependent smart grid networks regarding their impact on network survivability based on interdependency modeling [32]. Marashi & Sarvestani present an approach to model reliability in smart grid communication based on a preliminary Markov embedded systems and describe how it can be evolved to capture vulnerabilities [91].

Solutions to increase dependability in AMIs are, e. g., presented by Okabayashi *et al.* [117], who investigate the consequences of gateway failures in smart grid communication networks and their influence on the overall performance of smart grid services. A solution to the found challenges is presented in the form of an algorithm for dynamic selection of gateways in a multi-homing smart grid network. Varaiya *et al.* propose a new reliability concept based on real-time supply and demand information from the smart grid as well as the stochastic nature of renewable resources and demand-response [168]. While this model reflects the behavior of renewable energy sources well, there are no proposals how to decrease observed risks or how the reliability of a given smart grid infrastructure

could be improved. Nguyen & Flueck present a survivability strategy for communication networks in smart grids based on multi-agent systems [110]. The main contribution here is a detection scheme to track the status of communication links in smart grids. Apart from that, a strategy to handle failures in the communication links is presented that allows—in the case of permanent failures—to inform an administrator; in the case of temporary failures, message sending is delayed till the system function is restored.

Apart from such software-based solutions, Shi *et al.* propose a redundancy optimization method for smart grid AMI communication [146]; in particular, the redundancy optimization problem focus on deciding which data concentrator in an AMI requires redundancy to meet certain reliability goals. Based on a quantitative analysis model, which comprehensively considers both reliability and economy, an advanced redundancy deployment method based on genetic algorithms is developed to solve the proposed problem. Similarly, Niyato *et al.* also analyze the reliability of smart grids' data communication systems [114]. More specifically, the article focuses on a reliability analysis of the wireless connectivity between a smart meter and the MDMS, given random failures of system devices. In the end, a cost optimization is performed to determine a redundancy approach minimizing the cost of failure as well as the cost of deployment of the wireless communications system. More generally, Moslehi & Kumar [102] and Koziolek *et al.* [75] investigate reliability challenges in smart grids in a broader scope, focusing on the reliability of power supply under various challenges. ICT reliability or dependability are however not further discussed. Also, mainly basic ideas, e. g., the influence of renewable and demand-response on reliability/survivability, are analyzed using an abstract framework that deals with possible negative impacts on energy supply reliability.

Regarding dependability modeling, Alves *et al.* propose a fault tree-based methodology for smart grid dependability evaluation, which is suitable to be applied in early design stages to evaluate reliability and availability [5]. Also, potential critical regions and components of a smart grid can be identified. In contrast to the fault tree-based methodology of Alves *et al.*, this thesis uses a reward net approach, which is already discussed as a future extension in their paper. The identification of critical components, which uses a Birnbaum measure in the approach of Alves *et al.* is instead covered by a sensitivity analysis in this thesis. Nieße *et al.* present a process model that allows to contribute application-oriented research results for distributed control concepts in ICT for power systems [113]. The process model is set up with an initial conceptualization phase followed by a cycle of five phases with both analytical and experimental parts. While the process model presented is not directly used in this thesis, the iterative phases it describes (analysis of designed solutions, their implementation and subsequent experimentation) is also widely adopted here.

On the side of performance analysis, Priya & Saminadan propose a Worldwide Interoperability for Microwave Access (WiMAX) based traffic priority model for the NAN and HAN in smart grids [125]. In particular, three traffic types, differing in their QoS requirements are used to analyze the suitability of WiMAX for smart grid usage. A similar topic is covered by Neagu & Hamouda [109]. The investigation covers the performance of several different applications (e. g., metering and pricing, electric cars, video surveillance and voice support) under the influence of the impulsive noise over the communication layer. Quang *et al.* develop a testing framework as a guideline to assess the performance of several different communication mediums and technologies in smart grid communication networks [126]. Also, an evaluation of three different technologies in smart grids, i. e., WiMAX, Mesh RF and PLC, is done based on several performance metrics, i. e., reliability, latency and throughput. Stress tests are also carried out to determine the operating limit and capacity of traffic in the communication network. None of the articles however cover infrastructure disruptions in their analyses.

2.2.2 Virtualization for Performability/Dependability

Performability in combination with virtualization techniques has and is still receiving a lot of research attention. Thein *et al.* introduce a survivability framework for distributed systems through the use of virtualization technology and software rejuvenation to provide continued service and fast recovery [156]. While a detailed analysis regarding the survivability of the virtualized system is carried out, performance is disregarded in the article. Yeow *et al.* investigate the impact of substrate failures on virtual infrastructures, which may lead to cascading failures due to an overload induced to the remaining servers [179]. To guarantee a certain level of reliability, it is suggested to add redundant nodes and links that have sufficient capacities to the virtual infrastructure. This way, if substrate failures occur, sufficient computing resources are available and the virtual network topology is preserved. Kim *et al.* develop an availability model for virtualized system based on fault trees and homogeneous Continuous Time Markov Chains (CTMCs), which incorporates hardware and software failures, the latter also including virtualization failures (i. e., VMM and VM failures) [72]. The model is tested with high availability services and VM live migration settings. Nguyen *et al.* propose a cluster model of virtualized servers using a SRNs approach [111]. The developed model incorporates standby, VM live migration and failover techniques, as well as simplified failures and recovery behaviors of physical servers and VMs. Finally, several SRN models are developed based on different use cases to analyze ways to improve a system's overall availability. In contrast, Machida *et al.* develop a state-of-the-art approach on software rejuvenation in virtualized data centers to enable an effective mitigation of software aging in both VMs and VMM [86]. By optimizing the placement of VMs and the rejuvenation schedule, the performability of the overall system is increased. A similar goal is pursued by Silva *et al.*, where an approach for software rejuvenation based on automated self-healing techniques is introduced [150]. To detect software aging and transient failures, a continuous monitoring of system data and performability metrics is used, triggering an automatic rejuvenation action if abnormal behavior is detected.

The usage of virtualization within the smart grid is a relatively recent approach. Liberatore & Al-Hammouri proposes the usage of a system of heterogeneous networks to provide high levels of real-time performance, reliability, and security [82]. The creation and exact nature of the virtual networks is however not discussed, as well as their implications on performability. More recent than the development of virtualization based approaches for performability is the usage of NFV and Software-Defined Networking (SDN). However, these technologies gain more and more importance currently, not only to increase smart grid performability, but also in other environments. Especially in combination with cloud computing, NFV has evolved as a proposal from operators for hosting network services as VNFs. Yu *et al.* [182] and Cerroni & Callegati [29] state that NFV can be used to replace currently costly hardware middleware in clouds to provide similar, yet flexible and cost-effective services. Apart from NFV itself, the underlying substrate infrastructure, namely the NFVI has come to attention of researchers just recently. Cotroneo *et al.* [39, 38] evaluate dependability concerns in NFV by employing fault injection techniques. However, the focus of both articles is the reliability of NFVI under the influence of challenges originating from the usage of commodity software and hardware issues, which is not applicable in our scenario, where the NFV is executed and hosted on servers supervised by professionals. In addition to NFV, SDN is a related approach currently gaining significant attention, especially to increase performability measures; this is, e. g., presented by Di Mauro *et al.*, aiming at selecting the best redundancy scheme for an SDN controller [41]. The SDN model aims to compare various redundancy schemes under the influence of randomly occurring failures using a CTMC modeling approach.

2.2.3 Combining Approaches & Own Contributing Work

As the previously described approaches showed, there is significant interest in increasing the performability of smart grid communication infrastructures. However, the solution approach of this thesis is not yet introduced in other authors' works. However, there have already been approaches that use virtualization to increase performability in smart grid networks, which are discussed next.

Xin *et al.* develops a virtual smart grid architecture based on a cloud approach and suggest the virtualization of various elements within the smart grid and therefore directly support the approach of this thesis [177]. In contrast to the approach by Xin, the solution presented by the author suggests a concrete architecture for a AMI communication through virtualization. Similar approaches dealing with smart grid computations in cloud infrastructures are discussed by Sivapragash *et al.* [151] and Xiang *et al.* [175], among others. While cloud-based approaches offer advantages in flexibility and scalability, the downsides are possible security, privacy and data protection challenges [31, 145]. On the dependability side, clouds are certainly favorable in the computation domain compared to single servers due to their decentralized processing. Yet, the infrastructural dependability questions (especially in the LMN, HAN, and NAN domain) remain unsolved by such approaches. A framework based on network virtualization for smart grid communications is proposed by Lv *et al.* [85]. In the framework, real-time services are supported by virtual networks that are mapped to two physical networks simultaneously, i. e., wireless mesh and PLC networks. The enhanced transmission diversity through the two networks contributes to the reliability of real-time services. Since the virtual network mapping problem is $\mathcal{NP}$-hard, a heuristic solution is developed to solve the problem. Aydeger *et al.* presents an SDN solution for redundant communications in smart grids [10]. Specifically, multiple connection interfaces among distribution substations are developed to provide redundant network connectivity. An SDN approach to manage smart grid communication is also introduced by Rinaldi *et al.* [136]. The preliminary feasibility analysis of the approach seems promising, although a more detailed modeling and analysis of the system is needed due to the extreme heterogeneity of the network. Dorsch *et al.* presents and analyzes a flexible and dynamic network control approach based on SDN for meeting the specific communication requirements of both distribution and transmission power grid [44]. Results indicate the advantages of SDN compared to traditional routing and QoS mechanisms, providing a more reliable communication network, which is able to handle complex failures. Pfeiffenberger *et al.* evaluates the use of SDN for reliable communication, especially robust multi-cast between substations by providing one-link fault tolerance with little packet loss [122].

The solutions detailed here up till now mainly covered the challenges and possible solutions for communication links, yet not the services included in a smart grid system. The approaches employing NFV discussed previously focus on other application environments [182, 29] or on the underlying infrastructure, yet not the provided services [39, 38]. The next paragraph covers several papers which are (co)authored by this thesis' author and encompass the research topics of this thesis directly or partially. The papers are discussed in chronological order of their publication; also, the topics and results of each publication influencing this thesis are highlighted.

The challenges of interconnecting the legacy power grid with ICT and communication networks are investigated by Berl *et al.* [16]. The results show that the communication is challenging within smart grid communication networks due to numerous requirements, such as privacy, security, resiliency, and QoS. Particularly, the resilience of communication needs to be considered to maintain the power grid in a stable and controlled state. The paper suggests a Virtualized Energy Information Network (VEIN), where the energy information network is divided into multiple virtual networks that run over a common substrate. The results show that although smart grid related traffic is routed

over the public Internet together with other traffic in this approach, VEIN is able to provide a secure separation between traffic flows. Additionally, VEIN is able to provide a high level of dependability in terms of fault tolerance, as it allows for a transparent aggregation of access and transmission technologies within virtual links.

Apart from security and data protection challenges in the smart grid as a whole, the usage of network virtualization techniques and NFV in AMIs is discussed by Benze *et al.* [14]. Using both approaches, it is on the one hand possible to combine multiple substrate networking technologies into a single logical link, while on the other, using NFV, several services can be hosted on a single substrate hardware device. This can ensure certain QoS criteria on the network side and enables a rapid deployment of virtual smart grid services, offering a highly flexible and scalable overall architecture with superior performability.

Finally, the author discusses the usage of NFV technologies to construct a virtual AMI network to transmit information in a dependable and cost-effective way is illustrated [112]. After the discussion of dependability requirements of AMIs and the shortcomings of current approaches, the reliability and availability of a new architecture based on NFV is analyzed using a Markov Imbeddable Structure approach. Finally, a cost model is developed to compare the novel approach to current AMIs.

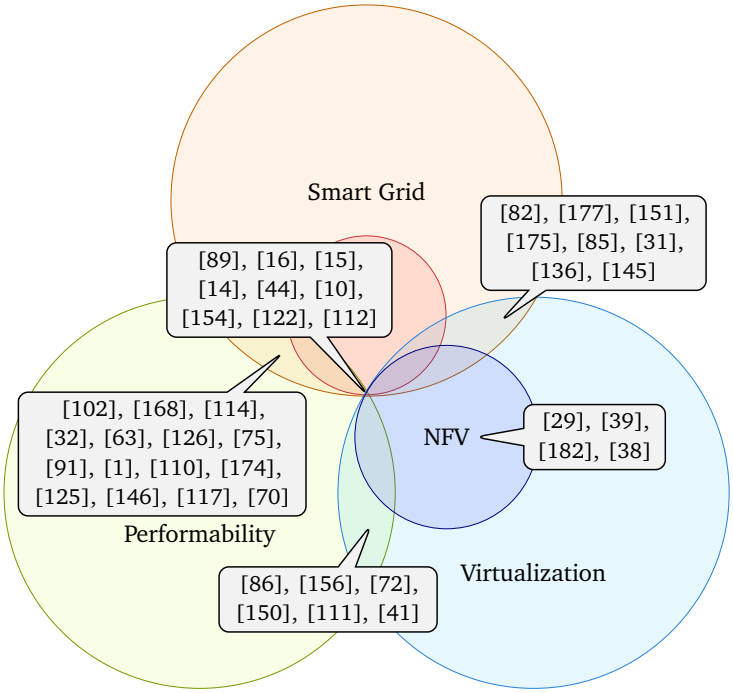


Figure 2.5.: Localization of related work in the research domains covered by this thesis

2.3 Summary

As the previous paragraph shows, there are—besides the own works of the author—currently mainly two general directions that suggest to employ virtualization in critical infrastructures such as the smart grid.

First, cloud-based architectures, as suggested by Xin *et al.* [177], Xiang *et al.* [175], and Sivapragash *et al.* [151] that use the on-demand flexibility and scalability to process the large amounts of smart grid data (mainly AMI related) effectively and efficiently. There are two remarks that need to be

mentioned in relation to these solutions. First, only the service provider's infrastructure benefits from a cloud approach, leaving the consumer's side with performability challenges. While the service provider certainly needs to fulfill high performability requirements, there are usually highly reliable and well-performing servers present at a utility provider's data centers. Also, there are legal issues, which are only mentioned briefly here. As AMI data is classified as highly personally identifiable data, there are legal restrictions on where and how such data may be stored, transferred and processed [176]. This directly contradicts the concept of cloud computing, which is based on (worldwide) distributed processing of data and subcontracting of data center providers from various countries, each following different legal specifications.

Second, there are approaches that suggest the usage of virtual networks in utility communication or virtualizing parts of the utility's hardware infrastructure, e. g., by Yeow *et al.* [179], Lv *et al.* [85], and Aydeger *et al.* [10]. While both research directions tackle existing challenges within critical infrastructures, the main issue is that the focus of all approaches lies on the provider's side of the utility infrastructure. Also, the goal of using virtual networks in AMIs is the preservation of privacy by isolation of different communication flows. To increase the performability of smart utilities by virtualization technologies is however a new research topic that has yet to be investigated in detail. Finally, in Figure 2.5, the related work is aligned with the research topics covered within this thesis.

3 Improvement of AMI Systems using Virtualization

In this chapter the question is discussed whether virtualization technologies are suitable to improve the performability of computing systems—especially AMIs. To do so, first, an overview of the currently proposed AMI architecture is given in Section 3.1.1; second, a preliminary dependability evaluation of the architecture is provided in Section 3.1.2. Thereafter, the dependability implications of virtualization technologies are assessed: In Section 3.2.1, the influence on reliability is highlighted, in Section 3.2.2 the impact on corrective and preventive maintainability is evaluated, and finally, in Section 3.2.3, the availability influence of virtualization is analyzed. After that, two scenarios for virtualization in context of performance/performability are evaluated: First, the possible overhead types induced by virtualization are investigated in Section 3.2.4.1, before the case of "bare" virtualization (Section 3.2.4.2) is discussed, highlighting the negative consequences of virtualization; next, the "applied" virtualization scenario (Section 3.2.4.3), that includes the positive influences of virtualization (i. e. increased availability through redundant service provision, load balancing, etc.). In the end, a summary is given in Section 3.3, weighing the results of the analyses to answer if virtualization is a suitable approach to improve smart grid AMIs.

Contents

3.1 Overview of Current AMI Architecture **40**
3.1.1 Architecture Overview 40
3.1.2 Preliminary Evaluation 40
3.2 Performability Impact of Virtualization **41**
3.2.1 Reliability 42
3.2.2 Maintainability 45
3.2.3 Availability 51
3.2.4 Performance/Performability 52
3.3 Summary **58**

3.1 Overview of Current AMI Architecture

Before a new AMI architecture is introduced in Chapter 4 of this thesis, it is important to first evaluate how the currently planned AMI performs regarding performance and dependability to identify current weaknesses. This is evaluated briefly in Sections 3.1.1 and 3.1.2.

3.1.1 Architecture Overview

The currently proposed AMI (as described by Eckert *et al.* [45]) is depicted in Figure 3.1. It is noteworthy that the figure does not depict an AMI with all its actors in its entirety; yet, the most relevant entities, including those in household, third-party and service provider domain, are included.

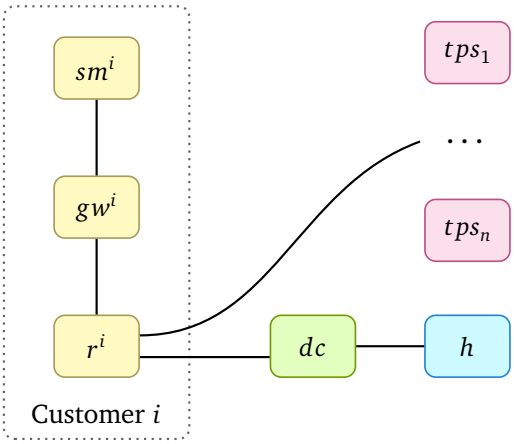


Figure 3.1.: Standard Smart Metering Architecture scheme

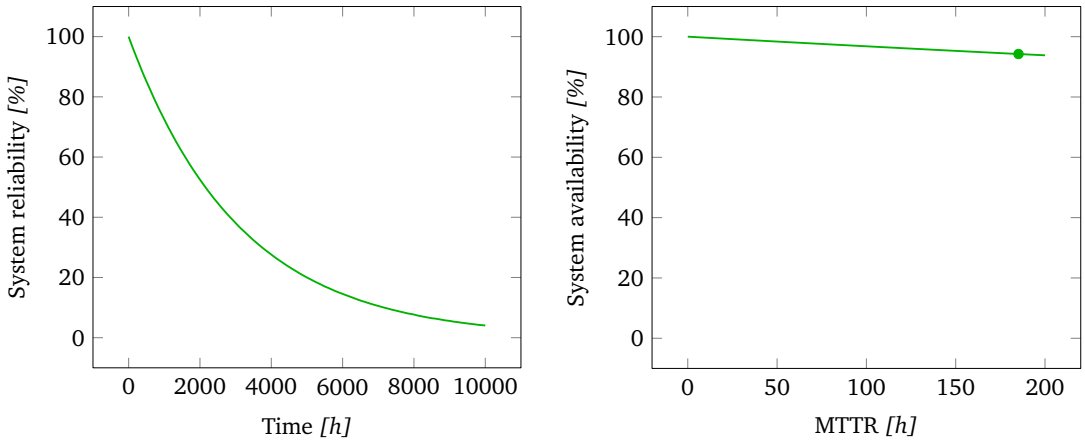
On the left hand of the figure, a common household and its AMI-related devices are depicted: sm^i is the smart meter belonging to consumer i. Similarly, gw^i is the gateway and r^i the router of the respective consumer i. Consumers are connected to a single data concentrator dc located in the household's neighborhood (i. e. its NAN). Beyond that, the WAN including additional services $(tps_1, \ldots, tps_n)$ and the AMI HES (h) is present. This AMI architecture is referred to as SSMA from here on.

3.1.2 Preliminary Evaluation

As the evaluation of SSMA is executed in detail in Chapter 6, within this section, only a brief description of the main challenges of SSMA is given, based on the results from the author's previous work [112]. The results were achieved by using a Markov Imbeddable Structure (MIS) approach to evaluate the reliability and availability of SSMA. For reasons of briefness, only the most important parts of the analysis' methodology are included here, while the focus lies on the achieved results. In the evaluation part, the same assumptions (see Section 6.1.1) as stated for the analysis performed in Chapter 6 apply.

For the SSMA MIS evaluation, first, the system's states need to be defined. As only a binary distinction of the system's $|C|$ components into "functional" (represented by a binary 1) or "failed" (represented by a binary 0) is done, the current state can be visualized by a $|C|$-dimensional binary vector $\mathbf{V}$. It follows that the overall state count is $2^{|C|}$. Next, a probability vector $\mathbf{\Pi_0}$ is defined,

$\mathbf{\Pi_0} = [Pr(Y_0 = \mathbf{V_0}), Pr(Y_0 = \mathbf{V_1}), \ldots, Pr(Y_0 = \mathbf{V_C})]^\top$. The i^{th} entry $Pr(Y_0 = \mathbf{V_i})$ illustrates the probability that the system's initial state is $\mathbf{V_i}$. $\top$ defines a vector transposition. Assuming that the system's initial state is fully functional ($\mathbf{V_0}$), it follows that $\mathbf{\Pi_0} = [1, 0, \ldots, 0]^\top$. The transition probabilities are defined for each component $c \in C$ as a transition matrix $\mathbf{\Lambda_c}$. In each matrix $\mathbf{\Lambda_c}$, the entry p_{ij} states the probability of the system switching from state $\mathbf{V_i}$ to $\mathbf{V_j}$ due to the failure of component c. Last, a $2^{|C|}$-dimensional binary vector $\mathbf{u}$ is defined, whose i^{th} state is 1, if state $\mathbf{V_i}$ is considered functional; and 0, if state $\mathbf{V_i}$ is considered failed. Using these variables, the overall reliability of the system can be expressed as $R = (\mathbf{\Pi_0})^\top \left(\prod_{c=1}^{C} \mathbf{\Lambda_c}\right)\mathbf{u}$. Using the described methodology, the analysis of SSMA yields the results illustrated in Figure 3.2.



(a) Reliability evaluation

(b) Availability evaluation

Figure 3.2.: Reliability and availability of SSMA

Using SSMA, most devices in the communication infrastructure are under end-user control (namely sm^i, gw^i, and r^i), which makes it hard to predict the Mean Time To Repair (MTTR). The reason behind this is that it cannot be assumed that failures in the user's domain are immediately detected, let alone repaired. Depending on the level of technical experience of a user, the time invested into repairing or replacing a failed device can vary greatly. Based on the failure rates given in the author's previous work [112] and Equation (3.2), the MTTR of SSMA is calculated as $\text{MTTR}_{SSMA} = 185.15\,\text{h}$. The availability of SSMA— depicted in Figure 3.2b—is derived using Equation (3.4) and the MTTR. For SSMA, the inherent availability is $A_{SSMA} = \frac{\int_0^\infty R_{SSMA}(t)dt}{\int_0^\infty R_{SSMA}(t)dt + \text{MTTR}_{SSMA}} = 94.250\,\%$.

While only reliability and availability are analyzed within this section, the results show quantitative evidence (given the input parameters from [112]) that the dependability of SSMA requires improvement, which are qualitatively generalizable under certain input parameter assumptions. These findings are also confirmed by Xiang *et al.* [174] and Xu *et al.* [178]. In the following Section 3.2, the utilization of virtualization within AMIs is investigated, which aims to increase their performability.

3.2 Performability Impact of Virtualization

Next, it is investigated if the underlying technological approach used in the proposed solution described in Section 1.2, namely virtualization, is at all an appropriate way to increase the performability of systems in general, and of critical infrastructures, such as AMIs, in particular. Currently, virtualization is applied in a large variety of ICT areas, with a prevalent tendency to be treated as a

panacea. However, at the moment, such one-sided approaches are often made without fully considering the impact on performability. Therefore, in accordance with the definition of performability (Definition 2.5) given in Section 2.1.1.3, the following discussion needs to clarify four questions, namely, if virtualization is able to enhance the:

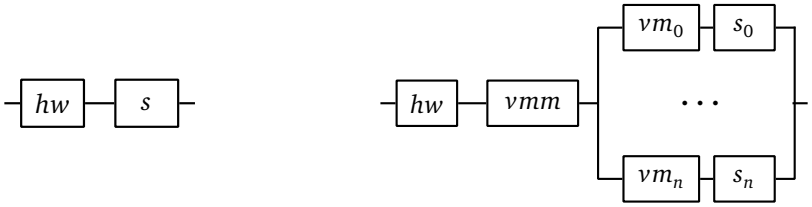
1. reliability (Section 3.2.1),
2. maintainability (Section 3.2.2),
3. availability (Section 3.2.3), and
4. performance (Section 3.2.4),

of systems.

3.2.1 Reliability

The evaluation regarding the influence of virtualization on reliability is based on the methodological approach and findings by Ramasamy & Schunter [127]. To be able to give a meaningful, yet not too lengthy approach, a combinatorial modeling is used in the reliability analysis (as opposed to, e. g., Markov modeling) to compare two scenarios where first, a single node is offering a service without virtualization, and second, with virtualization and several identical services running concurrently serving as backups. This way, it is possible to derive lower bounds on the VMM reliability and the number of VMs required for the virtualized scenario to achieve higher reliability than without virtualization.

In the combinatorial modeling approach used, a system is represented as a RBD. As a simplification, it is assumed that the failures appearing in different parts of the RBD are independent. While such assumptions are possibly not completely true for real environments, it needs to be noted that the reliability results obtained from combinatorial modeling represent upper bounds. Both different scenarios are depicted in Figure 3.3 below; the non-virtualized case in Figure 3.3a, the virtualized case in Figure 3.3b.



(a) Non-virtualized scenario **(b)** Virtualized scenario

Figure 3.3.: RBD models of non-virtualized (*left*) and virtualized (*right*) scenarios

In the first scenario, a single non-virtualized node is analyzed. This node itself contains a substrate (hw) and service part (s) (which includes both operating system and software-based services). Combining this coarse granularity and the previously made assumptions, the non-virtualized scenario can be represented as a simple serial system consisting of its substrate and software part. The reliability is calculated in a straightforward way as $R_{nv} = R^{hw}_{nv} R^{s}_{nv}$, where R_{nv} denotes the system's, R^{hw}_{nv} the substrate's and R^{s}_{nv} the (non-virtualized) service part's reliability, respectively.

In contrast to the non-virtualized case, Figure 3.3b depicts the RBD for a node consisting of a substrate (hw), a VMM (vmm) running in hardware-virtualization mode, and one or more virtual

machines (vm_i), each hosting—as assumed—an identical services (s_i). It is further assumed that the VMs provide identical service in a concurrent and independent manner; also, all virtual machines and identical virtual services are assumed to have the same reliability ($R_v^{vm_0} = R_v^{vm_1} = \ldots = R_v^{vm_n}$, $R_v^{s_0} = R_v^{s_1} = \ldots = R_v^{s_n}$). Furthermore, the reliability of non-virtualized services is expected to be the same as those of virtualized services (i. e., the *contents* of a virtual machine perform as reliably as the service part of a non-virtualized node; $R_{nv}^s = R_v^s$), same holds for the substrate in the non-virtualized and virtualized scenario ($R_{nv}^{hw} = R_v^{hw}$). The combinatorial solution of the resulting RBD can then be calculated by $R_v = R_v^{hw} R_v^{vmm}\big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big)$.

To evaluate the conditions required for a reliability increase by virtualization, the in-equality $R_v > R_{nv}$ needs to be analyzed further, which leads to:

$$R_v^{hw} R_v^{vmm}\big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big) > R_{nv}^{hw} R_{nv}^{s}. \tag{3.1}$$

Applying the previously established assumptions to In-equation (3.1) leads to $R_v^{vmm}\big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big) > R_{nv}^{s}$. As Ramasamy *et al.* also concluded, this directly yields several conclusions [127]. First, if it is assumed that $n = 1$, In-equation (3.1) results in $R_v^{vmm}\big(1 - (1 - R_v^{vm} R^s)\big) > R^s$. Because of that, it follows that under the generally acknowledged assert that reliability measures are < 1 under realistic conditions and the assumption that $n = 1$, a virtualized service cannot achieve the same reliability as a non-virtualized service. If $n > 1$, several other results become apparent. As n increases, the term $\big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big)$ converges, therefore $\lim_{n\to\infty}\big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big) = 1$. This further bears the result that the VMM reliability R_v^{vmm} is a limiting factor in scenarios where multiple redundant identical VMs are present. With these two results in mind, the following corollary is formulated.

Corollary 3.1: Reliability of virtualized systems

Given a fixed substrate reliability R^{hw}, the following conclusions can be drawn:

$$\begin{cases} R_v^{vmm} < R_{nv}^{s} \Rightarrow R_v < R_{nv} \\ R_v^{vmm} = R_{nv}^{s} \Rightarrow \begin{cases} \big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big) < 1 \Rightarrow R_v < R_{nv} \\ \big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big) = 1 \Rightarrow R_v = R_{nv} \end{cases} \\ R_v^{vmm} > R_{nv}^{s} \Rightarrow \exists\, n \in \mathbb{N} : \big(1 - \prod_{i=1}^{n}(1 - R_v^{vm_i} R_v^{s_i})\big) > R_{nv}^{s} \Rightarrow \exists\, n \in \mathbb{N} : R_v > R_{nv}, \end{cases}$$

where n is the number of VMs hosted on the VMM of the virtualized system.

Proof 3.1: Reliability of virtualized systems

The proof of Corollary 3.1 follows from In-equation (3.1).

■

To give an example showcasing the influence of virtualization on the reliability, the following parameters are assumed:

- $R_{nv}^{hw} = R_{v}^{hw} = 0.99$
- $R_{v}^{vm_i} = 0.8$
- $R_{v}^{s_i} = 0.8$
- $R_{nv}^{s} = 0.9$
- $R_{nv} = R_{nv}^{hw} R_{nv}^{s} = 0.891$

In the example, the number of virtual machines required under a certain level of VMM reliability is analyzed to match the level of reliability of a non-virtualized system.

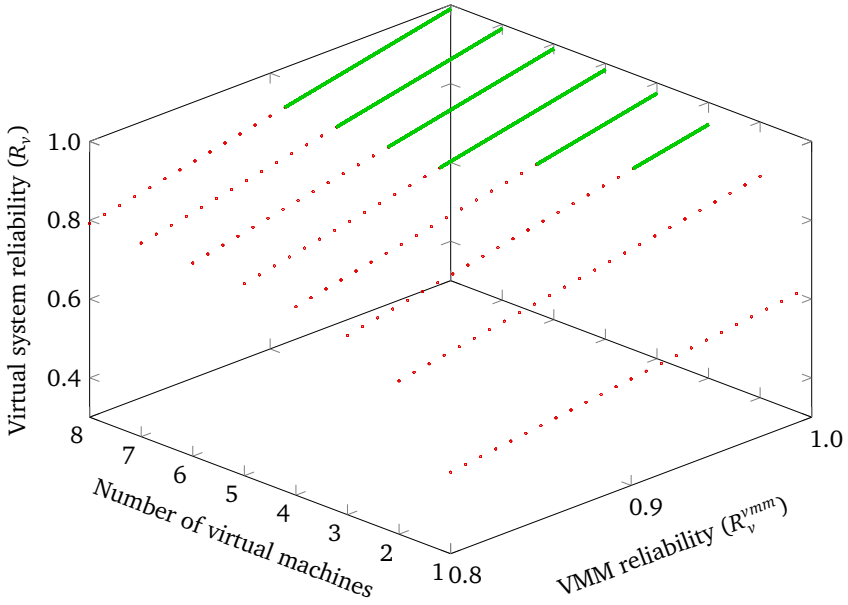


Figure 3.4.: Number of VMs required to surpass non-virtualized system reliability in relation to the VMM reliability

Figure 3.4 depicts the results of In-equation (3.1) for the parameters given above and varying number (1−8) of identical VMs and a VMM reliability in the range of 0.8 to 1.0. The figure also shows the practical application of Corollary 3.1. All parts of the graphs marked as dotted red lines depict reliability values that are lower than the non-virtualized system reliability, those marked in solid green have higher values. It is visible that—in case of a VMM reliability lower than 0.9 ($= R_{nv}^{s}$)—the reliability of a virtualized system (R_v) cannot be as high as that of a non-virtualized one (R_{nv}). In contrast, with R_v^{vmm} values being higher than 0.9, there exists a number of VMs to outperform the non-virtualized system regarding reliability.

Summarizing, it can be concluded that an increase in reliability is achievable using virtualization if certain preconditions are met, among them especially a high VMM reliability. Furthermore, several assumptions were used in the preceding analysis, which might not hold in certain scenarios, e. g., the independence of virtual machines hosted on a single substrate cannot necessarily be guaranteed, as common cause effects originating from either commonly suffered external effects or single failure causes originating from missing diversity may impair several VMs in a similar manner. While such effects require a more in-depth investigation in the analysis (Chapter 6), these possibilities are not analyzed here.

3.2.2 Maintainability

As the availability of a system is mainly dependent on its reliability and maintainability [21], this subsection deals with maintainability, before availability is covered in Section 3.2.3. A restoration process includes several steps required to restore a system's functionality, which not only consists of repairs required after the failure of a system, but also preventive maintenance as well as delays encountered in logistics (e. g., shipping of new hardware) and administration. The types of delays a system may face during its restoration are given in Figure 3.5.

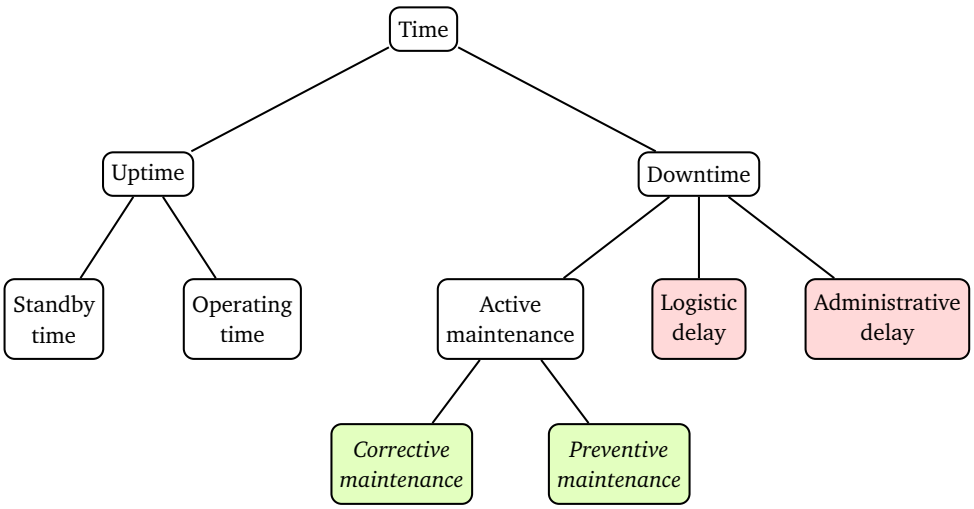


Figure 3.5.: Maintenance time relations (based on Blanchard *et al.* [21])

In the current analysis of maintainability in the context of virtualization, not all aspects are deeply investigated; main subject is the active maintenance, including corrective and preventive maintenance (*green*); both are further discussed in the next two paragraphs. The administrative delay and logistic delay (*red*) are disregarded in the following analysis.

3.2.2.1 Corrective Maintenance

For the assessment of a system's maintainability, the corrective maintenance time (M^{ct}) as well as—for a system-wide assessment—its mean, i. e. the mean corrective maintenance time ($\overline{M}^{ct}$, also known as MTTR) is often used. Equation (3.2) defines the (weighted) mean corrective downtime as:

$$\overline{M}^{ct} = \text{MTTR} = \frac{\sum \lambda_i M^{ct_i}}{\sum \lambda_i}, \tag{3.2}$$

where λ_i is the failure rate and M^{ct_i} is the corrective maintenance time of the ith element of a system. To be able to calculate maintenance times, a further look at the distribution of repair times is required. Usually, corrective maintenance times are characterized by one of the following three distributions [21]:

1. A **normal distribution** is usually a valid assumption for mechanical/electromechanical equipment with a well-defined concept for repair and replacement or relatively straightforward maintenance and repair actions (e. g., simple removal and replacement) which leads to a small amount of variations around the mean.

2. An **exponential distribution** applies to maintenance tasks and maintenance actions whose completion times are independent of previous maintenance experience. This e. g. happens in cases where several possible fixes of similar likelihood have to be exercised subsequently, until the correct solution is found.

3. A **log-normal distribution** approximates the repair behavior of electronic equipment without fixed repair strategy and the assumption that most maintenance and repair tasks comprise of several subsidiary tasks of both unequal frequency and time, which leads to an uneven distribution of repair times.

From the three possible distribution listed above, in the analysis of virtualization benefits in dependability, number 1. and 3. are not applicable due to the specialization of 1. on mechanical devices, which are clearly not the focus in the given scenario that mainly covers electronic devices. Number 3.'s applicability is limited to systems without a fixed repair strategy in place. However, such an assumption contradicts the operation of a critical infrastructure, where well-managed maintenance and repair strategies can be safely assumed. Therefore, an exponential distribution of corrective repair times, as described above in number 2., are assumed in the following. In a scenario with exponentially distributed corrective maintenance times, the maintainability $\big(M(t)\big)$ can be calculated using:

$$M(t) = \int_0^t \overline{\mu} e^{-\overline{\mu}t} dt = 1 - e^{-\overline{\mu}t},$$

where $\overline{\mu}$ is the system's repair rate and t the maximum allowed maintenance time. In the following comparison of non-virtualized and virtualized systems, the weighted $\overline{M}^{ct}$ $\big($as described in Equation (3.2)$\big)$ is going to be used to reflect the changing restoration frequency in the virtualized case. Therefore, to assess the influence of virtualization on a given system, the following two equations are employed to calculate $\overline{M}^{ct}$:

$$\overline{M}_{nv}^{ct} = \frac{\lambda_{nv}^{hw} M_{nv}^{ct_{hw}} + \lambda_{nv}^{s} M_{nv}^{ct_s}}{\lambda_{nv}^{hw} + \lambda_{nv}^{s}}, \text{ and}$$
$$\overline{M}_{v}^{ct} = \frac{\lambda_{v}^{hw} M_{v}^{ct_{hw}} + \lambda_{v}^{vmm} M_{v}^{ct_{vmm}} + \lambda_{v}^{vm} M_{v}^{ct_{vm}} + \lambda_{v}^{s} M_{v}^{ct_s}}{\lambda_{v}^{hw} + \lambda_{v}^{vmm} + \lambda_{v}^{vm} + \lambda_{v}^{s}},$$

where $\overline{M}_{nv}^{ct}$ stands for the mean corrective maintenance time of a non-virtualized and $\overline{M}_{v}^{ct}$ for the mean corrective maintenance time of a virtualized system, respectively. The equations above are however only applicable if a single VM and service is used in the virtualized system. The failure rate of multiple parallel elements (as depicted in Figure 3.3b) can be estimated by:

$$\lambda_{v}^{vm/s} = \frac{(n! \, \overline{\lambda}_{v}^{vm/s})^{q+1}}{(n-q-1)! \, (\overline{\mu}_{v}^{vm/s})^{q}},$$

where n is the overall number of parallel VMs/services in the system and q is the number of parallel VMs/services allowed to fail without causing a system failure (in the given case $n-1$).

In a similar scenario as previously used to showcase the behavior of reliability in a virtualized system, the maintainability is analyzed in the following. The parameters of the example system are assumed as:

- $\lambda_v^{vm} = 5.00\text{E}^{-4}$
- $M_v^{ct_{vm}} = 1.67\text{E}^{-2}\text{h}$
- $M_v^{ct_s} = 8.30\text{E}^{-3}\text{h}$
- $M_{nv}^{ct_s} = 3.34\text{E}^{-2}\text{h}$
- $\lambda_{nv}^{hw} = \lambda_v^{hw} = 1.00\text{E}^{-6}$
- $\lambda_{nv}^{s} = \lambda_s^{v} = 1.00\text{E}^{-3}$
- $M_{nv}^{ct_{hw}} = M_v^{ct_{hw}} = 96\,\text{h}$
- $\overline{M}_{nv}^{ct} = \frac{\lambda_{nv}^{hw} M_{nv}^{ct_{hw}} + \lambda_{nv}^{s} M_{nv}^{ct_s}}{\lambda_{nv}^{hw} \lambda_{nv}^{s}} = 1.2933\text{E}^{-1}\text{h}$

The corrective maintenance time of non-virtualized software components is assumed to require a longer restoration time than their virtualized counterparts, as a software failure may require a complete restart of the machine, while virtualized software can—in such cases—often be restored by restarting their hosting VM, which is also possible via remote access. Also, advanced recovery mechanisms available in virtualized environments, such as checkpointing the application state at the VM- or byte-code level, allow efficient restarting of the saved state [46]. Such technologies additionally decrease the software-sided corrective maintenance time in virtualized systems.

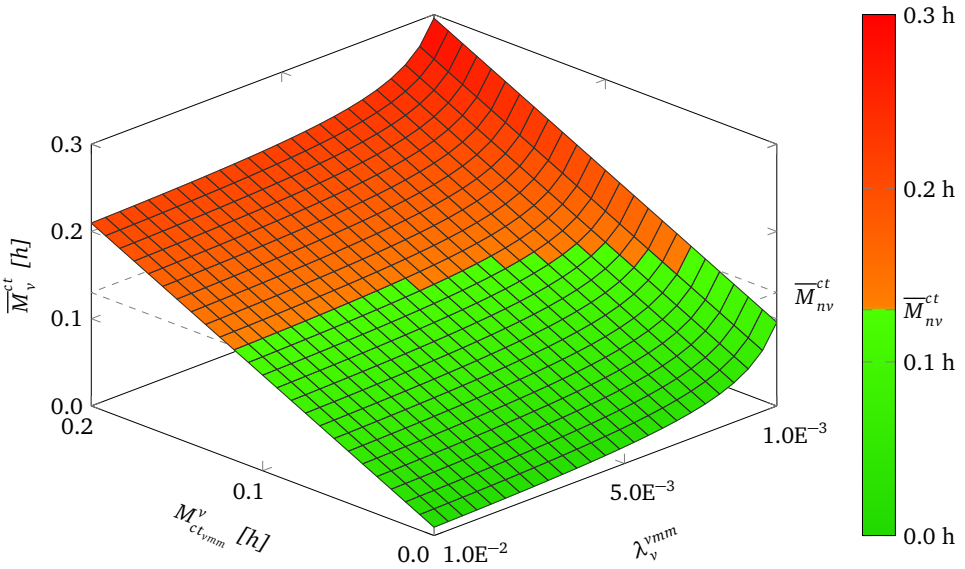


Figure 3.6.: Influence of λ_v^{vmm} and $M_v^{ct_{vmm}}$ on $\overline{M}_v^{ct}$ compared to $\overline{M}_{nv}^{ct}$

In the example, the influence of the VMM failure rate (λ_v^{vmm}) as well as the VMM corrective maintenance time ($M_v^{ct_{vmm}}$) on the mean corrective maintenance time ($\overline{M}_v^{ct}$) is analyzed and compared to a non-virtualized system. In the example scenario, it is assumed that two virtual machines (and their respective offered services) are working in parallel. Figure 3.6 depicts the results of the VMM analysis. All areas colored in green depict scenarios where $\overline{M}_v^{ct} < \overline{M}_{nv}^{ct}$, all areas colored in red depict scenarios where $\overline{M}_v^{ct} \geq \overline{M}_{nv}^{ct}$.

Apart from the properties of the VMM, the number and failure rates of the virtual machines running on a VMM and the respective influence on the $\overline{M}_{ct}^{v}$ was investigated. The resulting behavior is

depicted in Figure 3.7. The logarithmic x-axis represents a "failure rate multiplier" (m_λ), which is a parameter $m \in \mathbb{R}_0^+$ that is multiplied to the original failure rate of the VMs/services. This leads to Equation (3.3).

$$\overline{M}_v^{ct} = \frac{\lambda_v^{hw} M_v^{ct_{hw}} + \lambda_v^{vmm} M_v^{ct_{vmm}} + \frac{(n!\,\overline{\lambda}_v^{vm/s})^{q+1} m_\lambda}{(n-q-1)!\,(\overline{\mu}_v^{vm/s})^q} \overline{M}_v^{ct_{vm/s}}}{\lambda_v^{hw} + \lambda_v^{vmm} + \frac{(n!\,\overline{\lambda}_v^{vm/s})^{q+1} m_\lambda}{(n-q-1)!\,(\overline{\mu}_v^{vm/s})^q}}, \tag{3.3}$$

where m_λ is the failure rate multiplier.

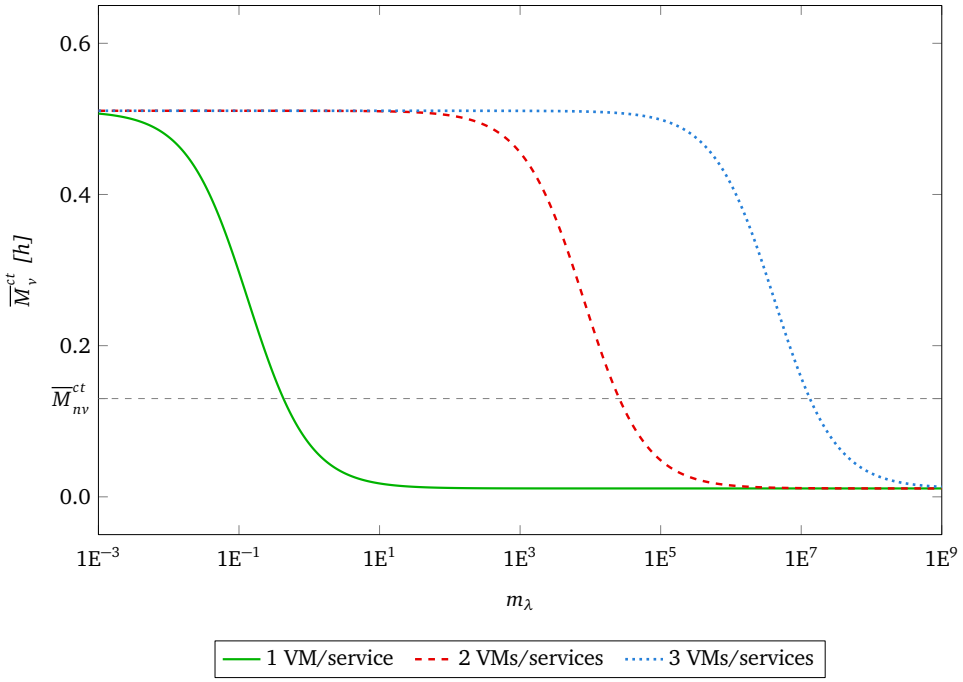


Figure 3.7.: Influence of virtual machine and service failure rate running in parallel on $\overline{M}_v^{ct}$ compared to $\overline{M}_{nv}^{ct}$

The results of all mean corrective maintenance time tests point to a similar behavior: A low failure rate of either the VMM or the VMs/services increases the mean corrective maintenance time; which is due to the relative increase of substrate failures having a longer corrective maintenance time.

In addition to the mean corrective maintenance time, using the relation of $\overline{M}^{ct}$, μ, and $M(t)$ the maintainability of both virtualized and non-virtualized systems is derived as:

$M(t) = 1 - e^{-\frac{t}{\overline{M}^{ct}}}$.

The maintainability results of the scenario are depicted in Figure 3.8. It is assumed here that $\lambda_v^{vmm} = 2.0\text{E}^{-4}$ and $M_v^{ct_{vmm}} = 3.33\text{E}^{-2}$h. In the figure, the solid white line indicates the maintainability of the non-virtualized system, the dashed white line represents a virtual system with a single VM/service, while the dotted white line shows the the maintainability of a virtualized system with two VMs/services running in parallel. The results indicate two main findings:

1. The properties of the VMM have major influence on the mean corrective downtime and the maintainability of a virtualized system. The mean corrective downtime is prolonged by either an increasing VMM corrective maintenance time ($M_v^{ct_{vmm}}$) or by a decreasing failure rate, leading to a relative increase in failure rates associated with longer corrective maintenance times, especially substrate failures.

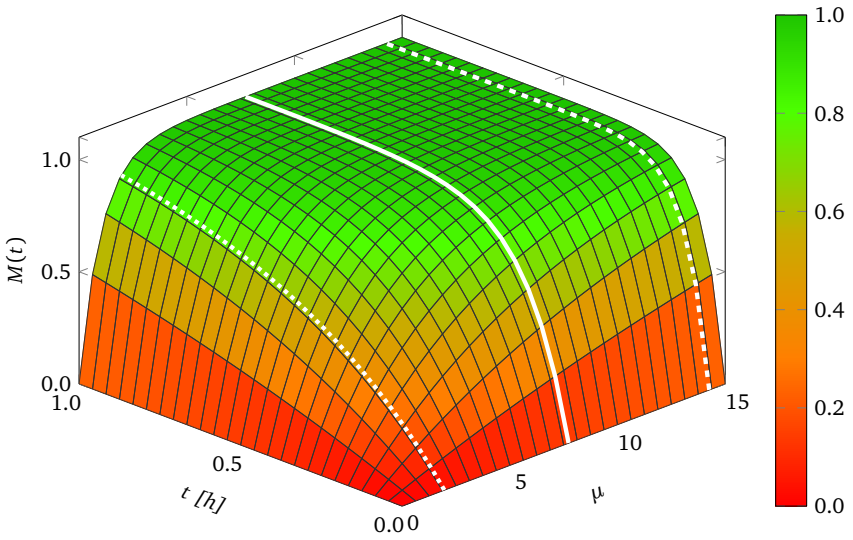


Figure 3.8.: Comparison of virtualized and non-virtualized systems' maintainability

2. The failure rate of VMs and their respective services can be arbitrarily reduced, depending on the number of VMs running in parallel. Despite the fact that this reduces a virtualized system's overall failure rate, the mean corrective maintenance time as well as the maintainability show a degradation with lower VM/service failure rates. This is a result of the relatively dominating substrate failures (having a long corrective maintenance time) in case of highly redundant software. However, it is noted here that a higher mean corrective maintenance time only implies that *in case of a failure*, its repair is likely to require a longer restoration time. If a single VM/service is hosted on a VMM, the same effect manifests in the opposite direction: The mean corrective maintenance time is decreased because of the higher failure rates in the VMM and VM/service parts of the systems leading to a quicker recovery on average.

3.2.2.2 Preventive Maintenance

Apart from the corrective maintenance analyzed in the previous paragraph, the influence of virtualization on the preventive maintenance time is discussed briefly next. Preventive maintenance is mainly used to avoid system deterioration, including both hard- and software. Preventive hardware maintenance is mainly conducted by regular inspections including the computation hardware itself and environmental variables, such as humidity and cooling adjusting systems. In contrast, preventive maintenance on the software side may include updates of software (both applications and the underlying OS) and drivers, as well as rejuvenation to prevent software aging caused, e. g., by integer overflows, data corruption, numerical error accumulation or resource leaks.

Theorem 3.1: Preventive maintenance performability
The usage of preventive maintenance measures does not bear a performability benefit for a system with exponential failure distribution, assuming its services are of non-aging nature.

Especially the latter statement is of special interest in a scenario where exponential failure and repair times are assumed. If a non-aging software is present (i. e., no performance loss after some time), preventive maintenance would not be reasonable, due to the constant failure rate and performance level, leading to Theorem 3.1, which is proven in Proof 3.2.

Proof 3.2: Preventive maintenance performability

Reliability: Imagine a system with $f_{pt} = f_0$ and exponential failure distribution, then

$$R_{npm}(t_0) = e^{-\lambda t_0}$$
$$R_{pm}(t_0) = e^{-\frac{\lambda}{f_0}} e^{-\lambda(t_0 - \frac{1}{f_0})} = e^{-\frac{\lambda}{f_0}} e^{-\lambda t_0 + \frac{\lambda}{f_0}} = e^{-\lambda t_0} = R_{npm}(t_0).$$

R_{pm} represents the reliability of a system with preventive maintenance, R_{npm} without, respectively.

Maintainability: The maintainability measure only regards corrective maintenance time (M^{ct}), preventive maintenance is of no importance. Therefore no further proof is required.

Availability: Depending on the availability metric employed, different effects are encountered due to preventive maintenance.

- Inherent availability: Does not consider preventive maintenance, so no proof is required.
- Achieved availability: The achieved availability is defined as:

  $A_a = \frac{\text{MTBM}}{\text{MTBM}+\overline{M}}$,

  where MTBM is the mean time between maintenance and $\overline{M}$ is the mean active maintenance time, $\overline{M} = \frac{\overline{M}^{ct} + \overline{M}^{pt}}{\#m^{ct} + \#m^{pt}}$; $\#m^{ct}$ is the number of corrective maintenances, $\#m^{pt}$ the number of preventive maintenances performed, respectively. It follows that:

  $$\begin{cases} \overline{M}^{pt} = 0 \Rightarrow A_a = A_i \\ \overline{M}^{pt} > 0 \Rightarrow A_a < A_i, \end{cases}$$

  assuming $\overline{M}^{ct}$, $\#m^{ct}$, and $\#m^{pt}$ being identical in both cases.

Performance: As non-aging services are assumed, the performance level is binary, i. e., a constant value p_c throughout the available phases of the system, and 0 during its unavailable phases. As $A_a \leq A_i$, it instantly follows that $p_{npm} \geq p_{pm}$.

■

While no benefit regarding dependability or performance under the assumption of non-aging software components may be observed, preventive maintenance may be successfully applied if this assumption is loosened. This way, degraded elements of a system may be restored to their initial performance level, as observed in software rejuvenation. Such measures are further discussed in Section 4.4.2.

To measure the preventive maintenance time required by a system, the mean preventive maintenance time is used, defined as

$$\overline{M}^{pt} = \frac{\sum_{i=1}^{n} f_{pt_i} M^{pt_i}}{\sum_{i=1}^{n} f_{pt_i}},$$

where f_{pt_i} is the preventive maintenance frequency and M_{pt_i} is the preventive maintenance time of system element i. Looking at the nature of preventive maintenance tasks, a virtualized system can circumvent most challenges that arise during the maintenance period. All VMs hosting virtual applications can be migrated during (planned) hard- and software maintenance procedures, enabling a through maintenance and testing period before re-migrating the VMs to their original substrate. This is not necessarily possible in non-virtualized systems. Here, tests in a non-productive environment

are used to reduce the expected downtime during a preventive maintenance; however, it is impossible to nullify the downtime if no redundant systems are being used.

3.2.3 Availability

Availability—while not being unrelated to reliability—does in contrast to reliability, which represents the probability of a system to perform its required functions for a desired period of time under specified conditions, include maintenance operations and represents the probability that the system is capable of fulfilling its required function when it is called upon.

This leads to the conclusion that availability is not only related to reliability, but also depends on the maintainability of a system, which are both covered in Sections 3.2.1 and 3.2.2, respectively. There are multiple definitions of availability, depending on what types of maintenance times are considered in the analysis. This results in several different classifications of availability, stated below.

- **Inherent availability:** The inherent availability represents the probability that a system operates satisfactorily at a given point in time under stated conditions *assuming an ideal support environment* (i. e., tools, spares, and personnel are instantly available). It explicitly excludes logistics and administrative delays, as well as preventive maintenance times. However, corrective maintenance times are included. Inherent availability is often derived by analyzing the design of a system and is calculated as

  $$A_i = \frac{\text{MTBF}}{\text{MTBF+MTTR}}. \tag{3.4}$$

- **Achieved availability:** The probability that a system operates with satisfactorily performance at a given point in time under stated conditions in an ideal support environment (again excluding logistics and administrative delays). However, achieved availability comprises—apart from corrective maintenance times—the complete active maintenance, which also includes preventive maintenance times. It is calculated as

  $$A_a = \frac{\text{MTBM}}{\text{MTBM}+\overline{M}},$$

  where $\overline{M}$ is the mean active maintenance time, as defined in Section 3.2.2.

- **Operational availability:** The operational availability is defined as the probability that a system operates satisfactorily at a given point in time when used in a realistic operating and support environment. This means that it includes logistics and administrative delays besides the previously included active maintenance time. It is calculated as

  $$A_o = \frac{\text{MTBM}}{\text{MTBM+MDT}},$$

  where MDT is the mean down time, defined as $\text{MDT} = \frac{\lambda\overline{M}_{ct}+f_{pt}\overline{M}_{pt}+f_{ld}\overline{M}_{ld}}{\lambda+f_{pt}+f_{ld}}$.

As virtualized systems can be safely assumed to have a lower $\overline{M}_{ct}$ and only corrective maintenance times were considered previously, in the following, the inherent availability is used; achieved availability is not considered because of the exponential failure assumption. Considering the previously discussed example scenario, a slight increase in inherent availability is visible, as shown in Table 3.1.

System type	Number of VMs	A_i
Non-virtual	n/a	0.99383
Virtual	1	0.99896
Virtual	2	0.99899

Table 3.1.: Inherent availability of non-virtualized and virtualized systems

Apart from the gains in inherent availability, additional gains are to be expected if other failure distributions are considered. Main reasons behind this are, e. g., that typically, preventive maintenance tasks, such as patch application, involve system restarts, and thus negatively effects service availability. A service hosted in a VM however provides a way to remove faults and vulnerabilities at run-time without affecting its availability, i. e. a copy of the VM is instantiated, the patch is applied, and a restart is performed. After that, the original VM is gracefully shut down and future service requests are redirected to the patched VM. Also, as already briefly described in the maintainability part of the analysis, proactive software rejuvenation can, e. g., be performed easily by rebooting a service or its hosting substrate. The downside of machine reboot is that the service is unavailable during the reboot process. A VMM introduces a convenient layer to proactively rejuvenate a service running inside a VM in a performance- and availability-preserving way. This can be realized by periodically reinstantiating a VM from a clean image. The booting of the reincarnation VM is done while the original VM still continues regular operation, thereby maintaining service availability. As mentioned above in the context of patch application, techniques based on VM checkpointing and live migration may be used to seamlessly transfer network connections and service state of the original VM to the clean VM.

3.2.4 Performance/Performability

Finally, after the dependability attributes have been covered in the previous paragraphs, this section deals with the performance/performability influence virtualization exerts on a system. First, the possible overheads encountered in virtualization are introduced. After that, two general use cases of virtualization are distinguished: First, the influence of virtualization on a single service is measured, to evaluate the overhead induced by the usage of virtualization technology. Second, virtualization is employed and assessed regarding its performance influence in realistic scenarios, such as load balancing and failure mitigation.

3.2.4.1 Virtualization Overheads

The overheads encountered using virtualization technologies are mainly originating from two sources. First, the overhead encountered due to the interference of the VMM acting as a link between the substrate and the VMs; second, the sharing of substrate resources among multiple VMs leads to competing requests for resources, causing additional overhead [157, 13]. The overheads encountered in virtualization are further elaborated next:

- **Central Processing Unit (CPU):** CPU overhead is produced by several actions. For example, whenever the virtual machine needs to access real hardware (causing a world switch from the VMM to the host) or during Input/Output (I/O) interrupts that potentially involve the VMM, the host and guest OS interrupt handlers. Moreover, network transmissions by guest OSs involve two device drivers, and, furthermore, an extra copy of the guest OS's memory is sent to the host OS's kernel buffers on a packet transmit. Additional examples are delivering

virtual Interrupt Requests (IRQs) to a guest operating system, handling Interrupt Return (IRET) instructions, and the Memory Management Unit (MMU) overheads associated with context switches. All these additional actions require computational cycles, causing a CPU overhead.

- **Random Access Memory (RAM):** RAM overhead originates from multiple sources. Examples are the additional time taken to get memory access from inside a VM or the RAM required by the VMM adding to the overall RAM consumption of each VM. Also, there is a static, system-wide memory overhead required by the VMM itself and further overhead introduced by virtualization data structures, such as shadow page tables and reserved memory. The RAM overhead increases with the number of Virtual Central Processing Units (vCPUs) used by a VM, too.

- **Disk I/O:** In order to virtualize an I/O device, a VMM intercepts the I/O operations from the guest OSs. These access requests are performed using privileged instructions, which are trapped and emulated by the VMM. This incurs overhead from world switches between the VMM and the host, and even from the expense of handling the privileged instructions used to communicate with the hardware. I/O overheads are created by virtualization due to additional data transfers of the involved device driver domains. Also, VMs running on a single VMM compete for the available I/O resources from the underlying substrate, leading to a decreased I/O performance.

- **Network:** There is an overhead attributed to the processing of VM network traffic by the VMM layer for network packet processing that performs two important operations: Network Interface Card (NIC) virtualization and virtual switching. Packet sending and receiving both involve these two operations of accessing Virtual Network Interface Cards (vNICs) and going through the virtual switching layer, the cost of which is directly added to the response time. When the guest accesses vNICs, for instance, to send a packet, the VMM intervenes to identify the NIC to communicate to. When a packet is received, the VMM processes that event and notifies the correct vNIC. All the processing done in VMM combined is network related overhead. Moreover, the CPU resources of the VM may be a limiting factor for network traffic. Due to the CPU overheads encountered during network traffic, a system that is naively capable of saturating a network link might instead become CPU bound when run within a VM.

3.2.4.2 Influence of “Bare” Virtualization

Before a performance/performability analysis is performed, it is both easily noticeable and important to understand that if virtualization is simply introduced to a service without leveraging any of its beneficial properties, such as, e. g., co-location of several virtual machines isolated among each other or load-balancing, the overheads (as described in Section 3.2.4.1) decrease the overall performance. Therefore, the next paragraph discusses a “bare” virtualization scenario, meaning a single service is virtualized to measure the influence of the previously mentioned overheads on its performance/performability. To evaluate the overhead present in the scenario, Equation (3.5) based on Li *et al.* is employed [81].

$$O = \frac{|B_m - B_b|}{B_b} \cdot 100\,\%, \tag{3.5}$$

where O represents the overhead; B_m denotes the benchmark result of a service; B_b indicates the baseline benchmark result of the service; $|B_m - B_b|$ represents the corresponding performance loss.

In the given evaluation, the performance of a service running directly on a substrate machine's OS is used as the baseline (i. e. 100 % performance $\widehat{=}$ 0 % overhead).

Given the description of overheads induced by virtualization in Section 3.2.4.1 and the Equation (3.5) above, it directly follows that:

$O_{nv} = \frac{|100\,\% - 100\,\%|}{100\,\%} \cdot 100\,\% = 0\,\%$, and

$O_{v} = \frac{|B_m - 100\,\%|}{100\,\%} \cdot 100\,\% \geq 0\,\%$,

where O_{nv} is the overhead present in a non-virtualized system, while O_v represents the overhead in a virtualized system. Assuming that the overheads introduced by virtualization are $\geq 0\,\%$, this leads to $B_m \leq B_b$. Therefore, $O_v \geq O_{nv}$.

To give a better example regarding the overheads induced by virtualization as well as to quantify them, several publications are available, e. g. by Morabito *et al.* [101], Tong *et al.* [158] and Li *et al.* [81]. Using a similar approach as employed by Morabito *et al.*, the overhead of virtualization is evaluated using benchmarks in the CPU, RAM, disk I/O and networking dimension [101]. Before that, the test system's specifications are given.

Test Environment. In the following, the test setup to evaluate the virtualization overheads is given. The test system used for the performance evaluation has the given specifications:

- **OS:** Windows 7 Professional Edition Service Pack 1 (64-bit)
- **CPU:** Intel Core i7-4770 @ 3.40 GHz
- **RAM:** 16 GB DDR3 SDRAM PC3-12800, 1.5 V, 800.0 MHz, 11-11-11-28
- **Disk:** Lite-On LCS-256L9S-11 256 GB, 512 bytes/sector, NTFS, cluster size 4 kB

For virtualization, VMware Workstation Pro 14.0 is used. The VM used during the tests is assigned the full amount of substrate resources during the tests to offer comparable results. The operating system installed inside the test VM is the same as on the substrate system[7].

CPU. To evaluate the CPU performance, many benchmark tools are available, which differ regarding the usage of multiple CPU cores, the focus on integer and/or floating point operations as well as the inclusion of multimedia content. For the virtualization overhead, two benchmark tools are used: To calculate the integer and floating point performance, the passmark benchmark[8] is used; to evaluate the multi-core efficiency, the y-cruncher[9] tool (a multi-threaded benchmark to calculate the value of Π) is employed, as suggested by Morabito *et al.* [101]. Figure 3.9 shows the results of the benchmark.

As visible, virtualization induces a slight decrease in all CPU tests, indicating a low CPU overhead. The integer computation performance loss due to virtualization is 1.77 %, for floating point computations 0.84 %, for multi-core efficiency 1.88 %.

---

[7] In real-world data centers and COTS servers, a different OS, such as Linux, may be present instead of Windows. While this case is not investigated in this thesis, the results using either of both OSs are similar, which is confirmed by the findings of Younge *et al.* [181].

[8] PassMark Software – PC Benchmark and Test Software, URL: `http://www.passmark.com`, last accessed: 30/01/2018

[9] y-cruncher – A Multi-Threaded Pi Program – NumberWorld, URL: `http://www.numberworld.org/y-cruncher`, last accessed: 30/01/2018

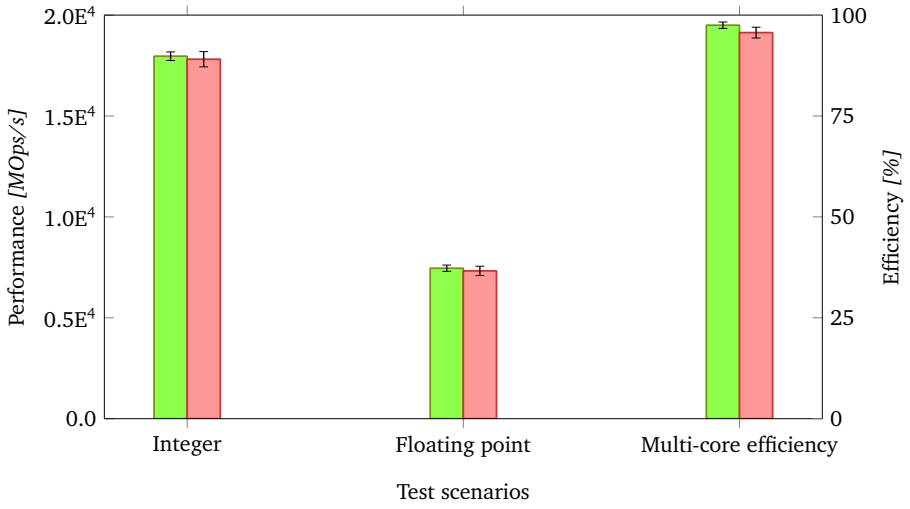


Figure 3.9.: Comparison of CPU performance in substrate and virtualized environment

RAM. In order to test the RAM performance and overhead by virtualization, again, the passmark benchmark is used. The basic RAM operations "read cached" (time taken to read a small block of RAM held entirely in cache), "read uncached" (time taken to read a large block of RAM too large to be held in cache), and "write" (time taken to write information into RAM), as well as its latency (time taken to access RAM) are tested. All tests use a combination of 32-bit and 64-bit data when reading or writing from or to RAM. The benchmark's results are depicted in Figure 3.10.

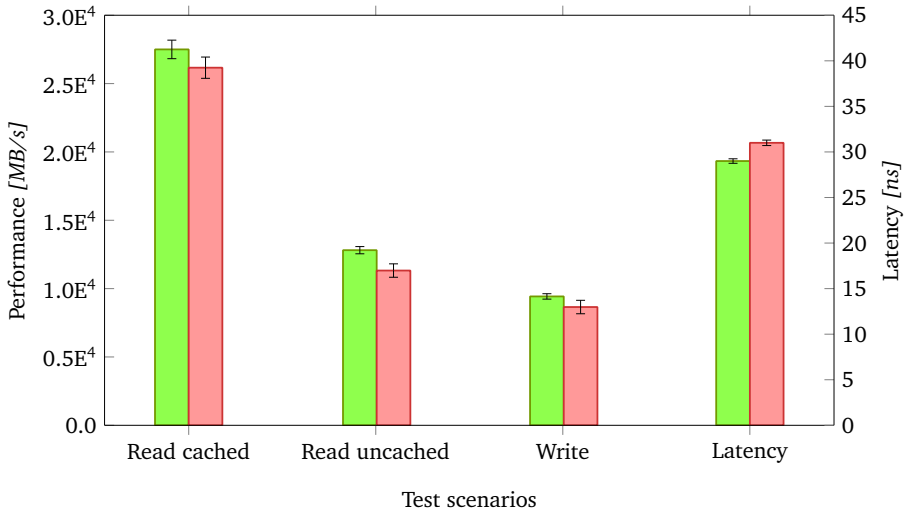


Figure 3.10.: Comparison of RAM performance in substrate and virtualized environment

The overhead induced by virtualization is more severe than in the CPU test; in detail, the performance decreased 4.86 % for "read cached", 11.63 % for "read uncached", and 8.23 % for "write". The latency in a virtualized environment increased by 6.90 % compared to the substrate.

Disk I/O. Similar to the evaluations performed by Valchanov, the performance impact of virtualization on disk I/O is assessed using the Iozone benchmark[10], which allows to test disk I/O with a broad range of file (128 kB to 512 MB) and record sizes (4 kB to 16 MB) [166]. The disk I/O is

[10] Iozone Filesystem Benchmark, URL: `http://www.iozone.org`, last accessed: 31/01/2018

compared using the sequential read and write operations of the disk, which represents the records being written to a new file, and read from an existing file, from beginning to end.

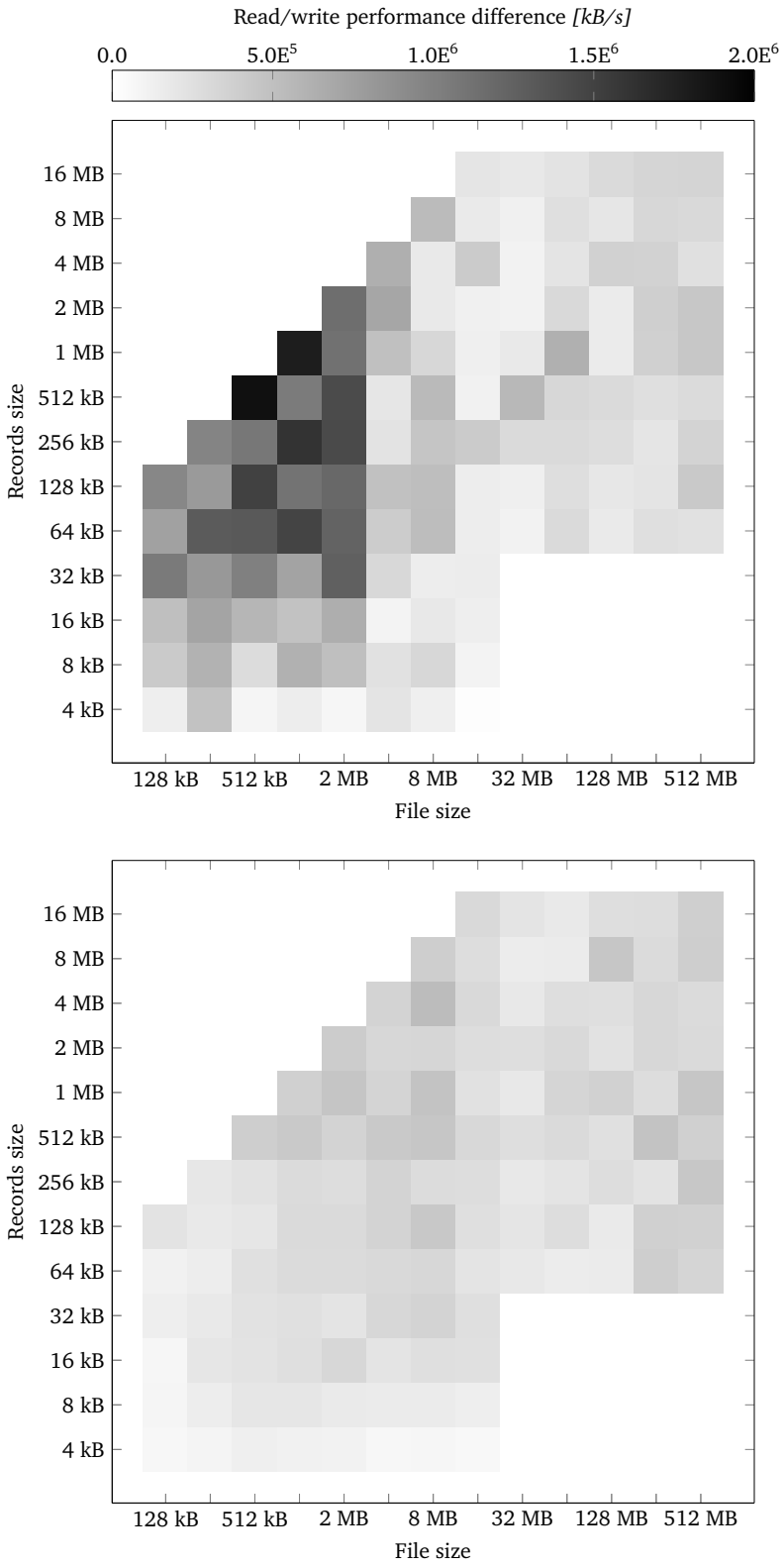


Figure 3.11.: Difference of substrate and virtual read (*upper*) and write (*lower*) I/O performance

The results depicted in Figure 3.11 show that the disk I/O performance in the virtualized environment is again slightly impaired in comparison to the substrate. The performance decrease in the I/O reading scenario is 11.41 % on average and 20.16 % at maximum. For the writing scenario, the loss is 10.48 % on average and 18.92 % at maximum.

Network. To measure the network performance overhead, the application Netperf[11] is used. Netperf offers several predefined tests to measure network performance between two hosts. For the evaluation of the virtualization overhead in networking, the following test setup was used:

- Two identical machines directly connected by 1 Gbps Ethernet link
- One host is running the netperf client (either on substrate OS or virtualized OS), the other the netperf server
- The default configuration for socket and message sizes are used
- Each test duration is 300 seconds
- The netperf tests TCP_STREAM, UDP_STREAM, TCP_RR, and UDP_RR are employed

The STREAM test scenarios measure the throughput of incoming data, while the RR tests are used to measure the request/response time. The test results are depicted in Figure 3.12.

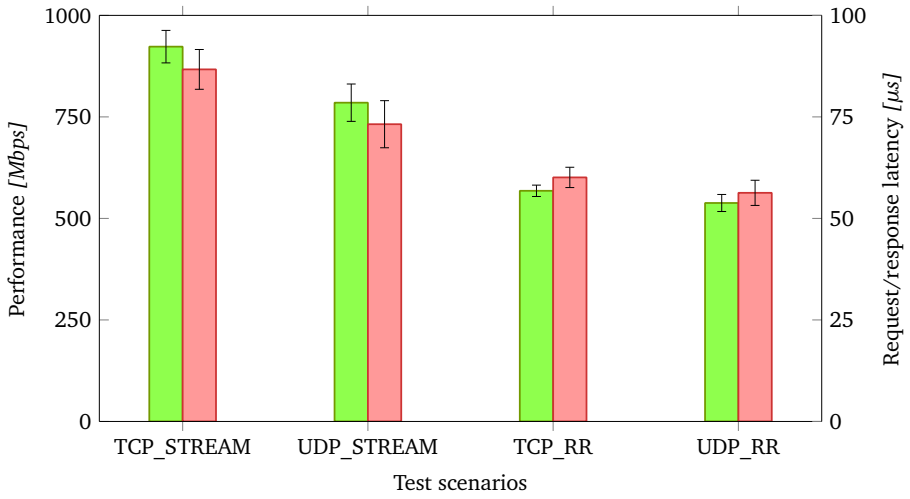


Figure 3.12.: Comparison of network performance in substrate and virtualized environment

The overhead of virtualization causes a performance decrease of 6.46 % in TCP_STREAM and 7.24 % in UDP_STREAM, in the TCP_RR scenario, a latency increase of 5.81 % is observed, in UDP_RR, the increase is 4.65 %.

3.2.4.3 Influence of "Applied" Virtualization

While the previous Section 3.2.4.2 showed that virtualization applied to a single service hosted on a single substrate entity induces overheads, the same is not true for "applied" virtualization, i. e. scenarios where virtualization is used in a proper context, such as load balancing scenarios or hot-spare usage of VMs to mitigate outages in certain geographical areas. In such application

[11] The Netperf Homepage, URL: `http://github.com/HewlettPackard/netperf`, last accessed: 31/01/2018

scenarios, the benefits of virtualization, which have already been covered in Sections 3.2.1 to 3.2.3, must be weighed against the overheads induced by virtualization solutions, as discussed in Section 3.2.4.2.

According to Wolter *et al.*, the main contributors to an improvement in performance are virtualization-based software restoration/rejuvenation (increasing performance) on the one hand, and VM replication techniques (increasing availability) on the other [172]; both are further discussed in Section 4.4. In terms of performance/performability, this has the following consequences:

- As stated in Section 3.2.4.2, the performance (used here as a generic term covering all of the four aforementioned dimensions, i. e. CPU, RAM, disk I/O, and networking) of a single service ($s1$) executed in a virtualized environment is (assuming the virtualization overhead is > 0) smaller than that of the same service $s1$ running directly on substrate $\Rightarrow P_{nv}^{s1} > P_{v}^{s1}$.
- The availability of $s1$ hosted in a virtualized environment is increased compared to a substrate hosting (assuming replicas of $s1$ are used), which is already argued in Section 3.2.3 $\Rightarrow A_{v}^{s1} > A_{nv}^{s1}$.
- Considering the performability measure of steady-state performability calculated as $Perf(hw1) = \sum_{q \in Q} p_q r(q)$ (where $hw1$ is the substrate $s1$ is hosted on), it is easy to see that the contributing factors to the performability measure are the steady-state probabilities of the structural process (p_q) and their associated rewards $\big(r(q)\big)$. Depending on the implementation of the virtualization (number of VMs running in parallel, usage of software rejuvenation, etc.) as well as other parameters (e. g., software aging and utilization), the results may lead to three results:
  1. The availability gain of virtualization outweighs the performance loss, leading to a higher steady-state performability $\Rightarrow \sum_{q_v \in Q_v} p_{q_v} r(q_v) > \sum_{q_{nv} \in Q_{nv}} p_{q_{nv}} r(q_{nv})$.
  2. The availability gain and performance loss are (nearly) equal, leading to a similar steady-state performability $\Rightarrow \sum_{q_v \in Q_v} p_{q_v} r(q_v) \approx \sum_{q_{nv} \in Q^{nv}} p_{q_{nv}} r(q_{nv})$.
  3. The availability gain is lower than the performance loss incurred by virtualization, thereby causing a decreased steady-state performability $\Rightarrow \sum_{q_v \in Q_v} p_{q_v} r(q_v) < \sum_{q_{nv} \in Q_{nv}} p_{q_{nv}} r(q_{nv})$.

3.3 Summary

As this section shows, virtualization offers capabilities to enhance the dependability and performability of a system by employing replication and fast restoration, as Sections 3.2.1, 3.2.2, and 3.2.3 show. More precisely, the reliability can be arbitrarily increased up to a threshold determined by the product of the substrate's and VMM's reliabilities. The expected maintenance times of virtualized services depend on the number of virtual machines running in parallel as well as the VMM's failure and repair rates. While the failure rate is decreasing with the number of VMs running in parallel, the estimated mean corrective maintenance time is increasing, due to the predominance of substrate failures in such scenarios, which tend to have long repair times. The availability, which is directly influenced by reliability and maintainability, is also increased due to lower overall failure and higher repair rates. Lastly, performance and performability investigations show that the introduction of virtualization in a system induces a slight overhead in CPU (average 1.497 %), RAM (average 7.905 %), disk I/O (average 10.945 %), and network performance (average 6.04 %). The achieved performability is defined by the ratio of the performance overhead induced by virtualization and the increase in performability.

4 Creating a Network Function Virtualized AMI

As Chapters 1 and 2 have shown, the current power grid needs to be revised in order to cope with challenges such as a lack of situational awareness in the grid, the increasing integration of renewable power sources, and rising power demand. While the smart grid offers significant improvements, the current AMI has severe drawbacks when it comes to its dependability, as seen in Section 3.1.2. The main reason behind this is a combination of two factors: the inherent complexity of AMIs and non-redundant design principles, which in combination lead to an increased proneness to failures on the communication infrastructure, which can—in turn—lead to challenges in the power grid.

This chapter describes the approach of an AMI architecture based on virtualization technologies. To do so, first, the general idea of virtualized AMI services and its overall implications on the AMI are discussed in Section 4.1. Second, in Section 4.2, it is evaluated if there is a possibility to virtualize AMIs, and, if so, which prerequisites exist so that an AMI service can be virtualized. Also, an evaluation is performed to identify where the novel virtualized AMI services are to be hosted to optimize performability. Third, the methodology for generating, composing and hosting softwarized AMI services is discussed in Section 4.3. The main focus in this section are the customer-specific service compositions as well as their subsequent embedding into the substrate. In Section 4.4, measures on substrate and service level to improve VNFA's performability are discussed. On substrate level, these measures include two server backup strategies allowing the migration of VMs in the case of hardware failures. On service level, software rejuvenation techniques are introduced to mitigate software-aging from negatively impacting the performability. Last, a summary of the chapter is given in Section 4.5.

Contents

4.1 General Idea **60**
4.2 Hardware Abstraction and Centralization **61**
4.2.1 Hardware Requirements and Abstraction 61
4.2.2 Relocation of Hardware 63
4.3 Softwarized Service Generation and Location **66**
4.3.1 Softwarization of AMI Services 66
4.3.2 Virtual Network Function Component Composition 67
4.3.3 Embedding of Virtual Network Function Forwarding Graphs 69
4.4 Introducing Performability-Enhancing Methods **74**
4.4.1 Substrate Enhancements 74
4.4.2 Software Enhancements 76
4.5 Summary **80**

4.1 General Idea

Figure 4.1 depicts both the physical and virtual entities of an AMI concept employing NFV. At the customer's side, there is a single smart meter (sm) located within his premises, which forwards its data via a router (r). On the provider's side of the infrastructure, the NFVI is realized by interconnected server sites, which can, e. g., be located inside of power substations that deliver VNFs to an area with multiple consumers. Each site houses a server that contains a virtualization layer, including a VMM for each server and an arbitrary number of VMs, hosting all required VNFCs. The three building blocks of NFV described in Section 2.1.3.3 are the VNFCs located on top of the virtualization layer forming VNFs (vnf^i), the NFVI including the server, its virtualization layer, and the interconnecting network infrastructure, as well as the AMI HES (h) hosting the NFV-MANO. Each VNF can be individually configured featuring a combination of VNFCs that are either required (such as gateway or aggregation functions) or optional (such as additional energy consumption analysis and optimization functions), if desired. It needs to be mentioned that Figure 4.1 only depicts a simple example of an infrastructure employing NFV. An architecture could be extended to feature an arbitrary number of different VNFCs, which could be chained to create numerous individual VNFs. This does not only lead to a softwarization of the functions previously realized in hardware, but also shifts the responsibilities (especially supervision and maintenance) from the user domain to the provider side, as the VNFs are running on servers located within provider premises [24].

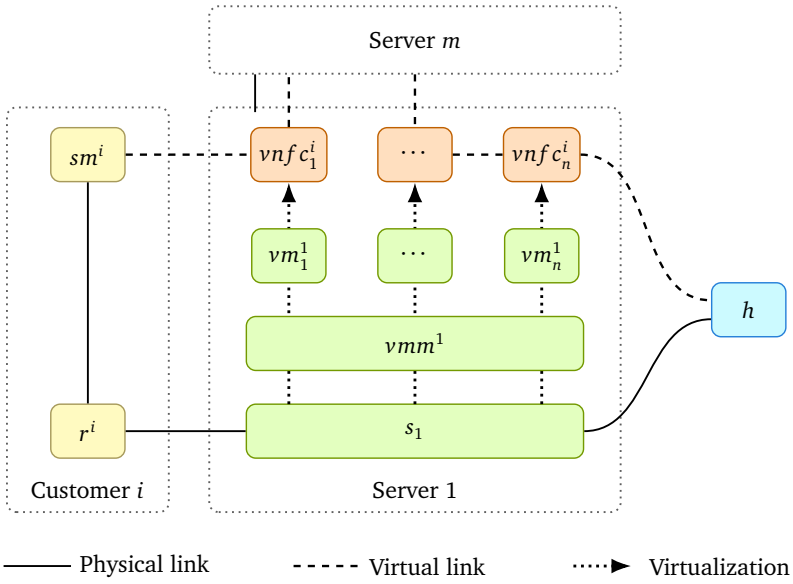


Figure 4.1.: Rough scheme of VNFA

The novel VNFA approach offers several benefits, such as, a) an increased dependability, because of the possibility to realize an adaptive level of service redundancy (through multiple VMs hosting the same service, located at the same or a different substrate); b) an improved adaptability, as services previously realized by proprietary devices are realized in a hardware-agnostic, softwarized manner, making them adaptable to various environments/locations by software alone; c) a decrease in costs, as most services are running detached from their original proprietary substrate in a virtualized environment.

The general idea explained here is investigated further in the following sections, where first, the hardware abstraction and centralization onto off-the-shelf-servers is described; second, the softwarization

of the previously proprietary device-bound services is elaborated on. After describing the details of hard- and software used in the VNFA, its performability-enhancing methods are discussed.

4.2 Hardware Abstraction and Centralization

In this section, two main subjects are covered: First, the prerequisites for the virtualization of previously hardware-specific services are discussed in Section 4.2.1; second, the benefits and downsides of two different location distributions of the required virtualization substrate are evaluated in Section 4.2.2.

4.2.1 Hardware Requirements and Abstraction

Hardware abstraction is required to allow the virtualization of multiple services on a common substrate. However, before a hardware abstraction can be performed, several prerequisites need to be clarified. As the common substrate for the virtualization is a set of COTS servers, several functions required within an AMI (which is a cyber-physical system) cannot be realized by a virtualized service running on an arbitrary substrate. Basically, all services directly interacting with the physical world, in a sense of measuring or actuating, can neither be relocated to a server or virtualized on another substrate; due to the necessity of (a) being in a specific location and/or (b) being equipped with the required sensors/actuators to interact with the physical world. In an AMI system, prime example of such a service is smart metering, which measures the power and energy demand of a costumer. Beyond that, customer-located in-house systems, such as smart thermostats, photovoltaic systems, in-house displays, etc. are also excluded from virtualization approaches due to the aforementioned required interaction with the physical world. Formally speaking, each service $s \in S$ is assigned three parameters that capture its requirements regarding hosting substrate (*type*), location (*loc*), and location tolerance (Δloc), allowing a service to be either:

- agnostic towards *type* and *loc*; making it a good candidate for virtualization,
- bound to a specific *type*, yet agnostic towards its *loc*; allowing a migration between similar substrate entities,
- bound to a specific *loc*, yet agnostic towards *type*; allowing in-place virtualization, yet only limited migration depending on Δloc,
- bound by both *type* and *loc*; only allowing virtualization on the exact substrate the service is running on in a non-virtualized scenario, making virtualization an unsuitable approach.

Looking at the systems present inside of an AMI, as described in Section 2.1.2.2, a classification in respect to the three previously mentioned parameters is done.

- **Smart meter:** A smart metering services is bound to a specific *type* (namely a substrate device offering the capabilities to both measure metering data obtained from a specific household, as well as store and send this data over a network link), as well as *loc* with no Δloc (the metering service needs to be hosted at a specific user's smart meter device).
- **Local user services:** Similar to smart meters, most local user services require a specific substrate *type*, such as, e. g., CLSs or In-Home Displays (IHDs). While it is arguable if, for example, IHDs could run as a virtualized service on any COTS server or Personal Computer (PC), the meaningfulness of such a virtualized service is highly questionable, as many in-browser

solutions or apps exist fulfilling such a purpose already. For *loc* and Δloc, similar restrictions apply as for smart meters.

- **SMGW:** SMGWs basically receive and forward information from or to smart meters. Therefore, no *type* restriction is given, except for being able to forward messages from and to smart meters via a network offering the required QoS and bandwidth. Regarding *loc* and Δloc, mainly the QoS requirements of an SMGW could restrict the placement of a virtualized SMGW service. Further restrictions could arise from the German BSI's requirements regarding security in SMGWs: There are proposals to enforce the usage of a so-called "security module" inside of SMGWs, which is basically a security chip similar to a Trusted Platform Module (TPM). If required, this would restrict the placement of such a service onto a certain substrate *type* (though COTS servers with TPMs are broadly available).
- **Data concentrator:** A data concentrator is a system that accumulates, stores, analyses, and forwards data from multiple smart meters to the AMI HES located within the NAN. It is independent of the substrate *type* as well as (within certain QoS restrictions) the *loc* and Δloc.
- **Third party services:** Usually, there are numerous third party services available within an AMI, ranging from customer services, such as energy management and optimization, price-signal triggered room heating services[12], and customer-site based demand-response systems[13]. To give a generic classification for all third party services is therefore impossible, due to the sheer amount of services available. However, there are two cardinally different third party services: those which are cyber-physically interacting with their environment, thereby being *type*- and mostly also *loc*-bound and low in Δloc, on the other hand, services which operate based on data alone, making them independent of the substrate *type* and *loc* and high in Δloc.
- **HES:** Due to the restriction of this thesis' topic in addition to the special protection needs of the HES and its deep interaction with the provider's infrastructure, it is generally not part of further analyses and considerations.

To improve readability, the classification above is summarized in Table 4.2.

type	*loc*	Δloc	Virtualization	Services
✗	✗	high low	✓ ✓	Non-physical provider field services, third party services (e. g. energy optimization)
✓	✗	high low	✓ ✓	Hardware-assisting services, e. g. TPMs
✗	✓	high low	✓ ✗	Real-time services, voltage monitoring and control systems, provider-side systems
✓	✓	high low	✗ ✗	Smart metering, sensor systems, local user services (e. g. CLSs, IHDs)

Table 4.2.: Virtualization possibilities of AMI services

[12] Nest | Create a Connected Home, URL: `https://nest.com`, last accessed: 12/20/2018

[13] EnerNOC – An Enel Group Company, URL: `https://www.enernoc.com`, last accessed: 12/21/2018

4.2.2 Relocation of Hardware

In contrast to SSMA, the virtualization performed to transform the current AMI into a VNFA offers the possibility to relocate the previously hardware-bound services to several locations (while respecting the restrictions identified in Section 4.2.1) as virtualized services. The relocation of substrate servers has however a non-negligible influence on the VNFA in multiple aspects, among others both of the major topic of this thesis, namely dependability and performance. Finding an appropriate location for entities within a network is a well-known topic discussed in several fields of study, e. g. management, energy-efficiency as well as networking. Examples for distribution approaches to achieve various goals are described by Develder *et al.* [40] and Jaumard *et al.* [69].

As pointed out in Section 4.1, the majority of services provided to customers in the VNFA is hosted on COTS servers. To be able to provide low-delay services and be able to be maintained both on a physical and cyber level, these servers' locations need to be carefully selected. In this section, the placement of the aforementioned servers is considered, pursuing two optimization goals in mind: First, a highly dependable distribution strategy is created; second, a performance-optimized approach is elaborated, before both methods are compared and discussed.

To optimize the locations of COTS servers, first, the assumptions for such a distribution are clarified.

- **Two-dimensional projection:** The location of failures as well as the substrate hardware is projected onto a two-dimensional (x, y) area, ignoring the z coordinate. All interaction between entities is expected to happen on ground level.

- **Fixed and variable locations:** The locations of households as well as their devices are considered fixed throughout this section. In contrast, the locations of servers may be chosen arbitrarily $\big($as long as the location is within the mission area (MA), where $MA = \{(p_1, p_2) \,|\, \text{Location } x, y \text{ is a part of the mission area}\}\big)$. Formally, this is represented by: $\forall\, s \in S : loc(s) \in MA$, where S is a set of all substrate servers, $loc : S \dot\cup \mathfrak{F} \rightarrow MA$, $i \in S \dot\cup \mathfrak{F} \rightarrow (p_1, p_2) \,|\, i$ is located at location (p_1, p_2), the location function.

- **Regarded failures:** Because only the distribution of substrate servers is considered in the following, the types of failures which are relevant in this context is intrinsically limited, too. To give an example, random hardware outages, which do not originate from a location-bound cause, are excluded from this analysis. Instead, only failures ($\mathfrak{F}$) that occur at a certain location (loc) within the mission area (MA) and are bound to a certain radius of influence ($r \in \mathbb{N}_0$). This allows the formalization of the points suffering from a localized outage as $rad : \mathfrak{F} \times r \rightarrow MA,\ \mathfrak{f}, r \rightarrow \{(p_1, p_2) \,|\, dist\big(loc(\mathfrak{f}), (p_1, p_2)\big) \leq r\}$, with $dist : p, q \in MA \rightarrow \mathbb{N}_0,\ (p, q) = |p_1 - q_1| + |p_2 - q_2|$, which is also known as the Manhattan distance. The regarded failures are then defined as $\forall\, \mathfrak{f} \in \mathfrak{F} : loc(\mathfrak{f}) \in MA \wedge rad(\mathfrak{f}) \neq \emptyset$. Examples of such failures are, e. g., power outages or detrimental environmental influences.

- **Failure impact:** All failures in the context of the placement of servers are assumed to cause a complete outage of a server if it is located within the failure's influence radius r, i. e. there is no degradation leading to an impaired, yet working system state, meaning: $\forall\, s \in S, \mathfrak{f} \in \mathfrak{F} : Perf(s) = 0 \,|\, loc(s) \in rad(\mathfrak{f})$.

- **Failure distribution:** The temporal distribution of failures is irrelevant within the context of positioning the substrate servers, therefore no assumptions are made. The failure locations are assumed to be distributed according a uniform distribution within the mission area, i. e.

failures hit each location with the same probability, which is $\forall \mathfrak{f} \in \mathfrak{F}, (p_1, p_2) \in MA : P[loc(\mathfrak{f}) = (p_1, p_2)] = |MA|^{-1}$.

4.2.2.1 Dependability

To increase the dependability of the substrate, the goal is to minimize the impact of failures on servers as far as possible. Due to the possibility of failures being located anywhere within the mission area, it is evident that it is impossible to avoid failures to impair the substrate entirely. To mitigate the impact of localized failures, a solution is to spread the required number of servers over a large distribution area, maximizing their pairwise distances [73]. This is known as the *max-sum diversification* problem.

Formally, problems of this type can be described as a graph $G = (V, E)$ with n vertices, and non-negative edge weights $w(v_1, v_2) = dist(v_1, v_2)$ for $(v_1, v_2) \in E$, where $dist(p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$, which is also known as the Euclidean norm. It is used throughout this thesis, unless specified otherwise. The *max-sum diversification* problem itself is formalized as follows: Given $k \in \{2, \ldots, n\}$, find a subset $S \subseteq V$ with $|S| = k$, such that $w(S) = \sum_{(v_i, v_j) \in E(S)} dist(v_i, v_j)$ is maximized. Being a weighted version of a generalization of the problem of deciding the existence of a k-clique, i. e., a complete subgraph with k vertices, the problem is strongly $\mathcal{NP}$-hard.

However, several heuristics exist providing feasible solutions to the *max-sum diversification* problem, among others described by Ravi *et al.* [131, 132] and Chandra & Halldórsson [30]. A straightforward approach, based on the work of Ravi *et al.* [132], is presented in Algorithm 4.1.

Algorithm 4.1: GMA max-sum heuristic

Input: Substrate network graph G, number of servers to place k
Output: Vertices with max-sum distance S

Function *main*(G, k)**:**
    V ← G.*getVertices*();
    select v_i, v_j ∈ V | ∀ v_i, v_j ∈ V: *getDistance*(v_i, v_j) = max;
    S ← {v_i, v_j};
    while |S| < k **do**
        select v_k ∈ V - S | $\sum_{v_l \in S}$ *getDistance*(v_k, v_l) = max;
        S ← S ∪ {v_k};
    return S;

Another possibility is the construction of the convex hull around all possible locations within the mission area MA(e. g. by using Graham-Scan or incremental algorithm). After that, the m points with maximum distance on the hull are selected. After that, if $k - m > 0$, the remaining $k - m$ points are selected from the interior, each maximizing the distance from the previously selected points. This process can additionally be sped up employing randomized search or simulated annealing heuristics.

4.2.2.2 Performance

The possible performance optimization of the infrastructure through a proper localization of servers is unlikely to be large. Under the assumption that the servers' performance properties are not

dependent on its location and the communication links' underlying technologies and properties do not change during the relocation of servers, the performance remains largely unchanged by the locations of servers. However, as the lengths of the interconnecting communication links changes depending on the server locations, the communication delay may be impacted. To calculate the communication delay, first, the predominant sources of delays in networks are briefly explained here.

1. **Processing delay:** The processing delay in a packet switching network is caused by routing operations which need to process the packet header to determine its destination and check for possible corruption during the transmission. The processing delay induced by state-of-the-art routers are typically in the order of microseconds or less [128], assuming no advanced processing, such as encryption, is performed.

2. **Transmission delay:** The transmission delay characterizes the time it takes to copy a packet into the first buffer and serialize it over the communication link. It is assumed that the packets inside of the AMI are forwarded in a first-come-first-serve manner, which is common practice in packet-switched networks [76]. Further, to estimate the transmission delay, the size of messages as well as the bandwidth of the transmission technology employed needs to be known.

3. **Propagation delay:** The propagation delay describes the time the physical movement of the signal requires and is on the one hand determined by the medium it moves through, on the other by the distance crossed. In wired mediums, such as fiber or copper wire, the speed is estimated to reach 0.59 c to 0.77 c; in wireless mediums, it is assumed that in environments free of obstructions blocking the signals, the propagation speed is c, i. e. the speed of light [76].

4. **Queuing delay:** If packets are arriving faster than they can be processed, a queue in form of a buffer is created. If packets keep arriving faster than the processing speed of the router, their queuing delay also rises. The averagely experienced queuing delay of packets is given by $\frac{1}{(\mu-\lambda)}$, where μ is the service rate and λ is the arrival rate.

As only the location of servers may be chosen, several of the aforementioned delays are unlikely to change if the infrastructure locations are altered. Under the assumptions given in the beginning of this paragraph, the processing, transmission, and queuing delays are of no further relevance. While the propagation delay changes with the distance between the servers and the other networked components inside the VNFA, the impact of these changes and the induced propagation delay are negligible. Even in large cities, this is the case. As an example, the world's ten most populated cities[14] on average each cover approximately an area of 4600 km^2, or, making the simplifying assumption of a square footprint of each city, an area of approximately 68 km × 68 km. Even if the propagation distance is pessimistically considered to be the diagonal of a city's square footprint using a point-to-point link, the propagation delay—in the worst case—is $\frac{96\,167\,\mathrm{m}}{0.59\,\mathrm{c\ m/s}} \mathrel{\widehat{=}} 0.32\,\mathrm{ms}$, which is, considering the maximum acceptable latency given in Table 2.1.2.2, inconsequential.

If the previously made assumptions would be loosened, it would be imaginable that all servers are hosted together in a single data center, interconnecting them with a higher bandwidth networking

[14] The Most Populated Cities of the World. World Megacities – Nations Online Project, URL: `http://www.nationsonline.org/oneworld/bigcities.htm`, last accessed: 10/05/2018

technology than before. While such a scenario is theoretically possible, it is excluded here from further analysis.

4.2.2.3 Comparison and Server Location Distribution Selection

Depending on the optimization goal pursued, the server positioning approach may differ, as previously shown. However, the influence of the servers' locations regarding the achieved performance is mostly negligible—mainly due to the short distances within cities, which renders the slightly increased propagation delays irrelevant in the given context. In contrast, the dependability of VNFA is greatly influenced by the server distribution.

Comparing both highlighted possibilities (max-dispersion vs. centralized server location), it is evident that while the probability of being impaired by a failure is relatively higher in the max-dispersion approach (denoted MDA in the following), the impact of a failure is much lower. This is evident and can be elucidated under the assumption that all servers share the same location in the centralized approach (denoted CA in the following). Then, $\sum_{s \in S} ps = 0 \,|\, \exists\, s \in S, \mathfrak{f} \in \mathfrak{F} : loc(s) \in rad(\mathfrak{f})$, i. e. all servers fail at once if a failure impairs the substrate. In the max-dispersion approach, $\sum_{s \in S} ps > 0 \,|\, \exists s \in S : \forall\, \mathfrak{f} \in \mathfrak{F} : loc(s) \notin rad(\mathfrak{f})$, i. e. as long as a single server is not impaired by failures, a part of the server performance is still available to VNFA.

If we compare the likelihood of a failure impairing the substrate in both approaches, the following equation is used: $\sum_{s \in S} E\big(loc(s) \in rad(\mathfrak{f})\big)$. It is easily visible that in the centralized approach, $\forall\, s_i, s_j \in S : loc(s_i) = loc(s_j)$, while in the max-dispersion approach, the opposite is true, namely $\forall\, s_i, s_j \in S : loc(s_i) \neq loc(s_j)$ (under the assumption that $|MA| \geq |S|$). Therefore, defining $P[\text{MDA}] = P[\text{CA}] = \sum_{s \in S} E\big(loc(s) \in rad(\mathfrak{f}_j)\big)$ as the probability of a failure impairing the servers of the max-dispersion and centralized approach. Considering that in the max-dispersion approach $|\sum_{s \in S} loc(s)| = |S|$, while in the centralized approach $|\sum_{s \in S} loc(s)| = 1$, it is apparent that $P[\text{MDA}] > P[\text{CA}]$.

Consequently, the choice is if a single point of failure in the system is tolerable or not. As VNFA offers strategies to mitigate the impact of few server failures through hard- (circle/all-for-all backup, see Section 4.4.1) and software mitigation strategies (VM warm start and live-migration, see Section 4.4.2), the only reasonable choice which is also pursued in the remainder of this thesis, is the max-dispersion approach.

4.3 Softwarized Service Generation and Location

This section deals with the generation of softwarized AMI services in Section 4.3.1, consisting of the VNFC composition (Section 4.3.2) and the subsequent VNF embedding (Section 4.3.3).

4.3.1 Softwarization of AMI Services

The softwarization of AMI services is a non-trivial matter. A VNF is composed of a number of VNFCs, through which a packet must pass to fulfill the VNF's function(s). The order in which the VNFCs composing a VNF are arranged has a strong influence on both the provided network function itself as well as its non-functional attributes, such as performance and dependability. Apart from that, before the usage of a VNF is possible, it has to be mapped onto a substrate network, which leads to the complex challenge of network embedding. The combination of these tasks can be formulated as a two-step process, being composed of:

1. **VNFC composition:** A VNF is a chain of VNFCs, where the order in which the VNFCs are arranged has to be carefully considered. While the order of VNFC may often be arbitrary (if the overall result of a VNF is similar, irrespective of the VNFCs' order), it may also happen that certain VNFCs require a specific order (e. g. a data concentrator service must not be placed in front of a service requiring individual, fine-granular metering data).

2. **VNF embedding:** After a suitable VNFC composition, the resulting VNF first needs to be mapped onto and second be embedded into the substrate network in an optimal manner. During the mapping process, an allocation of VNFCs onto substrate entities is done. The embedding is performed by allocating the services realized by the VNFCs to VMs and embedding them into the substrate network's entities, which has to respect the resource restrictions given by the substrate as well as the requirements of the VNFCs (being on a single service/entity level) and VNF (being on an overall service/path level). The optimization goals for the VNF embedding can thereby differ, including, but not being limited to optimization of QoS metrics, dependability, or cost minimization.

Both steps are presented in the upcoming Sections 4.3.2 and 4.3.3.

4.3.2 Virtual Network Function Component Composition

To be able to allocate resources for a VNF, first, its VNFCs need to be linked and placed, forming a Virtualized Network Function Forwarding Graph (VNF-FG), which is subsequently embedded into a substrate network. Finding a suitable VNFC composition and an appropriate embedding of its resulting VNF-FG is not a trivial task: While from a service side the goal of maximizing the performance of a VNF is pursued by composing its respective service in an optimal way, the substrate demands an embedding strategy focused on resource conservation, thereby maximizing the amount of embeddable VNFs.

To find the best solution for these two conflicting goals, the order of VNFCs can be exploited to form an optimal VNF-FG. While there may be some dependencies among VNFCs, other VNFCs (e. g. most TPSs, which are functionally independent VNFCs) are flexible. Consequently, the order of VNFCs realizing a VNF is not necessarily deterministic, but can be realized by several different VNFC compositions. To find the most suitable VNFC composition given a Virtualized Network Function Request (VNFR) and a substrate specification is known as the VNFC composition problem.

In Figure 4.2, a representation of a VNFR as suggested by Beck & Botero [12] and Ocampo *et al.* [115] is depicted. Instead of providing the VNF-FG, a client/service user only needs to provide the VNFR information, which in turn allows a service provider to derive an optimal VNFC composition tailored to achieve a predefined goal. Before an individual VNFR of a user is constructed, a generic VNFR is generated containing all *required* services $\{s \in S \mid \sigma^s = 1\}$, where σ^s is a priority flag, further elaborated in Section 5.3.3. To generate a generic VNF-FG from the set of required services, it is necessary to specify five elements:

- The initial bandwidth usage of the network service, which in this case is the data sent by the smart meter (b_{sm}) via the router.
- The VNFCs of the VNF, in this case $VNFC_c = \{s \in S \mid \sigma^s = 1\} = \{c_{gw}, c_{dc}, c_h\}$, i. e. the gateway, data concentrator and HES, along with their respective resource demands (d_c).
- The VNFCs where the VNF starts (c_{gw}) and ends (c_h).

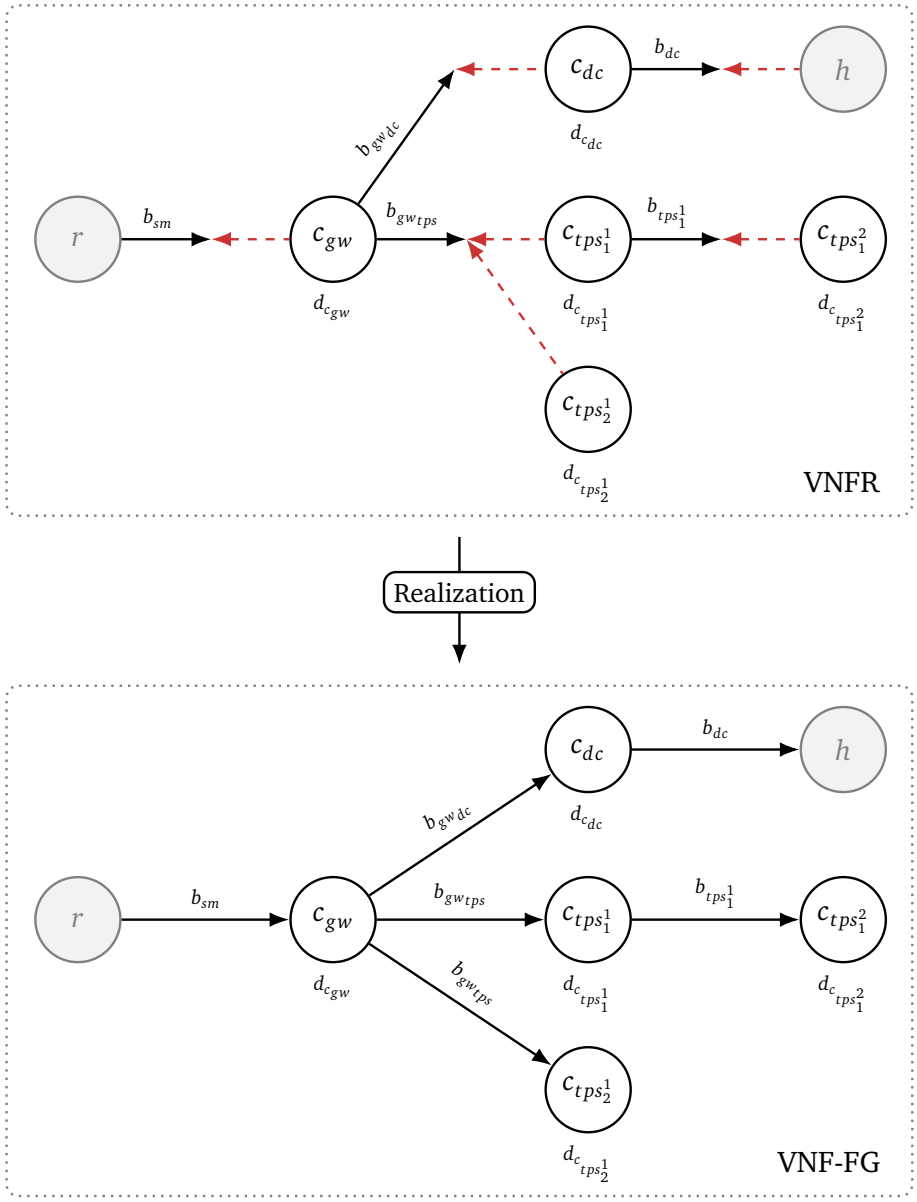


Figure 4.2.: VNFR *(upper part)* featuring required and optional services and corresponding VNF-FG *(lower part)*

- The outgoing links of each *required* VNFC (represented by solid black lines), with their associated relative bandwidths (b).
- Dependencies between different VNFCs (indicated by dashed red lines), which in this case are strict, as there is a precise order given for the required VNFCs: $c_{gw} \Rightarrow c_{dc} \Rightarrow c_h$.

To generate each user's individual VNFR, the generic VNFR is extended to include *optional* TPSs, which are—for each TPS—attached in a separate branch of VNFCs beginning at the gateway of a user. The properties of each VNFR branch is as follows:

- The initial bandwidth is—as before—the data sent by the smart meter (b_{sm}) via the router.
- The *optional* VNFCs of the VNF ($\{s \in S_{tps_i} \mid \sigma_s = 0\}$), where S_{tps_i} are services belonging to TPS i, along with their respective resource demands.

- The VNFCs where the VNF starts (the first VNFC of the TPS $c^1_{tps_i}$) and ends (the last VNFC of the TPS $c^n_{tps_i}$).
- The outgoing links of each VNFC of TPS i, with its associated relative bandwidth.
- Dependencies between different VNFCs of TPS i.

A possible individualized VNFR following the previously introduced scheme is depicted in the upper part of Figure 4.2, featuring several required as well as two optional services. After the VNFRs for all users are created using above methodology, the VNFRs are realized by constructing VNF-FGs fulfilling the VNFRs' requirements. Thereafter, an embedding of the VNF-FGs onto the substrate infrastructure is performed. To do so, the VNF-FG information is input into Algorithm 4.2, which in turn provides the required embedding. This is further described in the following Section 4.3.3.

4.3.3 Embedding of Virtual Network Function Forwarding Graphs

The embedding of the composed VNF-FGs is illustrated by first explaining the general Virtual Network Embedding Problem (VNEP) based on Rost & Schmid [139], before the additional restrictions induced by the requirements of a virtualized AMI are introduced.

To formalize the embedding process, first, the substrate network is represented as an undirected graph $G^{hw} = (V^{hw}, E^{hw})$. The resources in the substrate are given by the function $r^{hw} : V^{hw} \cup E^{hw} \Rightarrow \mathbb{R}^+_0 \cup \{\infty\}$. The capacity $r^{hw}(u)$ of node $u \in V^{hw}$ may represent for example the available CPU resources while the capacity $r^{hw}(u,v)$ of edge $(u,v) \in E^{hw}$ may represent the available bandwidth. By allowing to set substrate capacities to ∞, the capacity constraints on the respective substrate elements can be effectively disabled. We denote by P^{hw} the set of all simple paths in G^{hw}. A VNF-FG is similarly modeled as directed graph $G^{fg} = (V^{fg}, E^{fg})$ together with node and edge demands $d^{fg} : V^{fg} \cup E^{fg} \Rightarrow R \geq 0$.

The task is to find a mapping of the VNF-FG G^{fg} on the substrate network G^{hw}, i. e. a mapping of the VNF-FG's nodes to substrate nodes and a mapping of VNF-FG's edges to paths in the substrate. Virtual nodes and edges can only be mapped on substrate nodes and edges of sufficient capacity. Accordingly, we denote by $V^{hw}_i = \{u \in V^{hw} \mid r^{hw}(u) \geq d^{fg}(i)\}$ the set of substrate nodes supporting the mapping of node $i \in V^{fg}$ and by $E^{hw}_{i,j} = \{(u,v) \in E^{hw} \mid r^{hw}(u,v) \geq d^{fg}(i,j)\}$ the substrate edges supporting the mapping of virtual edge $(i,j) \in E^{fg}$. To solve this challenge known as the VNEP, it is required to be a *valid mapping*, leading to *allocations* forming a *feasible embedding*; all three terms as well as the VNEP itself are defined as follows.

Definition 4.1: Valid Mapping

A valid mapping of request G^{fg} to the substrate G^{hw} is a tuple $m = (m_V, m_E)$ of functions that map nodes and edges, respectively, so that the following holds:
The function $m_V : V^{fg} \Rightarrow V^{hw}$ maps virtual nodes to suitable substrate nodes, such that $m_V(i) \in V^{hw}_i$ holds for $i \in V^{fg}$.
The function $m_E : E^{fg} \Rightarrow P^{hw}$ maps virtual edges $(i,j) \in E^{fg}$ to simple paths in G^{hw} connecting $m_V(i)$ to $m_V(j)$, such that $m_E(i,j) \subseteq E^{hw}_{i,j}$ holds for $(i,j) \in E^{fg}$.

Definition 4.2: Allocations

We denote by $A_m(x) \in \mathbb{R}_0^+$ the resource allocations induced by valid mapping $m = (m_V, m_E)$ on substrate element $x \in G^{hw}$ and define
$A_m(u) = \sum_{i \in V^{fg} : m_V(i) = u} d^{fg}(i)$
$A_m(u,v) = \sum_{(i,j) \in E^{fg} : (u,v) \in m_E(i,j)} d^{fg}(i,j)$
for node $u \in V^{hw}$ and edge $(u,v) \in E^{hw}$, respectively.

Definition 4.3: Feasible Embedding

A mapping m is a feasible embedding, if the allocations do not exceed the available resources, i. e. $A_m(x) \leq r^{hw}(x)$ holds for $x \in G^{hw}$.

Definition 4.4: VNEP

Given is a single VNF-FG G^{fg} to be embedded on the substrate graph G_{hw}. The task is to find any feasible embedding or to decide that no feasible embedding exists.

The defined VNEP does however not account for the the additional restrictions that apply for VNFs used in the context of AMIs, as displayed in Sections 2.1.2.2 and 4.2.1, as well as Table 2.1.2.2. Therefore, to respect these requirements, two additional restrictions need to be included in the problem description, namely:

- **Node placement restrictions:** To account for both the *type* and *loc* restrictions imposed on several services in the AMI, as discussed in Section 4.2.1, the node placement needs to be restricted, which is realized by forbidding the placement of restricted services onto certain substrate nodes. Formally, this restriction is defined as:

  Definition 4.5: Node Placement Restrictions

  For each virtual network function component $c \in V^{fg}$ a set of forbidden substrate nodes $\overline{V}_c^{hw} \subset V^{hw}$ is provided. Accordingly, the set of allowed nodes V_c^{hw} is defined to be $\{u \in V^{hw} \backslash \overline{V}_c^{hw} \mid r^{hw}(u) \geq d^{fg}(c)\}$.

- **Latency restrictions:** Due to the strict timings imposed on several services within an AMI (as listed in Table 2.1.2.2), latency restrictions are used to ensure the VNFs and their associated paths do not lead to an excess of these timing requirements. These restrictions are defined as follows:

  Definition 4.6: Computational Latency

  For each virtual network function component $c \in V^{fg}$, the components' computational latency is given via $l_{cpu}(c) \in \mathbb{R}_0^+$. A network function's latency is the sum of its components' latencies, calculated as $l_{cpu}(vnffg_x) : \sum_{c \in vnffg_x} l_{cpu}(c) \in \mathbb{R}_0^+$.

Definition 4.7: Path Latency

For each substrate edge $e \in E^{hw}$ the edge's latency is given via $l_{net}(e) \in \mathbb{R}_0^+$. The path latency associated with $vnffg_x$ is the sum of the edges' latencies it is embedded into, calculated as $l_{net}(vnffg_x) : \sum_{e \in m_E(i,j)} l_{net}(e) \in \mathbb{R}_0^+$.

Definition 4.8: Latency Restrictions

The latency restrictions for virtual network functions are given by $l_d : vnffg_x \Rightarrow \mathbb{R}_0^+ \cup \{\infty\}$, such that the sum of latencies of both the virtual network function components and the path associated with the respective virtual network function forwarding graph x are no larger than $l_d(vnffg_x)$. Formally, the definition of feasible embeddings (see Definition 4.4) is extended by including that $l_{cpu}(vnffg_x) + l_{net}(vnffg_x) \leq l_d(vnffg_x)$.

Algorithm 4.2 is a realization of an algorithm solving a simplified VNEP for AMIs. More specifically, the algorithm performs an unrestricted initial embedding, i. e. the amount of hosting servers offering the resources can be adjusted arbitrarily. To increase the performance of VNF-FG embedding introduced in Algorithm 4.2, a heuristic can be employed that is shown in Algorithm 4.3. The idea behind the heuristic is to pack each user's VNF-FG completely onto a single server to avoid splitting the chains among multiple servers. This does not only simplify the embedding itself, but in addition avoids communication delays in the overall service chain (see Section 6.2.2.3 for further elaboration). As only whole chains get embedded, the resource requirements of the individual services of the chain can be summed up. Also, the link requirements within the chain are nullified, leaving only the link requirements from *type-/loc*-bound services to the rest of a user's VNF-FG. The simplified embedding therefore does not map individual VNFCs of a VNF-FG, but instead the whole chain of a user onto a single server, which reduces the complexity to a problem similar bin-packing:

Given a set of servers $s_1, s_2, \ldots$ with the same amount of resources r and a list of n VNF-FGs with resource demands $d_1, \ldots, d_n$ to pack, find an integer number of servers S and a S-partition $s_1 \cup \cdots \cup s_S$ of the set $\{1, \ldots, n\}$ such that $\sum_{i \in s_k} d_i \leq r$ for all $k = 1, \ldots, S$.

While the bin-packing problem itself is known to be $\mathcal{NP}$-hard, there are heuristics available here to solve the challenge in a more efficient manner, too. In the given case, a greedy best-fit decreasing heuristic is employed, sorting the VNF-FGs by their resource requirements in decreasing order first and subsequently embedding them onto the server providing enough resources to satisfy the VNF-FG's requirements, i. e. the one that will leave the least resources remaining. Doing so, the complexity decreases to $\mathcal{O}(n \log n)$ while maintaining an asymptotic worst-case performance ratio $R_{BFD}^{\infty} = \frac{11}{9}$. It needs to be noted here that such an approach is only possible in an offline scenario, i. e. during the first embedding. If new VNF-FGs are added during the mission time of VNFA, alternative heuristic solutions exist, such as first fit.

Algorithm 4.2: All-for-all backup algorithm

Input: Set of VNF-FGs vnffgSet, maximum relative server usage maxServerUsage
Output: Initial embedding of VNF-FGs onto a set of servers

```
Function main(vnffgSet, maxServerUsage):
    ServerSet S = ∅;
    while vnffgSet.getNotEmbeddedVnffgs() ≠ ∅ do
        // Select VNF-FG with highest resource requirements first
        embedVnffg(vnffgSet.getNotEmbeddedVnffgs().getBiggestVnffg(), S,
         maxServerUsage);
    // Substrate backup strategy as described in Algorithm 4.4 or 4.5
    applyBackUpStrategy(S);
Function embedVnffg(vnffg, S, maxServerUsage):
    Double vnffgRemainingDelay = vnffg.getMaxAllowedDelay();
    Server lastEmbeddingServer = ∅;
    while vnffg.getNextVnfc() ≠ ∅ do
        // Select starting VNFC first
        vnfc ← vnffg.getNextVnfc();
        if S == ∅ then
            s ← new Server();
            s.spareResources ← (Server.MAX_RESOURCES);
            S.add(s);
        double resourceDistance = Double.MAX;
        foreach s ∈ S do
            if 1 - ((s.spareResources - vnfc.requiredResources) / Server.MAX_RESOURCES)
             <= maxServerUsage then
                if 0 <= s.spareResources - vnfc.requiredResources < resourceDistance
                 then
                    resourceDistance ← s.spareResources - vnfc.requiredResources;
                    embeddingServer ← s;
            if resourceDistance == Double.MAX then
                s ← new Server();
                s.spareResources ← Server.MAX_RESOURCES;
                S.add(s);
            else
                vnffgRemainingDelay −= vnfc.getServiceDelay(embeddingServer) +
                 lastEmbeddingServer.getNetworkDelay(embeddingServer);
                if vnffgRemainingDelay ≥ 0 then
                    embeddingServer.embed(vnfc);
                    embeddingServer.spareResources ← (embeddingServer.spareResources
                     - vnfc.requiredResources);
                    vnfc.isEmbedded ← true;
    vnffg.isEmbedded ← true
```

Algorithm 4.3: VNF-FG embedding heuristic

Input: Set of VNF-FGs vnffgSet, maximum relative server usage maxServerUsage
Output: Initial embedding of VNF-FGs onto a set of servers

```
Function main(vnffgSet, maxServerUsage):
    ServerSet S = ∅;
    while vnffgSet.getNotEmbeddedVnffgs() ≠ ∅ do
        // Select VNF-FG with highest resource requirements first
        embedVnffg(vnffgSet.getNotEmbeddedVnffgs().getBiggestVnffg(), S,
         maxServerUsage);
    // Substrate backup strategy as described in Algorithm 4.4 or 4.5
    applyBackUpStrategy(S);
Function embedVnffg(vnffg, S, maxServerUsage):
    if S == ∅ then
        s ← new Server();
        s.spareResources ← (Server.MAX_RESOURCES);
        S.add(s);
    double resourceDistance = Double.MAX;
    foreach s ∈ S do
        if 1 - ((s.spareResources - vnffg.requiredResources) / Server.MAX_RESOURCES)
         <= maxServerUsage then
            if 0 <= s.spareResources - vnffg.requiredResources < resourceDistance then
                resourceDistance ← s.spareResources - vnffg.requiredResources;
                embeddingServer ← s;
        if resourceDistance == Double.MAX then
            s ← new Server();
            s.spareResources ← Server.MAX_RESOURCES;
            S.add(s);
        else
            embeddingServer.embed(vnffg);
            embeddingServer.spareResources ← (embeddingServer.spareResources -
             vnffg.requiredResources);
            vnffg.isEmbedded ← true;
```

4.4 Introducing Performability-Enhancing Methods

Performability-enhancing methods can be applied on several parts of the system, i. e. the substrate as well as the software part. Both methods used to increase performability are explained in the following, where Section 4.4.1 deals with substrate, while Section 4.4.2 discusses software enhancements.

4.4.1 Substrate Enhancements

To enhance the performability of AMIs, redundancy is a well-known and effective method. However, as already stated in Section 1.2, hardware redundancy needs to be carefully balanced against its costs. In the following, two redundancy schemes on substrate level are introduced which try to offer an adjustable level of redundancy, i. e. the service provider may choose the optimization criteria, being it availability or costs.

4.4.1.1 Circle Backup Network Layout

The backup strategy of the circle backup network is inspired by the author's previous work [112]. For each server there is exactly one other predefined server that offers its spare resources as backup. In this context, backup means offering resources to host virtual services of another server in case it fails. To determine the backup, every server is given a unique ID starting with zero. Let n be the number of server nodes in the network, then the backup of server with ID i is $(i+1) \bmod n$. This results in a circular backup strategy depicted in Figure 4.3.

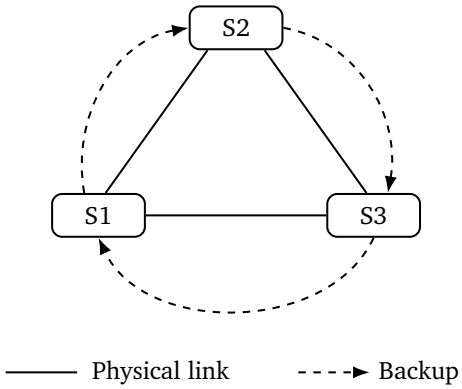


Figure 4.3.: Backup strategy in circle backup networks

In case of a backup node failure (due to either unavailable or missing spare resources), the virtual services of the failed server can no longer be hosted. However, before checking the available resources, existing non-critical services are temporarily removed from the backup server because critical services have priority over optional ones, as described in Section 4.3.2. If the virtual data concentrator service of a household is no longer hosted, the households are already disconnected even if their virtual gateways are still running. Their virtual gateway services can therefore be removed from the servers, freeing resources. After having freed as many resources as possible without deleting critical services of connected households, the circle backup algorithm tries to migrate as many neighborhoods as possible, one after another. First, the virtual data concentrator service is migrated. If this does not succeed, the neighborhood is skipped and no virtual gateways are migrated because the households are already disconnected if the virtual data concentrator service is not available anymore. If the data concentrator service can be migrated, as many of the

virtual gateways as possible are also migrated. The same approach is then applied to the next neighborhood. Last, after having migrated the mandatory services, as many optional services as possible are moved. Formally, this results in Algorithm 4.4.

Algorithm 4.4: Circle backup algorithm

Input: Set of servers S
Output: Circle backuped network of S

```
Function main(ServerSet S):
    init(S);
    while true do
        S_failed ← scanForFailure(S);
        if S_failed ≠ ∅ then
            foreach s ∈ S_failed do
                mitigateFailure(s, false);
        S_failed.empty();
Function init(ServerSet S):
    i ← 0;
    n ← S.getLength();
    foreach s ∈ S do
        s.setId(i);
        i++;
    for int j = 0; j < n; j++ do
        k ← j+1 mod n;
        S.getServerById(j).setBackupServer(S.getServerById(k));
Function mitigateFailure(Server s, boolean mem):
    b ← s.getBackupServer();
    if b.isAvailable() then
        if b.getSpareResources() ≥ s.getServices().getRequiredResources() then
            b.host(s.getServices());
        else if !mem then
            b.releaseLowPriorityServices();
            b.releaseUnboundGateways();
            mitigateFailure(s, true);
        else
            concentrator ← s.getServices().getConcentratorService();
            tryHosting(concentrator);
            foreach hpService ∈ s.getServices().getHighPriorityServices() do
                tryHosting(hpService);
            foreach lpService ∈ s.getServices().getLowPriorityServices() do
                tryHosting(lpService);
Function tryHosting(Server b, Service service):
    if service ≠ null && b.getSpareResources() ≥ service.getRequiredResources() then
        b.host(service);
```

4.4.1.2 All-for-All Backup Network Layout

The all-for-all backup networks are similar to the circle backup networks, the only difference being the employed backup strategy. In contrast to the circle backup, all-for-all backup networks utilize the spare resources of a server as backup for all other servers. This results in a backup strategy as illustrated in Figure 4.4.

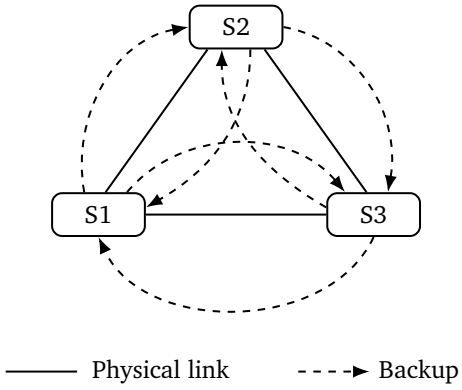


Figure 4.4.: Backup strategy in all-for-all backup networks

Despite being harder to implement because of the increased flexibility of potential backup nodes, there is only one major difference in the restoration process of the all-for-all backup network compared to the circle backup. In contrast to failures in circle backup networks, households that are disconnected can possibly be connected after another server (the backup server in the circle backup network) fails. This happens in the following scenario. Let u_1 be a disconnected user that still has a virtual data concentrator service dc_1 but its virtual gateway node g_1 could not be hosted due to lack of resources. Now another server fails that hosts a virtual data concentrator service dc_2 ($dc_1 \neq dc_2$) and there are not enough spare resources left on any server to host dc_2. Since the virtual gateway services of all users that are connected to dc_2 are also removed, similar to the approach in the circle backup network, the servers now have free resources to host gw_1, which is therefore migrated to the server offering spare resources. Formally, this results in Algorithm 4.5.

4.4.2 Software Enhancements

Generally, software enhancements are used in a wide variety of systems to various aspects of applications, yet foremost dependability and performance [155]. While to satisfy the dependability requirements of certain services, the usage of dependability-enhancing measures such as redundancy in more (e. g. replication via virtualization, N-version programming) or less (e. g. simple load balancing) advanced forms is often a sufficient solution, techniques to increase an application's performability are discussed in the following using service rejuvenation and replication techniques.

4.4.2.1 Software Rejuvenation

The importance of such performance-enhancing techniques has however grown over the last years, and given the ever increasing complexity of software, this trend is not likely to change in the future. To mitigate performance-impairments in applications, several techniques need to be applied in the testing/debugging phase and the operational phase of software. While the testing/debugging phase is out of scope of this thesis, the techniques applicable during the operational phase, i. e. measures to counter the impact of impairments originating from software aging, are going to be covered in the

following. More specifically, a preventive maintenance technique to deal with this challenge called software rejuvenation is going to be introduced.

Algorithm 4.5: All-for-all backup algorithm

Input: Set of servers S
Output: All-for-all backuped network of S

```
Function main(ServerSet S):
    init(S);
    while true do
        S_failed ← scanForFailure(S);
        if S_failed ≠ ∅ then
            foreach s ∈ S_failed do
                mitigateFailure(S, s, 0, s.getServices(), false);
        S_failed.empty();
Function init(ServerSet S):
    i ← 0;
    n ← S.getLength();
    foreach s ∈ S do
        s.setId(i);
        i++;
Function mitigateFailure(ServerSet S, Server s, int offset, ServiceSet services, boolean
 mem):
    b ← S.getServerById(offset);
    if b.isAvailable() then
        if b.getSpareResources() ≥ services.getRequiredResources() then
            b.host(services);
        else if !mem then
            b.releaseNonCriticalServices();
            b.releaseUnboundGatewayServices();
            mitigateFailure(S, s, offset, services, true);
        else
            concentrator ← services.getConcentratorService();
            tryHosting(concentrator);
            foreach hpService ∈ services.getHighPriorityServices() do
                tryHosting(hpService);
            foreach lpService ∈ services.getLowPriorityServices() do
                tryHosting(lpService);
            if services ≠ ∅ && offset < n-1 then
                mitigateFailure(S, s, offset++, services, false);
Function tryHosting(Server b, Service service, ServiceSet services):
    if service ≠ null && b.getSpareResources() ≥ service.getRequiredResources() then
        b.host(service);
        services = services.remove(service);
```

To prevent service degradation, software rejuvenation can be applied proactively as aging leads not only to reduced performance, but also increased failure rates in software systems. Software rejuvenation works by removing accumulated error conditions and freeing up system resources, for example by clearing OS kernel tables, using garbage collection, re-initializing data structures, or—maybe the simplest and most often applied solution—to reboot the system [62]. To develop appropriate rejuvenation strategies, three general dimensions of rejuvenation concepts are defined.

First, the rejuvenation *timing* needs to be set. Here two possibilities can be adopted: on the one hand, time-based techniques, which use a static time-schedule, or, on the other hand, inspection-based techniques, which initiate a rejuvenation after determining certain system parameters.

The *strategy* adopted during the rejuvenation is considered in the second dimension, discriminating between cold, warm and migrate rejuvenation. A cold rejuvenation approach is the most straightforward, not employing any features to decrease the restart time of services after a rejuvenation procedure. It is however important to notice that the restoration times encountered after a rejuvenation event are by far lower than after a failure, as the restoration of the system requires investigation, failure detection and repair, which is extensively covered in Section 5.3.1. In contrast, the system restoration after a deliberately performed rejuvenation action is much faster, as the effort of failure investigation and restoration planning is eliminated. It is noteworthy that both of the other rejuvenation strategies, namely warm and migrate rejuvenation, which are described in the next paragraphs, are only beneficial solutions in the case of VMM impairments, as either service or both the VMs and services maintain their previous states. Warm rejuvenation—in contrast to cold rejuvenation—uses persistent memory to suspend and re-initialize a VM's services after a rejuvenation to decrease the restart time. While a warm rejuvenation requires additional time for service suspension, the service restoration time is decreased in comparison to cold rejuvenation. Migration rejuvenation employs a live-migration strategy to move VMs, which would be impaired by a rejuvenation, to another substrate to minimize their downtime. During the rejuvenation event taking place on the original host, the migrated VM maintains its functionality (assuming the host it migrated to remains operational throughout the rejuvenation event on the original host). The re-migration of a VM can only be performed once it's hosting substrate and VMM are back online after a rejuvenation is performed.

The third dimension of a rejuvenation concept is its target. As highlighted in Section 5.3.1 later on, in the given case, the service application itself, its underlying VM or the VMM can suffer from Aging Related Mandelbugs (AMs). Therefore, the respective targets of the rejuvenation are either the service, the VM, or the VMM. All targets and their impact on the system are described in the following list.

- **Service:** The most basic type of software rejuvenation is the shutdown and restart of a service including all of its components [62]. The assumption behind this technique is that on OS-level, so-called state initialization mechanisms de-allocate all resources of a service once it is shut down, thereby freeing up memory and terminating any file handles and possibly remaining sockets. Furthermore, once the program is restarted, it is once more in a clean slate state, meaning any accumulated AMs are reset. It is important to note that a service rejuvenation does not influence other origins of performance degradation, such as VM, VMM and environmental reasons causing performance issues.
- **VM:** As a second software rejuvenation option, a restart of the VM hosting a service is possible. VMs also face the risks of software aging depending on the software running in the VMs, such as OS, middleware or user applications. A rejuvenation on VM level can be used to clear the

two main reasons for decreased performance, which originate from frequent VM migrations as well as by repeated VM suspend/resume operations [88].

- **VMM:** The final rejuvenation option is a restart of the virtual machine infrastructure, namely its VMM. Basically, before a VMM rejuvenation is executed, it needs to be determined if its VMs are to be re-run from a clean slate (cold rejuvenation), their execution states are to be restored after the VMM's restoration (warm rejuvenation), or—if continuous operation is required—the respective VMs are migrated to another host while the VMM is being rejuvenated, in order to keep them available during the restart process (migrate rejuvenation). Depending on the rejuvenation method chosen, the aging of the service running within a VM hosted on the VMM may also be rejuvenated (cold rejuvenation) or return to its aged state (warm or migrate rejuvenation). It is additionally noted here the the impairment state of a hosted VM also persists if migration rejuvenation is employed; this is not the case in cold or warm migration settings.

The best set of techniques to employ depends on several aspects, such as the storage and migrating performance, the statefulness of the services, the host capacities available, as well as the aging rates of services, VMs and VMMs [87].

4.4.2.2 Service Replication

To provide enhanced performability to system components, a commonly used strategy is to provide redundancy, either hard- or software based; this is also known as *replication*. Here, service replication is going to be discussed, which may be differentiated into two classes:

1. **Active replication:** Using this approach, one or more redundant systems receive copies of all inputs to the parent system and calculate the results of these inputs independently. If required, it is also possible to use the replica systems to detect incorrect behavior of the parent system by comparing its results to the ones generated by each replica (in case of ≥ 2 replicas, a Byzantine algorithm may be used in order to vote on the correct output). In case of a primary system failure, one of the replicas is promoted to parent system status and takes over its functionalities. Since the replica's state is identical to that of the parent system, this transition only requires negligible time. However, this type of replication incurs severe overheads and costs in case hardware is replicated (cf. Section 1.2), as the system costs nearly double without increasing the computational capacity of the system [160].

2. **Passive replication:** Here, a "cold spare" system is used as a replica for the parent. In contrast to active replication, this replica system typically resides in idle state, not receiving the inputs of the parent system. If the parent system fails, the replica first takes over the responsibilities of the parent system; however, this may incur some service interruption, depending on the idle state of the replica. Also, passive replicas are not able to constantly monitor the results of their parent systems due to remaining idle while not being called upon [160].

In the given context of virtualized AMIs, by using replication, it is possible to improve performability (see also Section 3.2.1). However, a certain resource overhead is incurred during the hosting. Depending on where service replicas are hosted, different failures may be mitigated:

- **Replica on different substrate:** Running replicas on a different substrate allows to mitigate all failures on the primary substrate, VMM, VM, and service. However, this might lead to

performance overheads due to additional data transfer between substrate entities, as data sent to the service on the primary hosting site needs to be forwarded to the second hosting site.

- **Replica on same substrate:** As multiple VMMs on the same substrate are not within the scope of this thesis, this case is excluded from further consideration.
- **Replica on same VMM:** Replicas hosted on a separate VM, but the same VMM lead to an avoidance of VM and service level impairments, yet VMM and substrate degradation still impacts such replicas.
- **Replica on same VM:** Running a service replica from within the same VM allows only minimal failure mitigation, i. e. service impairments are avoided.

4.5 Summary

In this chapter, the functional ideas behind VNFA are described in detail after an overview is provided in Section 4.1.

To realize a virtualized AMI system, first, the forced interconnection of substrate and software present in current AMIs needs to be loosened, which is described in Section 4.2. In Section 4.2.1, it is clarified if, and if so in which way, virtualization of AMI related services is possible. This results in the insight that services which are non-interacting with the physical world (i. e. bound to a certain *type* of substrate) as well as not heavily *location*-bound are potential candidates for virtualization. After that, in Section 4.2.2, it is discussed how the newly created virtualized services can be distributed onto a COTS server substrate to achieve optimal dependability and/or performance. Main result here is that a performance optimization would require all servers to be located in close proximity to one another to minimize the communication delays incurred. However, such a location strategy would lead to a single-point-of-failure system structure, thereby leading to a severe dependability degradation.

After the relocation of substrate entities, the virtualized services, their structure and embedding are further elaborated in Section 4.3. In Section 4.3.1, the softwarization of AMI services into forwarding graphs consisting of multiple VNFCs is explained, along with their subsequent embedding into a substrate network. Both steps are further detailed in Sections 4.3.2 and 4.3.3, respectively. The most important findings are that while the composition of VNFCs may be flexible in many scenarios, AMIs tend to have an almost completely fixed structure, providing little to no flexibility regarding the order of their VNFCs. In contrast, the embedding of VNFCs for VNFA users provides many options and optimization possibilities, especially due to the manifold embedding goals, e. g. maximization of acceptance rate, embedding of high priority services first, performance optimizations by embedding complete VNF-FGs of users onto the same server, et cetera.

Finally, in Section 4.4, the possibilities to enhance the performability of VNFA are introduced, which are split into improvements on the one hand on substrate (Section 4.4.1), and on the other hand on service level (Section 4.4.2). On the substrate level, two backup strategies are introduced, namely, the circle and all-for-all backup networks, which are further evaluated in Section 6.2.2.3. On service level, rejuvenation techniques for services, VMs and VMMs are described. The available techniques can be distinguished regarding their trigger (deterministic time- or dynamic performance threshold-based) and—on VMM-level—the behavior towards the hosted VMs: cold, warm or migrate rejuvenation. A further analysis of rejuvenation and its influence on VNFA's performability is given in Section 5.3.4.5.

5 Modeling a Virtualized AMI

The goal of this chapter is the development of an AMI performability model based on its physical and logical infrastructure, as well as the separation of substrate and service layers. In contrast to previous modeling approaches for AMI (e. g., by Xiang *et al.* [174]) focusing to represent a single property of an AMI (which are further elaborated in Section 5.1), this novel model includes multiple features such as the integration of virtualization of services on an arbitrary substrate and the modeling of performability measures.

To exactly determine the model features necessary, its requirements are derived from the shortcomings of previous models discussed in Section 5.1, the idea of the VNFA extensively illustrated in Chapter 4 and the overarching functional and non-functional requirements for AMIs in Section 5.2.

Based on that information, in Section 5.3, the hierarchically arranged parts of a novel AMI model are generated that allow a fine granular definition of hardware and network capabilities as well as a matching of services and their requirements onto the substrate. The overall model, which is the focus of Section 5.4, is made up of two hierarchical layers, namely a macro model (Section 5.3.3), which is used to represent the inter-system interactions of the virtualized AMI system and a micro model (Section 5.3.4), describing the system's behavior on an internal level. This allows a modeling of both substrate and virtualized AMIs and a later analysis of their performability.

The chapter's content is summarized in Section 5.5.

Contents

5.1 Current AMI Models and their Shortcomings **82**
5.2 Model Requirements **83**
5.3 Development of an AMI model **86**
5.3.1 Failure Behavior 86
5.3.2 Recovery Behavior 92
5.3.3 Macro Model (Inter-System) 94
5.3.4 Micro Model (Intra-System) 98
5.4 Overall Model **110**
5.4.1 Idea and Prerequisites 110
5.4.2 Generation of Overall Queuing Network Model 112
5.5 Summary **116**

5.1 Current AMI Models and their Shortcomings

Several approaches in the research domains of smart grids and especially smart metering are covered in Section 2.2. In this section, the focus is set on classifying the currently used models in AMI research and identify their shortcomings, before the actual AMI model of this thesis is described. To classify AMI models, a qualitative approach is used that orders the currently used AMI models according to seven dimensions using the rating none, low, medium, high, very high. The seven dimensions investigated are listed next.

- **Dependability:** The dimension dependability rates whether the model is capable to be used to calculate/estimate a system's reliability, availability or other dependability-related metrics.
- **Performance:** Similar to the dependability dimension, the performance rating is used to grade a model's suitability regarding the calculation of an AMI's performance-related metrics.
- **AMI context:** Here a model's specialization regarding the AMI context is qualified. If a model is perfectly suited to be used in an AMI-context, a high rating is present, if it is a very generic model that requires further adaptation to be usable for AMIs, a low rating is given.
- **Generality:** This dimension grades how generically applicable a model is. It needs to be mentioned that this rating is not the opposite of the "AMI context" dimension: A model that fits the AMI use case very well does not necessarily imply being non-general; vice-versa, "non-general" does not directly imply that the model is well-suited for usage in an AMI-related context.
- **Communication:** Depending on the level that communication is regarded in a model, the qualitative rating in the dimension communication is set. A low rating means that communication is disregarded, focusing on the substrate/hardware, while a high rating implies that a strong focus is set on the communication in between substrate nodes.
- **Specific model:** This dimension grades how specific a model is formalized. If only a vague modeling scheme is given, a low rating is given; in contrast, a highly formalized model leads to a high rating.
- **Virtualization:** In this dimension, a higher grading implies that virtualization is supported by the model in one or multiple ways (e. g. NFV, SDN, or other types of system or network virtualization).

Using the scheme described above to classify the currently employed AMI models, it becomes apparent that recent AMI models are either:

- **Abstract, non-quantitative models:** In this case only conceptual models are used, with a mainly explanatory intention. A prominent example is the Smart Grid Architecture Model (SGAM), which can be used to classify affected smart grid domains and zones if technological changes are introduced into AMIs (e. g. by Uslar *et al.* [165]). Apart from that, the NIST also provides a smart grid model that offers a well-structured overview of the smart grid as well as AMI domains and involved systems [108]; yet, they are not suitable for further analysis, due to missing evaluation possibilities and lack of employable quantitative metrics.

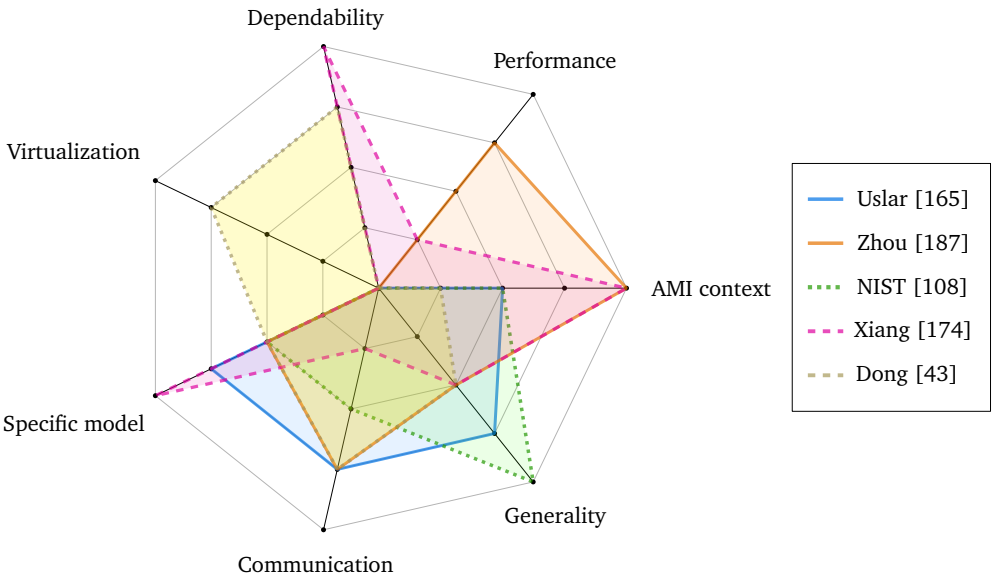


Figure 5.1.: Classification of currently used AMI models

- **Highly specialized, quantitative models:** Here, the modeling approach taken is highly focused on the specific domain and goal; typical models used include RBDs and SPNs for reliability/dependability analyses (e. g. by Xiang *et al.* [174]) and network graphs for topology/architectural analyses (e. g. by Zhou *et al.* [187]). The shortcomings in these models are often that they are either not specifically made for/adapted to the usage in AMI environments, leading to gaps within the approach, which is e. g. encountered in Dong *et al.*'s work [43], where SDN-based virtualization approaches in smart grid communications are presented which however fail to include details regarding AMIs as well as performance considerations. Similar shortcomings are present in Xiang *et al.*'s work [174], where the communication in AMI environments is not considered explicitly; instead, only substrate entities (i. e. smart meters) are used to predict the whole infrastructure's dependability.

A classification of the models mentioned in the previous paragraphs is depicted in Figure 5.1. For readability reasons, only five representative models are included in the figure. While there are more AMI-related models available, the resulting classification is similar as described before. Therefore, the requirements for an AMI model which supports both virtualization and performability evaluation are discussed in the upcoming Section 5.2.

5.2 Model Requirements

A suitable performability model for a NFV-based AMI requires a combination of several approaches that were previously not researched at all or only treated in isolation in AMI scenarios, namely virtual service embedding, dependability, as well as performance. The requirements to combine these three features are explained in the following.

The abstraction of network functions from their respective substrate, separating the previously merged substrate and service planes is enabled through the introduction of additional VMM and VM planes. The result of the mentioned separation of substrate and service planes is that the (previously fixed) static substrate can now be exchanged, leading to a dynamic and flexible indirect

interconnection through the VMM and VM planes, due to possible migration of services onto changing substrate entities. This leads to the requirement of a **dynamic mapping** between the network functions and links located within the service layer and the entities of the VM, VMM, and finally substrate layer. In addition to that, the model needs to be able to represent both bidirectional links (present at the substrate plane) as well as unidirectional links (at the service plane). The unidirectional links are required to give a fixed order to the services (VNFCs) forming a user's VNF, which cannot be arbitrary. Therefore, it is required that the model supports **unidirectional** links on the service and **bidirectional** links on the substrate layer, respectively. While this would be enough to represent a generic mapping of services, the model has to further respect several additional **requirements and limitations given by all layers** during their mapping. As the details of the considered values are given in Section 5.3.3.1, only several examples are briefly mentioned here. The substrate entities (comprising both nodes and links) are limited regarding their resources, which are given as CPU, bandwidth, et cetera. During a mapping, these limitations need to be considered and reconciled with the prerequisites given by the services, which each require a certain amount of resources. Therefore, a model is required to allow the **assignment of resource/requirement properties** to all entities.

To later be able to calculate the performability of the NFV AMI, several requirements emerge. The **granularity of the model needs to allow a performability estimation of all entities' states**. This includes both the nodes and links and the VNFCs themselves, their hosting VMs, as well as their respective VMMs. Main inputs are the **failure, aging, and repair rates associated with both services and substrate entities**, which have a significant impact on their performability. Therefore a proper representation of this property needs to be ensured. In addition, the model shall support both the calculation of **steady-state and transient measures**, to allow both general and time-dependent statements regarding the behavior of VNFA.

• Dynamic mapping	⇒	Separate model planes for services, virtualization and substrate entities that change over time: *dynamic* model
• Uni- and bidirectional links • Failure and repair rates on substrate, virtualization and service levels • Mapping requirements and limitations • Assign resource/requirement properties	⇒	Model has multiple, arbitrary properties assigned, allowing *explicit* modeling
• Performability estimation for substrate and/or services	⇒	Model allows both individual and combined plane calculations
• Steady-state and transient dependability and performance measures	⇒	Model needs to offer *quantitative* solutions
• Performability estimation for VNFs based on different VNFC properties	⇒	An *optimization* model is used

Table 5.2.: Requirements of the novel AMI model

To be able to estimate the overall AMI performability, a modeling of the **performability of each VNFC and their respective interconnections** is required, leading to an exegesis of performance into both **computational and network performance**. Both measures need to be modeled **respecting the influences of the service and substrate plane** (e. g., a virtual link could artificially restrict the

bandwidth of a substrate link). The overall model also needs to be able to calculate both **steady-state and transient performance measures**.

Looking at these prerequisites for a model capable of handling NFV AMIs, the model requirements are derived in Table 5.2.

It is apparent that the requirements identified in this section cannot be fulfilled by a single modeling mechanism. Based on the requirements, however, a modeling strategy can be selected. As a **quantitative evaluation** is required, either **measurements** or **abstract models** are possible options. Measurements are impossible to realize due to two reasons; first, the suggested VNFA does not yet exist; second, a construction of such a large scale installation is beyond the capabilities of the author. Therefore, abstract models are constructed for evaluation purposes, more precisely, both **discrete-event simulation** and **analytic approaches** are employed.

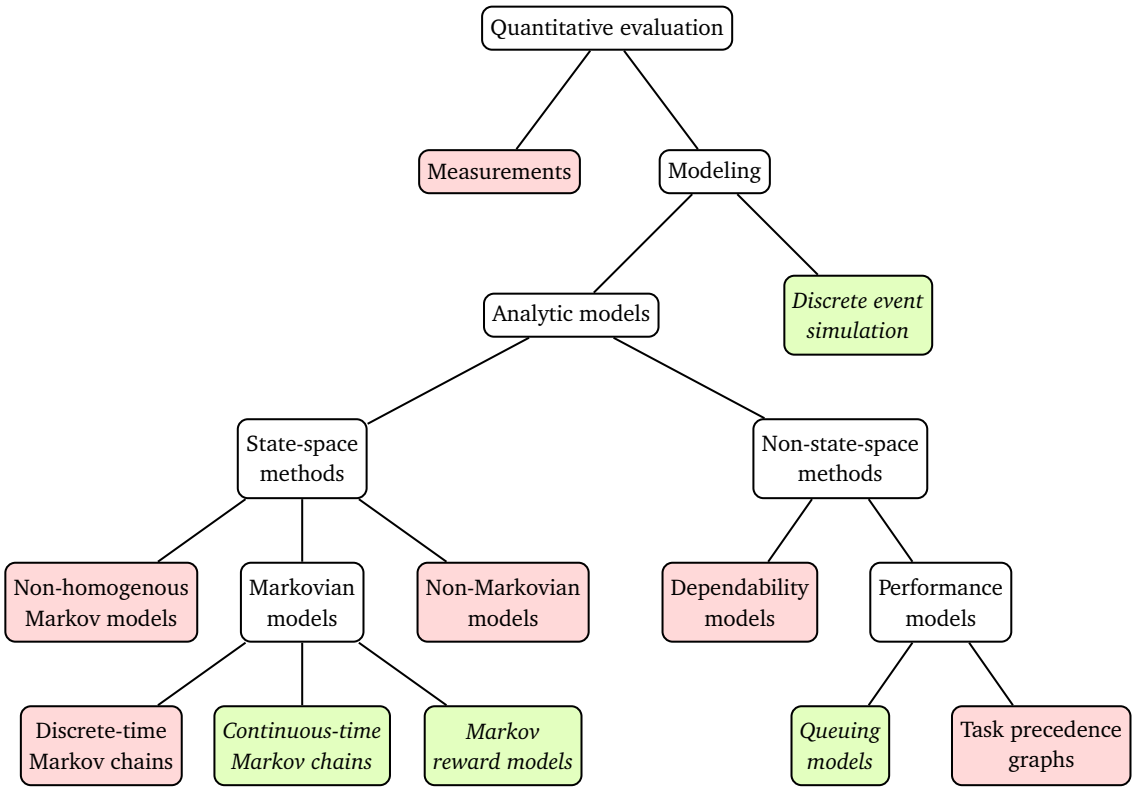


Figure 5.2.: Quantitative evaluation approaches (based on Trivedi [162])

The discrete-event simulation is required to evaluate scenarios where non-exponential transition rates are present in models, e. g. if deterministic and other non-exponential transition rates are required. While certain approaches exist to solve also non-exponential transition rates, such as Deterministic and Stochastic Petri Nets (DSPNs) for deterministic distributions, they are often limited, e. g. they only provide steady-state solutions. Therefore, discrete-event simulation is preferred in the aforementioned scenarios [84].

When analytic models are employed, both **state-space methods** and **non-state-space methods** are used: non-state-space methods in the macro model (representing the inter-system behavior, see Section 5.3.3); state-space methods in the micro model (representing the intra-system behavior, see Section 5.3.4), due to the manageable state-space size.

Within the domain of state-space methods, there are **non-Markovian**, **non-homogeneous Markovian**, and **Markovian models**. While non-homogeneous Markov models (i. e. semi-Markov and Markov regenerative processes) and non-Markovian models offer support for general and Weibull distributed event/failure distributions, due to the failure and repair rates (which are explained in Sections 5.3.1 and 5.3.2) Markovian models are chosen, more precisely **continuous-time Markov chains** (as well as their $\mathcal{F}_1$ models, i. e. Generalized Stochastic Petri Nets (GSPNs) [60]) and **Markov reward models**.

For **non-state-space methods**, both **dependability models** and **performance models** are available; however, the dependability models (i. e. RBDs and fault trees), while being easy to use, assume a statistical independence that is not given in the systems analyzed within this thesis. In contrast, the **performance models** are able to handle failure/repair dependencies, such as non-zero switching times or travel time of repair persons. Therefore, for performance assessments of the overall model, **queuing models** are employed.

An overview of the selected modeling formalisms is given in Figure 5.2.

5.3 Development of an AMI model

Before the macro and micro models are developed in Sections 5.3.3 and 5.3.4, respectively, several prerequisites need to be clarified. In Section 5.3.1 the failure behavior of hard- and software, as well as the resulting stochastic distributions assumed in the subsequently developed models are derived. Similarly, in Section 5.3.2 the properties of the recovery mechanisms are investigated.

5.3.1 Failure Behavior

In the following, an overview regarding the types of failures present in ICT systems as well as their importance to the overall failures of a device is given in Section 5.3.1.1. After that, the difference in failure behavior of hard- and software is discussed in Section 5.3.1.2.

5.3.1.1 Root Cause Breakdown

Before the failure and recovery behaviors of ICT systems are discussed, an obvious question when analyzing computer systems is what is the cause that triggers a system failure. Based on an analysis by Schroeder & Gibson the causes for a system outage are classified into six different origins: Hardware, software, human, environmental, network, and unknown [144]. Figure 5.3 depicts the results, where it is clearly visible that hardware failures are by far the biggest contributor to ICT system failures (60 %), followed by software (18 %), unknown (12 %), environmental (5 %), network (3 %) and human (2 %) failures.

It is further noteworthy that there is no significant change regarding the root cause breakdown over the lifetime of a system (i. e., as time progresses from initial deployment to later times of operation). According to Schroeder & Gibson, the main change is that in some systems the percentage of failures with unknown origin drops over time [144]; the relative frequency of the other types of failures however remains relatively unchanged.

As this brief analysis shows, the main contributors to the overall failures in ICT devices are hard- and software failures, accounting for over 75 % of outages. Therefore, in the following modeling and analyses, the regarded failures are restricted to both of the aforementioned failure types.

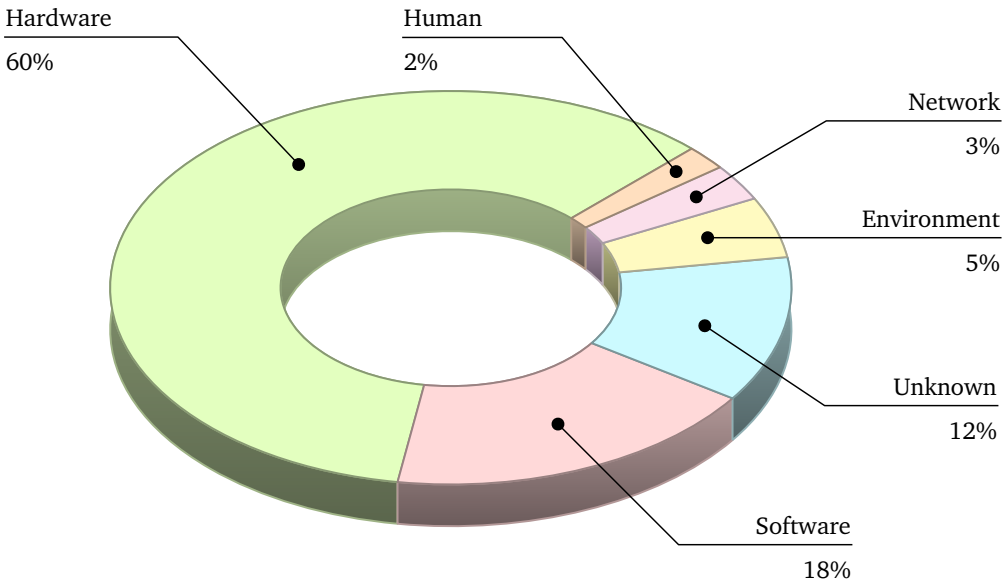


Figure 5.3.: Root cause breakdown chart

5.3.1.2 Hardware and Software Failures

After the main contributors to failures in ICT systems are identified, a clarification regarding the different failure behavior characteristics in hard- and software is regarded as necessary. While both hard- and software are subject to failures, their nature is quite dissimilar, which leads to unalike failure models and developments over time.

In the case of hardware, physical causes (e. g., random failures during useful life or wear-out failures) are the predominant cause of failures; in contrast, software malfunctions solely appear as a symptom of systematic failures (e. g., due to a wrongly defined interface in the software design phase) [6].

In addition to that, the behavior of hardware failure rates along the component's lifetime is well-known and follows a three-stage pattern. During the "infant mortality" phase, a decreasing failure rate can be observed, the "useful life" period is characterized by a quasi-constant failure rate (which are random failures), while the "wear-out" phase is showing an increasing failure rate due to wear-out until the system's end of life. This behavior is illustrated in Figure 5.4.

While the failure behavior of hardware is widely agreed upon [66, 159], this is not true in the case of software failure rate development. There are various theories and models regarding the behavior of software failure rates over time; these are going to be discussed here briefly.

It is agreed upon that software failures are inherently deterministic in nature, as they do not result from randomly appearing challenges, but rather from bugs present inside the software's code, which are triggered under specific conditions. This leads to software failures themselves being systematic and reproducible; however, when looked at from a temporal rather than a causal point-of-view, the times at which a bug is triggered leading to a system failure needs to be modeled using stochastic processes, as it is impossible to predict when a software failure will occur in a complex system.

The failure rate development of software over time therefore follows a three-stage pattern (debugging, useful life, obsolescence), too. While there is a consensus on the behavior during the debugging phase, being that possible bugs present during the first software deployment are identified and fixed,

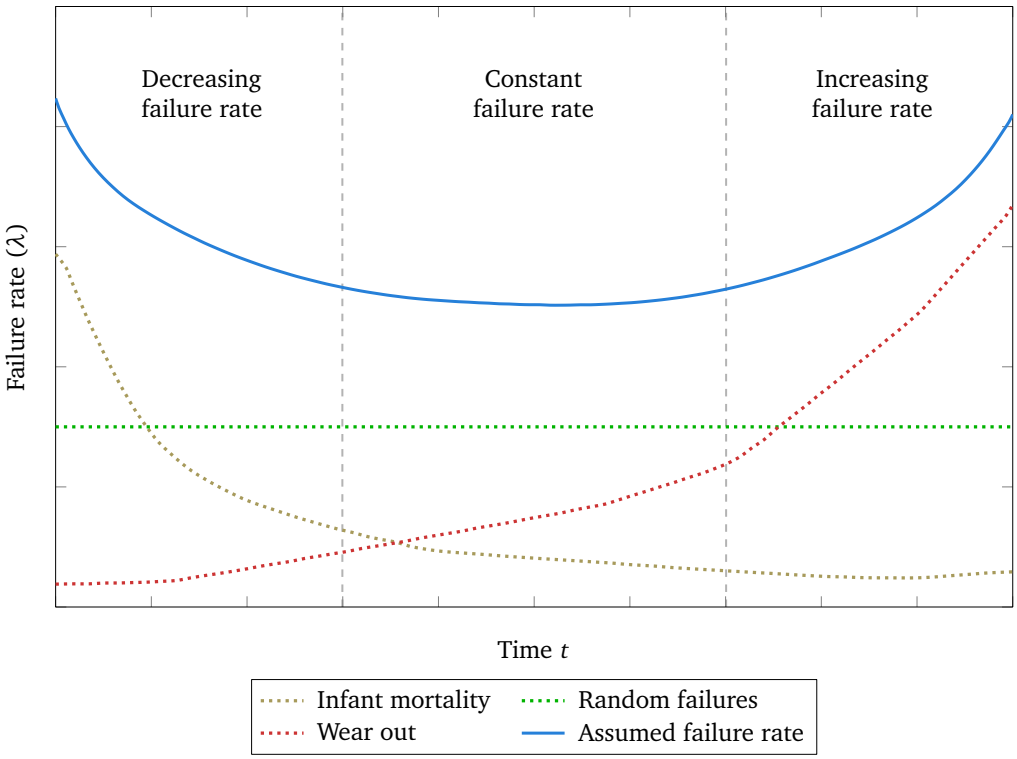


Figure 5.4.: Development of hardware failure rate over time, based on IEC 61508 [66]

leading to a decreasing failure rate, both following phases are still discussed by experts. The major points of discussion are:

- **Patching:** There are ongoing discussions if, and if so how much patching of software—which is required during the **useful life** phase of software to extend or adapt features—leads to new bugs that may lead to failures. Youn & Yi [180] and Rawat *et al.* [133] state that each introduction of a patch causes an initial increase in failure rate, as patches are causing new bugs to appear due to imperfect programming. The initially rising failure rate is then reduced by subsequent patches, which is captured in software reliability growth models, such as the Jelinski-Moranda, Littlewood and Goel-Okumoto models [137]. In contrast, Jalote & Murphy [68] and Rosenberg *et al.* [138] claim that such strong influences of patches are not present anymore in modern software patches. Rather a constantly decreasing failure rate is observable in software, somewhat resembling the shape of the "infant mortality" curve present in hardware.

- **Aging property:** There is a disagreement regarding the existence of an "aging property" in software with progressing **obsolescence**. Friedman *et al.* [56], Parnas *et al.* [120] and Rosenberg *et al.* [138] argue that software does not deteriorate like hardware does in a sense that the failure occurrence rate changes due to its own physical aging properties. This would lead to software failure rates being constant during time progression if it is not subject to any changes. In contrast, Ogheneovo [116] and Parnas [119] agree on the fact that while it is true that when software has reached a certain age (obsolescent state), there are no more upgrades made to the program (leading to nearly constant failure rate). However, several other reasons for deterioration lead to a similar "wear-out" effect as observed with hardware:

– **Changing operational profile**: The system is used according to the original specifications, however, the usage patterns change so that functions that were previously used infrequently are now invoked more often. Therefore, failures that were previously experienced only seldom are now occurring more frequently, necessitating system changes.

– **Changing requirements**: Business and functional requirements typically change over time. This is usually the result of innovation, security-related adaption and changes in the market, regulatory environment and competitive landscape.

– **Changing technology stack**: System software platforms such as operating systems and database management systems evolve with time. In order to avoid obsolescence, software systems must be migrated from old technology platforms to newer ones. This is true even if the functional and business requirements have not changed. There are situations where there is a business or functional requirement that requires that changes be made to the technology stack.

Figure 5.5 depicts the failure behavior of software over time. The dotted lines represent possible behaviors that are however *not* used during the modeling within this thesis; the assumed failure behavior is depicted as a blue, solid line.

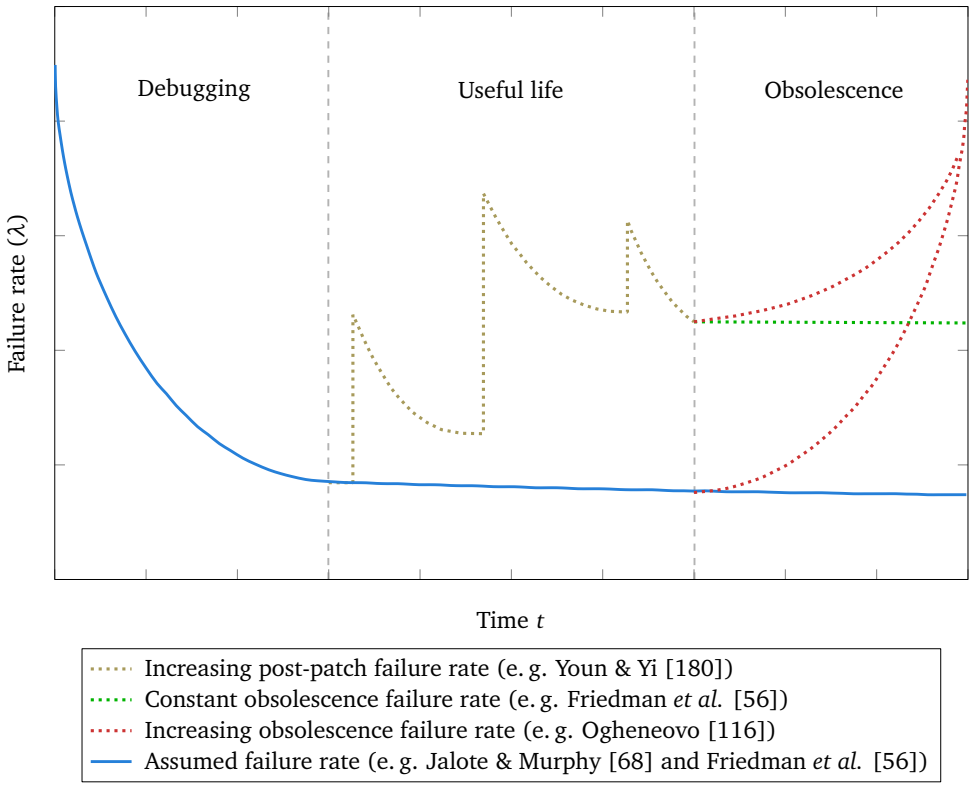


Figure 5.5.: Development of software failure rate over time

The reasons for the assumed software failure behavior are:

- Assuming an AMI environment as a part of a critical infrastructure, it is likely that the software services themselves are not subject to many changes, but rather restricted and constant in their functions. A high amount of modifications during their useful life phase is not realistic.

- In addition, if patches are applied in a critical infrastructure environment, it is probable that these software modifications are well-tested, leading to a lower probability of post-patch failure rate increases.
- In contrast to "normal" software, services in critical infrastructures are required to have a long support time, therefore an obsolescence time will be highly delayed.

After the failure behaviors of both hard- and software over time was discussed in this section, an exact definition of the types of failures regarded in this thesis is given in Section 5.3.1.3.

5.3.1.3 Failure Modes

To be able to subsequently model system failures, the term "failure", which was only informally used previously in this chapter, needs to formalized. In contrast to Ammar *et al.* [6], who stated that a system failure is only present if the actual output of the system for a specific input differs from its expected output, in this thesis, an extension to this definition is devised to include the QoS requirements of systems. First, the failure definition focusing on the function of a system—based on Ammar *et al.* [6]—are introduced, after that, failures stemming from unfulfilled non-functional requirements are discussed.

Functional Failures. Ammar *et al.* argued that defining a failure as a difference between the actual and expected output of a system for a specific input leads to two additional issues which need to be addressed [6]. First, it is required to investigate if, and if so how system behavior can be modeled by its input-output pairs; second, a system needs to have a well-defined expected behavior that is known a priori.

The first issue may be regarded as trivial for straightforward hard- and software components, such as basic mathematical operations. However, more complex systems, such as, e. g. OSs or similar are much harder to formalize in such manner. Skuce & Mili shows how such complex systems are represented by pairs of the form (h, y), where h is an input sequence and y is an output [152]. To model the expected behavior of a system, it is required to distinguish between two distinct definitions of the term failure discussed next.

- A failure to fulfill the function that the system needs to compute. The function of a system defines specific expected outputs to inputs. A failure is therefore present if any input leads to an output failing to fulfill the specified function.
- A failure to fulfill the specification that the system is built to satisfy. Using this definition, there is no check whether a system fulfills a function *correctly*, however, a failure of the system is detected if its specification (e. g., result out of range of possible results) is not satisfied.

The definitions of both types of failures are given in Definitions 5.1 and 5.2, respectively. For both definitions a system σ, which computes a function ϕ from space S to space S' (i. e., $\phi \subseteq S \times S'$) is considered.

Definition 5.1: Failure with respect to function

A failure of system σ with respect to its function ϕ is present if and only if for an initial state s the corresponding output state s' does not satisfy the condition $(s, s') \in \phi$.

Definition 5.2: Failure with respect to specification

A failure of system σ with respect to specification ρ is present if and only if for an initial state s the corresponding output state s' does not satisfy the condition $(s, s') \in \rho$.

As an example, a simple addition system for two positive integers (x, y) is considered. The expected behavior of such a system can straightforward be described as a function ϕ, defined as:

$$\phi = \{(\langle x, y\rangle, z) \mid z = x + y\}.$$

If the system function would not be clear a priori, a specification ρ could be:

$$\rho = \{(\langle x, y\rangle, z) \mid x \geq 0 \wedge y \geq 0 \wedge |z| = x + y\}.$$

Using the addition system, a function-based failure is for example present if for two given integers, $x = +15$ and $y = +12$, the output $z = -27$ is obtained; the system does no longer fulfill its function $\phi = \{(\langle x, y\rangle, z) \mid z = x + y\}$. Using the same example, however, no specification-based failure is detected, as the system still satisfies $\rho = \{(\langle x, y\rangle, z) \mid x \geq 0 \wedge y \geq 0 \wedge |z| = x + y\}$.

This can be generalized in the following way: A function-based failure detection is always a part of any valid specification-based failure detection. In most cases, a specification is going to be more coarse and allows more solutions than a function; this is also depicted in Figure 5.6. Here, the dashed circle represents valid (non-failing) solutions using a specification; the point marked by the arrowhead is the solution valid using a function-based failure detection.

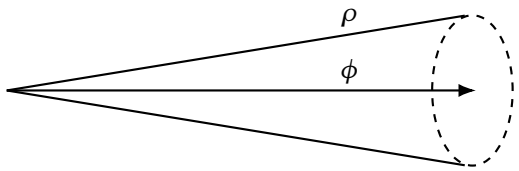


Figure 5.6.: Failure detection based on function ϕ and specification ρ (based on Ammar *et al.* [6])

Within Chapters 5 and 6, functional failures are included in the given failure rates λ. While a function/specification check would be possible for certain simple services, such as data concentrators, they are not done on a system-wide level. However, functional failures are considered in Section 6.3.3, where a proof-of-concept implementation of a virtualized SMGW is analyzed.

Non-functional Failures. In contrast to functional failures, a non-functional failure is present if a correct result is delivered by a system, however, at least one non-functional requirement (e. g. timing) for the result is not met. The specification of the non-functional requirements may include parameters such as latency and throughput, both in a minimum and maximum allowed amount. Mathematically, non-functional failures closely resemble the definition of specification-based failures, as they are only defining a minimum/maximum threshold, not a precise value to be fulfilled. It is noteworthy that non-functional requirements may include both maximum (ζ) and minimum (η) non-function requirements. Taking the addition system as an example again, a maximum non-functional requirement may include, e. g., a maximum latency (L_{max}). This results in:

$$\zeta = \{\sigma \mid L(\sigma) \leq L_{max}\}.$$

Definition 5.3: Failure with respect to non-function requirements

A failure of system σ with respect to non-function requirements is present if and only if for an initial state s the corresponding output state s' does not satisfy the condition $(s, s') \in \zeta \cap \eta$.

Non-functional failures are considered during the evaluation of the overall model (which is further elaborated in Section 5.4) performed in Section 6.2.3. Here, the performance achieved by SSMA and VNFA are compared to the non-functional requirements of AMIs.

Figure 5.7.: Failure detection based on non-functional requirements

5.3.2 Recovery Behavior

Upon a (sub)system failure (as explained in Section 5.3.1.3), a (hard-/software) system's performability may be impaired. The failures present in a system may be restored during a recovery phase. A system's recovery is mainly characterized in three dimensions: **possibility**, **quality**, and **structure** of the recovery.

5.3.2.1 Possibility

First, it needs to be distinguished if a system is able to be recovered from any state that requires repairs back to an acceptable performance level (**perfect recovery**) or if there is at least one non-performing state that cannot be recovered (**imperfect recovery**). Formally, a perfect recovery is defined as:

$$\forall\, i \in S_{down}, \exists k \in S_{up} : \exists \mathfrak{p} \in \mathfrak{P} \mid i \xrightarrow{\mathfrak{p}} k,$$

where S_{down} is a set of all "down" states of a system i. e. $\forall\, i \in S_{down} : P(i) < P_{threshold}$, where $P(i)$ is the performance during recovery phase i $\left(P(i) \in \mathbb{R}_0^+ \leq 1\right)$, $P_{threshold}$ the performance threshold which needs to be exceeded for a system to be considered operational. S_{up} is a set of "up" states, i. e. $S_{up} = S \setminus S_{down}$. $\mathfrak{p}$ is an arbitrary path within $\mathfrak{P}$, the set of all finite paths within the system states S. In contrast, an imperfect recovery is defined as:

$$\exists\, i \in S_{down}, k \in S_{up} : \nexists\, \mathfrak{p} \in \mathfrak{P} \mid i \xrightarrow{\mathfrak{p}} k.$$

5.3.2.2 Quality

The second characteristic of a recovery is its quality. Despite the possibility to recover from impaired system states, which was discussed in the previous Section 5.3.1.3, the resulting performance after a recovery may vary. If the system always offers a performance level similar to a new, i. e. never impaired, system after a recovery, it offers an **optimal recovery**. This is formalized as:

$$\forall\, \mathfrak{p} \in \mathfrak{P} \mid i \xrightarrow{p} k : P(k) = P_{max},$$

where P_{max} denotes the maximum performance level achievable by the system.

If after a recovery attempt the system shows a varying performance level—which in some cases is optimal, while in others, a decreased performance level persists—an **unstable recovery** is present, expressed as:

$$\exists\, \mathfrak{p}, \mathfrak{q} \in \mathfrak{P} : i \xrightarrow{\mathfrak{p}} k, P(k) = P_{max} \wedge j \xrightarrow{\mathfrak{q}} l, P(l) < P_{max},$$

where $i, j \in S_{down}$ and $k, l \in S_{up}$.

If no recovery attempt is able to restore the system's maximum, but instead only a lower performance level, a **deteriorated recovery** is present. This is mathematically expressed as:

$$\forall\, \mathfrak{p} \in \mathfrak{P} \mid i \xrightarrow{\mathfrak{p}} k : P(k) < P_{max}.$$

5.3.2.3 Structure

The structure of a recovery process expresses if multiple states are present during the recovery itself and if the performance levels do change. Two main types of recovery structures exist: single and multi-phase; both are described next.

Single Phase Recovery. In a single phase recovery approach, all required restoration actions are modeled into a single activity while assuming the performance being unchanging during this recovery. To be able to estimate the required restoration time appropriately, it is necessary to not only calculate the maximum, minimum, and mean restoration time required, but also its distribution; otherwise, inadequate assumptions regarding the recovery time may lead to under- or overprovisioning of recovery resources. Using a single phase recovery model, the recovery time must include all required actions to restore the system, including all recovery attempts until a successful recovery is achieved as well as all actions that are part of the recovery. Formally defined, this leads to:

$$|R| = 1 \wedge P(i) = 1 - \xi_i \wedge t(i) = \tau_i,$$

where R is a set of recovery phases; $|R|$ is the number of recovery phases; ξ_i is the performance loss during recovery phase i, where $i \in R$ and $\xi_i \in \mathbb{R}_0^+ \leq 1$; $P(i)$ is the performance during recovery phase i; $t(i)$ is the recovery time required for a single attempt to restore the system from recovery phase i.

Multi-phase Recovery. The multi-phase recovery allows changing performance levels during a system's restoration. This behavior is achieved by modeling the recovery in multiple, escalating phases of recovery, where each represents a new restoration attempt [96]. Usually, the simplest recovery approach (i. e. the one requiring the shortest time) is attempted first. If this approach is unsuccessful, the subsequent phases of recovery are attempted until the system is restored. Depending on the structure of the multi-phase recovery, two sub-classes may be distinguished:

- **Multi-phase, single level:** This recovery type assumes that the performability level remains unchanged during multiple recovery phases—similar to the single phase recovery. Effectively, this leads to multiple escalating recovery phases sharing the same performance level, which is formalized as:

  $$|R| > 1 \wedge \forall\, i, j \in R : P(i) = P(j) \wedge \exists\, k, l \in R : t(k) \neq t(l)$$

- **Multi-phase, multi-level:** In contrast, if the performance level may change for each phase during the recovery process, a multi-phase, multi-level process is present. A formal expression of such a recovery process is:

  $$|R| > 1 \wedge \exists\, i, j \in R : P(i) \neq P(j) \wedge \exists\, k, l \in R : t(k) \neq t(l)$$

  Examples for such a recovery are often found after complex systems suffered severe degradations. The recovery after such a failure is often divided into several recovery phases, each changing the performability level of the overall system. It needs to be noted that after each recovery phase, several possible outcomes need to be distinguished:

  - **Improved:** The system performance level is increased after the recovery phase is finished. Formally, this is the case if:

    $$i, i+1 \in R : P(i) < P(i+1)$$

  - **Unchanged:** The performance level remains unchanged if the restoration remains unsuccessful after the recovery phase finished, which is formalized as:

    $$i, i+1 \in R : P(i) = P(i+1)$$

  - **Degraded:** If another failure (either related to the recovery attempt or unrelated) occurs during the recovery, the performance level is decreased. This happens if:

    $$i, i+1 \in R : P(i) > P(i+1)$$

    An example for a failure caused by a restoration attempt is e. g. a software upgrade intended to restore a system, which may introduce another failure.

In the scenario investigated in this thesis, a *perfect, unstable, multi-phase, single level recovery* is assumed, which is further elaborated upon in Section 5.3.4.4.

5.3.3 Macro Model (Inter-System)

The macro model represents the interactions between several systems, the performance-enhancing methods on substrate level and the embedding of VNFs stretching across multiple substrate entities. To generate the model, it is first required to determine whether VNFA or SSMA is used. In the case of VNFA, a more detailed explanation is given in the following Section 5.3.3.1, while the generation of the macro model in SSMA's case is described later in Section 5.3.3.2.

5.3.3.1 VNFA Macro Model

In the VNFA macro model the VNF-FG of each user is embedded using the embedding technique formalized in Algorithm 4.2 and the substrate redundancy scheme to be employed (Section 4.4.1), a macro model can be generated which represents the hosting and interaction across multiple substrate entities, which—in turn—allows to generate the micro models for each substrate node. To cover the requirements of the macro model discovered in Section 4.3.3, a model based on networks graphs (see Definition 5.4 for a general definition), which are already actively used both in virtual network embedding as well as other network theoretical fields, is developed. In the given case, the substrate plane is modeled as a graph $G^{hw} = (V^{hw}, E^{hw})$ that is

- *undirected*, representing the bidirectional links of the substrate plane,

- *finite*, due to the inherently finite number of entities represented by the vertices and edges in the graph, and
- *weighted*, due to the resource limitations present in the substrate.

Definition 5.4: Network graph

A network graph is an ordered pair $G = (V, E)$ comprising a set V of vertices or nodes together with a set E of edges or lines, which are 2-element subsets of V. [18]

To represent the properties of an entity (being it a node or link), its "weight" is not defined as a single value, but as a weight vector. For vertices, this vector consists of its available hardware resources (CPU $r^{hw}_{cpu} \in \mathbb{R}^+_0$ and type $r^{hw}_{type} \in \mathbb{T}$, where $\mathbb{T} = \{sm, gw, r, dc, tps, g\}$ with g being a generic type representing no substrate type requirements) and an owner $o^{hw} \in \mathbb{N}_0$. In addition, there are three vectors present for the failure $\left(\boldsymbol{\lambda}^{hw} \in (\mathbb{R}^+_0)^3\right)$ and repair rates $\left(\boldsymbol{\mu}^{hw} \in (\mathbb{R}^+_0)^3\right)$ associated with the substrate entity, as well as a backup vector ($\boldsymbol{\nu}^{hw} \in S^*$). The vectors are defined as $\boldsymbol{\lambda}^{hw} = \begin{bmatrix}\lambda_{hos} & \lambda_{hom} & \lambda_{hol}\end{bmatrix}^\top$ and $\boldsymbol{\mu}^{hw} = \begin{bmatrix}\mu_{hrs} & \mu_{hrm} & \mu_{hrl}\end{bmatrix}^\top$, respectively, containing the variables on substrate level as detailed in Section 5.3.4. $\boldsymbol{\nu}^{hw}$ is a vector of arbitrary length containing all backup nodes of a substrate entity. Details regarding $\boldsymbol{\nu}^{hw}$ and its implications are further elaborated upon later in this section. The resulting substrate weight vector for vertices is given as $\mathbf{w}(v^{hw}_i \in V^{hw}) = \begin{bmatrix} r^{hw}_{cpu_i} & r^{hw}_{type_i} & o^{hw}_i & \boldsymbol{\lambda}^{hw}_i & \boldsymbol{\mu}^{hw}_i & \boldsymbol{\nu}^{hw}_i \end{bmatrix}^\top$. For edges, the weight is defined by the available link bandwidth $r^{hw}_b \in \mathbb{R}^+_0$, a failure rate $\lambda^{hw} \in \mathbb{R}^+_0$, and a repair rate $\mu^{hw} \in \mathbb{R}^+_0$. Formally, the weight is given as $\mathbf{w}(e^{hw}_{ij} \in E^{hw}) = \begin{bmatrix} r^{hw}_{b_{ij}} & \lambda^{hw}_{ij} & \mu^{hw}_{ij} \end{bmatrix}^\top$.

Both virtualization planes, namely the VMM and VM planes, are modeled in a similar way as the substrate plane, however, there are no edges defined for the virtualization layers. Two vectors in VMM vertices represent the failure rates $\left(\boldsymbol{\lambda}^{vmm} \in (\mathbb{R}^+_0)^3\right)$ and repair rates $\left(\boldsymbol{\mu}^{vmm} \in (\mathbb{R}^+_0)^2\right)$. The failure rate vector is defined as $\boldsymbol{\lambda}^{vmm} = \begin{bmatrix}\lambda_{vmmo} & \lambda_{vmmd} & \lambda_{vmmod}\end{bmatrix}^\top$, the repair rate vector as $\boldsymbol{\mu}^{vmm} = \begin{bmatrix}\mu_{vmmr} & \mu_{vmmrej}\end{bmatrix}^\top$. Instead of providing resources, the virtualization induces a certain overhead, as described in Section 3.2.4.2. This is represented by two parameters, namely the CPU overhead ($\Omega_{cpu} \in \mathbb{R}^+ \geq 1$) and the network overhead ($\Omega_{net} \in \mathbb{R}^+ \geq 1$). Formally the VMM weight vector for vertices is given as $\mathbf{w}(v^{vmm}_k \in V^{vmm}) = \begin{bmatrix} \boldsymbol{\lambda}^{vmm}_k & \boldsymbol{\mu}^{vmm}_k & \Omega_{cpu} & \Omega_{net} \end{bmatrix}^\top$.

While the VM plane is technologically deeply linked with the VMM plane, a significant difference distinguishes them: Multiple VMs may be hosted on a single VMM, therefore a $1 : n$ relationship is present between VMMs and VMs. This leads to the following definition regarding the VM plane: A CPU ($d^{vm}_{cpu} \in \mathbb{R}^+_0$) and type demand ($d^{vm}_{type} \in \mathbb{T}$) are defined; these values are inherited from the service plane. Again, there are two vectors in VM vertices that represent the failure rates $\left(\boldsymbol{\lambda}^{vm} \in (\mathbb{R}^+_0)^3\right)$ and repair rates $\left(\boldsymbol{\mu}^{vm} \in (\mathbb{R}^+_0)^2\right)$. The failure rate vector is $\boldsymbol{\lambda}^{vm} = \begin{bmatrix}\lambda_{vmo} & \lambda_{vmd} & \lambda_{vmod}\end{bmatrix}^\top$, the repair rate vector $\boldsymbol{\mu}^{vm} = \begin{bmatrix}\mu_{vmr} & \mu_{vmrej}\end{bmatrix}^\top$. The VM vertices weight vector is $\mathbf{w}(v^{vm}_l \in V^{vm}) = \begin{bmatrix} d^{vm}_{cpu_l} & d^{vm}_{type_l} & \boldsymbol{\lambda}^{vm}_l & \boldsymbol{\mu}^{vm}_l \end{bmatrix}^\top$.

After the definition of the substrate and virtualization planes of the AMI model, the service plane is defined; however, the properties of the service network graph $G^s = (V^s, E^s)$ are different compared to the previous graphs; G^s it is *finite*, *directed* and *weighted*. Compared to the substrate graph, the service plane requires a directed graph model to represent the logical order of services. The weights of service vertices include the required CPU resources of the service (CPU $d^s_{cpu} \in \mathbb{R}^+_0$), a required type $d^s_{type} \in \mathbb{T}$,

a service failure rate vector $\boldsymbol{\lambda}^s \in (\mathbb{R}_0^+)^5$, a service repair rate vector $\boldsymbol{\mu}^s \in (\mathbb{R}_0^+)^3$, a service priority flag $\sigma^s \in \{0,1\}$, and a service's user $u \in \mathbb{N}_0$: $\mathbf{w}(v_m^s \in V^s) = \begin{bmatrix} d^s_{cpu_m} & d^s_{type_m} & \boldsymbol{\lambda}^s_m & \boldsymbol{\mu}^s_m & \sigma^s_m & u^s \end{bmatrix}^\top$. The vectors are defined as follows: $\boldsymbol{\lambda}^s = \begin{bmatrix} \lambda_{so} & \lambda_{sol} & \lambda_{sd} & \lambda_{sod} & \lambda_{sold} \end{bmatrix}^\top$, and $\boldsymbol{\mu}^s = \begin{bmatrix} \mu_{sr} & \mu_{srl} & \mu_{srej} \end{bmatrix}^\top$. For service graph edges, the weight parameters are the required link bandwidth $d_b^s \in \mathbb{N}$, a failure rate $\lambda^s \in \mathbb{R}_0^+$, and a repair rate $\mu^s \in \mathbb{R}_0^+$. The respective definition is $\mathbf{w}(e^s_{mn} \in E^s) = \begin{bmatrix} b^s_{mn} & \lambda^s_{mn} & \mu^s_{mn} \end{bmatrix}^\top$.

In addition to the representation of the individual layers using network graphs, to formalize the relationship between the layers (i. e. the hosting relationship), a set of directed, finite, unweighted edges E^m is created, where:

$$e^m_{ij} \in E^m : \begin{cases} j \in V^{vm} & \text{for } i \in V^s \\ j \in V^{vmm} & \text{for } i \in V^{vm} \\ j \in V^{hw} & \text{for } i \in V^{vmm} \end{cases}$$

E^m includes edges that represent the mapping of an entity onto another: $\exists\, e^m_{ij} \in E^m : i$ is mapped to j. This mapping is unambiguous, i. e. the out-degree of each node regarding E^m that is not on the bottom-most layer is always 1; on the bottom-most layer, it is 0:

$$outdeg^{E^m}(i) : \begin{cases} 1 & \text{for } i \in \{V^s \cup V^{vm} \cup V^{vmm}\} \\ 0 & \text{for } i \in V^{hw}. \end{cases}$$

Also, the mapping induced by E^m needs to respect the type demands given by d^s_{type} and/or d^{vm}_{type}. This is enforced by requiring:

$$e^m_{ij} \in E^m : \begin{cases} r^{hw}_{type_j} \in V^{hw} = x & \text{for } d^s_{type_i} \in V^s = g \\ r^{hw}_{type_j} \in V^{hw} = y & \text{for } d^s_{type_i} \in V^s = y \neq g, \end{cases}$$

where $x, y \in \mathbb{T}$.

The scheme of the model is illustrated using an example including a single substrate and VMM entity, two VMs and three service entities in Figure 5.8. Using the substrate and service plane of the novel modeling approach, it becomes possible to represent both standard (non-virtualized) as well as virtualized AMIs. At this point, it should be noted that the substrate plane is only required to embed the logical entities of the service plane, i. e. the substrate plane itself could also be realized as virtual machines (in which case the embedding of logical services would lead to a nested virtualization). This way, it would be possible to achieve an even further abstraction from the bottom-most, physical substrate. However, it is also worth mentioning that every layer of virtualization induces an additional overhead leading to possible performance impairments, as discussed in Section 3.2. Because of that, the substrate plane is assumed to be realized as physical hardware throughout this thesis.

Concerning the modeling of either non-virtualized or virtual logical services, it can be seen that the usage of non-virtualized services implies that each service requires a separate substrate, leading to a $1:1$ relationship of service to substrate nodes ($|V^{hw}| = |V^s|$); in the virtualized case, this relationship does usually not hold, as many services can be embedded onto a single substrate, therefore, a $n:1$ relationship between service and substrate nodes is given. In contrast to the $|V^{hw}| = |V^s|$ relationship in the non-virtualized case, if virtualized services are hosted, no assumptions concerning the number of logical and substrate nodes are possible, as a dynamic redistribution of VMs is possible, leading, e. g., to the possibility of abandoned substrate nodes. In VNFA, in addition, it holds that $|V^{hw}| = |V^{vmm}|$ (a single VMM per substrate node) and a $m:1$ relationship of $|V^{vmm}|$ to $|V^{vm}|$.

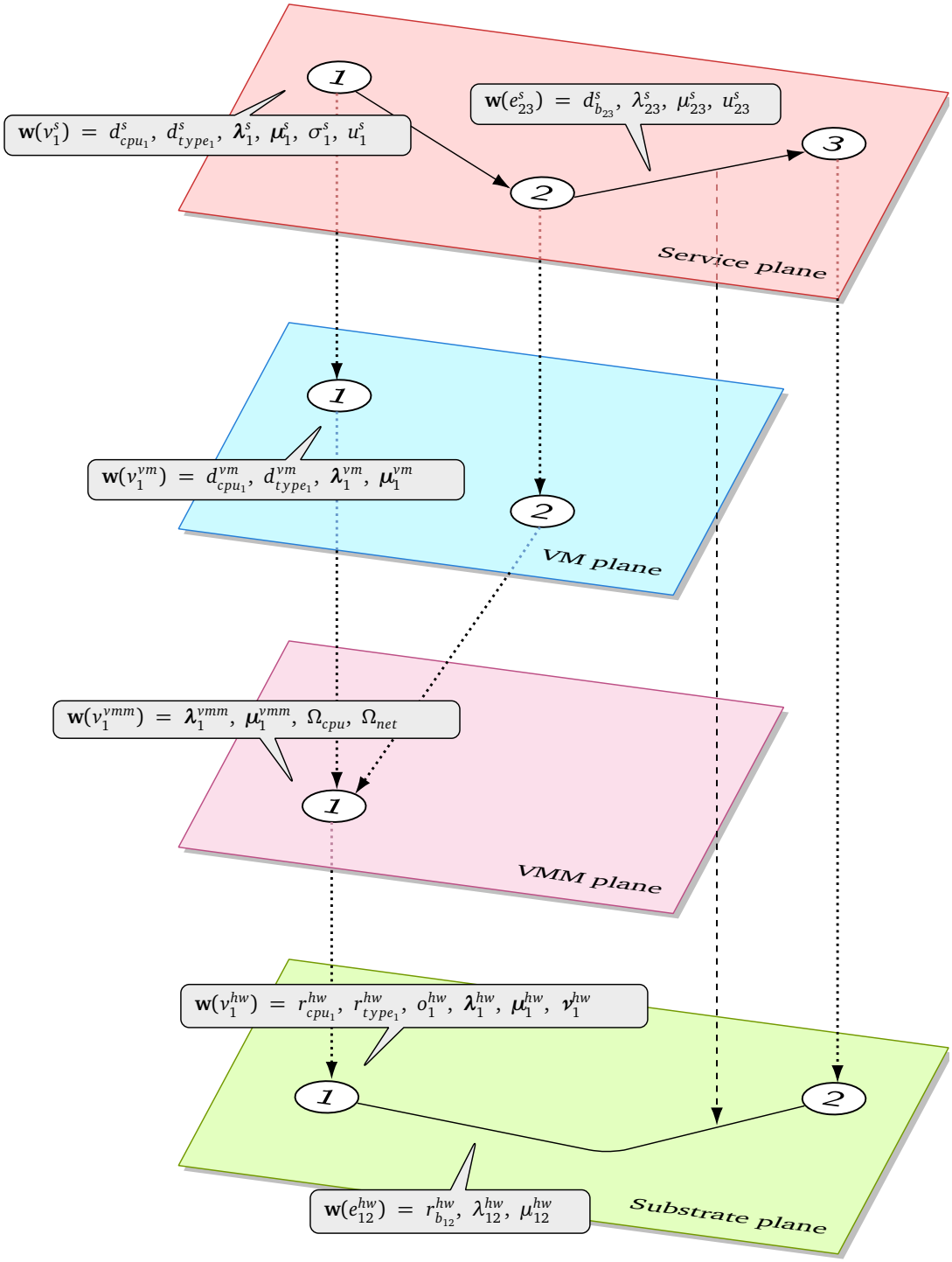


Figure 5.8.: Service, VM, VMM, and substrate plane of the VNFA macro model

As introduced in Section 4.4, on substrate level, there are three performance-enhancing methods, which focus on dependability by providing backup substrate nodes in case of outages. The three options are: (a) no substrate backup, (b) circle backup, and (c) all-for-all backup. In contrast to the performability-enhancing methods in the micro model, the macro model does not require extensive alterations to include the backup options discussed in Section 4.4. To specify which servers are to be used as backup nodes in case of a failure, the parameter $\boldsymbol{\nu}_i^{hw}$ is present for each substrate entity i. There are three cases to be distinguished for $\boldsymbol{\nu}_i^{hw}$:

$$|\boldsymbol{\nu}_i^{hw}| = \begin{cases} 0 & \text{if no backup is defined} \\ 1 & \text{if circle backup is used} \\ |V^{hw}| - 1 & \text{if all-for-all backup is used} \end{cases}$$

5.3.3.2 SSMA Macro Model

The SSMA macro model is realized in a similar manner as the VNFA model, mainly distinguishing itself in the removal of the VMM and VM layers. Furthermore, due to the non-virtualized system architecture, every service is directly "mapped" onto its corresponding substrate, with $d^s_{type_i} = r^{hw}_{type_i}$, which is not self-evident, as in the virtualized case, several services with different type demands may be hosted on the same substrate node. The backup strategy used in VNFA and represented by the vector $\boldsymbol{\nu}$ is also removed from the model in SSMA, as no servers hosting virtual machines are present. The resulting model is displayed in Figure 5.9.

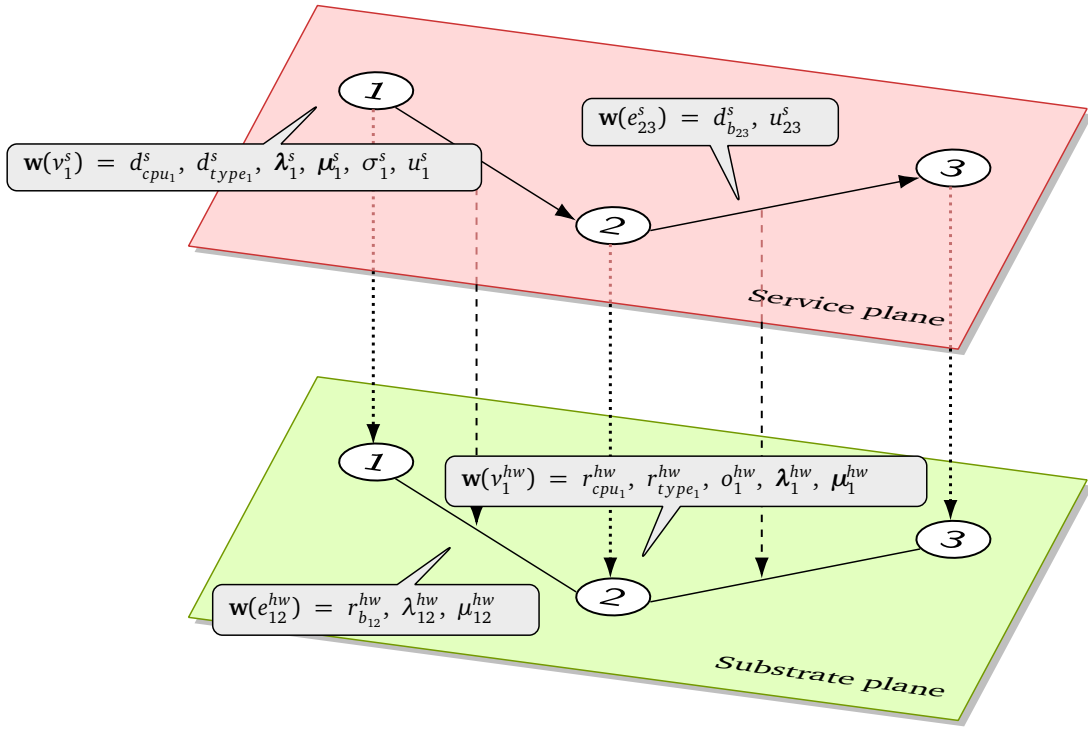


Figure 5.9.: Service, and substrate plane of the SSMA macro model

5.3.4 Micro Model (Intra-System)

The micro model represents the behavior of the virtualized AMI system on a system-internal level. This includes the modeling of failure and restoration behaviors, which are based on the findings in Sections 5.3.1 and 5.3.2, as well as the interplay of system components. After the basic model description, the performability-enhancing methods introduced to system components in Section 5.3.4.5 are included. Finally, in Section 5.3.4.6, the micro model of a virtualized AMI is generated. During the model generation, w.l.o.g. a single service being hosted in a VM is assumed, until stated otherwise. The micro model for SSMA is also presented at the end of Section 5.3.4.6.

5.3.4.1 Nomenclature

During the development of the micro models for VNFA and SSMA, several abbreviations are used to describe firing rates (such as repair and failure) as well as other parameters. Though these are elaborated further when they are first encountered, a list of the most important abbreviations used in the micro models is given in the following Table 5.3.

Abbreviation	Meaning
λ_{so}	Service failure rate
λ_{sd}	Service degradation/aging/impairment rate
λ_{sol}	Long service failure rate
λ_{sod}	Degraded/aged/impaired service failure rate
λ_{sold}	Long degraded/aged/impaired service failure rate
μ_{sr}	Service repair rate
μ_{srl}	Long service repair rate
λ_{vmo}	VM failure rate
λ_{vmd}	VM degradation/aging/impairment rate
λ_{vmod}	Degraded/aged/impaired VM failure rate
μ_{vmr}	VM repair rate
λ_{vmmo}	VMM failure rate
λ_{vmmd}	VMM degradation/aging/impairment rate
λ_{vmmod}	Degraded/aged/impaired VMM failure rate
μ_{vmmr}	VMM repair rate
λ_{hos}	Short hardware failure rate
λ_{hom}	Medium hardware failure rate
λ_{hol}	Long hardware failure rate
μ_{hrs}	Short hardware repair rate
μ_{hrm}	Medium hardware repair rate
μ_{hrl}	Long hardware repair rate
τ_d	Failure detection time
τ_{dl}	Long failure detection time
δ	Deterministic rejuvenation interval
∇	Dynamic rejuvenation threshold

Table 5.3.: Micro model nomenclature

5.3.4.2 Assumptions

Several assumptions are taken during the establishment of the micro models, which are listed in the upcoming paragraph.

- Impairments due to aging-related Mandelbugs are assumed to cause the same amount of performance loss each time. Also, the performance loss associated with either a VMM, VM or

service impairment, as well as any combination of the aforementioned do not immediately lead to a non-functional failure as stated in Definition 5.3. This is due to the AMI performance constraints being present on VNF-, not VNFC-level.

- Performability impairments of VMMs, VMs, and services are assumed to be additive, i. e. the combined performability impairment (l_{Perf}) is obtained by $l_{Perf} = \sum_{i \in S_{imp}} \big(1 - r(i)\big)$, where $S_{imp} = \{j \in S \mid 0 < r(j) < 1\}$. It is further assumed that $\max(l_{Perf}) = 1$.
- While the overall model of VNFA is dynamic, in a sense that it captures changes to the structure of the model due to failures, VM migrations, etc., as stated in Section 5.2, the micro model in itself is static, i. e. apart from changing firing rates, the structure of the model is static.

5.3.4.3 NFV Failure Behavior

Using the information from the previous Section 5.3.1, the behavior of an infrastructure employing NFV is further analyzed here. Failures in VNFs can either originate from the NFVI level (affecting the host, consisting of hardware, VMMs, or VMs) or the service level. Both are explained further next:

- **Substrate failures** occur because of failing hardware and impair the NFVI. Well-known failures related to hardware are CPU/memory damage or defects in other parts of the hardware, such as power supply or Hard Disk Drives (HDDs). Hardware failures are expected to occur less often than most other failures; however, they also require a long repair time. On substrate level, three levels of failure difficulties are further distinguished, as hardware failures may have highly different likelihoods and respective required repair effort.

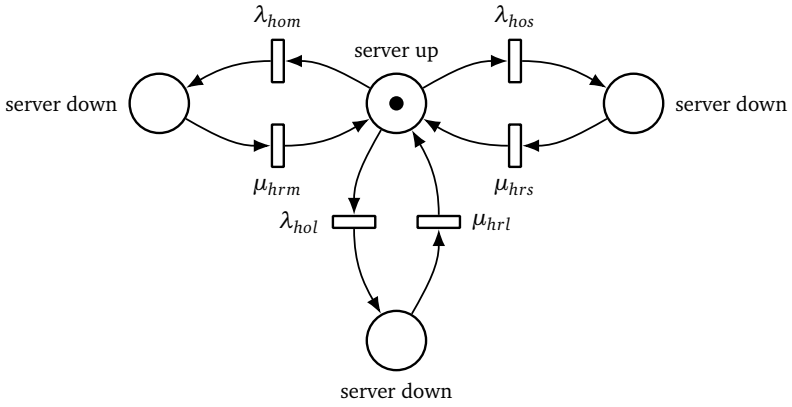
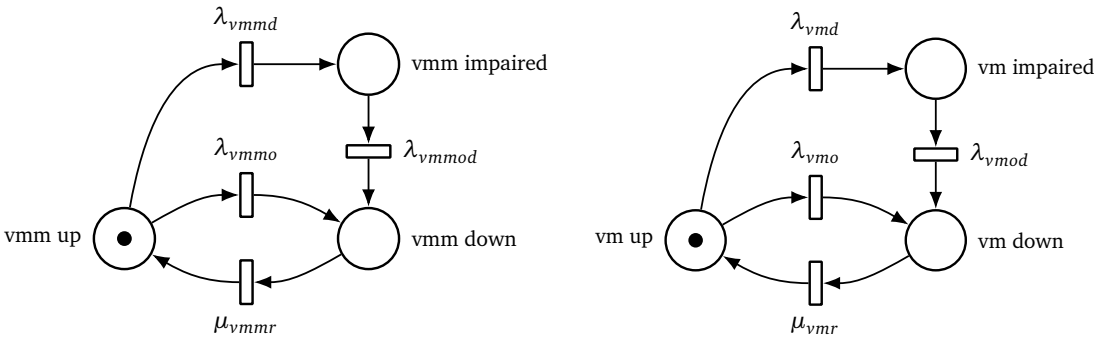


Figure 5.10.: Modeling on substrate plane

Short hardware failures (e. g. failure of a single component fixable by simple replacement) have a failure rate of λ_{hos} and a repair rate of μ_{hrs}. Medium hardware failures are present if, e. g., components fail that require a partial hardware disassembly such as CPU or mainboard. In this case, the failure rate is λ_{hom}, the repair rate is μ_{hrm}. Last, there are long hardware failures, which are events such as the destruction of multiple hardware components by common cause failures, such as over-voltage. The associated failure rate is λ_{hol}, the respective repair rate is μ_{hrl}. Also, on the substrate plane, there is only a difference between *working* and *failed* states, a further performance distinction of failure states is not done. Similar to availability, on the substrate plane, the hardware either runs smoothly, performing at its optimal level, or fails completely. While there are certain states that could be seen as impaired, such as, e. g., single HDD failures in a Redundant Array of Independent Disks (RAID) array, these do not impair

the system's performance significantly—unless a repair that requires the system to be shut down is performed. Also, as previously explained, the repair times need to be considered in the analysis, which is covered in Section 5.3.2. Figure 5.10 illustrates the behavior on substrate level as a Petri net model.

- **Virtualization failures** can either manifest on VMM level, where multiple VMs are impaired at once, or only impact a single VM. The failure modes which are usually considered in virtualization scenarios are Mandelbugs and Bohrbugs, where the former describe software-related failures which go unnoticed while the VMMs/VMs are deployed and running; in contrast, the latter are failures discovered during the deployment of VMMs/VMs. For the given scenario, Bohrbugs are excluded from the analysis, as the deployment is not within the scope of this thesis.



(a) Modeling on VMM level

(b) Modeling on VM level

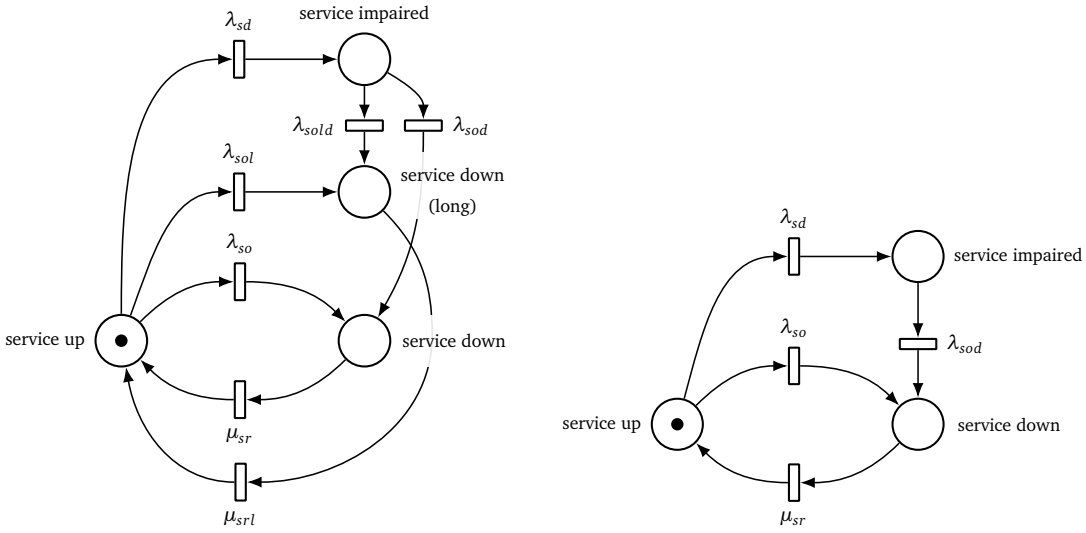
Figure 5.11.: Modeling on virtualization plane

Regarding the Mandelbugs, there is a further difference made between Non-Aging Related Mandelbugs (NAMs) and AMs. NAMs are considered as software *failures* in a classical sense, leading to an outage, while AMs are *impairments*, downgrading the performance of an entity. In case of an AM appearing on VMM level, the performance is reduced from $P = 1$ (in a fully functional state) to $P = 1 - \alpha$, where $\alpha \in [0, 1]$. In case of a VM AM, the resulting performance is $P = 1 - \beta$, where $\beta \in [0, 1]$. Figure 5.11a illustrates the behavior on VMM, Figure 5.11b on VM level.

- **Services failures** behave quite similar to the ones on VMM/VM level, as services can run in a degraded, aged state, too. This leads to a performance loss of γ, so the remaining performance in such a state is $1 - \gamma$, where $\gamma \in [0, 1]$. Depending on the interconnection with other services, it might be possible that the overall availability of the NFV-based AMI is impaired if services become unavailable or not. To represent this in a model, an additional distinction has to be made to separate *critical* services (e. g. mandatory AMI-related services, such as gateway components) from *non-critical* ones (e. g., third-party energy optimization consultation services), as introduced in Section 5.3.3.1. The failure of a *critical* service directly leads to an outage of the main functionality of the AMI for a user, the unavailability of a *non-critical* service represents a minor impairment.

  Regarding the failure behavior present on the service level, an additional distinction is made regarding the type of failure present. The failures on service level can be classified by the persistence of failures in regard to restarts of the service. This represents the fundamentally different behavior between a service failure due to a *transient/intermittent* challenge in the

service, e. g., caused by very high loads or a single bit-flip in memory and a *persistent* challenge, which requires to patch/update the service to run it again. To model this, two versions including both failure persistence behaviors are depicted in Figure 5.12.



(a) Modeling of persistent failures

(b) Modeling of non-persistent failures

Figure 5.12.: Modeling on service plane

In Figure 5.12a, persistent failures are modeled, which are not cleared with a repair rate of μ_{sr}, but take a much longer time (μ_{slr}) to be resolved. In contrast to that, Figure 5.12b illustrates a non-persistent failure model. Here, all failures are repaired in the same short amount of time (μ_{sr}).

Apart from the individual failures in each layer, there are interdependencies between them that may cause failure propagations. The concrete influences the planes have upon each other are further investigated in Section 5.3.4.6. In the following Section 5.3.4.4, the recovery model for a NFV-based AMI is illustrated.

5.3.4.4 Recovery

As stated in Section 5.3.2, the recovery of the system is perfect, unstable, multi-phase, single level, which is justified in the following. It is assumed that the system is repairable (as most ICT systems are [20]), more precisely speaking it has a perfect recovery, i. e., the system has no absorbing down-states, which is a valid assumption for a regularly maintained system present in a critical infrastructure, such as the smart grid's AMI [19]. Apart from that, a multi-phase, single level restoration process is implemented by a recovery starting with the easiest recovery option and escalated further until a restoration of the system is achieved. In the given case, four escalating levels of recovery are considered:

1. **Service restart:** The simplest recovery approach is a service restart. While only non-persistent service failures are fixed doing this, the time required to restart a service (τ_{sr}) is very short. A service patch is not part of the first recovery level.

2. **VM restart:** If a service restart does not restore the operation, it is assumed that the next restoration measure is the restart of the VM hosting the failed service. To do so, the time τ_{vmr} is required.

3. **VMM restart:** After an unsuccessful recovery attempt using VM restart, the VMM hosting the failed service is restarted. This operation consumes a time of τ_{vmmr}.

4. **Hardware repair/software patch:** After all other restoration attempts have failed, another check for the cause of the outage is performed, requiring a time of τ_{dl}. Depending on the result, either a hardware repair or service patch process is issued to recover the service operation. The former is expressed by the times τ_{hrs}, τ_{hrm}, or τ_{hrl} (depending on the severity of the hardware failure), the latter takes τ_{srl}.

In addition, before any action is taken, a time τ_d is required to detect the presence of a failure. This leads to the following mapping of repair times (τ) to recovery rates (μ):

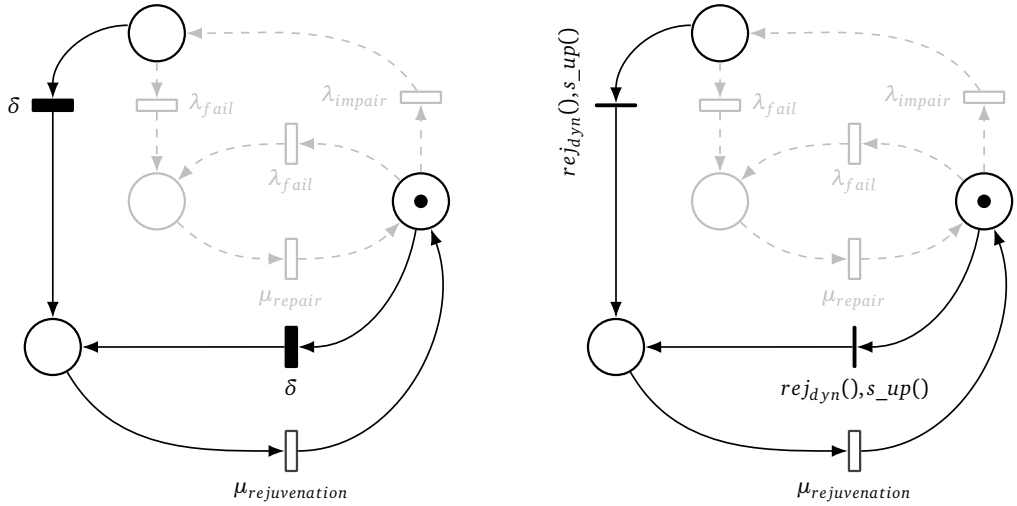
- $\mu_{sr} = (\tau_d + \tau_{sr})^{-1}$
- $\mu_{srl} = (\tau_d + \tau_{sr} + \tau_{vmr} + \tau_{vmmr} + \tau_{dl} + \tau_{srl})^{-1}$
- $\mu_{vmr} = (\tau_d + \tau_{sr} + \tau_{vmr})^{-1}$
- $\mu_{vmmr} = (\tau_d + \tau_{sr} + \tau_{vmr} + \tau_{vmmr})^{-1}$
- $\mu_{hrs} = (\tau_d + \tau_{sr} + \tau_{vmr} + \tau_{vmmr} + \tau_{dl} + \tau_{hrs})^{-1}$
- $\mu_{hrm} = (\tau_d + \tau_{sr} + \tau_{vmr} + \tau_{vmmr} + \tau_{dl} + \tau_{hrm})^{-1}$
- $\mu_{hrl} = (\tau_d + \tau_{sr} + \tau_{vmr} + \tau_{vmmr} + \tau_{dl} + \tau_{hrl})^{-1}$

During the recovery, the service is assumed to be unavailable and non-performing, because a failure of either component (service, VM, VMM, or substrate) leads to a non-available service, i. e. a performance of 0. This holds as long as no redundancy options (such as replication) are considered. The influence of such measures is discussed in Section 5.3.4.5.

5.3.4.5 Modeling Performability-Enhancing Methods

In the upcoming paragraphs, the modeling of performability-enhancing methods is described, featuring service rejuvenation techniques first; service replication second.

Service Rejuvenation. As described in Section 4.4.2, the introduction of a service rejuvenation may lead to an overall service performance enhancement. However, finding an appropriate rejuvenation interval requires a careful analysis, as every service restart may reduce the infrastructure's availability, which needs to be balanced against the possible performance benefits, particularly in critical infrastructure scenarios. For the given scenario, the rejuvenation policies introduced in Section 4.4.2 are further specified by differentiating them in two dimensions (the third dimension "target" is given implicitly by the Petri net model). In the first dimension ("timing") two different rejuvenation activation intervals are analyzed. The first activation threshold is implemented statically, meaning the rejuvenation interval (δ) is set to a deterministic value. The second is a dynamic activation threshold based on the currently achieved performance of the system, which is realized as a guard function $\left(rej_{dyn}()\right)$ in the Petri net model. $s\_up()$ represents a guard function, which blocks the activation of rejuvenations until the system is once again in an operational state after a rejuvenation happened. Both the deterministic and dynamic rejuvenation are depicted in Figure 5.13a and Figure 5.13b, respectively. The transitions and rates belonging to the original VMM model without rejuvenation are displayed in gray to improve readability. $rej_{dyn}()$ can be defined in several ways, depending on the rejuvenation behavior that is to be implemented. In the given case, once the reward function $r()$ undershoots a certain threshold value ∇, a rejuvenation is triggered (a more formal definition on the reward function is given later on in this section). Possible alternatives would, e. g., be a



(a) Deterministic rejuvenation

(b) Dynamic rejuvenation

Figure 5.13.: Deterministic *(left)* and dynamic *(right)* rejuvenation models

sliding window technique to get a time-averaged performance measure, which is however not further elaborated in this thesis. Both guard functions $rej_{dyn}()$ and $s\_up()$ are defined as follows:

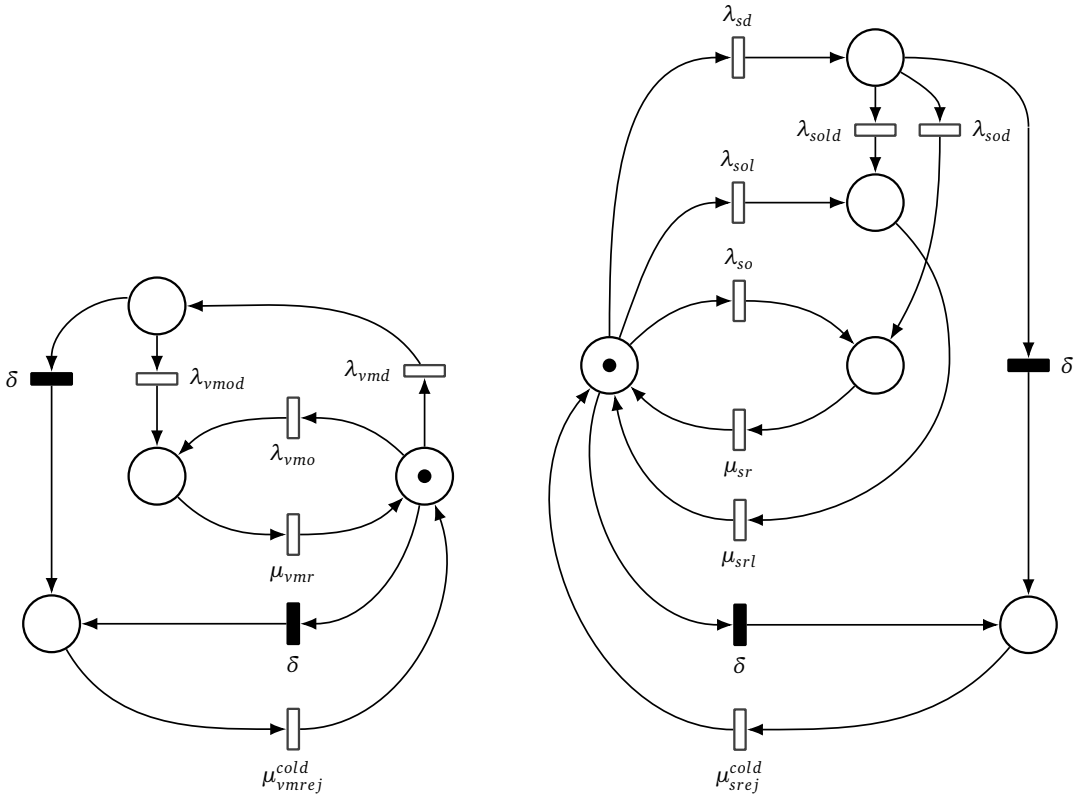
$$rej_{dyn}() = \begin{cases} 1 & \text{for } r \leq \nabla \\ 0 & \text{otherwise} \end{cases}$$

$$s\_up() = \begin{cases} 1 & \text{for } mark(\text{"}serviceDown\text{"}) == 0 \\ 0 & \text{otherwise} \end{cases}$$

The $mark()$ function is used in Petri/reward nets to indicate the number of tokens present in a certain place.

The second dimension ("strategy") differentiates between cold, warm and migrate rejuvenation. The base case is a cold rejuvenation, which is described in Section 4.4.2 and depicted in Figure 5.14. As previously highlighted, it is noteworthy that the repair rate ($\mu_{rejuvenation}$) introduced in the rejuvenation model is higher than the repair rates after a failure is encountered (μ_{repair}).

The usage of persistent memory in warm rejuvenation is depicted in Figure 5.15. To model this behavior, four new rates are introduced: On VM level, η_{vm}, representing the VM suspending time and μ^{warm}_{vmrej} for the re-instantiation of a VM after its warm rejuvenation, where $\frac{1}{\mu^{warm}_{vmrej}} + \frac{1}{\eta_{vm}} < \frac{1}{\mu^{cold}_{vmrej}} \ll \frac{1}{\mu_{vmr}}$. On service level, η_s represents the service suspending time and μ^{warm}_{srej} stands for the service restoration after a warm rejuvenation, where $\frac{1}{\mu^{warm}_{srej}} + \frac{1}{\eta_s} < \frac{1}{\mu^{cold}_{srej}} \ll \frac{1}{\mu_{sr}}$. In addition, both μ^{warm}_{vmrej} and μ^{warm}_{srej} are not constant values, but distribution functions, which have a variable rate depending on the level of rejuvenation currently running: If a VMM level rejuvenation is active, the fast re-initialization of a VM's services is possible, in cases of a VM or service level rejuvenation, this is not the case, so the restoration will proceed slower. The distribution functions $\mu_{vmrej}()$ and $\mu_{srej}()$ are defined as follows:



(a) Cold VM rejuvenation

(b) Cold service rejuvenation

Figure 5.14.: Cold (deterministic) service rejuvenation models

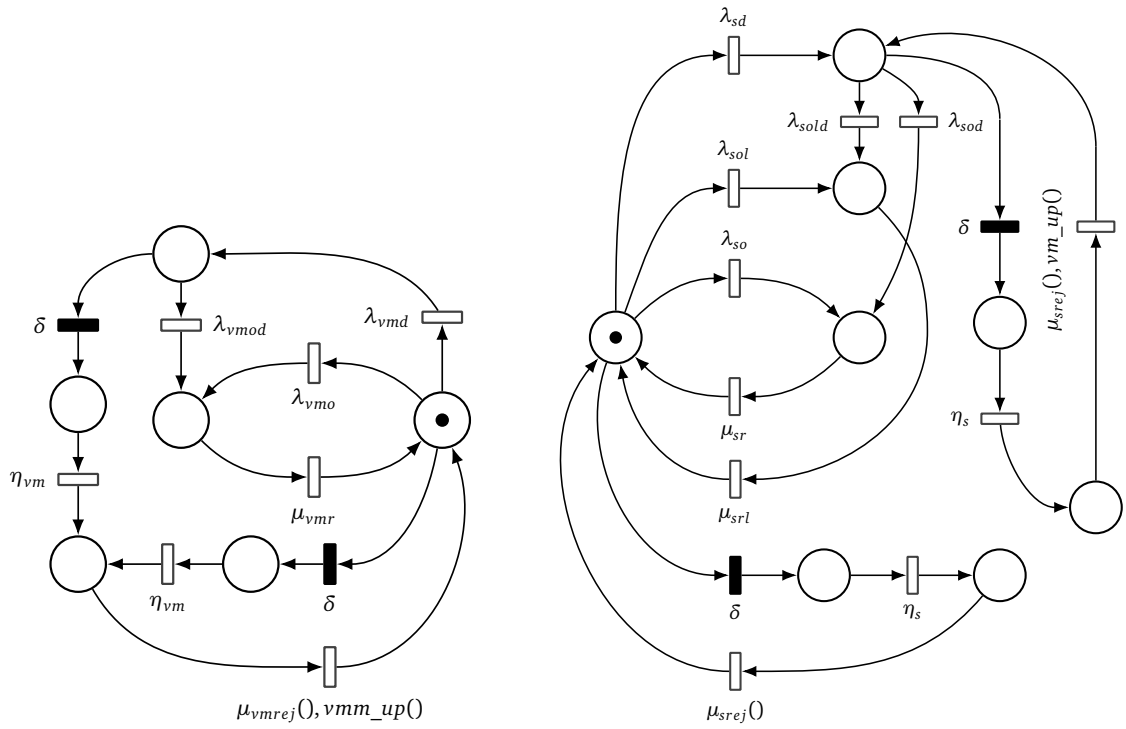
$$\mu_{vmrej}() = \begin{cases} \mu^{warm}_{vmrej} & \text{for } mark(\text{"}vmmRejuvenation\text{"}) == 1 \\ \mu^{cold}_{vmrej} & \text{otherwise} \end{cases}$$

$$\mu_{srej}() = \begin{cases} \mu^{warm}_{srej} & \text{for } mark(\text{"}vmmRejuvenation\text{"}) == 1 \\ \mu^{cold}_{srej} & \text{otherwise} \end{cases}$$

The live-migration events performed during a migration rejuvenation is modeled in Figure 5.16, where ψ_{vm} and ψ_s are the virtual machine's and service's transfer times encountered during live migration. The requirement that both the substrate and the VMM hosting the migrated VM and its service need to be restored before a re-migration takes place is realized by the guard functions $vmm\_up()$ and $vm\_up()$, defined as:

$$vmm\_up() = \begin{cases} 1 & \text{for } mark(\text{"}vmmUp\text{"}) == 1 \,||\, mark(\text{"}vmmImpaired\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$

$$vm\_up() = \begin{cases} 1 & \text{for } mark(\text{"}vmUp\text{"}) == 1 \,||\, mark(\text{"}vmImpaired\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$



(a) Warm VM rejuvenation **(b)** Warm service rejuvenation

Figure 5.15.: Warm (deterministic) service rejuvenation models

To be able to employ a migration rejuvenation, another prerequisite is the existence of a backup server, i. e. $|\nu| \geq 1$. It directly follows that in an AMI with no substrate backups, no migration rejuvenation is usable.

Service Replication. Several assumptions regarding the usage of replication mechanisms are given in the following. For the remainder of this thesis, the assumptions stated next are present (in case replication is employed):

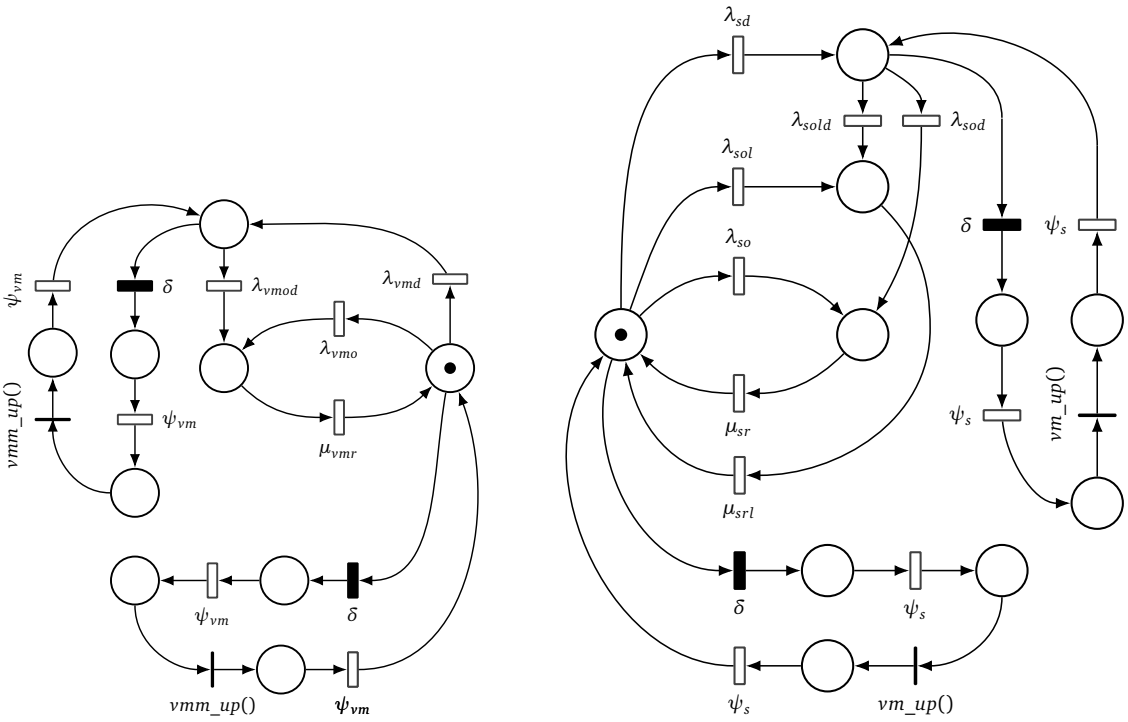
1. Replicas are assumed to feature the same failure, impairment and repair parameters as their parent service.

2. All services of similar type use the same amount of replicas.

3. All replicas employ the same rejuvenation as their parent service (timing & strategy).

4. Replicas are not used for load-balancing purposes. While investigating the benefits of load-balancing is certain a worthwhile endeavor, it is not within the scope of this thesis. Instead, research in this area is part of future work.

Also, it is argued here that the most suitable approach for replication discussed in Section 4.4.2 is a hosting of replica system on the same VMM. The reason is that substrate failures are mitigated by migration mechanisms of VNFA, which redistribute VMs/services once they get impacted by a substrate failure. Therefore, hosting a replica on another server to mitigate substrate failures would lead to additional communication overheads. Because of that, replication is only used to mitigate VM and service impairments in the following.



(a) Migration VM rejuvenation

(b) Migration service rejuvenation

Figure 5.16.: Migration (deterministic) service rejuvenation models

Due to the separation of substrate, VMM, VM, and service layer in the DSPN model (cf. Figure 5.17), replication may easily be integrated by duplicating the respective parts of the model. Including the assumptions stated in this paragraph, replication is modeled by doubling the VM and service part for each replica.

5.3.4.6 Micro Model Generation

In the upcoming section, the generation of the VNFA and SSMA micro models is explained in detail.

VNFA Model Generation. The performability micro model needs to be able to estimate both a service's performance and its availability. While the availability analysis only distinguishes between two possible system states (namely available or unavailable), the performance model needs to include several levels of performance degradation. To do so, the separately defined Petri net models of substrate, VMM, VM and service layer (Figures 5.10, 5.11a, 5.11b, and 5.12) are merged, extended, and—by adding reward functions to measure the achieved performability—converted to an SRN. Furthermore, if a deterministic timing strategy is employed, it is required to alter the used SRN to a DSRN, allowing the definition of immediate, exponential and deterministic transitions. Its underlying stochastic process is thereby also changed from a CTMC to a Markov Regenerative Process (MRGP).

The resulting model is depicted in Figure 5.17, where the substrate part is located in the lower right, VMM part in the lower left, VM part in the upper left and the service part in the upper right. To merge the separate models, the dependencies between the different layers are added through immediate transitions on VMM, VM, and service layer which represent the results of happenings on

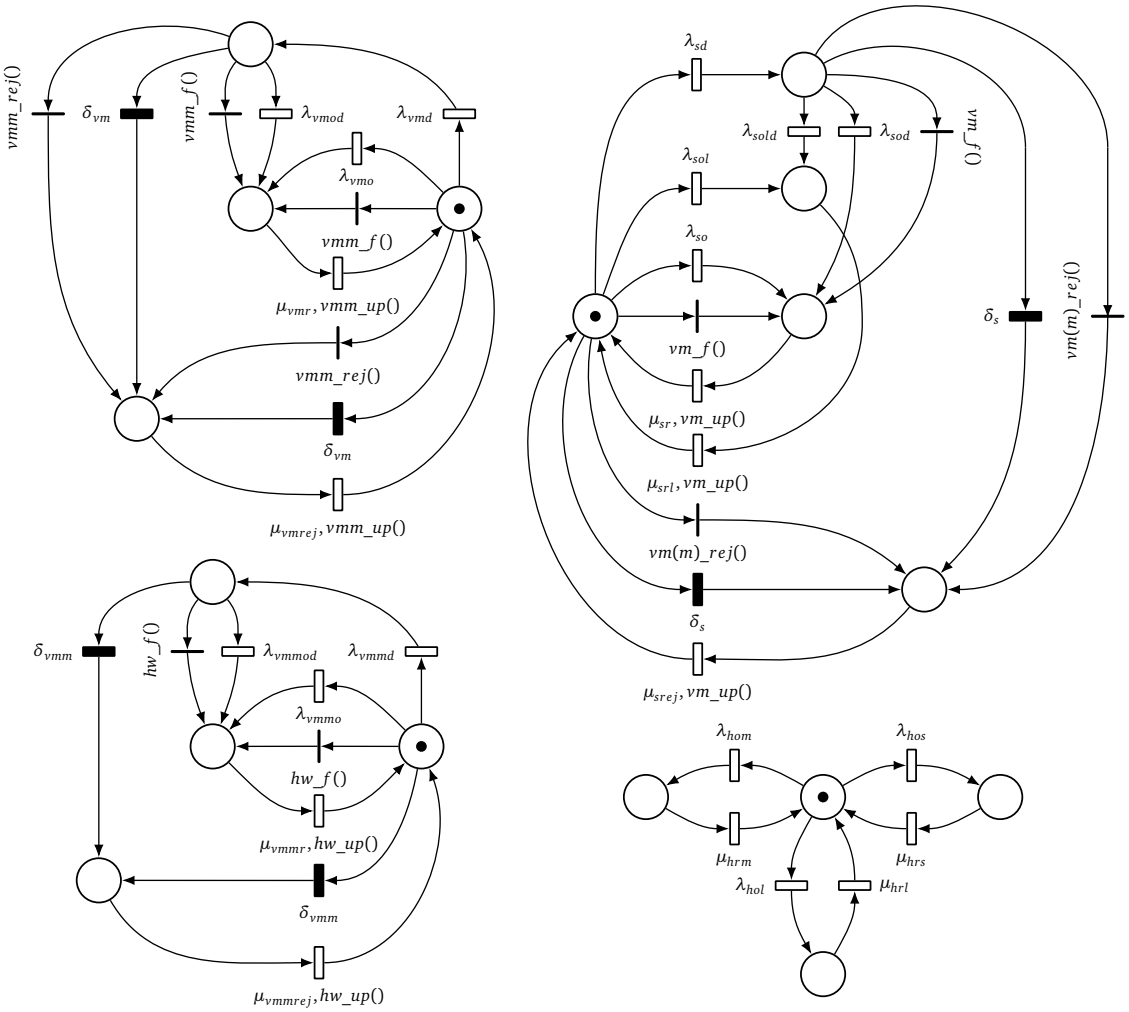


Figure 5.17.: Merged VNFA performability stochastic reward net model (cold, deterministic rejuvenation)

the lower layers. This leads to the following definitions of the guard functions $hw\_f()$, $vmm\_f()$, $vmm\_rej()$, $vm\_f()$, and $vm(m)\_rej()$:

$$hw\_f() = \begin{cases} 1 & \text{for } mark(\text{"}substrateUp\text{"}) == 0 \\ 0 & \text{otherwise} \end{cases}$$

$$hw\_up() = \begin{cases} 1 & \text{for } mark(\text{"}substrateUp\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$

$$vmm\_f() = \begin{cases} 1 & \text{for } mark(\text{"}vmmDown\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$

$$vmm\_up() = \begin{cases} 1 & \text{for } mark(\text{"}vmmDown\text{"}) == 0 \\ 0 & \text{otherwise} \end{cases}$$

$$vmm\_rej() = \begin{cases} 1 & \text{for } mark(\text{"}vmmRejuvenation\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$

$$vm\_f() = \begin{cases} 1 & \text{for } mark(\text{"}vmDown\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$

$$vm\_up() = \begin{cases} 1 & \text{for } mark(\text{"}vmDown\text{"}) == 0 \\ 0 & \text{otherwise} \end{cases}$$

$$vm(m)\_rej() = \begin{cases} 1 & \text{for } mark(\text{"}vmmRejuvenation\text{"}) == 1 \,||\, mark(\text{"}vmRejuvenation\text{"}) == 1 \\ 0 & \text{otherwise} \end{cases}$$

This is achieved by first adding additional states to the VMM and service parts of the model, resulting in the model in Figure 5.17.

To measure the performability metrics introduced in Section 2.1.1.3, a reward function r, where $r : S \to \mathbb{R}$ is introduced. Because the reward function is later on used in the performance analysis in Chapter 6, at this point, only a brief introduction is given. As also stated in the performability background, reward functions can either be rate- or impulse-based. The reward function r in this case is rate-based, with the reward gained per unit of time for each state $i \in S$ is denoted by $r(i)$. In the given model, the performance measure P is used as the reward rate per unit of time. This results in the reward rates displayed in Equation (5.1).

$$r(i) = 1 - \begin{cases} 0 & \text{if state } i \text{ is fully operational} \\ \alpha & \text{if state } i \text{ has an impaired VMM} \\ \beta & \text{if state } i \text{ has an impaired VM} \\ \gamma & \text{if state } i \text{ has an impaired service} \\ (\alpha+\beta) & \text{if state } i \text{ has an impaired VMM and VM} \\ (\alpha+\gamma) & \text{if state } i \text{ has an impaired VMM and service} \\ (\beta+\gamma) & \text{if state } i \text{ has an impaired VM and service} \\ (\alpha+\beta+\gamma) & \text{if state } i \text{ has an impaired VMM, VM and service} \\ 1 & \text{else} \end{cases} \tag{5.1}$$

To investigate the performance of a VNFA, the model from Figure 5.17 is used to model the performability (as defined in Section 2.1.1.3) of services in a VNFC. For the transient case, Equation (5.2) is used, Equation (5.3) for the steady-state case.

$$Perf(j,t) = \prod_{\substack{hw \in S_j^{hw}, vmm \in S_j^{vmm}, \\ vm \in S_j^{vm}, s \in S_j^{s}}} \pi_{hw}(t) r(hw) \pi_{vmm}(t) r(vmm) \pi_{vm}(t) r(vm) \pi_s(t) r(s) \tag{5.2}$$

$$Perf(j) = \prod_{\substack{hw \in S_j^{hw}, vmm \in S_j^{vmm}, \\ vm \in S_j^{vm}, s \in S_j^{s}}} \pi_{hw} r(hw) \pi_{vmm} r(vmm) \pi_{vm} r(vm) \pi_s r(s), \tag{5.3}$$

where j is a VNFC, $S_j^{hw}, S_j^{vmm}, S_j^{vm}$, and S_j^s are j's hardware, VMM, VM, and service states, respectively. $\pi_j(x,t)$ is the transient, $\pi_j(x)$ the steady-state probability of j's $x \in S_j^x$ being non-empty.

SSMA Model Generation. In a similar manner as the VNFA model, the stochastic reward net model for SSMA is generated using the information stated in Section 5.3.4.6 and stripping the VMM and VM parts from the model. The substrate part of the model stays structurally identical, while the service part faces two alterations: First, the immediate transitions leading to failure states due to outages of the lower layer are now considering substrate instead of VM failures; second, rejuvenation actions in the VMM or VM parts of the model, which previously lead to a transition to the rejuvenation state of the model, are removed. This leads to the DSRN depicted in Figure 5.18. The guard function $hw\_down()$ is identical to the one defined in the VNFA micro model previously.

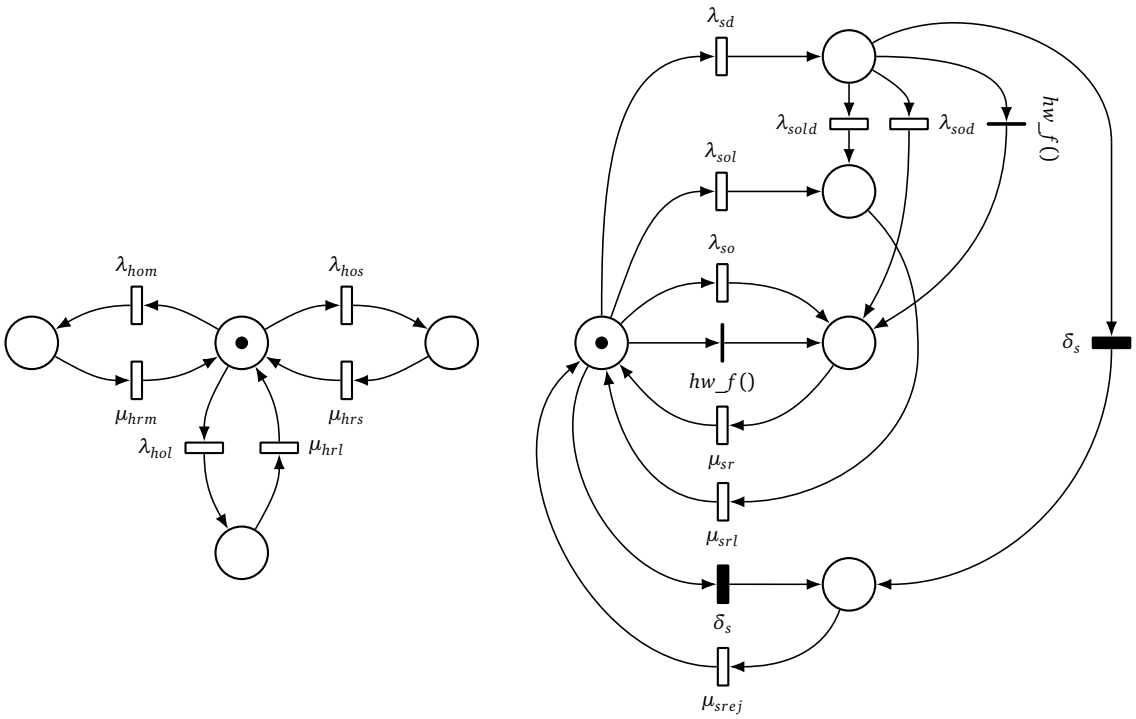


Figure 5.18.: SSMA performability stochastic reward net model (cold, deterministic rejuvenation)

The reward function $r()$ is basically the same as in the VNFA model, however, all performance impairments due to VMM or VM aging are stripped from the function, resulting in Equation (5.4).

$$r(i) = 1 - \begin{cases} 0 & \text{if state } i \text{ is fully operational} \\ \gamma & \text{if state } i \text{ has an impaired service} \\ 1 & \text{else} \end{cases} \tag{5.4}$$

5.4 Overall Model

To construct the overall AMI and service model, first, the main ideas and prerequisites are covered in Section 5.4.1, before the actual model generation is described in Section 5.4.2.

5.4.1 Idea and Prerequisites

To generate the overall model, the macro (Section 5.3.3) and micro model (Section 5.3.4) need to be combined, as well as—if VNFA is used—the network embedding algorithm (Section 4.3.2).

Using the macro model, the micro models for each substrate entity are generated. The resulting micro models are in turn analyzed to determine the performability measures required to generate the overall model, which itself is a queuing network.

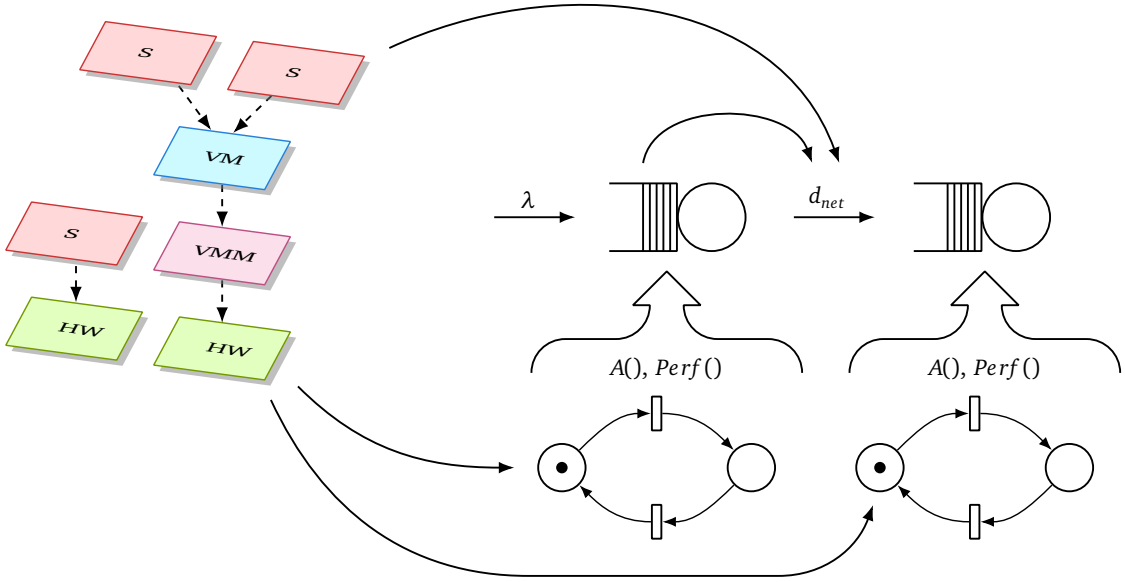


Figure 5.19.: Overall VNFA AMI model

Moreover, the network delays encountered during the data transfer in between different servers use the information from the macro model—to determine the queuing network's structure and link bandwidth—as well as the results of the queuing network itself to estimate the data rate sent over a link. A scheme of the overall model is depicted in Figure 5.19.

To construct the overall model, several steps are to be taken beforehand, listed in the following:

1. **Realization of network request:** As detailed in Section 4.3.2, using the VNFR of all users, their VNF-FGs can be constructed applying the methodology described in the aforementioned section. For SSMA, the network request is already inherently present.

2. **Embedding of request onto substrate graph:** The mapping algorithm of the VNF-FGs in case of VNFA or the network requests in SSMA is introduced in Section 4.3.3. The remainder of this thesis postulates that the initial embedding is completed successfully.

3. **Generation of macro model:** After the embedding is performed, the information is used to construct the macro model by adding the failure and repair rates for each of the services, (VMs and VMMs in case of VNFA) and substrate entities. In addition, the links between the substrate nodes are given their respective bandwidths as well as failure and repair rates.

4. **Generation of micro models:** Using the macro model information, micro models are generated for each substrate node. This in turn allows the estimation of the performability and availability measures for each substrate node and its hosted services, which is subsequently handed over to the overall model.

5.4.2 Generation of Overall Queuing Network Model

Queuing systems and networks, as briefly introduced in background Section 2.1.1.2 are used to form the overall model and estimate the performance of the SSMA and VNFA AMIs. To create the queuing network used for the analysis, the macro and micro models generated before are transformed into queuing networks. This process is described next, first covering the properties of the queuing network models in general, second differentiating between and adapting the method to SSMA and VNFA, respectively.

5.4.2.1 General Properties

Several properties apply for both VNFA and SSMA. First, the queuing networks are open, modeling the traffic arriving from smart meters, which exit the system at the provider's HES (represented by a sink). As smart meters generate packets in regular time intervals, the source's arrival rate λ_{sm_o} follows a deterministic distribution.

5.4.2.2 Converting Non-Virtualized Networks

Due to the structure of the non-virtualized SSMA networks, the SSMA macro model layout can almost completely be used in the queuing networks with few exception listed in the following.

- Smart meters are not directly included in the queuing networks, instead they act as the source of data arriving into the queuing network. Since the generated queuing networks only possess a single source shared among all users, the arrival rate is adjusted by altering the deterministic distribution (λ_{sm_o}) to a uniform arrival rate (suggesting non-synchronized sending of metering data). This results in $\lambda_{sm} = \frac{1}{\lambda_{sm_o}/|\{V^s \mid d^s_{type}=\text{"}smart\ meter\text{"}\}|-0}$. It is assumed that $|\{V^s \mid d^s_{type} = \text{"}smart\ meter\text{"}\}| \geq 1$, which is a safe assumption, as $|\{V^s \mid d^s_{type} = \text{"}smart\ meter\text{"}\}| < 1$ would imply that no user would own a smart meter. Such a case is neither relevant nor further regarded in this thesis.
- A drain is added to the generated queuing networks.
- Third party services are the most complex part of the conversion process as they require the generation of copies of the packets sent to the queuing system representing the data concentrator of an SSMA infrastructure. This is modeled by the usage of forks, which allows to create multiple tasks out of a single packet. Depending on the number of TPS used by a user, the number of outgoing links of the fork defines the number of tasks the packet is split into. Using k TPSs, the overall number of tasks is k+1, where k tasks are forwarded to the respective TPSs, while a single task remains to be sent to the data concentrator.

The path from the households to the HES is modeled in two sections. The queuing network Q_h models the connection from the data concentrators to the HES and is separated from the network Q_{sm} that connects the smart meters to the data concentrators. The steps highlighted above are only necessary for Q_{sm}. For Q_h the generation of the queuing network is more straightforward. The combined concentrators act as the source of packages. Similar to the source in Q_{sm}, their arrival rates λ_{dc_o} are deterministic, leading to a combined uniform distribution of $\lambda_h = \frac{1}{\lambda_{dc_o}/|\{V^s \mid d^s_{type}=\text{"}data\ concentrator\text{"}\}|-0}$, similar to the calculation of the overall arrival rate λ_{sm} in the HAN. The only other node is the HES which is connected directly to the drain. This part about Q_h is mainly included for the sake of completeness because the HES is not within the scope of the studies done in this thesis. This leads

to the queuing network generation algorithm in Algorithm 5.1 for Q_{sm}. The generation of Q_h is not further elaborated upon.

Algorithm 5.1: Generation of Q_{sm} of SSMA AMIs

Input: Macro model graph G, micro model petri nets P
Output: SSMA queuing network Q_{sm}

```
Function main(G, P, Q_sm):
    senders ← 0;
    stationConnections ← ∅;
    foreach s ∈ G.getServices() do
        switch s.type do
            case "smart meter" do
                senders++;
                break;
            case "gateway" do
                qg ← new QueuingSystem();
                Q_sm.addSystem(qg);
                qg.addForks(|s.getOutgoingConnections()|);
                stationConnections.add(qg, s.getConnections());
                break;
            otherwise do
                q ← new QueuingSystem();
                Q_sm.addSystem(q);
                q.setServiceRate(μ_s · P.getPerformance(s));
                stationConnections.add(q, s.getConnections());
                break;
    qs ← new Source();
    Q_sm.addSource(qs);
    qs.setArrivalRate(senders · λ_sm);
    qd ← new Drain();
    Q_sm.addDrain(qd);
    foreach q ∈ Q_sm do
        q.setConnections(stationConnections.getConnections(q));
```

5.4.2.3 Converting Virtualized Networks

As described in the previous section, converting non-virtualized networks is mostly straightforward. Converting virtualized networks, however, presents itself as a more challenging task that is explained next. After describing the general idea of the algorithm, first, an approach ignoring TPSs is introduced, as they introduce significant complexity to the networks that would hinder the basic understanding. The addition of TPSs is explained subsequently after the basic algorithm is defined. Again, as previously described, the path from smart meter to HES is split in two queuing network Q_{sm} and Q_h.

The main challenges which lead to complexity during a conversion of VNFA AMI systems are:

1. Multiple types of services run on the same queuing station.

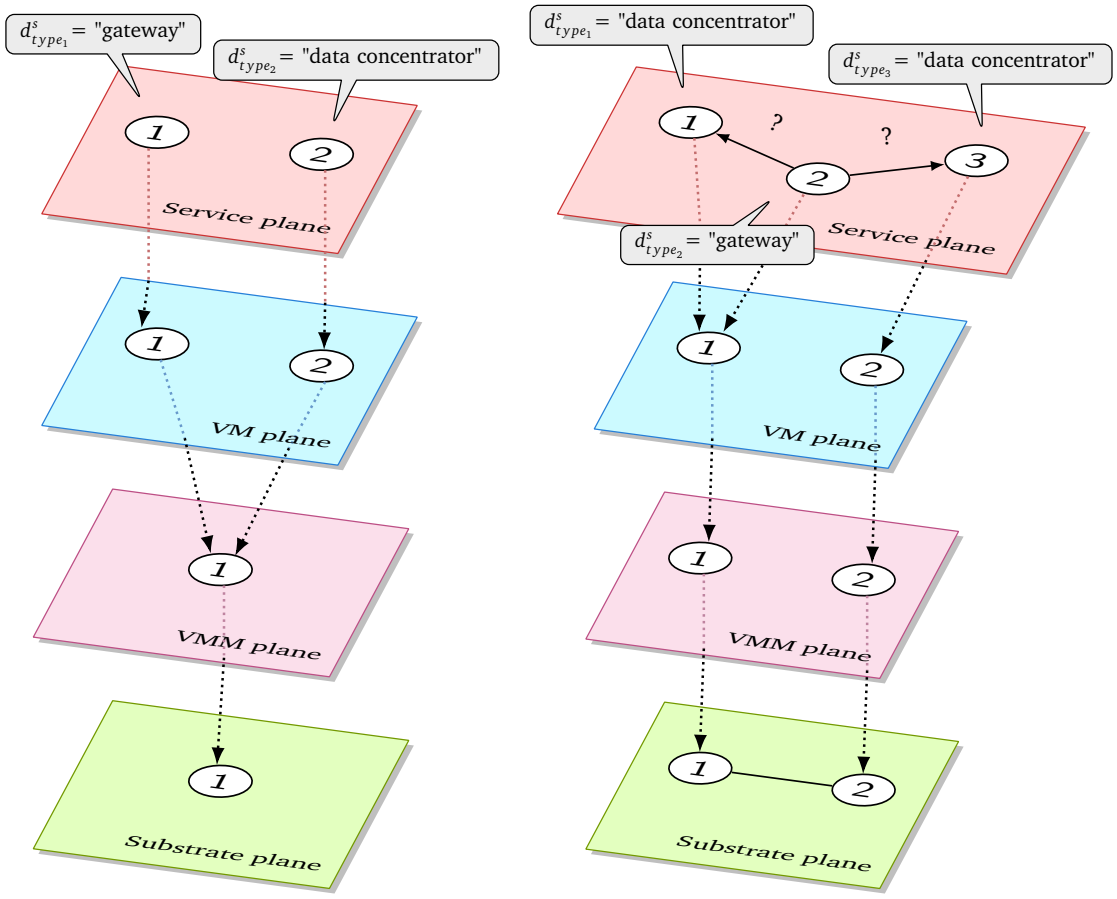
2. Queuing networks do not support explicit destination addresses. Their routing behavior is limited to probabilistic distributions and processing by the station they arrive to.

The first challenge (see Figure 5.20a) occurs if at least two different services are hosted on the same substrate node. Since every type of service may require different service times, the queuing network has to be extended to use multiple classes—one class for each service type. This introduces the challenge that the conversion of packages from one class to another is performed if (and only if) it is required.

Second, a challenge is present if substrate nodes are shared among multiple users (see Figure 5.20b). This results in the need to route packages through the queuing network, which is not possible, as packages are processed and sent to the next queuing system (if multiple links exist, a stochastic distribution is used to determine the target of a package). For SSMA AMI queuing networks this is irrelevant, as only a single link to sent the package to does exist. For VNFA, in contrast, multiple possible links exist, with

$$\sum_{e \in E_q^o} P(e) = 1,$$

where E^o is the set of outgoing links in the whole network, $E_q^o \subseteq E^o$ is the set of outgoing links for queuing station q and $P : E^o \to [0, 1]$ is the function that assigns probabilities to an outgoing link. By adjusting those probabilities, the package still cannot be routed exactly, however the number of packages that are sent to each queuing station can be adjusted to fit in steady-state.



(a) Service rate challenge in VNFA AMI conversion

(b) Routing challenge in VNFA AMI conversion

Figure 5.20.: Challenges during VNFA AMI conversion

Variable	Description
S	Set of server nodes
T	Set of TPS types
H	Set of users
$H_s \subseteq H$	Set of users with virtual gateway in node $s \in S$
$H_{s,s'} \subseteq H_s$	Set of users with virtual gateway in node $s \in S$ and virtual data concentrator in $s' \in S$
$H^t_{s,s',s''} \subseteq H_{s,s'}$	Set of users as in $H_{s,s'}$ with substrate node of TPS $t \in T$ in node $s'' \in S$
$H^t_{s,s',\_} \subseteq H_{s,s'}$	Set of users as in $H_{s,s'}$ with no TPS of type $t \in T$ assigned
F	Set of forks
$f_{s,s'} \in F$	Fork that multiplies packages that go from $s \in S$ to $s' \in S$

Table 5.5.: Variables used for calculating the link probabilities

If only high priority services are considered, the model contains at least two different classes. Packages arrive at the network in the *gateway* class. As the name suggests, packages of this class are processed by a virtual gateway service. After being processed at the gateway, packages are converted to the *data concentrator* class, which is required to forward the packages to the provider's HES. In addition to those mandatory classes, additional classes may be required, depending on the number of third party services employed. Because every TPS may require a different service time, a separate class is needed. The packages are therefore converted to a TPS's class after being processed and forked at the virtual gateway node. To enable class changes, class switcher nodes are added to the queuing network. Every switcher node contains a conversion matrix M_c defining stochastic probabilities p_{ij} representing the likelihood of a package changing from class i to j. Since all values are probabilities, for a $m \times n$ conversion matrix it holds that: $\sum_{i=0}^{m} p_{ij} = 1$. Because the virtual machines belonging to the VNF-FGs of a user may be distributed all over the substrate network, each connection is given its own class switchers.

After explaining the queuing network class switching process, in the following, the calculation of the link probabilities is covered in detail. Table 5.5 defines all variables used. Several link probabilities are trivial, namely all links with the property $|\{E^o_s\}| = 1$, i. e. all queuing stations which have an *outdegree* of 1. However, server nodes most likely feature multiple outgoing links—one link per server node. In the following, the calculation of the probabilities is explained.

The goal is to find an equation to calculate the probability $P_t(s'', f_{s,s'}), t \in T, s, s', s'' \in S$ that a package destined for TPS type t is routed to the server node s'' if the package currently is in fork $f_{s,s'}$. As all users send the same amount of packages, the problem reduces to counting the number of users. The number of households that send packages to fork $f_{s,s'}$ is $|H_{s,s'}|$. For every TPS type a package is either sent to a virtual TPS node or the household has no TPS of type t. This leads to the following equations:

$\forall\, t \in T : \left(\sum_{s'' \in S} |H^t_{s,s',s''}|\right) + |H^t_{s,s',\_}| = |H_{s,s'}|$, and

$\forall\, t \in T : \left(\sum_{s'' \in S} \frac{|H^t_{s,s',s''}|}{|H_{s,s'}|}\right) + \frac{|H^t_{s,s',\_}|}{|H_{s,s'}|} = 1.$

The required probabilities are obtained by:

$P_t(s'', f_{s,s'}) = \frac{|H^t_{s,s',s''}|}{|H_{s,s'}|}$.

After the overall model generation, the service rates (μ) of each queuing station may be adjusted based on the package type processed and the performability measures obtained from the micro model. Based on this model, the analysis is performed in the upcoming Chapter 6.

5.5 Summary

In this chapter, first, the currently suggested AMI models are discussed and classified. After that, their shortcomings are discussed, leading to the finding that current models are either abstract, non-quantitative or highly specialized, quantitative models. However, the the best of the author's knowledge, no appropriate model fulfilling the requirements of virtualized AMIs currently exists. The requirements for a model which supports the analysis of performability in virtualized AMIs are thereafter specified explicitly and realized by the separation of the model into three hierarchical parts, being the macro, micro and overall model.

The macro model employs a network-graph based model to offer a separation of service, VM, VMM, and substrate plane. This enables a dynamic embedding of services into the virtualization planes which are in turn embedded into the substrate. This way, a flexible assignment of services to virtual machines and substrate hardware is possible.

The micro model allows the calculation of performability in AMIs after the initial embedding finished. To do so, a DSPN-based approach is used that enables the modeling of failures and their respective influences on the overall infrastructure. This is extended by additionally including performability measures, thereby forming a DSRN model. As all software-based entities (i. e. VMMs, VMs, and services) are prone to performance impairments (e. g., due to aging), the micro model adds impaired states to the model that merely influence the achieved reward of the services. The resulting reward nets can properly represent the performability of VNFA AMIs.

The overall model combines the information of the macro and micro models to offer a queuing network-based representation of the overall AMI system. This enables the assessment of the performance that is achieved by the AMI.

6 Analysis

The goal of this chapter is the analysis of VNFA and its comparison to SSMA. In Section 6.1, the assumptions made during the analysis are stated. Next, three scenarios used in this chapter are described: In the first, a single service hosted on substrate (i. e. SSMA) is compared to a virtualized VNFC realizing the same function (i. e. VNFA). The results are used to compare the performability as well as to highlight how to find the optimal rejuvenation/replication method for a single service. The second scenario analyzes the performability of a single user that uses either VNFA or SSMA, respectively. The third scenario extends the use case from a single user to the whole city of Passau. Subsequently, a performability assessment of the three example scenarios is executed in Sections 6.2.1, 6.2.2, and 6.2.3, respectively. Each of the evaluations first focuses on the dependability and performance assessment of SSMA, then VNFA. The results of the analyses are evaluated and conclusions are drawn in Section 6.2.4. To validate the feasibility of virtualizing AMI components, in Section 6.3, a vSMGW is developed and analyzed. Using a proof-of-concept implementation of a vSMGW, both its dependability and performance are evaluated in Section 6.3.3. Results and conclusions are presented in Section 6.3.4. Furthermore, the costs of SSMA and VNFA are calculated and compared in Section 6.4, which is done by first deriving a cost function that is in turn used to evaluate both AMI architectures. A summary of the chapter is given in Section 6.5.

Contents

6.1 Description of Assumptions and Scenarios **118**
6.1.1 Assumptions 118
6.1.2 Single Service Scenario 122
6.1.3 Single User Scenario 122
6.1.4 Passau City Scenario 123
6.2 Performability Assessment **124**
6.2.1 Evaluation: Single Service Scenario 125
6.2.2 Evaluation: Single User Scenario 136
6.2.3 Evaluation: Passau City Scenario 142
6.2.4 Result Comparison 154
6.3 Proof-of-Concept: Virtualized Smart Meter Gateway **156**
6.3.1 Gateway Task Description 157
6.3.2 Proof-of-Concept Implementation 158
6.3.3 vSMGW Performability Analysis 158
6.3.4 Results 164
6.4 Cost Analysis **165**
6.4.1 Deriving a Cost Function 165
6.4.2 Evaluation and Results 167
6.5 Summary **169**

6.1 Description of Assumptions and Scenarios

This section contains three main parts: First, the assumptions made throughout the example scenarios are described in Section 6.1.1. After that, the three example scenarios used to analyze and compare VNFA to SSMA are explained: In Section 6.1.2, a scenario analyzing and optimizing a single service is defined. Second, a scenario analyzing a single user is described in Section 6.1.3. Third, in Section 6.1.4, a scenario comprising an imaginary, VNFA-enabled AMI is assumed to be implemented in the city of Passau, Germany, and compared to SSMA.

6.1.1 Assumptions

In this section, the assumptions for the analyses performed later are illustrated. The main areas assumptions are taken in contain the failure and recovery properties (Section 6.1.1.1), the properties of the communication networks (Section 6.1.1.2) and location distributions (Section 6.1.1.3). Further assumptions of the analyses are covered in Section 6.1.1.4.

6.1.1.1 Failure and Recovery Properties of Entities

- **Useful life:** During the upcoming performability analysis, the system is expected to perform within its useful life phase, as discussed in Section 5.3.1. The restriction of the analysis to the useful life period is done to show only the failure rate during the operational period of the devices. This is of particular interest, because it allows to make estimations concerning long-term failure rates and the system's dependability. In the given analysis, the assumption is made that all systems are comprised of electronic components, whose lifetime and failure rate can be estimated using the "bath tub curve", depicted in Figure 5.4. The figure also depicts the development of the failure rate λ over the life time of a system and the position of the useful life phase.

- **Failure types:** The evaluations are based solely on the system model created in Chapter 5, the failure and repair rates illustrated in Tables 6.3, 6.5, 6.7, and 6.9, respectively, as well as the assumptions stated in this paragraph. Other influences, such as operation conditions (temperature, humidity and similar effects) are not respected in the assessment. Failure of end devices in the HAN are—except for smart meters—not considered in the evaluation, because the failure of such end devices is not expected to cause a non-negligible disturbance in the function of the AMI.

- **Failure rates:** Also, during the useful life, the failure rate is assumed to have a quasi-constant value [71]. The failures that occur during this period are random in nature, i. e. neither caused by infant mortality or wear out. To estimate the numeric failure rate values for the example scenarios, the estimations given by Kim *et al.* [72], Almeroth *et al.* [3], Matos *et al.* [94], Shojaee *et al.* [148], and Manel *et al.* [90] are used as a guideline; however, the articles vary regarding the information provided and the failure as well as repair rates. All relevant information is summarized in Table 6.1. The information is ordered by publication date of the source, no implication regarding data quality is intended. A general assumption is that the virtualization of a service does not impair the quality of the software itself, i. e. $\lambda^s_{nv} = \lambda^s_v$. This is not to be confused with the additional failures originating from the VMM or VM, which are to be regarded in addition to the service failures.

- **Repair rates:** The repair rates differ depending on two factors: First, there is a difference between entities under direct supervision of the service provider and those under control of a user (i. e. smart meter, router, and—in case of SSMA—the gateway). Second, depending on being virtualized or not, services experience varying repair rates, as virtualized environments offer more restoration options.

- **Statelessness:** Moreover, all VMs and services are assumed to be stateless, so they can be re-instantiated after a failure. The assumption is that no large amounts of data are gathered in virtual machines between the sending of packages. Due to the immediate forwarding of packages after data reception, a negligible amount of data is lost so that this assumption can be made without major impairment of the model.

Source	λ_{ho}	λ_{hr}	λ_{vmmo}	λ_{vmmd}	λ_{vmmr}	λ_{vmo}	λ_{vmd}	λ_{vmr}	λ_{so}	λ_{sr}
[156]	-	-	-	-	-	$4.5E^{-4}$	$4.0E^{-3}$	$5.0E^{-1}$	-	-
[72]	-	-	$3.5E^{-4}$	-	$1.0E^{0}$	-	-	$2.0E^{0}$	$3.0E^{-3}$	$3.0E^{0}$
[87]	-	-	-	$1.5E^{-3}$	$1.0E^{0}$	-	$6.0E^{-3}$	$2.0E^{0}$	-	-
[170]	$1.0E^{-4}$	$4.0E^{-2}$	$1.5E^{-3}$	-	$1.5E^{-1}$	$6.0E^{-3}$	-	$1.5E^{-1}$	-	-
[94]	$4.0E^{-4}$	$6.0E^{-1}$	-	-	-	$3.5E^{-4}$	-	$2.0E^{0}$	$5.5E^{-3}$	$6.0E^{1}$
[148]	$2.5E^{-4}$	$2.0E^{-2}$	$1.5E^{-3}$	-	$4.0E^{-2}$	$4.0E^{-3}$	-	$1.5E^{-1}$	-	-
[186]	-	-	-	$1.5E^{-3}$	$1.0E^{0}$	$1.5E^{-2}$	$1.5E^{-2}$	$2.0E^{0}$	-	-
[110]	$1.0E^{-4}$	$1.5E^{-2}$	$4.0E^{-4}$	$1.5E^{-3}$	$6.0E^{-1}$	$3.5E^{-4}$	$6.0E^{-3}$	$2.0E^{0}$	-	-
[90]	-	-	-	$1.5E^{-3}$	$1.0E^{0}$	-	$6.0E^{-3}$	$2.0E^{0}$	-	-

Table 6.1.: Failure and repair rate information stated in related work

6.1.1.2 Communication Networks Properties

- **Technologies used:** From the technologies available within the example scenario, the currently most commonly suggested combination for AMI installations is assumed. Within both the LMN and HAN, the ZigBee standard is used. The NAN is equipped with PLC links; inside the WAN, DSL links are present.

- **Bandwidth:** The bandwidth in a real-life scenario is expected to fluctuate depending on the daytime and current load situation, however, such variances are disregarded within the example scenario. Static link bandwidths increase both the performance of the analysis and the reproducibility of results (due to lower amount of stochastically derived parameters). Because of that, static link bandwidths are used, which can be estimated using three different approaches: minimum, average, or maximum bandwidth; all three are derived as Algorithm 6.1 illustrates. In the upcoming scenario, a worst-case assumption is taken, therefore, the strategy parameter is set to *min*, resulting in the minimum expected bandwidth for each communication medium to be selected.

Algorithm 6.1: Algorithm to estimated a link's bandwidth

Input: Link medium, bandwith calculation strategy
Output: Bandwidth of the medium

```
minBandwidth ← medium.getMinBandwidth();
maxBandwidth ← medium.getMaxBandwidth();
switch strategy do
    case min do
        return minBandwidth;
    case avg do
        Δ ← maxBandwidth - minBandwidth;
        // α ∈ [0,1], indicating a lower or higher average bandwidth
        return minBandwidth + αΔ;
    case max do
        return maxBandwidth;
```

- **Delay:** The delays encountered during the data transfers from the users' smart meters to the service provider's HES are twofold: the processing delays which are dependent on the performance of the services on the one hand; the network delays encountered during the data transmission (see Section 4.2.2), on the other. For the network delay, only the transmission and queuing delays are regarded, as the processing and propagation delays can be safely assumed to cause a combined delay of < 1 ms, making them negligible in comparison to the other delays encountered (cf. Section 4.2.2). Also, usually, it is expected that there are various types of traffic present within communication networks. This may lead to additional latency compared to a network that only serves one service. However, it is assumed that only AMI services exist in the network because the traffic of other services cannot be predicted reliably. In-network package routing and associated delay besides the service time directly encountered in the household's router are ignored in the model.

- **Reachability:** Every substrate node is assumed to be able to reach any other substrate node that is available, even if the connection is not explicitly modeled. This assumption represents packages being routed through the Internet even without a direct connection between two substrate nodes.

6.1.1.3 Location Distributions

- **Smart meters:** The locations and number of deployed smart meters is derived by assuming that every household is equipped with a single smart meter, usually located within a switchbox. In the case that several households are situated within a single building, sub-switchboxes are used to house the different meters [27]. The distribution of smart meters is therefore logically bound to that of households, which applies for both SSMA and VNFA.

- **Gateways:** In the example scenario it is assumed that gateways are located within household premises, therefore, both the number and distribution is similar to those of the previously discussed smart meters. This assumption applies only for SSMA.

- **Servers:** The server locations and their respective influence on the VNFA is discussed extensively in Section 4.2.2. Following the results obtained in that section, servers are distributed

using a maximum dispersion scheme described in the same section. If the maximum dispersion scheme would place a server within an area that is inherently unsuited (e. g. due to other buildings or rivers covering that area), the placement is shifted to the nearest possible suitable location. The shifting is performed after the initial placement is calculated and does not influence the positions of other servers. As COTS server are only used in VNFA, this assumption does not apply to SSMA.

- **Data concentrators:** The data concentrators are located within the NAN part of an AMI network. The main purpose of data concentrators are to compress and communicate metering information to the service provider's AMI HES. The provider in turn can use this information for billing services, but also enhanced consumer services such as real-time energy analysis and communication of usage information.

- **Third party services:** In both the single user and Passau city scenario, two third party services are assumed to be available to users: an energy optimization advisor service (tps_{eo}), offering valuable insights by analyzing and evaluating users' energy demand and giving advice regarding possible ways to save energy, and a car monitoring service (tps_{cm}), allowing users to monitor the battery status of their electric cars. Third party services are not expected to be located within the mission area of the scenario; rather, they are located in a data center/servers of the third party that offers a specific service. Due to the privacy-relevant nature of the energy data sent to the third party services, it is assumed the servers processing it are located within Germany. Because third party services are only present in a virtualized form in VNFA, the assumptions made are only relevant for SSMA.

6.1.1.4 Further Assumptions

- **Message arrivals:** Inside an AMI system, several types of messages may be exchanged, such as, e. g., metering data, pricing information, firmware updates, as well as demand-response or signaling messages. While these messages are of major importance for the system and its functionality, apart from the metering data, none are part of the upcoming analysis, as they do not undergo any complex processing. For the smart metering data of each user, a deterministic arrival time with $\delta = 60$ is assumed (i. e. the user's smart meter data is sent every minute, as given in Section 2.1.2.2). It is noted here that though each user's smart metering data is arriving in a deterministic process, the messages are not sent out in a synchronized manner, which leads to an overall arrival process that is distributed in a uniform manner with $\alpha = 0, \beta = \frac{60}{n}$, where n is the number of users of VNFA. The data concentrator is sending data every 15 minutes (see Section 2.1.2.2) to the HES at the providers side, leading to a uniform distribution with $\alpha = 0, \beta = \frac{900}{m}$, where m is the number of data concentrators in the VNFA infrastructure.

- **Queuing stations:** Every queuing station is expected to model a single substrate server. The queuing network is comprised of queuing stations having an exponential service rate. In addition, it is assumed that all queues have infinite queuing capacity and use a First In – First Out (FIFO) strategy.

- **Initial state:** For every entity in the models used, the assumption is that it starts in an unimpaired, fully available state, achieving its optimal performance, unless specified differently.

- **Model analysis method:** Depending on the rejuvenation method chosen, the resulting model is either an SRN (dynamic rejuvenation timing) or a DSRN (deterministic rejuvenation timing).

SRN models are analyzed using a numerical solution approach by constructing the SRN's underlying CTMC and solving it by successive over-relaxation for steady-state and uniformization for transient analysis. DSRNs in contrast are solved by employing a simulation approach. Each simulation is repeated 50 times initially. To estimate whether these simulation runs suffice, Equation (6.1) is used.

$$R \geq \left(\frac{Z_{1-\alpha/2} \cdot s_0}{\epsilon \cdot \overline{x}_0}\right)^2, \tag{6.1}$$

where R is the number of required simulation runs, $Z_{1-\alpha/2}$ is the $1-(\alpha\,/\,2)$ quantile of the standard normal distribution, α is the allowed level of uncertainty (here 0.02), s_0 is the standard deviation of the R_0 simulation results, R_0 is the default number of runs (in the given analysis $R_0 = 50$), $\overline{x}_0$ is the mean value of the R_0 simulation results, and ϵ is the allowed level of error (here 0.02).

6.1.2 Single Service Scenario

The single service scenario is used to identify the optimal configuration of a single service in SSMA and VNFA, mainly focusing on the rejuvenation methods employed. More specifically, this means in case of VNFA, both the rejuvenation *timing* and *strategy* (as defined in Section 4.4.2) are assessed. However, it is assumed that only a single server is present in the scenario, i. e. no backup servers are present, or formally expressed: $|\boldsymbol{\nu}_i^{hw}| = 0$, where i is the server used. In case of SSMA, the rejuvenation *strategy* is limited to a cold rejuvenation approach, due to missing virtualization. In this scenario, a virtualizable SMGW service is assumed. As only a single service is evaluated in the scenario, no formal embedding is performed for VNFA, as it is assumed that $r_{cpu}^{hw} >> d_{cpu}^{s}$ and $r_{type}^{hw} = d_{type}^{s}$. The parameters used for the service are depicted in Table 6.3 for both VNFA and SSMA. Furthermore, the benefit of using service replicas is investigated.

The evaluation of the single service scenario is continued in Section 6.2.1.

6.1.3 Single User Scenario

In the single user scenario, the difference between SSMA and VNFA for a single user is evaluated. To do so, the services required for a fully functioning VNFA and SSMA are set up (i. e. a smart meter (sm), an SMGW (gw), a router (r), a data concentrator (dc), as well as two additional, optional TPSs (tps)). Besides the comparison of VNFA and SSMA, a goal of the evaluation of this scenario is to analyze and judge if the performability of both architectures is able to fulfill the requirements of AMIs elaborated in Section 2.1.2.2. A formal embedding process is also skipped here, due to the fact that the resources provided by a single server (r_{cpu}) by far exceed the resources required to serve a single user's VNF-FG ($\sum_{i \in vnffg} d_{cpu_i}^{s} \Omega_{cpu}$), i. e. $r_{cpu}^{hw} >> \sum_{i \in vnffg} d_{cpu_i}^{s} \Omega_{cpu}$, where Ω_{cpu} is the CPU overhead induced by the virtualization. Also, in contrast to the single service scenario, not only the case where no additional substrate server is present is investigated, but also the circle and all-for-all backup strategies. For reasons of simplicity, a generalization of the failure rates in the scenario is applied, which groups services (and, in case of SSMA, substrate) into two classes: simple (gr_s) or complex (gr_c). This classification is based on two contributing factors: the complexity of the functions provided as well as the amount of user interaction required. This leads to the following grouping: $gr_s = \{sm, gw, r\}$ and $gr_c = \{dc, tps\}$. The parameters assumed in the single user scenario are given in Table 6.5.

VNFA							
λ_{hos}	6.0E^{-5}	λ_{hom}	3.0E^{-5}	λ_{hol}	1.0E^{-5}		
μ_{hrs}	2.0E^{-1}	μ_{hrm}	5.0E^{-2}	μ_{hrl}	1.0E^{-2}		
λ_{vmmo}	4.0E^{-5}	λ_{vmmod}	8.0E^{-5}	λ_{vmmd}	8.0E^{-5}		
μ_{vmmr}	1.0E^{0}	μ_{vmmrej}	2.0E^{1}				
λ_{vmo}	1.0E^{-3}	λ_{vmod}	2.0E^{-3}	λ_{vmd}	1.0E^{-3}		
μ_{vmr}	2.0E^{0}	μ^{cold}_{vmrej}	3.0E^{1}	μ^{warm}_{vmrej}	6.0E^{1}		
λ_{so}	2.0E^{-3}	λ_{sol}	4.0E^{-5}	λ_{sod}	4.0E^{-3}	λ_{sd}	4.0E^{-3}
μ_{sr}	3.0E^{0}	μ_{srl}	3.0E^{1}	μ^{cold}_{srej}	6.0E^{1}	μ^{warm}_{srej}	2.4E^{2}
η_{vm}	9.6E^{2}	η_{s}	9.6E^{2}	ψ_{vm}	1.8E^{3}	ψ_{s}	1.8E^{3}
α	1.0E^{-1}	β	2.0E^{-1}	γ	3.0E^{-1}		
SSMA							
λ_{hos}	6.0E^{-5}	λ_{hom}	3.0E^{-5}	λ_{hol}	1.0E^{-5}		
μ_{hrs}	1.0E^{-1}	μ_{hrm}	2.0E^{-2}	μ_{hrl}	6.0E^{-3}		
λ_{so}	2.0E^{-3}	λ_{sol}	4.0E^{-5}	λ_{sod}	4.0E^{-3}	λ_{sd}	4.0E^{-3}
μ_{sr}	5.0E^{-1}	μ_{srl}	1.0E^{-2}	μ^{cold}_{srej}	6.0E^{1}		
γ	3.0E^{-1}						

Table 6.3.: Single service scenario VNFA and SSMA parameters

In Section 6.2.2, the evaluation and results of the single user scenario are discussed further.

6.1.4 Passau City Scenario

The investigated scenario area is estimated to be similar to the current coverage of the power supply provided by Stadtwerke Passau GmbH. The power grid of Passau supplies an approximate area of 85 km^2. The number of users (= households) in the city of Passau is 13 835[15]. While there is currently no AMI present in Passau, several different communication technologies are available, which would allow the construction of AMIs. Among others, there are Internet access options via optical fiber, DSL and cable line networks, PLC/Broadband over Power Lines (BPL), as well as wireless options such as GPRS, Universal Mobile Telecommunications System (UMTS), or Long Term Evolution (LTE) available. These options allow the realization of the link technologies mentioned in the assumptions in Section 6.1.1. There are two main goals present in this scenario: First, the dependability of a city-size AMI infrastructure is investigated especially focusing on the impact of persistent substrate failures—as introduced in Section 4.2.2—on the three backup topologies discussed in Section 4.4.1. Second, it is investigated whether VNFA/SSMA are capable to fulfill

[15] Data is taken from the 2018 annual report of Stadtwerke Passau GmbH.

VNFA							
λ^{s}_{hos}	$6.0\mathrm{E}^{-5}$	λ^{s}_{hom}	$3.0\mathrm{E}^{-5}$	λ^{s}_{hol}	$1.0\mathrm{E}^{-5}$		
μ^{s}_{hrs}	$2.0\mathrm{E}^{-1}$	μ^{s}_{hrm}	$5.0\mathrm{E}^{-2}$	μ^{s}_{hrl}	$1.0\mathrm{E}^{-2}$		
λ_{vmmo}	$4.0\mathrm{E}^{-5}$	λ_{vmmod}	$8.0\mathrm{E}^{-5}$	λ_{vmmd}	$8.0\mathrm{E}^{-5}$		
μ_{vmmr}	$1.0\mathrm{E}^{0}$	μ_{vmmrej}	$6.0\mathrm{E}^{0}$				
λ_{vmo}	$1.0\mathrm{E}^{-3}$	λ_{vmod}	$2.0\mathrm{E}^{-3}$	λ_{vmd}	$1.0\mathrm{E}^{-3}$		
μ_{vmr}	$2.0\mathrm{E}^{0}$	μ^{cold}_{vmrej}	$3.0\mathrm{E}^{1}$	μ^{warm}_{vmrej}	$6.0\mathrm{E}^{1}$		
λ_{so}	$2.0\mathrm{E}^{-3}$	λ_{sol}	$4.0\mathrm{E}^{-5}$	λ_{sod}	$4.0\mathrm{E}^{-3}$	λ_{sd}	$4.0\mathrm{E}^{-3}$
μ_{sr}	$2.0\mathrm{E}^{0}$	μ_{srl}	$3.0\mathrm{E}^{1}$	μ^{cold}_{srej}	$6.0\mathrm{E}^{1}$	μ^{warm}_{srej}	$2.4\mathrm{E}^{2}$
λ_{so}	$2.0\mathrm{E}^{-3}$	λ_{sol}	$4.0\mathrm{E}^{-5}$	λ_{sod}	$4.0\mathrm{E}^{-3}$	λ_{sd}	$4.0\mathrm{E}^{-3}$
μ_{sr}	$2.0\mathrm{E}^{0}$	μ_{srl}	$3.0\mathrm{E}^{1}$	μ^{cold}_{srej}	$6.0\mathrm{E}^{1}$	μ^{warm}_{srej}	$2.4\mathrm{E}^{2}$
η_{vm}	$9.6\mathrm{E}^{2}$	η_{s}	$9.6\mathrm{E}^{2}$	ψ_{vm}	$1.8\mathrm{E}^{3}$	ψ_{s}	$1.8\mathrm{E}^{3}$
Ω_{cpu}	$1.02\mathrm{E}^{0}$	Ω_{net}	$1.06\mathrm{E}^{0}$				

Table 6.5.: Single user scenario VNFA parameters

the requirements of AMIs regarding performability. During this scenario, it is assumed for both SSMA and VNFA that the optimal choices regarding rejuvenation are used, i. e., for SSMA: dynamic rejuvenation with $\nabla^{s}_{nv} = 0.7$; for VNFA: dynamic migration rejuvenation with $\nabla^{s}_{v} = 0.7$, $\nabla^{vm}_{v} = 0.8$, and $\nabla^{vmm}_{v} = 0.9$, until stated differently[16].

The results of the Passau city scenarios evaluation are investigated in Section 6.2.3.

6.2 Performability Assessment

The performability assessment is executed to analyze and compare both architectures in Sections 6.2.1, 6.2.2, and 6.2.3. After the evaluation, the results are compared in Section 6.2.4 and conclusions are drawn. As for the hardware, for the performability analysis, a single server is used. Apart from that, neither is further equipment required, nor used. The server has the following properties:

- **OS:** Windows 10 Professional Edition Version 1803 (64-bit)
- **CPU:** Intel Core i7-2600K @ 3.40 GHz
- **RAM:** 24GB DDR3 SDRAM PC3-10600, 1333.0 MHz, 11-11-11-28
- **Disk:** Lite-On LCS-256L9S-11 256 MB, 512 bytes/sector, NTFS, cluster size 4 kB

[16] How the optimal rejuvenation configurations for both SSMA and VNFA are derived is explained in the evaluation Sections 6.2.1 and 6.2.2.

SSMA							
gr_s							
λ_{hos}	3.0E^{-5}	λ_{hom}	2.0E^{-5}	λ_{hol}	1.0E^{-5}		
μ_{hrs}	1.5E^{-1}	μ_{hrm}	3.5E^{-2}	μ_{hrl}	8.0E^{-3}		
λ_{so}	2.0E^{-3}	λ_{sol}	4.0E^{-5}	λ_{sod}	4.0E^{-3}	λ_{sd}	4.0E^{-3}
μ_{sr}	2.0E^{0}	μ_{srl}	3.0E^{1}	μ_{srej}^{cold}	6.0E^{1}	μ_{srej}^{warm}	2.4E^{2}
gr_c							
λ_{hos}	1.2E^{-4}	λ_{hom}	4.0E^{-5}	λ_{hol}	2.0E^{-5}		
μ_{hrs}	1.5E^{-1}	μ_{hrm}	3.5E^{-2}	μ_{hrl}	8.0E^{-3}		
λ_{so}	4.0E^{-3}	λ_{sol}	8.0E^{-5}	λ_{sod}	8.0E^{-3}	λ_{sd}	8.0E^{-3}
μ_{sr}	2.0E^{0}	μ_{srl}	3.0E^{1}	μ_{srej}^{cold}	6.0E^{1}	μ_{srej}^{warm}	2.4E^{2}

Table 6.7.: Single user scenario SSMA parameters

6.2.1 Evaluation: Single Service Scenario

The results of the single service evaluation are used to evaluate the optimal rejuvenation strategy to achieve the highest possible performability while maintaining the required level of availability.

6.2.1.1 Test Environment

For the evaluation process and comparison of VNFA and SSMA as well as its various variants differing in their rejuvenation details, the Stochastic Petri Net Package (SPNP)[17] is used for the evaluation of the micro model. This package allows the creation and analysis of SRNs and was developed by Ciardo *et al.* [35]. SPNP uses a C based SPN Language (CSPL) for the specification of SRN models. During the analysis process, SPNP automatically converts SRN specifications to Markov reward models, which are subsequently solved to compute a variety of transient, steady-state, cumulative, and sensitivity measures.

For both numerical and simulative analyses, the standard configuration of the application is used, except for the analytical iterations/simulation length (which is adjusted to a higher value to enable the evaluation of large δ values). The parameters of both evaluation methods are given in the following two listings:

- **Numerical analysis:**
  - Approach: CTMC
  - Stead-state method: Successive overrelaxation
  - Transient method: Uniformization Fox & Glynn

[17] SPNP | Duke High Availability Assurance Laboratory (DHAAL), URL: `https://trivedi.pratt.duke.edu/software_packages/spnp`, last accessed: 01/03/2019

VNFA							
$r^{hw}_{cpu_{sm}}$	$2.0\mathrm{E}^{0}$	$r^{hw}_{cpu_{r}}$	$2.0\mathrm{E}^{0}$	$r^{hw}_{cpu_{gw}}$	$3.0\mathrm{E}^{0}$	$r^{hw}_{cpu_{dc}}$	$1.0\mathrm{E}^{2}$
$r^{hw}_{cpu_{s}}$	$5.0\mathrm{E}^{1}$						
$d^{s}_{cpu_{sm}}$	$1.5\mathrm{E}^{0}$	$d^{s}_{cpu_{r}}$	$1.5\mathrm{E}^{0}$	$d^{s}_{cpu_{gw}}$	$2.5\mathrm{E}^{0}$	$d^{s}_{cpu_{dc}}$	$5.0\mathrm{E}^{0}$
$d^{s}_{b_{sm\rightarrow}}$	$1.4\mathrm{E}^{1}$	$d^{s}_{b_{dc\rightarrow}}$	$1.4\mathrm{E}^{0}$				
μ_{sm}	$1.0\mathrm{E}^{1}$	μ_{gw}	$1.0\mathrm{E}^{1}$	μ_{r}	$1.0\mathrm{E}^{1}$	μ_{dc}	$1.0\mathrm{E}^{2}$
$\mu_{tps_{eo}}$	$1.0\mathrm{E}^{1}$	$\mu_{tps_{cm}}$	$1.0\mathrm{E}^{1}$				
$l_{tps_{eo}}$	$4.0\mathrm{E}^{-1}$	$l_{tps_{cm}}$	$7.0\mathrm{E}^{-1}$				
$U^{hw}_{s_{max}}$	$8.0\mathrm{E}^{-1}$						
Ω_{cpu}	$1.02\mathrm{E}^{0}$	Ω_{net}	$1.06\mathrm{E}^{0}$				
SSMA							
$r^{hw}_{cpu_{sm}}$	$2.0\mathrm{E}^{0}$	$r^{hw}_{cpu_{r}}$	$2.0\mathrm{E}^{0}$	$r^{hw}_{cpu_{gw}}$	$3.0\mathrm{E}^{0}$	$r^{hw}_{cpu_{dc}}$	$1.0\mathrm{E}^{2}$
$r^{hw}_{cpu_{s}}$	$5.0\mathrm{E}^{1}$						
$d^{s}_{cpu_{sm}}$	$1.5\mathrm{E}^{0}$	$d^{s}_{cpu_{r}}$	$1.5\mathrm{E}^{0}$	$d^{s}_{cpu_{gw}}$	$2.5\mathrm{E}^{0}$	$d^{s}_{cpu_{dc}}$	$5.0\mathrm{E}^{0}$
$d^{s}_{b_{sm\rightarrow}}$	$1.4\mathrm{E}^{1}$	$d^{s}_{b_{dc\rightarrow}}$	$1.4\mathrm{E}^{1}$				
μ_{sm}	$1.0\mathrm{E}^{1}$	μ_{gw}	$1.0\mathrm{E}^{1}$	μ_{r}	$1.0\mathrm{E}^{1}$	μ_{dc}	$1.0\mathrm{E}^{2}$
$\mu_{tps_{eo}}$	$1.0\mathrm{E}^{1}$	$\mu_{tps_{cm}}$	$1.0\mathrm{E}^{1}$				
$l_{tps_{eo}}$	$4.0\mathrm{E}^{-1}$	$l_{tps_{cm}}$	$7.0\mathrm{E}^{-1}$				
$U^{hw}_{s_{max}}$	$8.0\mathrm{E}^{-1}$						

Table 6.9.: Passau city scenario VNFA and SSMA parameters

  - Maximum # of iterations: 50 000
  - Minimum precision: 0.000 001
  - M0 return rate: 0.0

- **Simulative analysis:**
  - Simulation method: Discrete event simulation with independent replications
  - Number of replications/batches: 50
  - Required confidence of the simulation: 95 %
  - Length of each simulation iteration: 50 000
  - Used for comparing the fluid places content: 0.000 001
  - Used for comparing the firing time conflict: 0.000 001

6.2.1.2 Single Service Scenario: Evaluation of SSMA Model

For SSMA, the rejuvenation strategy is simplified, as both the *warm* and *migrate* strategies are not applicable because of the missing virtualization. Therefore, the rejuvenation dimension *strategy* is skipped here, reducing the analysis to a distinction between no, deterministic and dynamic rejuvenation in this case. The results for no rejuvenation is described next, the deterministic and dynamic approaches are depicted in Figures 6.1 and 6.2, respectively.

In the no rejuvenation case, while a relatively high availability $\left(A(\text{SSMA}) = 99.326\,\%\right)$ is achieved, the performability suffers from the relatively high residual times in an impaired state ($\pi_{s_{imp}} = 51.471\,\%$), leading to $Perf\,(\text{SSMA}) = 84.682\,\%$.

In contrast, for deterministic rejuvenation, the results heavily depend on the time intervals in which the rejuvenation is triggered, as the results show.

$$A(\text{SSMA}) = \begin{cases} 0\,\% & \text{for } \delta^s_{nv} = 0 \\ \leq 99.320\,\% & \text{for } 0 < \delta^s_{nv} \leq 10\,000 \\ 99.326\,\% & \text{for } \delta^s_{nv} \to \infty \end{cases}$$

$$Perf\,(\text{SSMA}) = \begin{cases} 0\,\% & \text{for } \delta^s_{nv} = 0 \\ \leq 98.468\,\% & \text{for } 0 < \delta^s_{nv} \leq 10\,000 \\ 84.682\,\% & \text{for } \delta^s_{nv} \to \infty \end{cases}$$

If the rejuvenation interval is short, the increasing number of restarts leads to a decreasing availability and performability; an extreme δ^s_{nv} of 0 causes constant rejuvenation actions, leading to an availability and performability value of 0. In contrast, high values of δ^s_{nv} achieve a similar behavior as no rejuvenation, i. e. high availability and low performability. An optimal performance using deterministic rejuvenation is achieved in between both extremes.

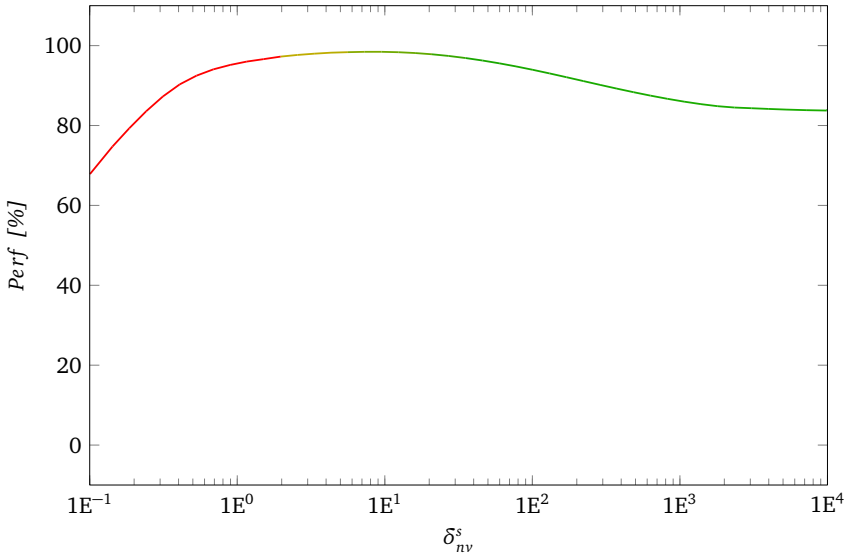


Figure 6.1.: Influence of parameter δ^s_{nv} in deterministic, cold rejuvenation (non-virtualized case)

To find the optimal deterministic rejuvenation parameters, first, it is required to define an optimization goal for an AMI system. Two goals are possible in the given scenario, namely availability or performance. This results in a non-trivial multi-objective optimization challenge, where trade-offs

between the two conflicting objectives are required. To solve this challenge, two basic methodological approaches are possible; either a simulation-based optimization or multi-objective optimization approaches (such as a Pareto-optimization or an optimization under constraints) are feasible. As the DSPNs are solved using a simulative approach in this thesis, a simulation-based optimization approach based on indifference zone methods is chosen.

For a multi-objective optimization approach, the steady-states of the DSPNs could be calculated using solution approaches as applied by Lindemann [83]. Thereafter an optimization such as Karush-Kuhn-Tucker (KKT) could be used to find the optimal value of δ^s_{nv} given the hard constraint that the AMI's availability may not drop under 98 %: maximize $Perf(\delta^s_{nv})$ subject to $A(\delta^s_{nv}) \geq \sqrt[n]{0.98}$, where $A(\delta^s_{nv}) \geq \sqrt[n]{0.98}$ is the availability inequality hard constraint that needs to be satisfied assuming that n entities (with $\sigma = 1$) having the same availability each are present in the AMI of a user. In the given single service scenario, however, $n = 1$.

The simulative indifference zone method optimization chosen is a multi-stage KN procedure, which operates using the following method [74]:

1. **Setup:** Select a confidence level $(1-\alpha)$, an indifference parameter δ and the first-stage sample size n_0. Set $\eta = \frac{1}{2}\left(\left(\frac{2\alpha}{k-1}\right)^{\frac{-2}{n_0-1}} - 1\right)$.

2. **Initialization:** Let $I = \{1,2,...,k\}$ be the set of systems in contention. Set $h^2 = 2\eta(n_0-1)$. Next, obtain the n_0 outputs $Perf_{ij}$, where $j = 1,\ldots,n_0$ from each system. Calculate the sample means of observations from each system and sample variance of differences between observations from systems i and l, using Equation (6.2).

$$S^2_{il} = \frac{1}{n_0-1}\sum_{j=1}^{n_0}\left(Perf_{ij} - Perf_{lj} - \left(\overline{Perf}_i(n_0) - \overline{Perf}_l(n_0)\right)\right)^2 \quad \forall\, l \neq i. \tag{6.2}$$

   Set $r = n_0$.

3. **Screening:** Set $I^{old} = I$. Define the new set of systems in contention as $I = \{i : i \in I^{old}$ and $\overline{Perf}_i(r) \geq \overline{Perf}_l(r) - W_{il}(r) \quad \forall\, l \in I^{old}, l \neq i\}$, where

$$W_{il}(r) = \max\{0, \frac{\delta}{2r}\left(\left(\frac{hS_{il}}{\delta}\right)^2 - 1\right)\}. \tag{6.3}$$

4. **Selection:** If $|I| = 1$ then stop the procedure and return $i \in I$ as the best system. Otherwise, take an additional output $Perf_{i,r+1}$ from each $i \in I$, set $r = r+1$ and return to the screening stage.

The following parameter definitions are used in the equations above:

$$\begin{aligned}
(1-\alpha) &= \text{Confidence level} &&= 0.01\\
\delta &= \text{Indifference parameter} &&= 0.0001\\
n_0 &= \text{First-stage sample size} &&= 10\\
k &= \text{Number of systems to choose between} &&= 10\\
Perf_{ij} &= j^{\text{th}} \text{ performability observation of } i^{\text{th}} \text{ system}
\end{aligned}$$

resulting in:

$$\begin{aligned}
\eta &= \frac{1}{2}\left(\left(\frac{1.98}{9}\right)^{\frac{2}{9}} - 1\right) = 0.2\\
h^2 &= 0.4 \cdot 9 = 3.6
\end{aligned}$$

i	0	1	2	3	4	5	6	7	8	9
δ^s_{nv}	3	4	5	6	7	8	9	10	11	12
$\overline{Perf}_i$ [%]	97.850	98.155	98.301	98.380	98.4316	98.450	98.448	98.433	98.398	98.362
$\overline{A}_i$ [%]	98.675	98.786	98.839	98.869	98.878	98.885	98.893	98.896	98.899	98.901

Table 6.10.: $\overline{Perf}_i$ and $\overline{A}_i$ calculation

Table 6.10 shows the intermediate results for $\overline{Perf}_i$ and $\overline{A}_i$. Thereafter, it is possible to calculate S^2_{il} using Equation (6.2), as depicted in Table 6.11. After that, using Equation (6.3), the values of W_{il} are derived, allowing to determine a set of candidate systems I (marked in *green* in Table 6.11).

i/l	0	1	2	3	4	5	6	7	8	9
0	-	$2.50E^{-7}$	$7.11E^{-8}$	$7.11E^{-8}$	$1.34E^{-7}$	$1.60E^{-7}$	$7.11E^{-8}$	$1.60E^{-7}$	$1.87E^{-7}$	$4.00E^{-8}$
1	$1.00E^{-8}$	-	$7.11E^{-8}$	$1.11E^{-9}$	$9.00E^{-8}$	$5.44E^{-8}$	$1.00E^{-8}$	$1.87E^{-7}$	$3.21E^{-7}$	$9.00E^{-8}$
2	$1.00E^{-8}$	$1.77E^{-8}$	-	$4.44E^{-9}$	$1.00E^{-8}$	$4.44E^{-9}$	$5.44E^{-8}$	$1.11E^{-9}$	$2.77E^{-8}$	$4.44E^{-8}$
3	$1.11E^{-9}$	$1.77E^{-8}$	$4.44E^{-9}$	-	$1.00E^{-8}$	$5.44E^{-8}$	$1.11E^{-9}$	$1.00E^{-8}$	$1.11E^{-9}$	$2.33E^{-8}$
4	$5.44E^{-8}$	$4.00E^{-8}$	$1.60E^{-7}$	$4.00E^{-8}$	-	$9.00E^{-8}$	$1.34E^{-7}$	$7.11E^{-8}$	$2.17E^{-7}$	$1.11E^{-7}$
5	$4.00E^{-8}$	$4.00E^{-8}$	$1.88E^{-7}$	$1.11E^{-9}$	$1.11E^{-9}$	-	$1.11E^{-9}$	$1.00E^{-8}$	$5.44E^{-8}$	$1.00E^{-8}$
6	$1.877E^{-7}$	$4.00E^{-8}$	$1.00E^{-8}$	$9.00E^{-8}$	$4.44E^{-9}$	$7.11E^{-8}$	-	$4.44E^{-9}$	$1.00E^{-8}$	$7.11E^{-8}$
7	$3.21E^{-7}$	$3.21E^{-7}$	$9.00E^{-8}$	$1.11E^{-7}$	$1.878E^{-7}$	$4.00E^{-8}$	$7.11E^{-7}$	-	$4.00E^{-8}$	$4.00E^{-8}$
8	$1.00E^{-8}$	$2.77E^{-8}$	$2.18E^{-7}$	$1.00E^{-8}$	$5.44E^{-8}$	$4.44E^{-9}$	$2.77E^{-8}$	$5.44E^{-8}$	-	$2.77E^{-8}$
9	$9.00E^{-8}$	$1.11E^{-9}$	$1.77E^{-8}$	$7.11E^{-8}$	$1.11E^{-9}$	$4.44E^{-9}$	$4.00E^{-8}$	$5.44E^{-8}$	$9.00E^{-8}$	-

Table 6.11.: Results of S^2_{il} calculations

Using the aforementioned approach, the result yields $\delta^s_{nv} = 8.25\,\text{h}$.

The results show that, in the dynamic rejuvenation case:

$$A(\text{SSMA}) = \begin{cases} 0\,\% & \text{for } \nabla^s_{nv} \geq 1 \\ 99.323\,\% & \text{for } 0.7 \leq \nabla^s_{nv} < 1 \\ 99.328\,\% & \text{for } \nabla^s_{nv} < 0.7 \end{cases}$$

$$Perf(\text{SSMA}) = \begin{cases} 0\,\% & \text{for } \nabla^s_{nv} \geq 1 \\ 99.323\,\% & \text{for } 0.7 \leq \nabla^s_{nv} < 1 \\ 84.682\,\% & \text{for } \nabla^s_{nv} < 0.7 \end{cases}$$

Failure Sensitivity Analysis. In the sensitivity analysis, the influence of the various failure rates on the overall performability of SSMA and VNFA is investigated. While especially the impact of λ_{sol} and $\lambda_{hos}, \lambda_{hom}$ and λ_{hol} is similar for both architectures, software failures λ_{so} have a much higher impact in SSMA, which is attributable to the longer repair times encountered. These originate from the fact a software failure can sometimes not be remotely solved in a non-virtualized environment and requires an employee to fix the failure on-site, which induces delays. This is depicted in Figure 6.3.

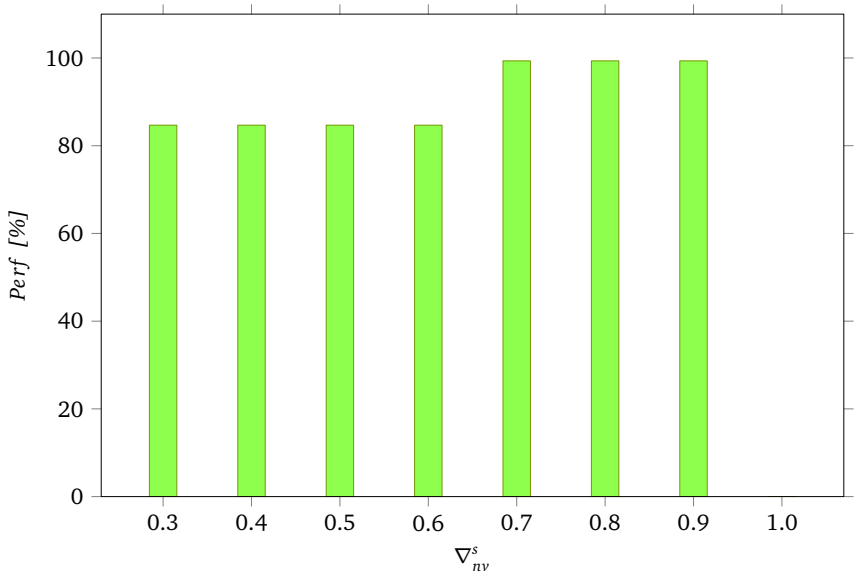
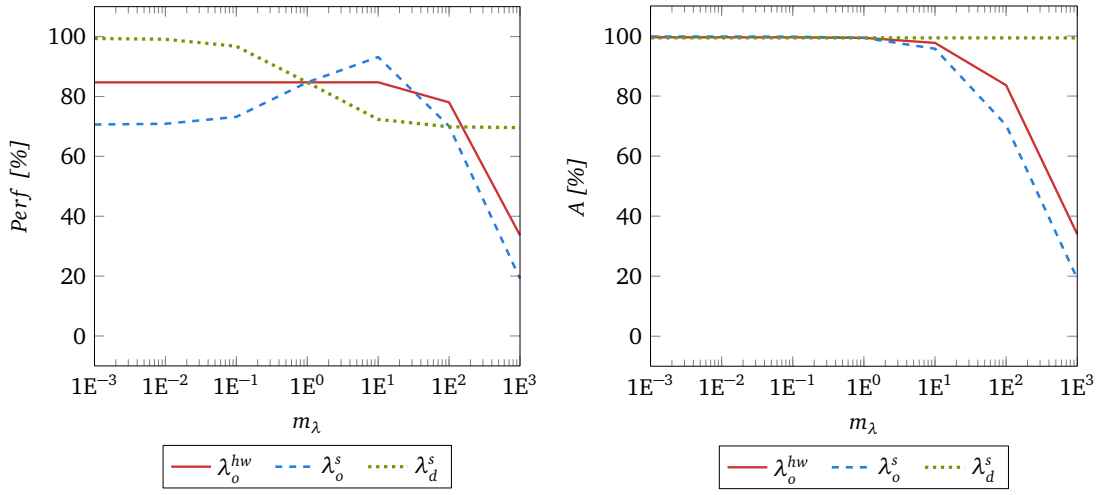


Figure 6.2.: Influence of parameter ∇^s_{nv} in dynamic, cold rejuvenation (non-virtualized case)



(a) Influence of failure rate on *Perf* of SSMA

(b) Influence of failure rate on *A* of SSMA

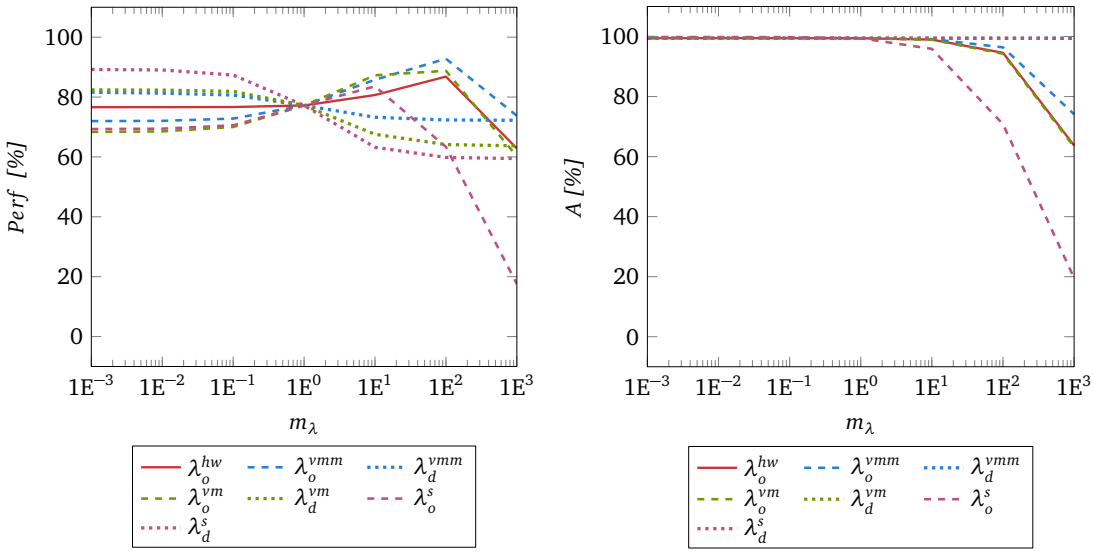
Figure 6.3.: Failure sensitivity analysis of SSMA regarding performance *(left)* and availability *(right)*

6.2.1.3 Single Service Scenario: Evaluation of VNFA Model

NFV AMI Architecture, No Rejuvenation. As the analysis of the no rejuvenation has shown in the SSMA case, it is similar to using a dynamic rejuvenation with ∇-parameters set to a value lower than $1-(\alpha+\beta+\gamma)$. Therefore, the no rejuvenation case is not regarded separately in this analysis, instead, it is referred to the respective part of the VNFA dynamic, cold rejuvenation analysis.

Failure Sensitivity Analysis. For VNFA, the parameters' influences can be classified into two basic groups, more specifically into those parameters showing negligible and those having strong influence on the overall availability/performability of the architecture. The main parameter difference usable for the classification is the repair rate associated with the respective failure, where a lower repair rate implies a higher impact of a parameter on the system's dependability. Therefore, the influence of hardware and software failures requiring a longer restoration time to fix (λ_{sol} and λ_{hom} and λ_{hol}) have by far the strongest impact, followed by λ_{hos}, λ_{vmmo} and λ_{vmo}; increasing any of the impairment

rates results in comparably small availability losses of $< 1\,\%$, while performability is moderately impacted (on average 11.320 % loss). These results are depicted in Figure 6.4.



(a) Influence of failure rate on *Perf* of VNFA

(b) Influence of failure rate on *A* of VNFA

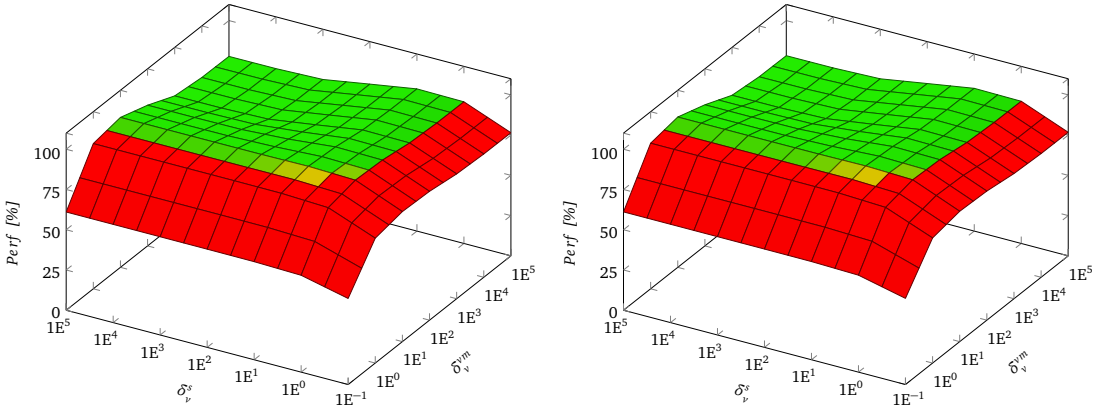
Figure 6.4.: Sensitivity analysis of VNFA regarding performability *(left)* and availability *(right)*

NFV AMI Architecture, Cold Rejuvenation Strategy. Regarding VNFA, in case of a cold rejuvenation strategy, it is visible that especially high frequencies of service rejuvenation cause a significant drop in both availability and performance. The results are depicted in Figure 6.5, where the areas of the graph marked in red represent parameter settings that do not achieve the required availability of 98 %. All areas marked in yellow have an availability of $\geq 98\,\%$, but $< 99\,\%$. Areas marked in green represent an availability of $\geq 99\,\%$. This color scheme is also used in the upcoming evaluations.

For deterministic rejuvenation, generally speaking, if δ_ν^s or $\delta_\nu^{vmm} \leq 1$, the availability decreases to $< 98\,\%$, making it not suitable. In scenarios where δ_ν^s, δ_ν^{vm}, and $\delta_\nu^{vmm} > 1$, optimization methods (as shown in Section 6.2.1.2) lead to a peak performance at $\delta_\nu^{vmm} = 43.25\,\text{h}$, while δ_ν^{vm} and δ_ν^s is to be set to ∞, resulting in $A(\text{VNFA}) = 99.186\,\%$ and $Perf\,(\text{VNFA}) = 99.045\,\%$.

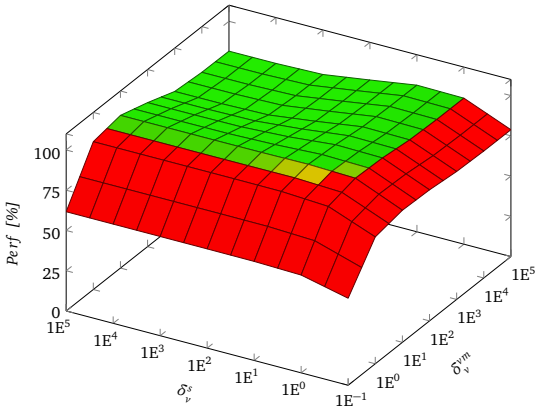
The results show that, in the dynamic rejuvenation case, both availability and performance are 0 in scenarios where either ∇_ν^{vmm}, ∇_ν^{vm} or $\nabla_\nu^s \geq 1$. The reason behind this outcome is that any ∇_ν value of ≥ 1 leads to a constant triggering of the rejuvenation. Second, it is apparent that especially high ∇_ν^s in combination with low ∇_{vm} values have a high impact on both availability and performance. This is due to the fact that in the given scenario, $\alpha = 0.1$, $\beta = 0.2$ while $\gamma = 0.3$. Therefore, a high ∇_ν^s value leads to unnecessary reboots which do however not fix the impairments on VM and VMM levels. The highest performance values that fulfill the availability requirements are observable in scenarios where $\nabla_\nu^{vmm} = 0.9$, $\nabla_\nu^{vm} < 0.9$, and $\nabla_\nu^s < 0.9$. These results are depicted in Figure 6.6.

NFV AMI Architecture, Warm/Migration Rejuvenation Strategy. Generally speaking, the availability and performability improvements for both warm and migration rejuvenation behave in a similar manner: If a VMM rejuvenation is triggered, warm and migration rejuvenation offer an alternative restoration to cold rejuvenation, which is faster, thereby preserving a higher performability. However, as described in Section 5.3.4, during a VMM rejuvenation event, warm rejuvenation does not trigger a service rejuvenation automatically. In the case of migration rejuvenation, VM and service do not get rejuvenated, respectively. The exact influences of these behaviors are described next. The

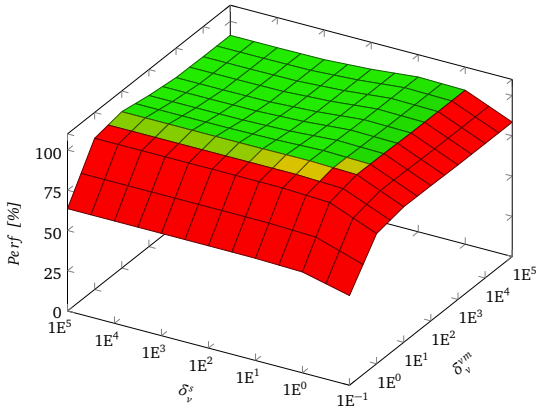


(a) Deterministic, cold rejuvenation ($\delta_v^{vmm} = 1E^5$)

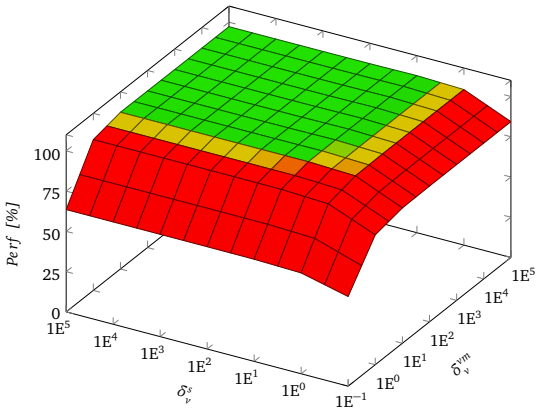
(b) Deterministic, cold rejuvenation ($\delta_v^{vmm} = 1E^4$)



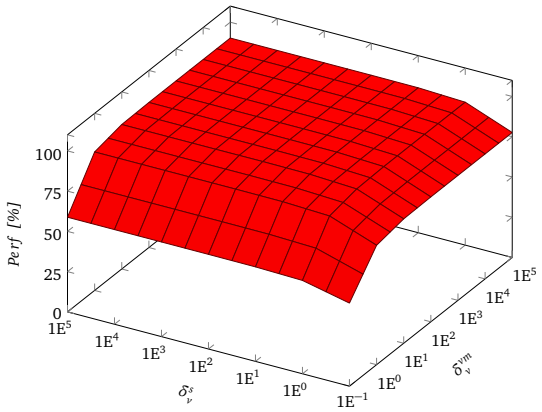
(c) Deterministic, cold rejuvenation ($\delta_v^{vmm} = 1E^3$)



(d) Deterministic, cold rejuvenation ($\delta_v^{vmm} = 1E^2$)



(e) Deterministic, cold rejuvenation ($\delta_v^{vmm} = 1E^1$)

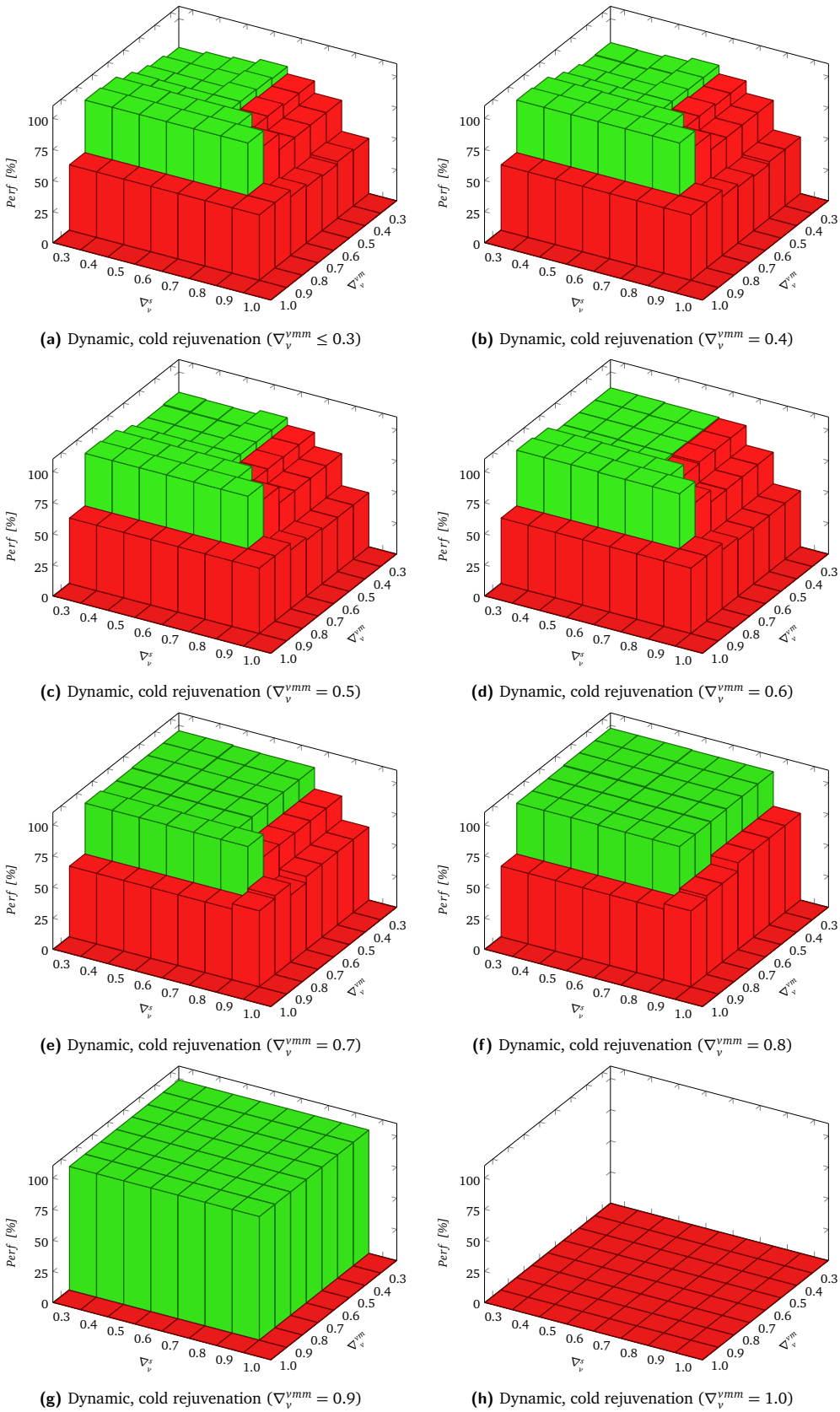


(f) Deterministic, cold rejuvenation ($\delta_v^{vmm} = 1E^0$)

Figure 6.5.: Influence of rejuvenation parameters on performability (deterministic, cold rejuvenation, virtualized case)

performability changes are always given for the peak availability and performability being achieved, i. e., for the difference between cold and warm rejuvenation: $\Delta A = \max(A_{vnfa}^{cold}) - \max(A_{vnfa}^{warm})$ and $\Delta Perf = \max(Perf_{vnfa}^{cold}) - \max(Perf_{vnfa}^{warm})$.

- **Deterministic timing:** Using deterministic warm rejuvenation, the availability gain is 0.083 %, the performability increase is 0.149 %. A similar phenomenon can be observed in migration rejuvenation, however, the effect is even stronger compared to warm rejuvenation. Using



(a) Dynamic, cold rejuvenation ($\nabla_v^{vmm} \leq 0.3$)

(b) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 0.4$)

(c) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 0.5$)

(d) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 0.6$)

(e) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 0.7$)

(f) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 0.8$)

(g) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 0.9$)

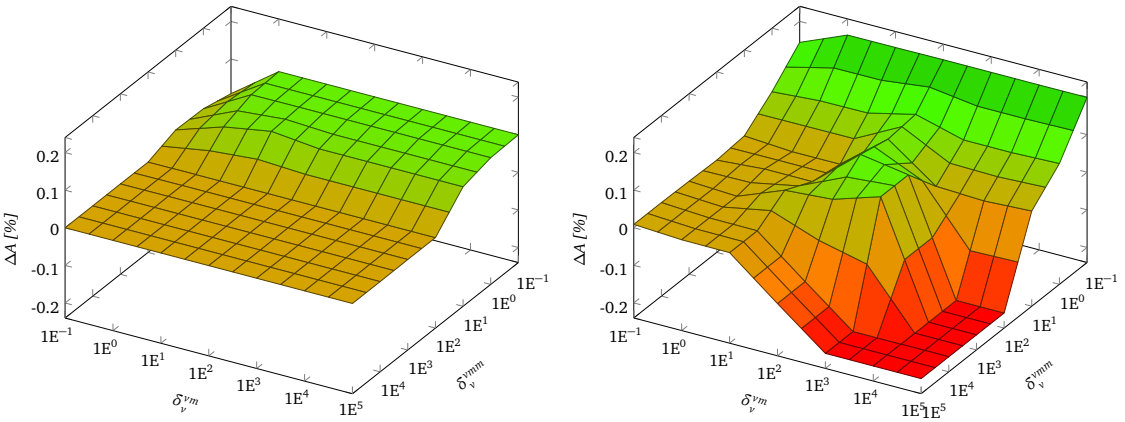
(h) Dynamic, cold rejuvenation ($\nabla_v^{vmm} = 1.0$)

Figure 6.6.: Influence of rejuvenation parameters on performability (dynamic, cold rejuvenation, virtualized case)

deterministic rejuvenation, the availability gain is 0.121 %, the performability increase is 0.187 %.

- **Dynamic timing:** Warm rejuvenation offers a benefit of 0.105 % in availability and 0.105 % in performability, using dynamic rejuvenation. Migration rejuvenation offers a benefit of 0.219 % in availability and 0.213 % in performability, using dynamic rejuvenation. In migration rejuvenation, a special case needs to be discussed, namely if $\nabla_{vmm} \geq 1$. In such a case, a migrated VM will never return to its original host, as there is a constant triggering of VMM rejuvenation. While the difference in availability and performability in such cases is severe (ΔA and $\Delta Perf \geq 90\,\%$), setting $\nabla_{vmm} \geq 1$ is not feasible, as the original host would constantly migrate all VMs to other substrate hosts. Therefore, this case is disregarded from hereon forward.

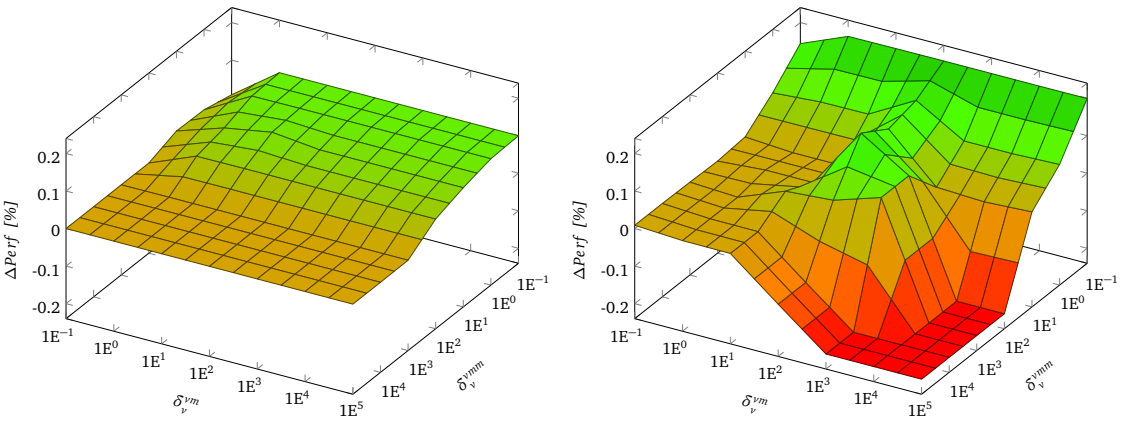
The obtained results are visualized in Figures 6.7 and 6.8 for deterministic and Figures 6.9 and 6.10 for dynamic rejuvenation, respectively. Here, the color scheme implies a decrease (colored *red*) or increase (colored *green*) of A or $Perf$, respectively.



(a) ΔA of warm compared to cold rejuvenation

(b) ΔA of migration compared to cold rejuvenation

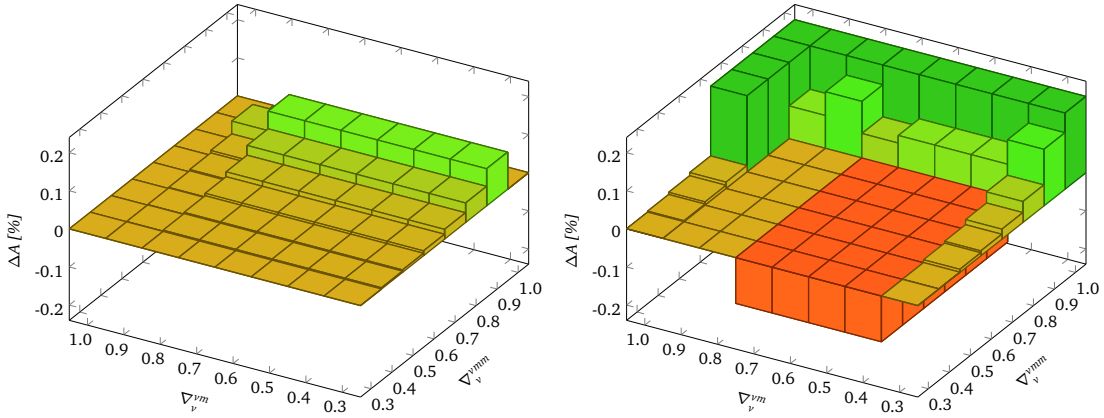
Figure 6.7.: ΔA of warm *(left)* and migration *(right)* compared to cold rejuvenation (deterministic, virtualized case)



(a) $\Delta Perf$ of warm compared to cold rejuvenation

(b) $\Delta Perf$ of migration compared to cold rejuvenation

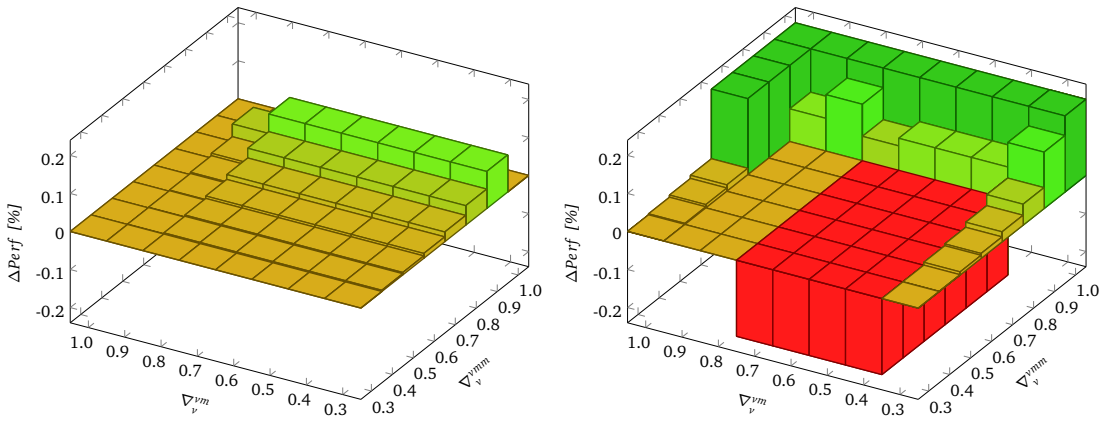
Figure 6.8.: $\Delta Perf$ of warm *(left)* and migration *(right)* compared to cold rejuvenation (deterministic, virtualized case)



(a) ΔA of warm compared to cold rejuvenation

(b) ΔA of migration compared to cold rejuvenation

Figure 6.9.: ΔA of warm *(left)* and migration *(right)* compared to cold rejuvenation (dynamic, virtualized case)



(a) $\Delta Perf$ of warm compared to cold rejuvenation

(b) $\Delta Perf$ of migration compared to cold rejuvenation

Figure 6.10.: $\Delta Perf$ of warm *(left)* and migration *(right)* compared to cold rejuvenation (dynamic, virtualized case)

		Number of replicas		
		0	1	2
No rejuvenation		99.424 %	99.915 %	99.932 %
Cold rejuvenation	deterministic	99.186 %	99.746 %	99.658 %
	dynamic	99.423 %	99.923 %	99.934 %
Warm rejuvenation	deterministic	99.269 %	99.821 %	99.857 %
	dynamic	99.528 %	99.931 %	99.947 %
Migration rejuvenation	deterministic	99.307 %	99.934 %	99.985 %
	dynamic	99.642 %	99.969 %	99.992 %

Table 6.12.: Influence of replicas on service availability in VNFA

Replica Usage. Besides the employed rejuvenation method, a VNFC's dependability also depends on whether and if so, how many replicas are used to offer service backups (the performance is not

affected by the usage of replicas as stated in the assumptions). To evaluate the benefit of replicas, the dependability gain of introducing an increasing number of replicas is analyzed. Using one replica, VNFA with no rejuvenation the availability rises from to 99.424 % to 99.915 %. Increasing the number to two replicas only bears a minor increase in availability to 99.932 %, which is to be expected, as gross of the remaining failures occurs at substrate or VMM level. The availability development using replicas behaves in a similar manner with both deterministic and dynamic rejuvenation in place, as well as cold, warm or migration rejuvenation. Table 6.12 summarizes the findings.

Despite the apparent gain in dependability, it is noted that each additional replica also demands resources. Therefore, improving each service's dependability by employing multiple replicas is not an economically feasible solution, as discussed in Section 1.2. To determine the proper amount of replicas, two requirements need to be evaluated. The most important is upholding the 98 % availability requirement posed by AMIs. Secondly, there are the costs incurred by the additional replicas, which—in turn—lead to increased resource demands and substrate nodes to provide them. Therefore, an optimization that minimizes the costs while maintaining an availability $\geq 98\,\%$ is performed. Finding an optimal number of replicas for each service individually is not part of this thesis. However, the influence of employing system-wide replicas (i. e. a static number of replicas per service entity of a specific type) is analyzed in both single user and Passau city scenario. If replicas are employed, the service featuring the highest in-degree and $\sigma = 1$ in the macro model is identified and replicated to offer the achieved dependability benefit to a maximum amount of users [184]. To weight the dependability gain against the additional resources required to realize replicas, a cost analysis is provided in Section 6.4.2.

6.2.2 Evaluation: Single User Scenario

While the results of the single service evaluation are partially applicable in the single user scenario, it is important to once more review the rejuvenation strategy that achieves optimal results in the single service scenario. Here, dynamic rejuvenation results in the optimal performability under a set availability threshold. However, in the single user scenario, multiple services are working serially aligned, i. e. if a single service is under rejuvenation while the other services are performing at full capacity, the performance of the user's service chain is insufficient. Therefore, it needs to be evaluated if a synchronized, deterministic rejuvenation of all services offers a higher overall performability than dynamic rejuvenation, or if the usage of replicas is superior. To maximize the performability of services, a migration rejuvenation strategy is used (if applicable), as shown in Section 6.2.1.3.

6.2.2.1 Single User Scenario: Test Environment

In addition to the SPNP tool described in the single service scenario, for the performability analysis of the overall model, the open source suite Java Modelling Tools (JMT)[18] [17], consisting of tools for performance evaluation and modeling of computer and communication systems, is used. The suite implements several state-of-the-art algorithms for the exact, approximate or simulative analysis of queuing network models. To be more precise, the JMT tool JSIMgraph is used, which is a discrete-event simulator for the analysis of queuing network and Petri net models. The application supports numerous probability distributions (e. g. deterministic, Erlang, exponential, gamma, normal) for characterizing service and inter-arrival times as well as state-independent and state-dependent routing strategies.

[18] Java Modelling Tools – Introduction, URL: `http://jmt.sourceforge.net/`, last accessed: 05/01/2019

6.2.2.2 Single User Scenario: Evaluation of SSMA Model

Before the actual calculations to analyze the behavior and properties of a single user's service chain in SSMA are done, it needs to be clarified which rejuvenation *timing* and *strategy* offer an optimal performability. Because the only applicable *strategy* in case of SSMA is cold rejuvenation, the variants break down to the *timing* dimension. In this case it is required to evaluate whether a synchronized deterministic timing might outperform a dynamic rejuvenation, which is triggered based on the performance of each service individually. To do so, both rejuvenation timings are tested on a set of n sequentially aligned services. δ^s_{nv} is assumed to be set to the optimal setting of 8.25 h; ∇^s_{nv} is set to 0.7, respectively.

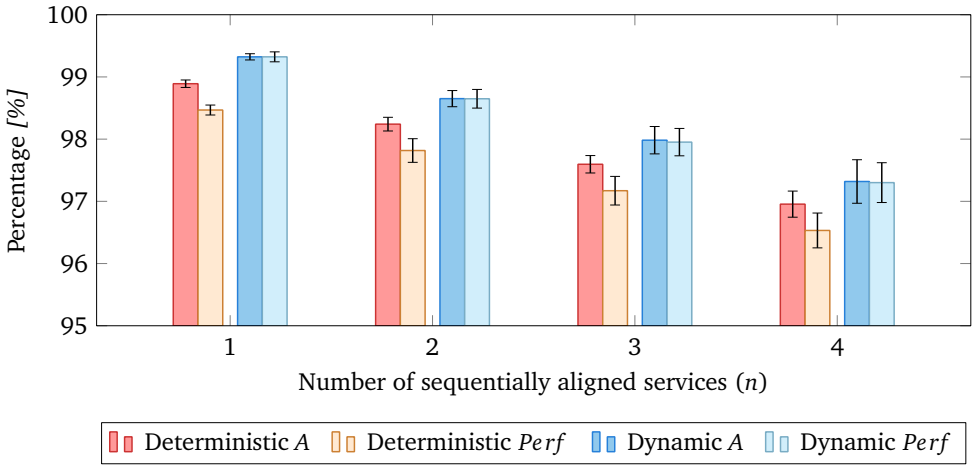


Figure 6.11.: Comparison of deterministic and dynamic rejuvenation timing in SSMA

Figure 6.11 depicts the findings of the experiment, showing that with an increasing number of sequential services, a synchronized, deterministic rejuvenation offers a less steep dependability decrease than a dynamic approach, which is also verified by Myint & Thein [105]. Though the performability difference of deterministic and dynamic rejuvenation timing is decreased using synchronized deterministic rejuvenations, it cannot be equalized, therefore dynamic rejuvenation still offers a higher performability in the given scenario.

As the analysis in Section 6.2.1.2 already revealed, the results for gr_s are: $A = 99.323\,\%$, for $Perf = 99.323\,\%$. For gr_c, the evaluation results in $A = 98.749\,\%$, for $Perf = 98.745\,\%$. This results in the following service rates for the SSMA services:

$$\begin{aligned}
\mu(gw) &= Perf(gw) \cdot \mu_o(gw) = 0.99323 \cdot 40 &&= 39.73 \\
\mu(r) &= Perf(r) \cdot \mu_o(r) = 0.99323 \cdot 1000 &&= 993.23 \\
\mu(dc) &= Perf(dc) \cdot \mu_o(dc) = 0.98745 \cdot 60 &&= 59.25 \\
\mu(tps) &= Perf(tps) \cdot \mu_o(tps) = 0.98745 \cdot 30 &&= 29.62
\end{aligned}$$

μ_o is the maximum unimpaired performance achieved by a service. Employing the service rates introduced in Table 6.7, the overall performance/delay for a single SSMA user x is calculated by: $d(x) = \sum_{i,j \in S | u^s_i, u^s_j = x} d(i) + d(i,j)$, where $d(i)$ is the delay introduced by service i, while $d(i,j)$ is the delay caused by traffic traversing the link between services i and j. $d_{nc}(x)$ is the delay encountered

by traffic traversing from the smart meter to the TPSs for user x, $d_c(x)$ respectively from the smart meter to HES of user x.

$$d_{nc}(x) = d(sm, gw) + d(gw) + d(gw, r) + d(r) + d(r, tps) + d(tps)$$

$$d_c(x) = d(sm, gw) + d(gw) + d(gw, r) + d(r) + d(r, dc) + d(dc) + d(dc, h),$$

where $d(x)$ denotes the processing delay incurred by service x, $d(x, y)$ is the network delay of traffic traversing from services x to y.

Due to the delay calculations only respecting transmission and queuing delays (see assumptions in Section 6.1.1) and the disregard of in-network routers, the network delays calculate as follows:

$$\begin{aligned}
d(sm, gw) = d(gw, r) &= \tfrac{1600}{250000} &&= 0.0064\,\text{s} = 6.4\,\text{ms} \\
d(r, tps) &= \tfrac{1600}{1000000} &&= 0.0016\,\text{s} = 1.6\,\text{ms} \\
d(r, dc) &= \tfrac{1600}{500000} &&= 0.0032\,\text{s} = 3.2\,\text{ms} \\
d(dc, h) &= \tfrac{1600}{15000000} &&= 0.0001\,\text{s} = 0.1\,\text{ms}
\end{aligned}$$

This in turn leads to the following delays:

- $d_{nc}(x) = d(gw) + d(r) + d(tps) + d(sm, gw) + d(gw, r) + d(r, tps) = 25.2\,\text{ms} + 1\,\text{ms} + 34.3\,\text{ms} + 6.4\,\text{ms} + 6.4\,\text{ms} + 1.6\,\text{ms} = 74.9\,\text{ms}$, and
- $d_c(x) = d(gw) + d(r) + d(dc) + d(sw, gw) + d(gw, r) + d(r, dc) + d(dc, h) = 25.2\,\text{ms} + 1\,\text{ms} + 17.4\,\text{ms} + 6.4\,\text{ms} + 6.4\,\text{ms} + 3.2\,\text{ms} + 0.1\,\text{ms} = 59.7\,\text{ms}$.

To evaluate the availability of the overall SSMA for a single user, two cases need to be distinguished, similar to the delay calculation before. As not all services running within an AMI are necessarily required to provide its core functionality, the priority of each service, indicated by the priority flag σ, is of importance. Based on this, the following two metrics are distinguished:

- **Critical availability** (A_c), defined as the percent of operational time of the critical AMI path, encompassing all *required* services $\left(s \in S : \sigma(s) = 1\right)$ and the respective links connecting them.
- **Non-critical availability** (A_{nc}), defined as the percent of operational time of the *optional* AMI path, which are the paths ending in optional services $\left(s \in S : \sigma(s) = 0, outdeg(s) = 0\right)$ and the respective links connecting them.

To find the complete availability of an AMI, the individual availabilities of the services comprising the AMI have to be multiplied. Equations (6.4) and (6.5) illustrate the required calculations for the critical and non-critical availability of an AMI of user x.

$$A_c = \prod_{i \in S} A(i) : u_i^s = x \wedge \sigma(v_i^s) = 1 \tag{6.4}$$

$$A_{nc} = \prod_{i \in S} A(i) : u_i^s = x \wedge \left(\sigma(v_i^s) = 0 \mid outdeg(v_i^s) = 0\right) \tag{6.5}$$

For SSMA, this results in $A_{nc} = A(sm) \cdot A(gw) \cdot A(r) \cdot A(tps) \Rightarrow A_{nc} = A(g_s)^3 \cdot A(g_c)$ and $A_c = A(sm) \cdot A(gw) \cdot A(r) \cdot A(dc) \Rightarrow A_c = A(g_s)^3 \cdot A(g_c)$, leading to $A_c = A_{nc}$, which calculates as: $0.99323^3 \cdot 0.98749 = 0.96757$.

6.2.2.3 Single User Scenario: Evaluation of VNFA Model

For VNFA, the evaluation methodology is similar to SSMA; however, two additional influences need to be included within the evaluation:

1. **Hosting substrate influence:** Depending on the *spread* of a user's embedded VNF, i. e. how many VMs, VMMs, substrate servers, and connecting links are involved in the hosting of a user's VNF, the performability is changing. In the following, two cases are going to be analyzed: (a) all virtualized VNFs of a user are embedded on a single substrate server, (b) all virtualized VNFs of a user are distributed onto different substrate servers. Scenario (a) is thereby the optimal, scenario (b) the worst case from a performability perspective.
2. **Substrate backup strategy:** As extensively explained in Section 4.4.1, three possibilities for the substrate backup strategy are available: no, circle, or all-for-all backup. Depending on the strategy employed, the performability of a user's VNF changes. Before both backup strategies are compared, the no backup case is analyzed first.

The obtained results are discussed to answer these aforementioned questions.

1. In a similar manner as in Section 6.2.2.2, the network delay can be calculated; however, the virtualization overhead needs to be respected, which results for cases (a) and (b) in:

   (a) In the first scenario, the network-induced delays are minimized, as all VNFCs are located on the same server. This leads to the following network delays between services:

   $$\begin{aligned}\Omega_{net}\big(d(sm,r)\big) &= \Omega_{net}(\tfrac{1600}{250000}) &&= \Omega_{net}(0.0064\,\mathrm{s}) = 6.784\,\mathrm{ms}\\ \Omega_{net}\big(d(r,s)\big) &= \Omega_{net}(\tfrac{1600}{1000000}) &&= \Omega_{net}(0.0016\,\mathrm{s}) = 1.696\,\mathrm{ms}\\ \Omega_{net}\big(d(s,h)\big) &= \Omega_{net}(\tfrac{1600}{15000000}) &&= \Omega_{net}(0.0001\,\mathrm{s}) = 0.106\,\mathrm{ms}\end{aligned}$$

   The overall *network* delay (currently ignoring service delays) results in:

   $$d_{nc}^{net}(x) = \Omega_{net}\big(d(sm,r)+d(r,s)\big) = 6.784\,\mathrm{ms}+1.696\,\mathrm{ms} = 8.48\,\mathrm{ms}$$

   $$d_{c}^{net}(x) = \Omega_{net}\big(d(sm,r)+d(r,s)+d(dc,h)\big) = 6.784\,\mathrm{ms}+1.696\,\mathrm{ms}+0.106\,\mathrm{ms} = 8.586\,\mathrm{ms}$$

   (b) The second scenario induces an increased network delay due to transmissions required between various servers. The resulting network delays between services are:

   $$\begin{aligned}\Omega_{net}\big(d(sm,r)\big) &= \Omega_{net}(\tfrac{1600}{250000}) &&= \Omega_{net}(0.0064\,\mathrm{s}) = 6.784\,\mathrm{ms}\\ \Omega_{net}\big(d(r,s)\big) &= \Omega_{net}(\tfrac{1600}{1000000}) &&= \Omega_{net}(0.0016\,\mathrm{s}) = 1.696\,\mathrm{ms}\\ \Omega_{net}\big(d(s_i,s_j)\big) = \Omega_{net}\big(d(s,h)\big) &= \Omega_{net}(\tfrac{1600}{15000000}) &&= \Omega_{net}(0.0001\,\mathrm{s}) = 0.106\,\mathrm{ms}\end{aligned}$$

   The overall *network* delay, again ignoring service delays is therefore:

   $$d_{nc}^{net}(x) = \Omega_{net}\big(d(sm,r)+d(r,s_{gw})+d(s_{gw},s_{tps})\big) = 6.784\,\mathrm{ms}+1.696\,\mathrm{ms}+0.106\,\mathrm{ms} = 8.586\,\mathrm{ms}$$

   $$d_{c}^{net}(x) = \Omega_{net}\big(d(sm,r)+d(r,s_{gw})+d(s_{gw},s_{dc})+d(s_{dc},h)\big) = 6.784\,\mathrm{ms}+1.696\,\mathrm{ms}+0.106\,\mathrm{ms}+0.106\,\mathrm{ms} = 8.692\,\mathrm{ms}$$

   As the results show, a slight network delay increase is suffered if services are distributed among multiple servers (1.250 % in the non-critical path, 1.235 % in the critical path). In comparison to SSMA, the hosting of multiple services on a single substrate may help to mitigate the virtualization overhead on network level.

The encountered network delays in VNFA scenario (a) are 41.111 % shorter than in SSMA in the critical path, 46.671 % in the non-critical, in scenario (b), the decrease is 40.375 % in the critical path and 46.012 % in the non-critical path, respectively.

2. To evaluate the influence of the introduction of hardware redundancy, more specifically the circle backup redundancy introduced in Section 4.4.2, it is assumed that the backup system has similar properties to the main system. In the case of the single user scenario, again, a check for resource availability and embedding possibility is skipped, due to the assumptions given in Section 6.1.3.

   The assumption for the all-for-all backup analysis are the same as for the circle backup. However, to estimate $Perf$ and A for this type of substrate redundancy, the number of employed servers in the VNFA infrastructure need to be known. As in the single user scenario, the number of required substrate servers is limited to one by default, in the case of all-for-all backup, the number of available backup servers (which is $n-1$, where n is the number of overall servers) influences how often services may migrate in the presence of substrate failures.

   Due to these restrictions, to give a meaningful availability estimation for VNFA (which is impossible using only a single server, thereby effectively nullifying any substrate redundancy options) an evaluation using one to three available substrate servers is done. The results are displayed in Tables 6.13 and 6.14.

The results show two key findings: First, if either circle or all-for-all backup are used, the required availability $\geq 98\,\%$ can be achieved in both scenarios (a) or (b) (assuming that two or more servers are present). Second, if no substrate redundancy is employed, however, the required availability is not reached (if no other dependability-enhancing measures are used, such as replicas).

The usage of replicas can enhance the dependability of VNFCs, however, the limit of replica usage can be estimated using the results of Sections 3.2.1, 3.2.3 and 6.2.1. This leads to the conclusion that the availability of a VNFA AMI can—without the introduction of either higher quality smart meters/routers or the usage of hardware sided replicas, which is not part of this thesis—never exceed 98.651 %.

		Number of servers								
		1			2			3		
		Replicas			Replicas			Replicas		
		0	1	2	0	1	2	0	1	2
Scenario (a)	No redundancy	97.724	98.126	98.142	97.724	98.126	98.142	97.724	98.136	98.152
	Circle backup	97.724	98.126	98.142	97.955	98.255	98.285	97.955	98.255	98.285
	All-for-all backup	97.724	98.126	98.142	97.955	98.255	98.285	97.994	98.297	98.326
Scenario (b)	No redundancy	97.712	98.113	98.129	97.712	98.113	98.129	97.712	98.123	98.139
	Circle backup	97.712	98.113	98.129	97.943	98.242	98.272	97.943	98.242	98.272
	All-for-all backup	97.712	98.113	98.129	97.943	98.242	98.272	97.981	98.284	98.313

Table 6.13.: VNFA availability *[%]* in single user scenario using 0, 1, and 2 replicas for (either dc or tps)

		Number of servers								
		1			2			3		
		Replicas (each)			Replicas (each)			Replicas (each)		
		0	1	2	0	1	2	0	1	2
Scenario (a)	No redundancy	97.724	98.533	98.565	97.724	98.533	98.565	97.724	98.533	98.565
	Circle backup	97.724	98.533	98.565	97.955	98.574	98.615	97.955	98.574	98.624
	All-for-all backup	97.724	98.533	98.565	97.955	98.574	98.615	97.994	98.615	98.657
Scenario (b)	No redundancy	97.712	98.520	98.552	97.712	98.520	98.552	97.712	98.157	98.552
	Circle backup	97.712	98.520	98.552	97.943	98.587	98.611	97.943	98.561	98.602
	All-for-all backup	97.712	98.520	98.552	97.943	98.587	98.611	97.981	98.561	98.602

Table 6.14.: VNFA availability *[%]* in single user scenario using 0, 1, and 2 replicas each (gw and either dc or tps)

6.2.3 Evaluation: Passau City Scenario

Using the results of the single service and single user evaluations, the Passau city scenario is analyzed to offer further insights regarding the performability of VNFA in a larger application environment and a comparison between SSMA and VNFA.

6.2.3.1 Passau City Scenario: Test Environment

In order to evaluate different virtualization models and their impact on the AMI performance, a set of tools is used to create and analyze suitable test networks. As there exists—to the best of the author's knowledge—no application that can be used for every single aspect of the necessary studies, multiple existing programs are used. To be able to automatically generate, analyze and compare VNFA scenarios, a software framework called *AMIgo* is implemented that can create models of the studied AMI networks and convert them from their model representation into queuing networks to be subsequently evaluated. A description of this software and its interaction with other tools is provided in the following. In *AMIgo*, the Algorithms for Embedding of Virtual Networks (ALEVIN) framework is used for both *mapping* and *validation* purposes, which are both explained in the upcoming paragraphs.

Embedding. ALEVIN is used for embedding when creating the virtualized AMI networks. The virtual gateway, data concentrator and third party nodes of the neighborhoods need to be mapped onto the substrate servers. Every type of virtual node has its own demands that need be respected. For the embedding, the algorithm detailed in Algorithm 4.3 is used.

Validation. In non-virtualized networks, ALEVIN is used for validation. As the name suggests, no virtual machines are used in this topology. Nevertheless, they are introduced artificially to have ALEVIN check the CPU and bandwidth requirements of the AMI services. This ensures that the network has enough resources to serve all its customers. The actual embedding is done by *AMIgo* as the network structure defines exactly which virtual machines are supposed to be hosted on which substrate node. The resource model employed in this VNF embedding simulation regards two types of resources, namely CPU cycles and link bandwidth.

***AMIgo* Program Flow.** When running *AMIgo*, at first, a plan for all neighborhoods, each consisting of multiple users, is generated. The plan respects the settings that are made in the network definition file (see Appendix A.1). The result of this step is a template for each neighborhood that follows the basic architecture of an AMI as depicted in Section 2.1.2.2. Every neighborhood has its own data concentrator and multiple users. Each user in a neighborhood has one smart meter, one router, one gateway, and optionally TPSs.

Depending on the topology, the generated network nodes are either marked as virtual or physical. The plan for each of the neighborhoods is the same for all network topologies to provide comparability. Three copies of this plan are sent to the different topology handlers, where at first the neighborhoods are converted so that ALEVIN can map the virtual nodes.

After the Virtual Network Embedding (VNE) is finished, the embedded networks are converted into an eXtensible Markup Language (XML)-based format required by JMT and analyzed. To account for possible failures in the substrate network and analyze their impact, after the networks are created and assessed, substrate servers are eliminated according to a crash plan. The crash plan determines the order in which servers are eliminated from the substrate network. *AMIgo* subsequently determines

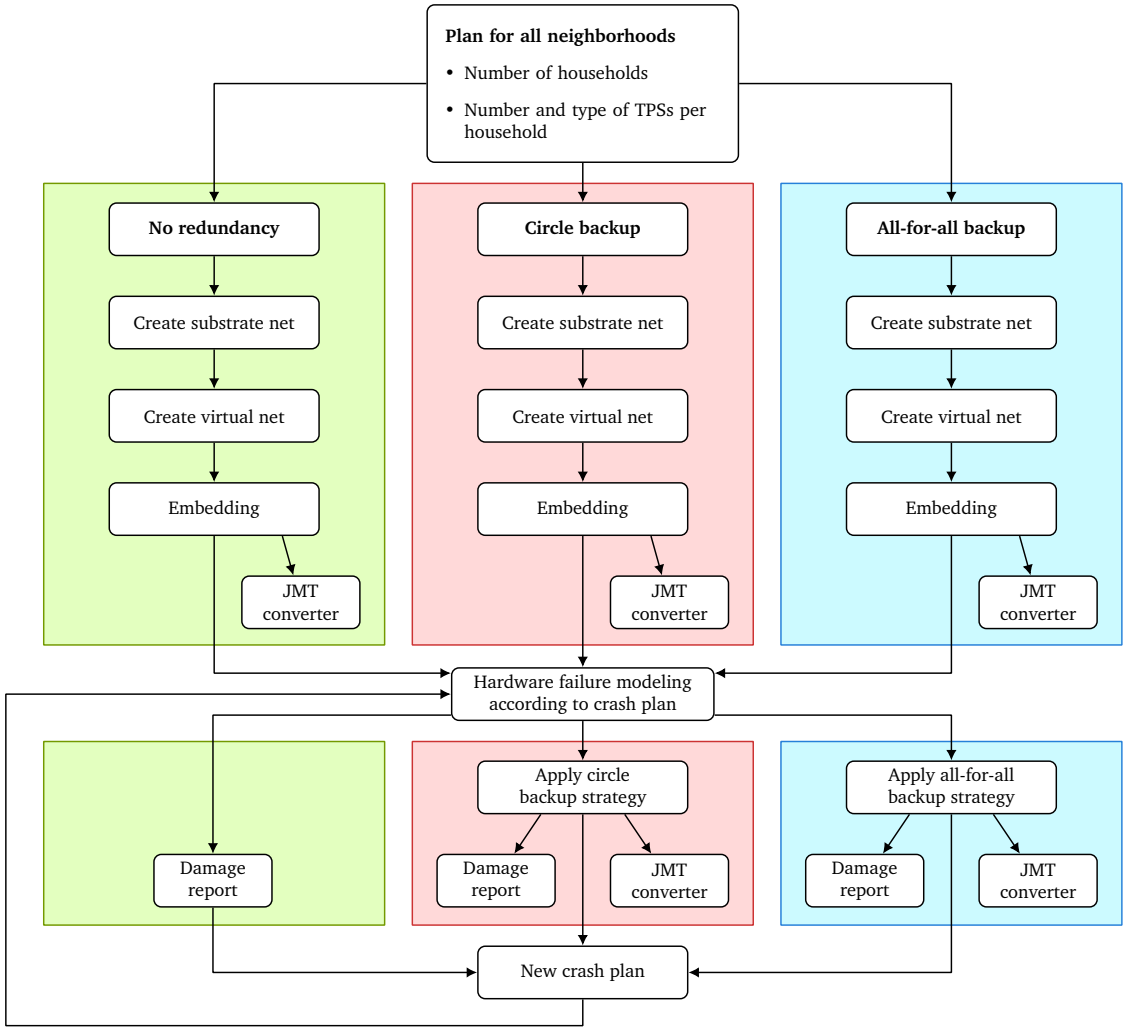


Figure 6.12.: *AMIgo* program flow

the amount of damage that is inflicted to the network and checks whether the virtual machines can be migrated to other servers. The resulting networks are again converted to queuing networks for performance measurements. Finally, another crash plan is constructed in which one additional server fails and the process is repeated until all servers have failed. Figure 6.12 depicts the algorithm as a flow chart.

6.2.3.2 Passau City Scenario: *AMIgo* Performance Evaluation

In order to subsequently be able to analyze complex AMI scenarios, first, the *AMIgo* simulator itself is evaluated regarding its computational performance using the following scenario: An increasing number of neighborhoods (one to 20) is created, every neighborhood has a random number (between five and ten) of users. For each number of neighborhoods, the tests are repeated ten times to counter random effects. The values in the figures depict the mean values of those ten runs. In the tests, the simple Dijkstra algorithm is not used, as none of the embeddings succeeded, which is due to its too simplistic embedding strategy making it unsuited for complex networks.

At first, the analysis focuses on finding the most computationally complex part of the simulation process. Figure 6.13 illustrates the results of this analysis (using an example AMI network with twelve neighborhoods, each featuring ten users). The legend of those figures contains four entries.

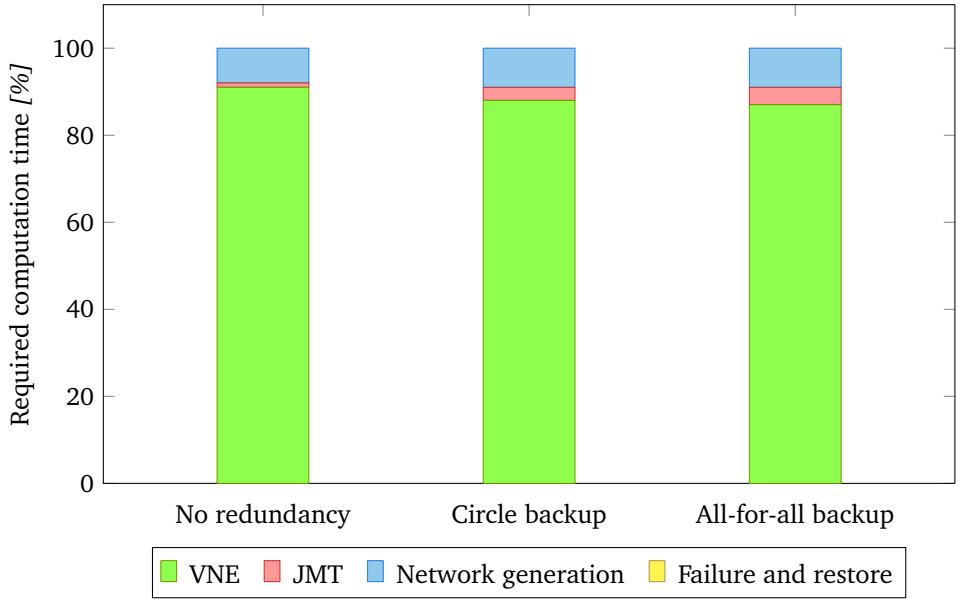


Figure 6.13.: *AMIgo* runtime distributions

VNE describes the time that ALEVIN requires to perform the VNE, JMT summarizes the run time of the queuing network converter, the export into the JMT XML format and writing the files to disk. Network generation includes the time to convert the neighborhood template into a finished network. Failure and restore is the maximum time required to simulate the failure impact including applying the backup strategy of the topology.

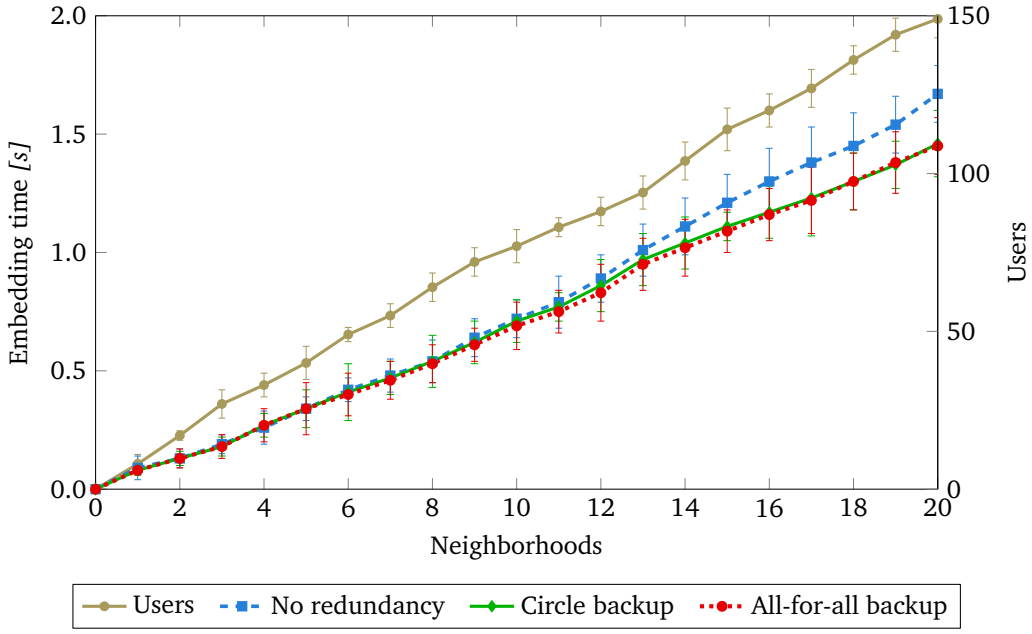


Figure 6.14.: *AMIgo* embedding time requirements

The results clearly indicate that the VNE induces the highest computational burden, therefore it is further analyzed. The other steps are insignificant in comparison: The network generation has a similar duration for all topologies and increases slightly with a growing number of neighborhoods up to 160 ms (95 % Confidence Interval (CI) 15.32 ms). This is one order of magnitude lower than

the embedding time. The conversion to queuing networks takes 90 ms (95 % CI 8.49 ms). The failure simulation and restoration is also negligible taking less than 10 ms (95 % CI 2.14 ms). The results of the embedding benchmark are shown in Figure 6.14. In addition to the embedding times it shows the number of users which grows linearly as expected.

6.2.3.3 Simulation Parameters

The simulation parameters used during the assessments are described next.

- **Investigated AMI topologies:** The three backup options introduced in Section 4.4.1 are implemented so that they can be compared. Figure 6.15 depicts them in a graph: For SSMA, only the no redundancy option is available, while for VNFA, all three topologies (no redundancy, circle, all-for-all backup) are investigated.

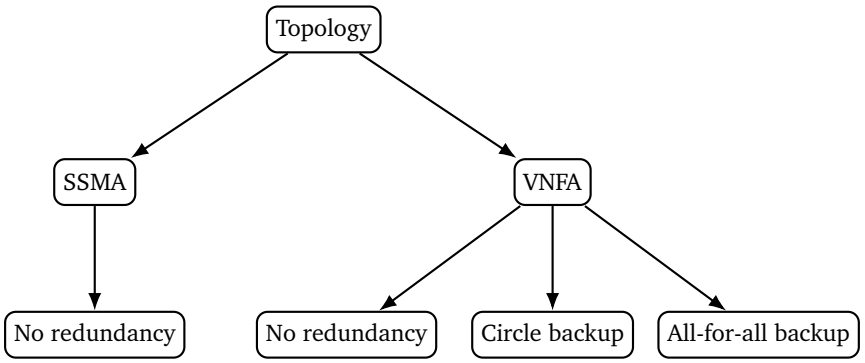


Figure 6.15.: Investigated AMI topologies

  The no redundancy topology is similar to the currently suggested SSMA AMI systems, which are elaborated in detail in Section 2.1.2.2. In this architecture every neighborhood has its own physical data concentrator node; every user has its own physical smart meter, gateway and router; every third party service has its own physical server. As the name suggests, no backups exist that can be used if any nodes fail. Every failure results in users that are no longer connected to the AMI network or that have no access to third party services.

  In addition to the no redundancy topology, the two VNFA backup approaches are analyzed. In the circle backup topology, every neighborhood has a virtual data concentrator; every user still has a physical smart meter and router, but the gateway and third party services are virtualized. All the virtual nodes are placed on COTS servers that host services. Every server has exactly one predefined node which offers its spare resources as backup. The all-for-all backup is closely related to the circle backup approach. It also uses virtualization but instead of a fixed physical backup server, every other server offers its spare resources as backup to all other servers.

- **ALEVIN:** The offered and demanded resources for ALEVIN nodes are directly transferred off the macro model. As described in Section 6.1.1, PLC is used in NAN, DSL in the WAN environment, while ZigBee is employed in the HAN. The links demand different bandwidths depending on their usage purpose, as depicted in Table 2.1.2.2. From the smart meters to the data concentrators, which includes third party services, 9 kbps are required; the WAM part of the networks requires 6 kbps in SSMA. In VNFA, the overhead is included by multiplying these values with Ω_{net} each.

  The values of CPU for the nodes represent a relative value, not real-life CPU cycles; they are used to compare the resources and demands of different nodes. Depending on the AMI

architecture present (SSMA or VNFA), different substrate node types are available (cf. Section 5.3.3): In SSMA networks, $r^{hw}_{type} \in \mathbb{T}, \mathbb{T} = \{sm, gw, r, dc, tps\}$ are available, whereas in VNFA networks, $r^{hw}_{type} \in \mathbb{T}, \mathbb{T} = \{sm, r, g\}$ can be found. Table 6.15 gives a summary of the offered and demanded CPU cycles. Because the main task of a data concentrator is assumed to be similar to that of a gateway (en-/decryption of data), the CPU demand of a data concentrator is assumed to be a multiple of a gateway's CPU demand depending on the users served. Equation (6.6) is used to estimate the parameter $d^s_{cpu_{dc}}$.

$$d^s_{cpu_{dc}} = d^s_{cpu_{gw}} E(u)\chi, \tag{6.6}$$

where $d^s_{cpu_{dc}}$ is the data concentrator's CPU demand, $d^s_{cpu_{gw}}$ the CPU demand of the gateway, $E(u)$ the expected number of users and χ a correction factor. The expected number of users is based on the numbers in Section 2.1.2.2. χ is chosen to represent that not all smart meters send data at the same time, but unsynchronized. The numerical value for $d^s_{cpu_{dc}}$ is therefore $2.5 \cdot 100 \cdot 0.8 = 200$. The TPSs have same CPU demand as the gateway. The resources present are calculated as follows:

The resources that a TPS substrate node in a non-virtualized network needs to offer are calculated by *AMIgo*. This is done by considering the maximum number of users that are assigned to a TPS substrate node, the resources a single user demands and by respecting the maximum load that the server is allowed to have. This is formalized as:

Let T be the set of TPS types. Let $C_t \in \mathbb{N}, t \in T$ be the maximum number of users that a substrate node of TPS type t can serve. Let $d^s_{cpu_t} \in \mathbb{R}^+_0, t \in T$ be the amount of demanded CPU resources per customer by TPS type t. Let $l_{max} \in \mathbb{R}^+ \leq 1$ be the maximum load that is allowed on a substrate server. Then the offered resources of a substrate node hosting TPS type t, designated Λ_t, need to be:

$$\Lambda_t : T \Rightarrow \mathbb{R}^+_0, (t) \Rightarrow \frac{d^s_{cpu_t} \cdot C_t}{l_{max}}$$

If more users are assigned to a TPS of type t, additional substrate nodes of the same TPS type are created by *AMIgo* automatically during the initial embedding. In the Passau city scenario both C_{cm} and C_{eo} are assumed to be 1000 each, leading to $\Lambda_{cm} = \Lambda_{eo} = \frac{3 \cdot 1000}{0.8} = 3750$.

Node type	r^{hw}_{cpu}	d^s_{cpu}
sm	2	1.5
r	2	1.5
gw	3	2.5
dc	200.5	200
tps_{cm}	*	3
tps_{eo}	*	3
g	1000	-

Table 6.15.: Offered and demanded CPU resources of nodes in *AMIgo*

- **Queuing Networks:** Two types of rates must be defined for the queuing networks: Arrival and service rates. As described in the assumptions (cf. Section 6.1.1), the arrival rates are given as uniformly distributed random variables λ_{sm} and λ_h, respectively. The estimated service rates for SSMA are taken from Section 6.2.2.2; for VNFA, service times and rates highly depend on

three main factors: the substrate resources dedicated to a virtualized service, the virtualization overhead, and the service's performability.

Node type	τ **[s]**	$\mu\,[\frac{1}{s}]$ **(at** $Perf$ **= 1.0)**
r	0.001	1000
gw	0.025	40
dc	0.0167	60
tps_{cm}	0.0333	30
tps_{eo}	0.0333	30

Table 6.16.: Service times and rates of queuing stations

Except for ALEVIN, *AMIgo* utilizes JMT for the performance analysis. To be able to process the previously generated and mapped networks, it is important to state that ALEVIN networks cannot be directly read by queuing network tools; therefore, a conversion is required that is handled by *AMIgo* and not further elaborated here, though it is noted that the conversion of virtualized networks into queuing networks presents itself as a challenging task. For further details regarding this, it is suggested to refer to Brand [23].

6.2.3.4 Passau City Scenario: Evaluation of SSMA Model

After the analysis of the run time performance of the *AMIgo* simulation framework and the parameter definition, the dependability and performance of SSMA are analyzed.

Dependability Results. The dependability analysis for SSMA shows expected results. As failures impact specific substrate entities (as no "common" substrate nodes, such as servers in VNFA, exist), the resulting impact on the AMI depends on the substrate entity failing. The impairments suffered are depicted in Figures 6.16 (sm, r and gw failures), 6.17 (dc failures), 6.18 (tps_{cm} failures), and 6.19 (tps_{eo} failures).

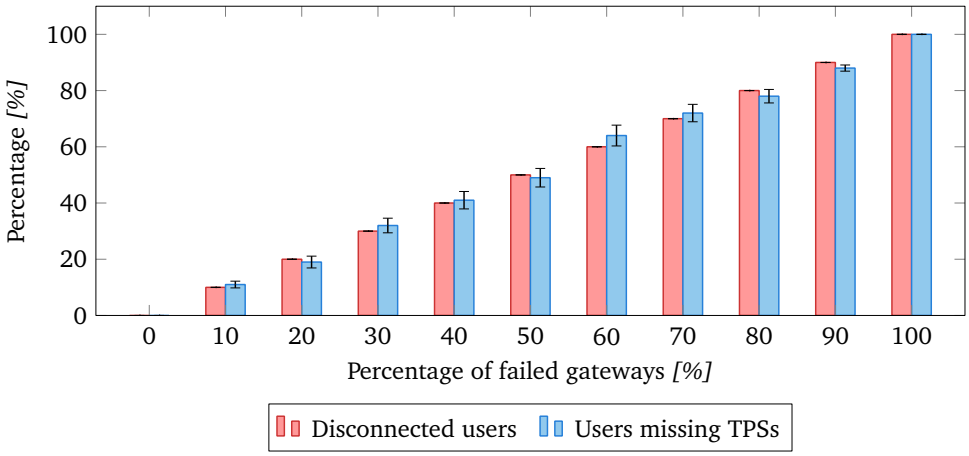


Figure 6.16.: Influence of sm, r, and gw failures on SSMA dependability

As the results show, the impact of sw, r and gw outages is deterministic regarding the number of disconnected users, which is to be expected, as each of these entities is required for a single user to transfer his data to a TPS and/or the dc and finally the service provider's HES. Therefore, each sw, r or gw failure results in exactly one user being disconnected. The result differs for the number of users losing their TPSs: As only 40 % of users use a tps_{cm} and 70 % a tps_{eo} service, the number of

users missing a TPS has a slight variation and is not as consistent with the number of failed gws as the amount of disconnected users, however, the trend is similar.

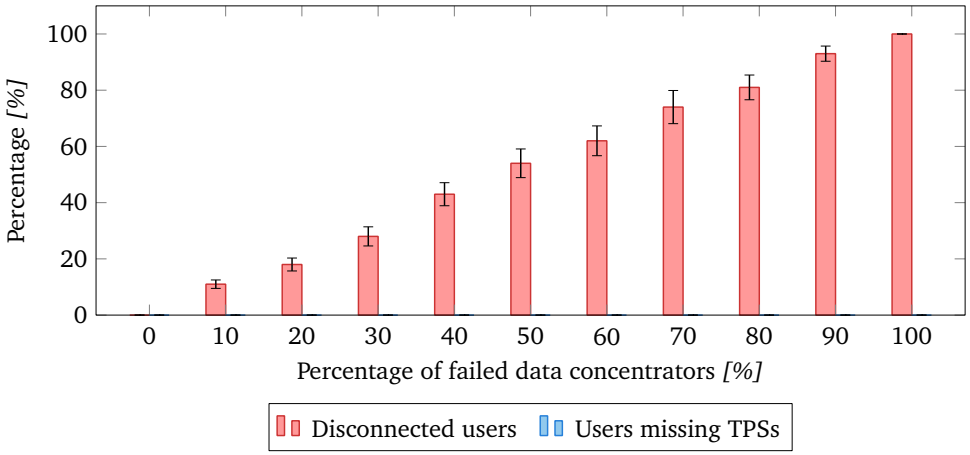


Figure 6.17.: Influence of dc failures on SSMA dependability

A failure in a dc effects a single neighborhood, each containing 70 to 100 users. These users get disconnected once their neighborhood's dc fails. TPSs are generally not impaired if dc nodes fail. Due to the varying number of users within each neighborhood, the overall number of disconnected users slightly varies in subsequent simulation runs, however it keeps monotonically rising.

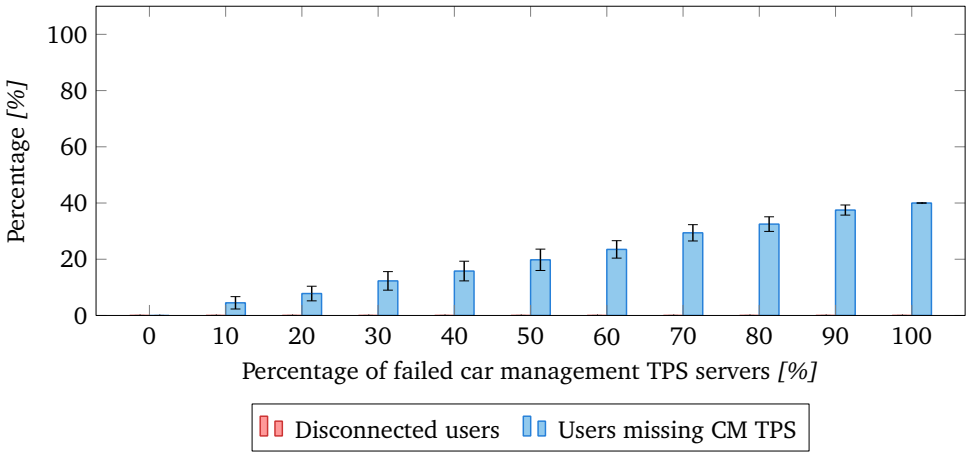


Figure 6.18.: Influence of tps_{cm} failures on SSMA dependability

Finally, looking at failures in either tps_{cm} or tps_{eo}, it is apparent that neither do effect the connections of users to the dc and the provider's HES. Instead, as previously noticed in this paragraph, the maximum amount of users impaired by outages—in the case of tps_{cm}—are $\lceil 0.4|U| \rceil = 5534$, where U is the set of users. For tps_{eo}, the maximum amount of affected users is therefore $\lceil 0.7|U| \rceil = 9686$. Assuming the server capacity being calculated as Λ (cf. Section 6.2.3.3), the server count accounts to $\lceil \frac{0.4|U|d^s_{cpu_{cm}}}{\Lambda_{cm}} \rceil = 6$ for tps_{cm}; in case of tps_{eo}, this results in $\lceil \frac{0.7|U|d^s_{cpu_{eo}}}{\Lambda_{eo}} \rceil = 10$. These results are reflected in Figures 6.18 and 6.19, depicting the rising number of users missing TPSs as more and more servers of the respective TPS type fail.

With the dependability results of the single user scenario present, it is possible to estimate the amount of disconnected users in a transient manner, enabling a calculation of how many users' metering data is unavailable to the service provider at each point in time.

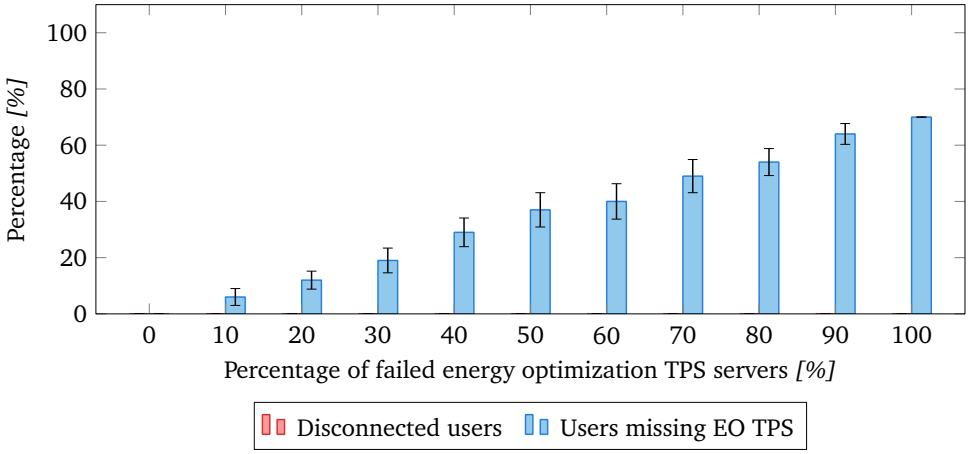


Figure 6.19.: Influence of tps_{eo} failures on SSMA dependability

Performance Results. Table 6.18 shows the results of the JMT calculations. All values are means of the ten simulation runs in JMT and have a relative error of 0.02. tps_{cm} and tps_{eo} are the car management and energy optimization TPSs that are used as example services in this scenario. The offered CPU resources and demands as well as service rates are as defined in Tables 6.15 and 6.16. The 95 % CIs are given in brackets.

Node	Queuing time *[ms]*	Response time *[ms]*	Utilization *[%]*
r	0.0 (± 0.0)	1.6 (± 0.2)	0.01 (± 0.0)
gw	0.0 (± 0.0)	26.3 (± 3.7)	0.17 (± 0.0)
dc	0.5 (± 0.1)	17.6 (± 2.5)	5.68 (± 0.6)
tps_{cm}	8.6 (± 1.4)	36.2 (± 4.9)	27.98 (± 4.3)
tps_{eo}	5.2 (± 0.8)	34.5 (± 4.4)	17.33 (± 3.7)

Table 6.18.: Results of *AMIgo* simulation for SSMA (mean values)

Comparing the previous results of the single user scenario with Table 6.18, two conclusions can be drawn: First, the results of both analyses closely match; second, the latency calculated in the Passau city scenario are slightly higher (3.740 % in the non-critical path, 3.360 % in the critical path) than the single user results, which is attributed to queuing delays due to more users being present:

- **Single user scenario results:**
  $d_{nc}(x) = 74.9\,\text{ms}$
  $d_c(x) = 59.7\,\text{ms}$
- **Passau city scenario results:**
  $d_{nc} = d(gw) + d(r) + d(tps) + d_{nc}^{net} = 1.6\,\text{ms} + 26.3\,\text{ms} + 35.4\,\text{ms} + 14.4\,\text{ms} = 77.7\,\text{ms}$
  $d_c = d(gw) + d(r) + d(dc) + d_c^{net} = 1.6\,\text{ms} + 26.3\,\text{ms} + 17.6\,\text{ms} + 16.1\,\text{ms} = 61.6\,\text{ms}$

For the estimation of the non-critical path latency, the mean values of both TPSs are used. This calculation style is also employed in the upcoming VNFA analysis.

6.2.3.5 Passau City Scenario: Evaluation of VNFA Model

The dependability analysis of VNFA and the performance measurement using queuing networks is described in the following sections. In the performability evaluation, it is on the one hand investigated whether VNFA is able to satisfy AMI's performance requirements; on the other hand, it is tested how different backup approaches influence VNFA's dependability.

Dependability Results. The dependability analysis of VNFA is much more complex than of SSMA, mainly due to two reasons. First, the hosting of various service types belonging to multiple users on a single substrate server leads to more unpredictable results upon a substrate failure. Second, the backup strategies used in VNFA lead to a more sophisticated handling of failures, which in turn allows to cope with a higher number of substrate failures. To be able to capture all of these effects, *AMIgo* generates detailed reports regarding the effects of substrate failures. This includes, among other metrics:

- the neighborhoods and users that are affected by the failure,
- the VMs (gateway, data concentrator, TPSs) that are re-instantiated at other hosts,
- the users with missing TPSs because of failures,
- the mandatory and optional services that have to be shut down due to insufficient system resources, and
- the server load.

The example network consists of 160 neighborhoods, each having between 70 and 100 users, which results in a total of 13 835 users in the scenario. A detailed analysis and comparison of the three investigated VNFA topologies (see Figure 6.15) is performed in the following. To be able to compare the behavior of the three network topologies, again, persistent failures are used to gradually reduce the amount of available substrate servers. Doing so, the impact of an increasing number of failures on each topology can be measured and compared. In these test, only ten repetitions are used; this way, it is deliberately taken into account that the standard deviations remain higher than in other evaluations to estimate the stability of each approach.

If there are no substrate backups available, failures in the hosting servers lead to a similar amount of disconnections and lost TPSs among users. As no redistribution actions are triggered to migrate services from a failed server to another after a substrate failure, each failure causes the unavailability of a certain amount of households and TPSs. The exact number of failed services and disconnected users depends on the service composition hosted on each server. The server load is relatively constant, disregarding the outages on substrate level. This is to be expected, because no services are redistributed to the remaining servers, therefore, the server load is indifferent towards failures on substrate level. Figure 6.20 depicts these findings.

For the circle backup topology, the system load behaves similarly to the no redundancy case except for its higher CIs. This is due to the premise that only one specific server may act as a backup. If this node has no spare resources or has also failed already, no restoration is possible. This leads to an instant disconnection of users—their numbers rise monotonously. No clear relation between missing TPSs and disconnected users can be drawn, but the number of users with missing TPSs decreases at one point (60 to 50 available servers). This happens if one or more virtual data concentrator services can no longer be hosted. Then, the virtual gateways are removed and their resources are freed, which are offered to the TPSs that had to be suspended formerly. A similar effect is described

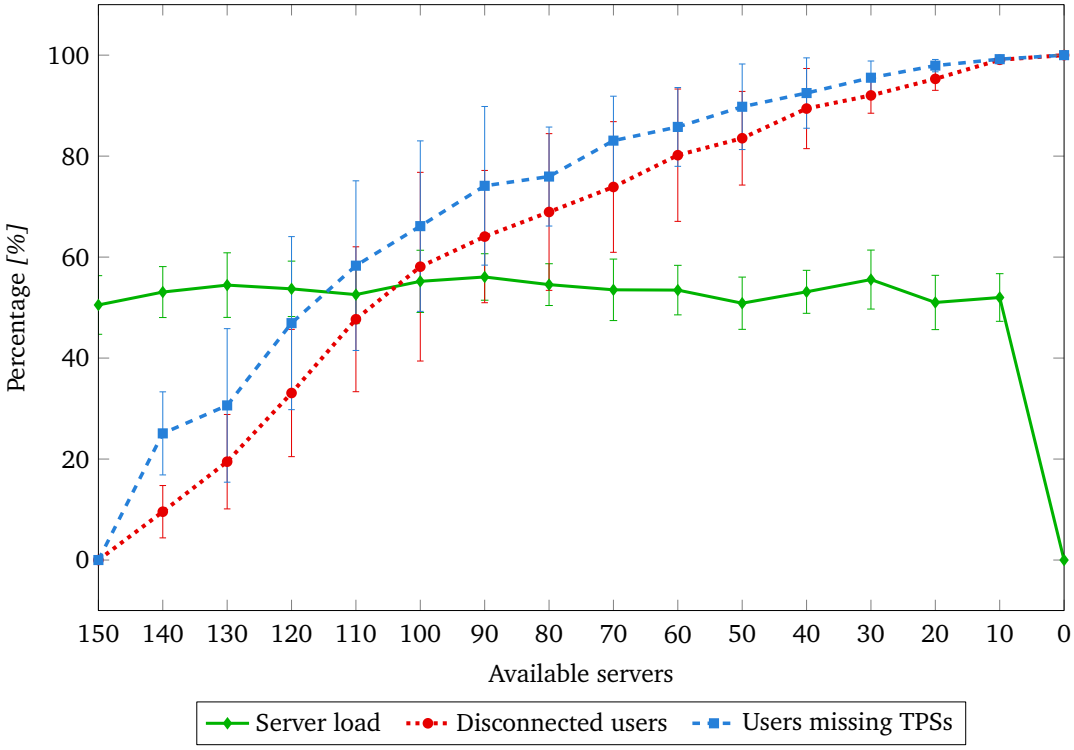


Figure 6.20.: No redundancy performability test

in Section 4.4.1, where the re-connection of formerly disconnected users is explained. These results are visible in Figure 6.20.

Figure 6.21.: Circle backup performability test

The all-for-all backup topology behaves strongly different compared to both previous topologies. In Figure 6.22, three phases can be distinguished. In the beginning, the spare resources of servers can compensate the failure of substrate nodes up to a point where the load reaches 100 % (available servers ≥ 110). Until then, no users become disconnected and all TPS services can be kept operational. However, once more servers fail and no spare resources are available, the service priorities are used to decide that the optional TPSs are gradually removed to free resources for the users' mandatory services (available servers ≥ 70). In the last phase users start to become disconnected. At this point, no TPSs are left that can be removed, therefore virtual data concentrators and gateways can no longer be hosted.

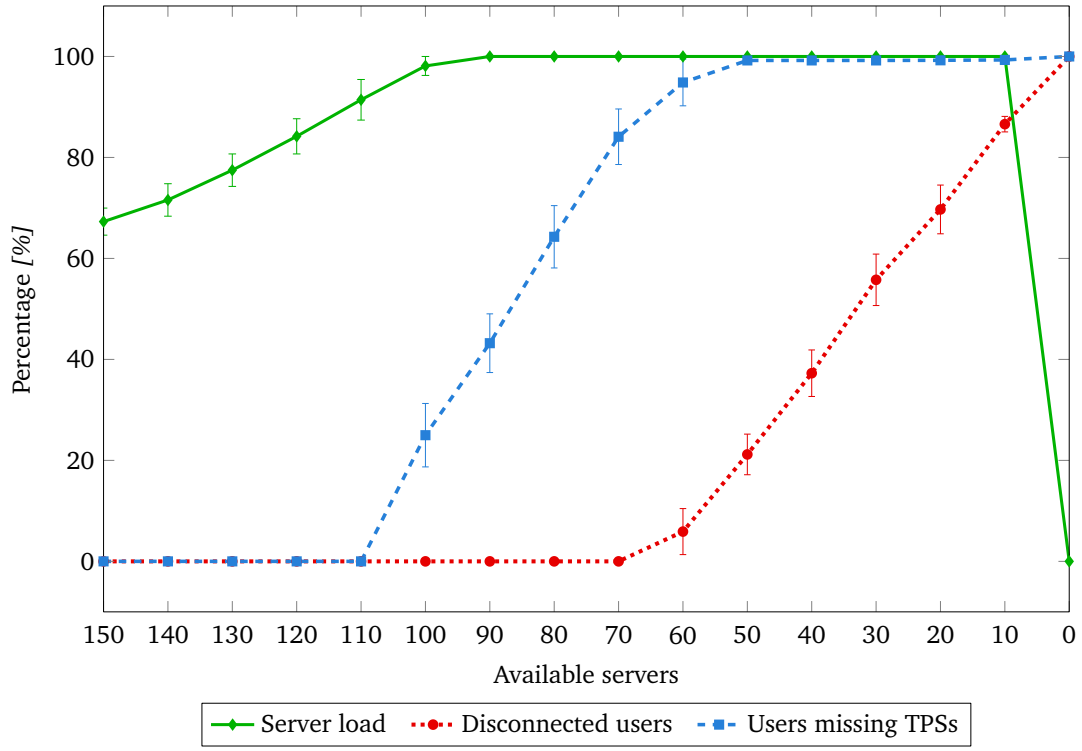


Figure 6.22.: All-for-all backup performability test

The results show that the behavior of all-for-all backups are stable, i. e. the amount of disconnected users, missing TPSs and the server load is predictable. Although suffering from much larger CIs, the circle backup topology also offers a better dependability as the no redundancy topology. The system load does not reach 100 % although users become disconnected, which is the main disadvantage compared to the all-for-all backup strategy, besides the more unstable results achieved.

Performance Results. To assess the performance of VNFA, *AMIgo* obtains the following metrics during each JMT simulation run:

- System response time
- System throughput
- Information about metrics of an individual node both per class and for all classes combined:
  - Number of packages
  - Queuing time of packages

- Response time of packages at one single station (= queuing time + service time)
- Utilization
- Throughput

In VNFA the virtualization overheads Ω_{cpu} and Ω_{net} apply. The arrival rate for both queuing networks is the same. In Table 6.20, the results of the JMT analysis are displayed using ten randomly selected servers as an example. Dashed table cells represent that a server did not process any packages of the respective service class. The results of the VNFA evaluation show that the response times approximately equal the sum of queuing and service times. An exact match cannot be expected because all service rates are modeled as exponentially distributed random variables.

It is notable that the queuing times in all classes are higher than in the SSMA approach, which is a result of the co-location of numerous virtual services on a single substrate server. This challenge could be fixed by using multi-processor stations in the queuing network, which would not be unrealistic, as multi-core systems are widely adopted in real life environments. Although a multi-core server analysis is not part of this thesis, it is a worthwhile topic for future work. Thus, the mean response times of VNFA service are slightly higher than those of SSMA services, which is further discussed in Section 6.2.4.

Using the performance results of VNFA in combination with the network delays as calculated in Section 6.2.2.3, the overall delays encountered can be estimated. It is noteworthy at this point that the results for the VNFA no redundancy and circle backup strategies are not listed in their entirety; instead, only the results are given in the following. Again, the two scenarios (a) and (b) from Section 6.2.2.3, where in the first, all VNFCs are located on the same host, while in the second, all VNFCs are distributed and therefore hosted on different substrate nodes.

- **VNFA, all-for-all backup:**
  (a) Combining the network delays $d_{nc}^{net}(x)$ for non-critical paths and $d_{c}^{net}(x)$ for critical paths of a user x, the following overall delays are derived:
  $$d_{nc}(x) = d(r) + d(gw) + d(tps) + d_{nc}^{net} = 1\,\text{ms} + 27.5\,\text{ms} + 44.1\,\text{ms} + 8.48\,\text{ms} \approx 81.1\,\text{ms}$$
  $$d_{c}(x) = d(r) + d(gw) + d(dc) + d_{c}^{net} = 1\,\text{ms} + 27.5\,\text{ms} + 23.7\,\text{ms} + 8.59\,\text{ms} \approx 60.8\,\text{ms}$$
  (b) As already established in Section 6.2.2.3, the differences of scenarios (a) and (b) are minimal in the given scenario, as the upcoming calculations show:
  $$d_{nc}(x) = d(r) + d(gw) + d(tps) + d_{nc}^{net} = 1\,\text{ms} + 27.5\,\text{ms} + 44.1\,\text{ms} + 8.59\,\text{ms} \approx 81.2\,\text{ms}$$
  $$d_{c}(x) = d(r) + d(gw) + d(dc) + d_{c}^{net} = 1\,\text{ms} + 27.5\,\text{ms} + 23.7\,\text{ms} + 8.69\,\text{ms} \approx 60.9\,\text{ms}$$
- **VNFA, circle backup:**
  (a) $d_{nc}(x) \approx 81.3\,\text{ms}$, $d_{c}(x) \approx 61.0\,\text{ms}$
  (b) $d_{nc}(x) \approx 81.4\,\text{ms}$, $d_{c}(x) \approx 61.1\,\text{ms}$
- **VNFA, no redundancy**[19]**:**
  (a) $d_{nc}(x) \approx 81.6\,\text{ms}$, $d_{c}(x) \approx 61.3\,\text{ms}$
  (b) $d_{nc}(x) \approx 81.8\,\text{ms}$, $d_{c}(x) \approx 61.4\,\text{ms}$

[19] As no migration rejuvenation is available in the no redundancy topology, a warm rejuvenation strategy is employed.

Metric	Node	Service class			
		gw	dc	tps_{cm}	tps_{eo}
Queuing time *[ms]*	$S0$	2.8 (± 0.3)	17.9 (± 1.7)	18.8 (± 2.1)	17.9 (± 2.1)
	$S1$	2.3 (± 0.1)	14.7 (± 1.1)	15.2 (± 1.4)	15.4 (± 1.5)
	$S2$	2.7 (± 0.2)	-	16.7 (± 1.4)	16.6 (± 1.8)
	$S3$	2.1 (± 0.1)	18.4 (± 2.1)	15.7 (± 1.6)	15.3 (± 1.2)
	$S4$	2.2 (± 0.2)	15.8 (± 1.9)	16.1 (± 1.9)	16.5 (± 1.6)
	$S5$	1.8 (± 0.1)	17.4 (± 2.2)	15.9 (± 1.8)	15.0 (± 1.1)
	$S6$	2.4 (± 0.2)	18.6 (± 2.4)	16.6 (± 2.0)	16.3 (± 2.0)
	$S7$	2.3 (± 0.3)	15.1 (± 1.4)	16.4 (± 1.8)	16.2 (± 1.8)
	$S8$	1.9 (± 0.1)	-	16.0 (± 1.5)	16.1 (± 1.7)
	$S9$	1.9 (± 0.2)	16.6 (± 1.7)	15.8 (± 1.6)	15.9 (± 1.4)
	Mean	2.2 (± 0.2)	16.8 (± 1.8)	16.3 (± 1.7)	16.1 (± 1.6)
Response time *[ms]*	$S0$	28.6 (± 3.1)	23.8 (± 2.7)	41.3 (± 4.2)	42.1 (± 3.8)
	$S1$	26.7 (± 2.7)	23.5 (± 2.3)	43.1 (± 4.1)	42.2 (± 4.2)
	$S2$	26.9 (± 2.7)	-	47.9 (± 5.0)	46.2 (± 4.7)
	$S3$	27.1 (± 2.8)	23.7 (± 2.5)	44.1 (± 5.4)	45.8 (± 4.9)
	$S4$	27.9 (± 3.3)	22.8 (± 2.0)	41.2 (± 3.8)	40.2 (± 3.6)
	$S5$	28.3 (± 2.4)	23.0 (± 1.8)	45.4 (± 4.6)	44.2 (± 4.8)
	$S6$	26.5 (± 1.9)	23.3 (± 2.3)	46.7 (± 5.2)	47.8 (± 3.9)
	$S7$	27.0 (± 3.3)	22.9 (± 2.9)	42.2 (± 3.7)	43.1 (± 4.3)
	$S8$	27.2 (± 3.1)	-	45.9 (± 4.3)	47.2 (± 4.8)
	$S9$	27.8 (± 2.6)	23.4 (± 1.9)	41.5 (± 5.5)	42.8 (± 3.8)
	Mean	27.5 (± 2.8)	23.5 (± 2.3)	43.9 (± 4.6)	44.2 (± 4.3)
Utilization *[%]*	$S0$	0.20 (± 0.0)	3.43 (± 0.4)	3.49 (± 0.2)	3.46 (± 0.2)
	$S1$	0.21 (± 0.0)	3.43 (± 0.3)	3.48 (± 0.3)	3.52 (± 0.2)
	$S2$	0.17 (± 0.0)	-	3.46 (± 0.2)	3.47 (± 0.1)
	$S3$	0.19 (± 0.0)	3.47 (± 0.3)	3.39 (± 0.2)	3.33 (± 0.3)
	$S4$	0.15 (± 0.0)	3.35 (± 0.4)	3.55 (± 0.3)	3.50 (± 0.2)
	$S5$	0.20 (± 0.0)	3.41 (± 0.2)	3.46 (± 0.3)	3.47 (± 0.2)
	$S6$	0.22 (± 0.0)	3.71 (± 0.3)	3.48 (± 0.1)	3.49 (± 0.2)
	$S7$	0.23 (± 0.0)	3.49 (± 0.2)	3.50 (± 0.2)	3.52 (± 0.2)
	$S8$	0.17 (± 0.0)	-	3.46 (± 0.2)	3.44 (± 0.3)
	$S9$	0.21 (± 0.0)	3.29 (± 0.1)	3.57 (± 0.3)	3.48 (± 0.1)
	Mean	0.19 (± 0.0)	3.45 (± 0.2)	3.48 (± 0.2)	3.47 (± 0.2)

Table 6.20.: Results of JMT calculations for VNFA (mean values)

6.2.4 Result Comparison

In this chapter the VNFA approach introduced in Chapter 4 and modeled in Chapter 5 is analyzed and compared to SSMA regarding its performability. To do so, three scenarios (single service, single user, and Passau city) are introduced in Section 6.1 along with several assumptions made during

the evaluation. After that, the performability assessments are covered in Section 6.2. The results of the analysis are compared in the following by first analyzing the properties of the *AMIgo* simulation framework; second, conclusions are drawn from the simulation findings of each AMI architecture separately for all three scenarios. Third, the overall consequences resulting from these individual findings are drawn.

- ***AMIgo* simulation framework:** Regarding the performance of *AMIgo*, it became apparent that small simulations are solvable within a matter of seconds; however, the Passau city scenario, comprising more than 10 000 users, requires the generation and analysis of "city-sized" queuing networks, which leads to a very high computational complexity in JMT. However, to counter this effect, only minor changes to the generated queuing networks would be required (e. g. the simplification of networks by committing an explicit modeling of routers). The *AMIgo* simulation also revealed that the dependability of the all-for-all backup topology is higher in comparison to both the circle backup and no redundancy topologies. Also, employing service priorities (σ) during the embedding algorithm has positive effects on dependability in cases where a resource shortage is present in the substrate.

- **SSMA:** Looking at the results of SSMA, several properties of the infrastructure become apparent. Depending on the concrete realization of an SSMA service, its performability properties vary. If no replica and no rejuvenation is present, an SSMA service offers a steady-state availability of 99.326 %. While this is reasonably high, the performability is severely degraded, achieving no more than 84.682 %. Using rejuvenation, the performability may be enhanced to 99.323 % while maintaining an availability of 99.323 % using dynamic rejuvenation and—using deterministic rejuvenation—a performability of 98.468 % with an availability of 98.891 %, respectively.

  Due to the limited availability of a single service in SSMA, in the single user scenario SSMA is unable to reach an availability higher than 96.815 %, i. e. the overall availability is not able to achieve the required 98 %. In the preliminary analysis given in Section 3.1.2, the availability of SSMA for a single user is calculated as 94.25 %, which aligns well with the analysis made here, resulting in an SSMA availability of 96.815 %. The performance of SSMA shows that it is able to achieve the required AMI performance in a single user scenario, having a latency of 74.9 ms for non-critical and 59.7 ms for critical traffic.

  In the Passau city scenario, the dependability of SSMA shows straightforward results if permanent failures of substrate entities are considered. Depending on the type of failing substrate node, varying effects are observable: In case of gateway failures, traffic directed at both TPSs and the HES is impaired; if data concentrators fail, only HES, if TPS servers fail, only TPS traffic is impaired, respectively. The steady-state availability of SSMA in the Passau city scenario is mainly influenced by the "shared" entities of the AMI, i. e. the data concentrators (each serving up to 100 users) and TPS servers (each serving up to 1000 users)[20].

  Regarding the performance of SSMA, the observations of the single user scenario translate to the Passau city scenario, as stated in Section 6.2.3.4 with a slight increase in the observed overall latency. This is due to queuing effects caused by multiple users sending jobs to the same substrate server. The results show that the 100 ms requirements are fulfilled in all cases for SSMA, though the maximum latency reaches up to 77.7 ms for non-critical and 61.6 ms for critical traffic.

[20] If the service provider's HES would be part of the performed dependability analysis, it would need to be listed here too; however, the HES is excluded as stated in the assumptions given in Section 6.1.1.

- **VNFA:** For the VNFA approach, in addition to the deterministic and dynamic timings, three different rejuvenation strategies are compared: cold, warm, and migration. The results show that warm and migration rejuvenation perform slightly better than the cold rejuvenation strategy in both availability and performance comparisons. Using a combination of deterministic and warm rejuvenation has an availability increase of 0.083 % and a performance increase of 0.149 %. With dynamic rejuvenation the availability increases by 0.105 % and performance is increased by 0.105 %.

  The availability increase of migration rejuvenation is 0.121 % and a performance increase of 0.187 % using a deterministic rejuvenation timing. A combination of dynamic timing and migration rejuvenation yields an availability increase of 0.219 % and the performability increase of 0.213 %.

  In the Passau city scenario, VNFA's dependability behavior severely differs from SSMA; even using different VNFA backup strategies have a strong influence on the results. The comparison of the no redundancy, circle backup and all-for-all backup shows that the behavior of all-for-all backups behaves more stable (i. e. much lower CIs), thus having a more predictable amount of disconnected users, missing TPSs and server load. Although suffering from much larger CIs, the circle backup topology also offers a higher dependability than the no redundancy topology. In circle backups, the system load does not reach 100 % although users become disconnected, which is the main disadvantage compared to the all-for-all backup strategy, besides the more unstable results achieved. Using the no redundancy topology, each failing server leads to missing TPSs and disconnected users, the severity of each failure depends on the service composition hosted on each substrate server. Thus, the consequences of a server failure are hardly predictable if no a priori knowledge regarding the service composition is given, which cannot be safely assumed as services get distributed without a predetermined composition for each substrate node. Furthermore, the servers' load is almost independent from the amount of failures occurring, which is due to the fact that no services are migrated in the event of failures. Similar to the SSMA case, the steady-state availability in the Passau city scenario is majorly influenced by the loss of "shared" entities of the AMI.

  Performance-wise VNFA behaves in a similar manner as SSMA: The overall latency of VNFA is 81.1 ms for non-critical and 60.8 ms for critical traffic in cases where all services are hosted on the same substrate server using the all-for-all backup strategy (the latency increase using circle backup is 0.250 % for non-critical and 0.330 % for critical traffic, using no redundancy 0.620 % for non-critical and 0.820 % for critical traffic, respectively). In cases where all services are distributed among different substrate nodes, the respective latency is 81.2 ms for non-critical and 60.9 ms for critical traffic (increasing 0.250 % for non-critical and 0.490 % for critical traffic using circle backup, 0.740 % for non-critical and 0.990 % for critical traffic if no redundancy is employed).

6.3 Proof-of-Concept: Virtualized Smart Meter Gateway

As a full implementation and subsequent validation of the performability results of VNFA is impossible, an example of a VNFA service—namely a vSMGW—is demonstrated and analyzed in the next sections to answer if:

- vSMGWs can provide the functionalities required in AMIs,
- the performance of vSMGWs is sufficient, and
- vSMGWs offer an acceptable dependability.

To analyze the properties of the suggested vSMGWs, a prototypical implementation of a vSMGW is assessed regarding its network and computational performance as well as its dependability properties. Before that, the tasks of an SMGW are briefly summarized and the implementation of a vSMGW is explained.

6.3.1 Gateway Task Description

Before the virtualization of the gateway devices is performed, in the following, their architecture and functionality is analyzed. As the BSI states, gateways may differ in their architecture, depending on the features they implement (e. g. communication capabilities can either be provided by the gateway or by another device, such as a router) [25]. Figure 6.23 depicts a scenario that uses the minimal architecture presented by the BSI, where the gateway does not include an internal communication subsystem, but relies on an external communication device (here a user's router) [25]. The inner organization of a gateway realizing a minimal architecture is depicted in Figure 6.23. The gateway module requires connectivity to three independent communication interfaces, which in our case are located inside the router. These communication interfaces interconnect the gateway with the LMN, HAN and NAN/WAN. The security module offers additional security-related functions to the gateway. The gateway is also the central component that collects, processes, and stores meter data. It is in charge of handling meter data and submission to authorized external entities (e. g. a TPS provider).

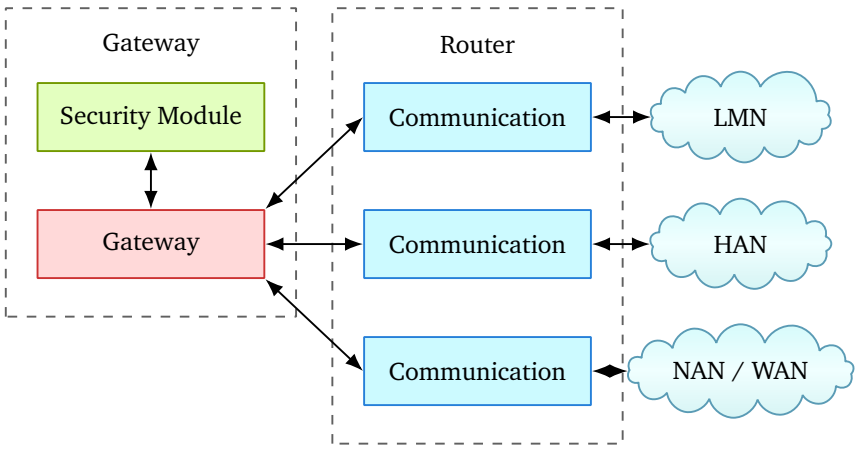


Figure 6.23.: Example of a minimal gateway design, based on the BSI [25]

Summarizing, the following features and tasks have to be fulfilled by gateway devices and need to be implemented in their virtual counterparts.

- **LMN interface:** At the LMN interface, the measured values of the attached meter(s) are collected in regular intervals and timestamped. It is determined which data needs to be derived from measured data based on a set of rules. Finally, the derived data is sent to eligible external participants.

- **HAN interface:** The HAN interface provides three internal communication interfaces: First, a CLS interface through which CLS can communicate with external market participants in the WAN over a secure TLS connection. The second interface is for end users through which the client is able to retrieve his stored information; third an interface for service technicians to view configuration profiles and system logs for diagnostics.
- **NAN/WAN interface:** The wide-area interface is used for the transmission of measured values on the basis of analysis and communication profiles. It provides pseudonymization functions for not billing-relevant data to protect the identity of customers. In addition, it is required for the reception of administration and configuration information from smart meter gateway administrators, as well as the handling of firmware updates. Lastly, the NAN/WAN interface offers a wake-up service, which, upon receipt of a particular data packet of the smart meter gateway administrator, sets up a communication connection to the same.

6.3.2 Proof-of-Concept Implementation

The vSMGWs are implemented using a Java runtime environment, realizing the following key features:

- Creation of three isolated network interfaces (LMN, HAN, NAN/WAN)
- Reception and storage of energy consumption information originating from LMN interface
- Output of energy consumption information through HAN interface
- Establishment of connection to NAN/WAN (by request of gateway administrator)
- Output of energy consumption information through NAN/WAN interface (after connection establishment)
- Modification of CLS power state from NAN/WAN interface

Moreover, each SMGW runs within an isolated VM preventing mutual influence of vSMGWs. The vSMGWs use three different LAN-ports of the substrate server to ensure not only logical, but also physical isolation of the LMN, HAN and NAN/WAN interfaces.

6.3.3 vSMGW Performability Analysis

The performability analysis of the vSMGWs is executed by first selecting a suitable test environment in Section 6.3.3.1. After that, the different test scenarios and subsequent evaluations are covered.

6.3.3.1 Test Environment

To evaluate vSMGWs, a test environment is set up consisting of two hosts h_1 and h_2 (h_1: Windows 7, Genuine Intel CPU U4200 @ 1.30 GHz, 4 GB RAM; h_2: Windows 7, Intel Core 2 Duo CPU T9400 @ 2.53 GHz, 4 GB RAM), and a server s_1 (Debian 9.0.0, Intel XEON @ 2.53 GHz, 4 GB RAM). h_1 sends data to h_2 via server s_1 over a network path composed of Ethernet links l_i, each having a length of 1.5 m and a bandwidth of 100 Mbps. For the data transfer, HTTP/2 was used in conjunction with TCP/IP.

For the evaluation, several vSMGWs ($vSMGW_i^1$) are spawned at server s_1. If the entire memory is made available to s_1, a large number vSMGWs (depending on the memory requirements of the

vSMGWs) can be hosted. To limit the number of creatable vSMGWs and thereby decreasing the evaluation time, the memory available on s_1 for the creation of vSMGWs is limited to 512 MB. The test environment is depicted in Figure 6.24.

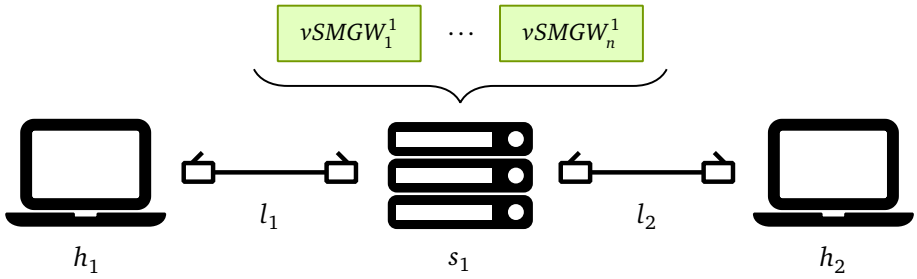


Figure 6.24.: vSMGW test environment

The maximum number of simultaneously hostable vSMGWs can be estimated by dividing the available memory (512 MB) by the required memory of each vSMGW, which is, using a minimal installation of Linux as a base for a vSMGW, 36 MB. This results in a maximum of 14 co-hosted vSMGWs at a time.

6.3.3.2 Performance Evaluation

The performance of the vSMGWs is analyzed in two ways: First, the startup performance is evaluated to test how long a vSMGW takes from execution to actual readiness; Second, the throughput and latency of vSMGWs is assessed using several test scenarios with focus on the behavioral changes if many vSMGWs are active simultaneously.

- **Startup time test:** This test assesses the startup times of vSMGWs. The performed assessments are organized in two scenarios: First, it is investigated how long the generation vSMGWs takes starting one after another in sequential order. Precisely, the time from the generation request to the point where the vSMGW is booted up and in an operational state is measured. The test is run while the hosting router is running an increasing number of additional vSMGWs at the same time.

  As the results in Figure 6.25 show, the startup time increases if more vSMGWs are hosted, which is a logical consequence of the fewer available resources. Precisely, on average, the first gateway takes 2.8 s to start, while the 14^{th} takes 5.3 s, which is an increase of 89.3 %. The sequential startup time test results clearly have an exponential growth, as the approximation function $f(x) = 0.000005355 \exp(x) + 0.081x + 8.335$ obtained by symbolic regression shows.

  Second, the simultaneous startup of several vSMGWs and its influence on the overall startup time is investigated. To do so, the number of simultaneously starting routers is varied from two to 14. The first observation is that a parallel start of several gateways takes shorter than a sequential start. For example, starting 14 vSMGWs in parallel takes 38.7 s, while the sequential start would need 44.9 s, which is approximately 16.0 % faster. The results are illustrated in Figure 6.26. The parallel startup time test suggest a nearly linear growth, as visible in the approximation function $f(x) = 7.76x - 3.24$.

- **Throughput test:** The data throughput is a critical value of a server, especially in its functionality as a vSMGW, where QoS restrictions apply. Therefore, in this test, it is determined how big the impact of a different number of active vSMGWs is on T. In the assessment, the T

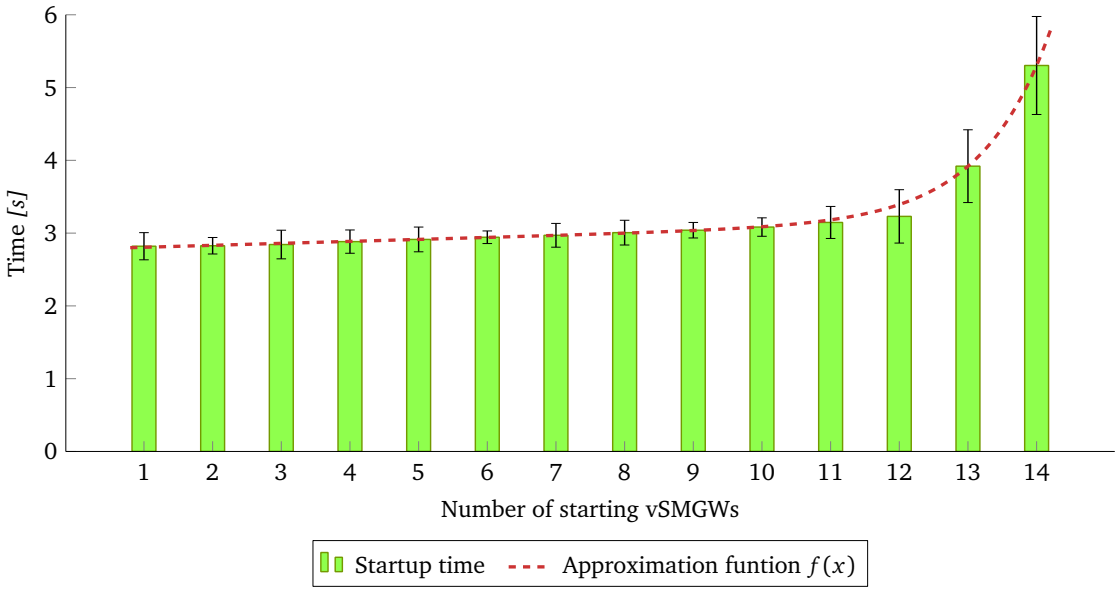


Figure 6.25.: Results of the startup time test (sequential starting)

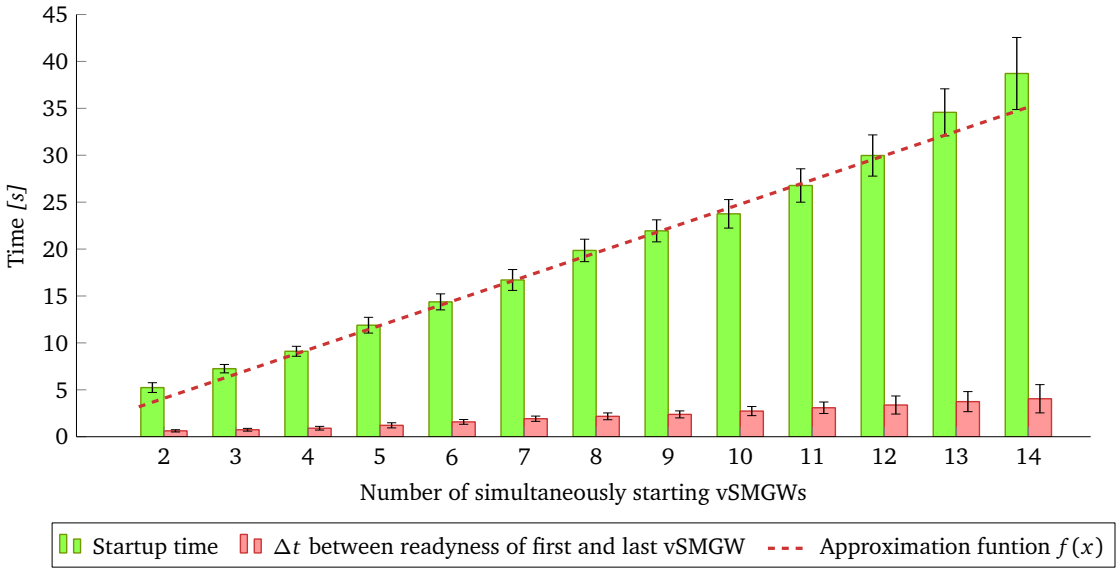


Figure 6.26.: Results of the startup time test (parallel starting)

from h_1 to h_2 is measured and the impact originating from the number of active vSMGWs is observed. As might be expected, T decreases with each additional active vSMGW. To test the impact of active vSMGWs on T, a 30 MB file is transferred via Secure Copy (SCP). This test is carried out with up to 14 simultaneously active vSMGWs. The evaluation shows that the number of active vSMGWs impacts T on a vSMGW to host level. The transfer time while 14 vSMGWs are active is 144.500 % higher than with one active vSMGW on average. However, the throughput does not decrease linearly, rather, a polynomial time requirement increase is approximated by $f(x) = 0.000825x^4 + 24.3822$, as depicted in Figure 6.27.

- **Latency test:** To investigate L, the Round Trip Time (RTT) is measured to calculate the system's response time. As before, the tests are conducted using up to 14 active vSMGWs. In each test,

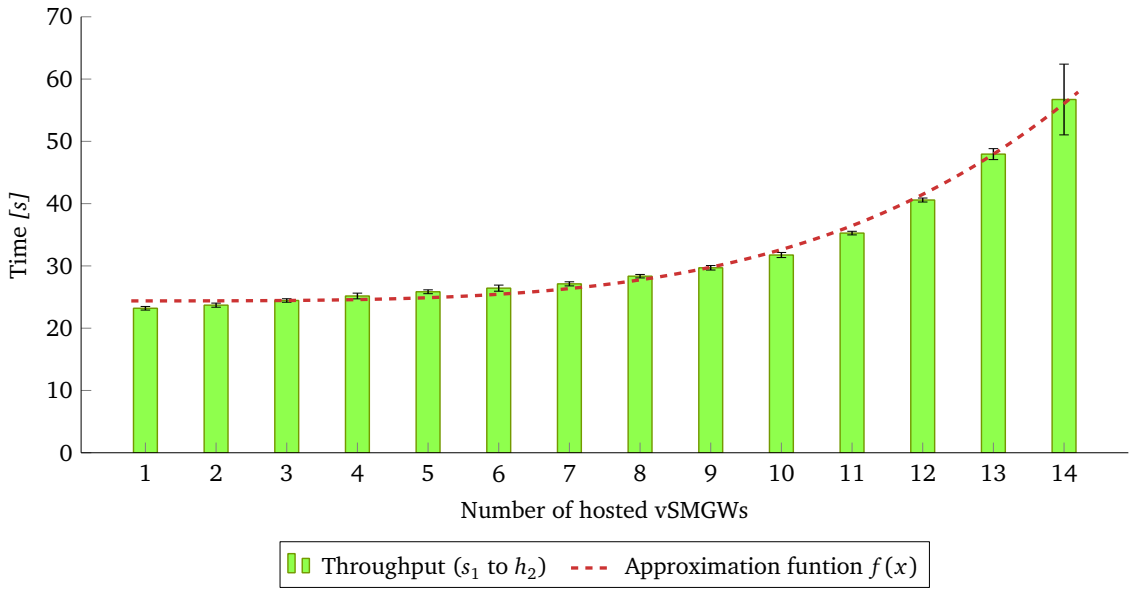


Figure 6.27.: Results of the throughput test (s_1 to h_2)

50 pings of 65 535 B are sent with the tool hrPING[21] to the receiving system using the following commandline:

```
hrping.exe 1.1.1.2 -l 65535 -n 50 -W
```

For each number of running vSMGWs ten test repetitions with 50 pings are performed. In addition, it is evaluated whether the latency of the system itself changes due to a different number of active vSMGWs. For this, a connection to the vSMGW HAN interface is established and a data transfer is initiated. Here, the time between the issuing of the "transfer data" command and the complete reception of data is measured. This corresponds to the sum of the processing time and transmission time (the connection establishment time is ignored). The test is repeated 10 times for one active vSMGW and each additionally running vSMGW.

The results are illustrated in Figure 6.28. It becomes visible that L is only marginally influenced by a rising number of vSMGWs. More precisely, the latency in case of ping testing with 14 active vSMGWs is 12.3 % higher than with one vSMGW, using data requests, 14 active vSMGWs have a 20.9 % higher L value than with one vSMGW. The approximation function $f(x) = 0.1563x + 16.17$ indicates a linear increase in latency with rising number of active vSMGWs. Also, the latency and throughput requirements stated in 2.1.2.2 can be fulfilled using vSMGWs, as the influence of virtualization on latency is negligible in the assessment and required throughput of 100 kbps can even be met with 14 active vSMGWs.

6.3.3.3 Dependability Evaluation

The dependability evaluation assesses both reliability and availability of the developed vSMGWs. To evaluate the dependability of a software application, several approaches are available. There is no comprehensive list of methodologies given here, because of briefness on the one hand, and the following restrictions, which further reduce the amount of applicable analysis methodologies, on the other hand:

[21] Ping Utility hrPING, URL: `http://www.cfos.de/en/ping/ping.htm`, last accessed: 06/14/2018

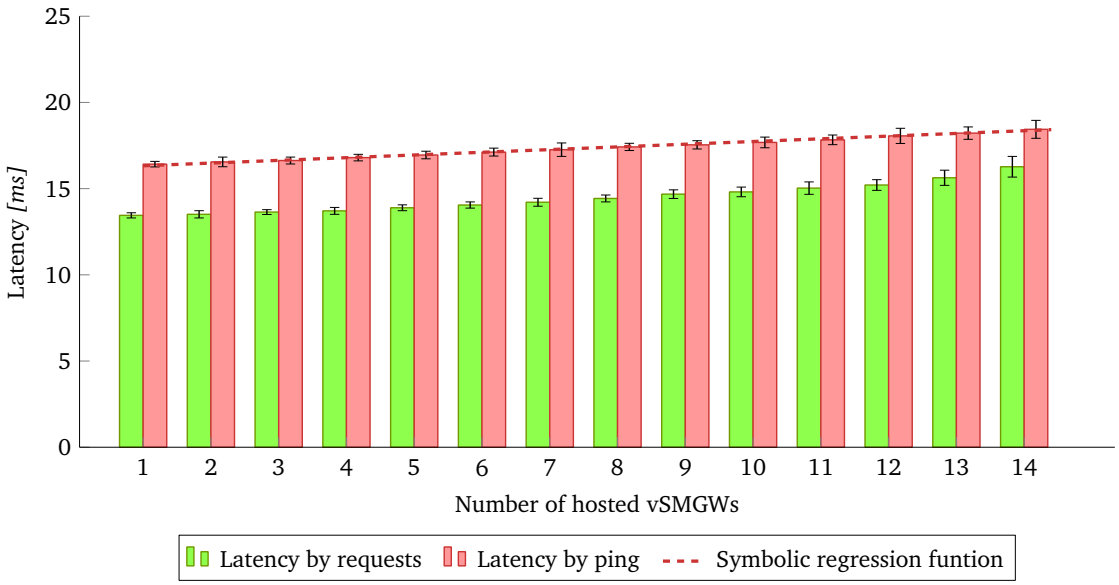


Figure 6.28.: Results of the latency test

- A source code testing of the vSMGW software is not performed, as it could not represent the failure behavior of vSMGWs in a real-life scenario properly. This is due to the lack of real-life input test data as well as the focus on the observable software behavior, which cannot be captured using source code-based white-box tests alone.
- As SMGWs—as well as vSMGWs—do not offer a possibility to interact with end users directly (i. e. via a user interface), no user-centric analysis is feasible/possible. Instead, analytic, automated processes are required for testing.
- While there were tests performed on a vSMGW application for a total time of 600 h, no longer term tests were possible due to time restrictions making a purely measurement-based analysis nearly impossible.

Therefore, to estimate the dependability of vSMGWs, the failure data obtained during the initial tests[22] conducted with vSMGWs is used as input into a software reliability growth model, more precisely a logarithmic Musa-Okumoto model, to predict the future failure rate, reliability and availability of vSMGWs. Musa-Okumoto models are based on several assumptions briefly listed in the following [104]:

- At time $t = 0$ no failures have been observed, i. e. $Pr\big(M(0) = 0\big) = 1$.
- The cumulative number of failures follows a Non-Homogeneous Poisson Process (NHPP) by its expected value function $\mu(t)$.
- The failure intensity decreases exponentially with the expected number of failures observed. This can be formally expressed as $\lambda(t) = \lambda_0 e^{-\Theta\mu(t)}$, where by λ_0 and Θ the initial failure rate and the failure rate decay is defined, respectively.

[22] The initial tests for the vSMGW were performed in a lab environment under constant supervision of the system. Once a system failure occurred, the vSMGW was restarted, using the escalating, multi-phase recovery approach described in Section 5.3.4.4. A detailed list of failure and recovery times are given in Appendix A.2.

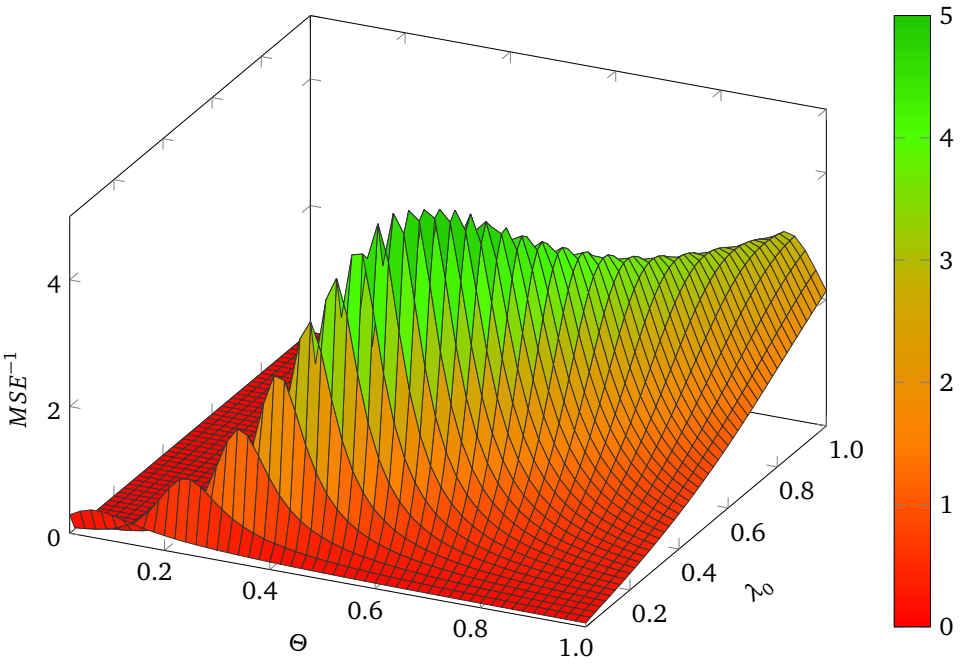
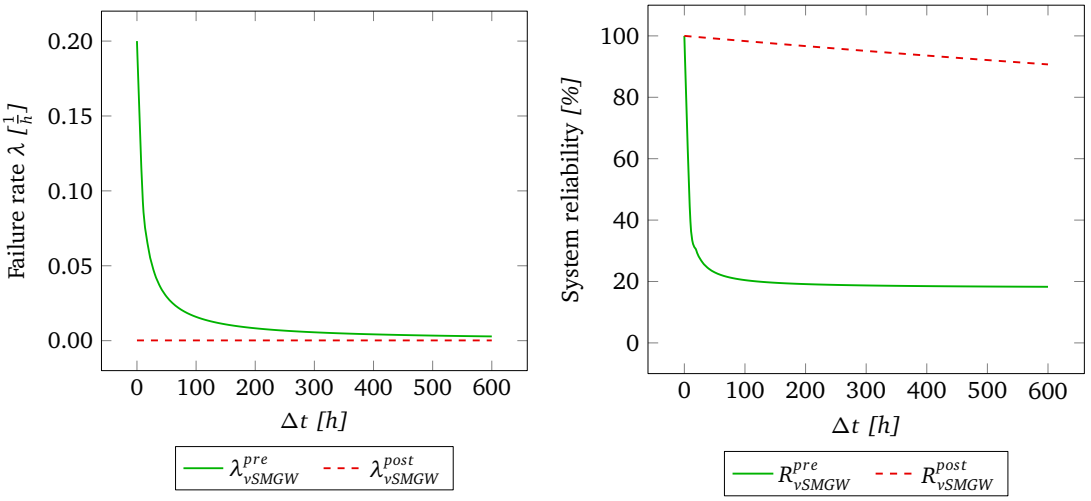


Figure 6.29.: MSE-based prediction of Musa-Okumoto Θ and λ_0 parameters

The logarithmic Musa-Okumoto model predicts the failures at time t using the equation $\mu(t) = \Theta^{-1} ln(\lambda_0 \Theta t + 1)$. To estimate the Θ and λ_0 parameters of the model, a fitting using a Mean Squared Error (MSE) approach is used, which is depicted in Figure 6.29. The figure displays an inverse MSE for the sake of easier visibility of the result. This yields $\Theta = 0.58$ and $\lambda_0 = 0.2$. While the result for Θ seems unusually high at first, it is reminded here that the vSMGW software is a minimal example of a vSMGW implementation, which only contains approximately 3000 lines of code. Therefore, the amount of bugs are also limited and mostly easy to fix, which leads to a high failure rate decay. Using



(a) vSMGW failure rate (pre- and post-burn-in)

(b) vSMGW reliability (pre- and post-burn-in)

Figure 6.30.: Development of vSMGW failure rate and reliability

the estimated Θ and λ_0 parameters, the expected future failure rate (using Equation (6.7)) as well as reliability (using Equation (6.8)) is possible. Two different scenarios are used to provide a good insight regarding the behavior of vSMGWs. On the one hand, the failure rate as well as the reliability

are analyzed right after the initial start ($t_{start} = 0\,\text{h}$, leading to $\lambda = \lambda(0\,\text{h}) = 0.2$); on the other hand, both reliability and failure rate are evaluated with the expected parameters after roughly one year $\left(t_{start} = 10\,000\,\text{h}$, leading to $\lambda = \lambda(10\,000\,\text{h}) = 1.723\text{E}^{-4}\right)$. The results are depicted in Figures 6.30a and 6.30b.

$$\lambda(t) = \frac{\lambda_0}{\lambda_0 \Theta t + 1} \tag{6.7}$$

$$R_{vSMGW} = 1 - e^{-\lambda(t)t} \tag{6.8}$$

For the evaluation of the availability, Equation (6.9) is used. As described previously in Section 5.3.1, the availability estimation is focused on the useful life phase of the application, meaning a nearly constant failure rate is reached. While a constant failure rate can never be reached using a Musa-Okumoto model (due to the constant Θ parameter), the availability is calculated assuming $\lambda = \lambda(10\,000\,\text{h})$ in Equation (6.9), yielding the results visible in Figure 6.31. Assuming the MTTR stays constant during the lifetime of the vSMGW application (which results in a MTTR of 1.833 h), the resulting availability is 99.968 %.

$$A_{vSMGW} = \frac{\lambda^{-1}}{\lambda^{-1} + \text{MTTR}} \tag{6.9}$$

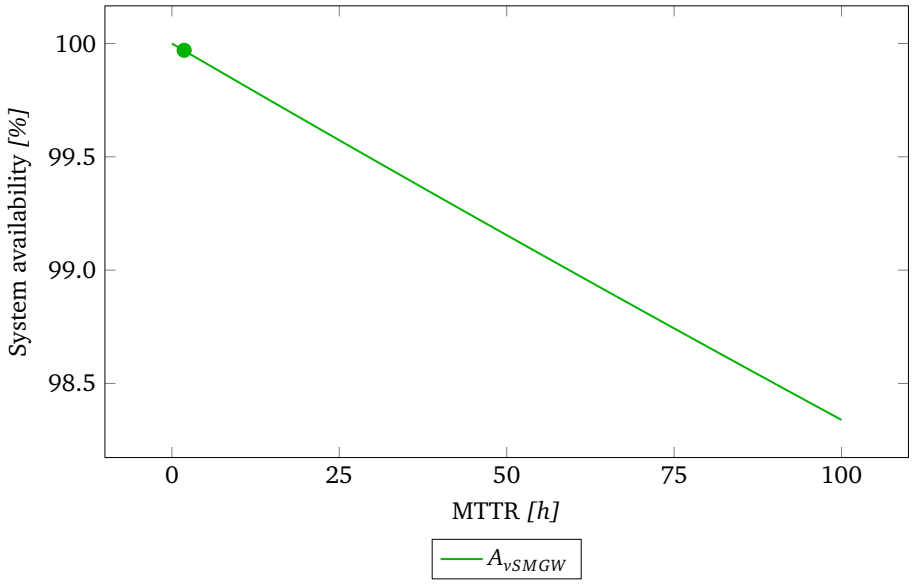


Figure 6.31.: Availability of vSMGWs in relation to expected MTTR

6.3.4 Results

Although the evaluation of vSMGWs in Section 6.3.3 only uses a non-optimized, minimal prototype, the results show that the approach works as a proof-of-concept implementation. Both the dependability and performance evaluations show results that are well within the acceptable levels stated in Table 2.1.2.2 in Section 2.1.2.2.

The performance analysis shows two main findings: The simultaneous hosting of multiple vSMGWs is both possible and feasible. The startup performance of vSMGWs is almost unimpaired up until

the resources of the hosting substrate are running low. The startup jitter between the first and the 12th starting vSMGW is only 14.54 %. Once the resources are depleted to over ≈ 90 %, severe increases become visible. The 13th vSMGW startup is 39.01 %, the 14th 88.06 % slower than the first one, respectively. The second result is that the network performance shows contradicting behavior comparing throughput and latency. While the influence of multiple vSMGWs on throughput is significant (maximum increase 144.5 %), the latency stays nearly constant, disregarding the number of hosted vSMGWs (maximum increase 12.3 % (ping)/20.9 % (request)).

In the dependability assessment, the failure data obtained during the tests conducted with a proof-of-concept vSMGW is used and input into a logarithmic Musa-Okumoto software reliability growth model. This yields the Musa-Okumoto parameters $\Theta = 0.58$ and $\lambda_0 = 0.2$. Using the Θ and λ_0 parameters, the current and expected future failure rates are calculated as $\lambda(0\,\text{h}) = 0.2$ and $\lambda(10\,000\,\text{h}) = 1.72265\text{E}^{-4}$. The estimation of dependability yields that the reliability is 351.3 % higher on average comparing a prototype at starting time $t_{start} = 0\,\text{h}$ and after one year of service $t_{start} = 10\,000\,\text{h}$ over a service time of 600 h. The availability after one service year—assuming the MTTR stays constant during the lifetime of the vSMGW application at 1.833 h—is 99.968 %.

6.4 Cost Analysis

After the performability analysis and a proof-of-concept implementation of virtualized AMIs, another important aspect is the monetary costs associated with an AMI implementation[23]. First off, it is required to state that the overall costs as well as benefits of an AMI system, being it a standard or virtualized solution, are hardly predictable if all factors are considered. This would include not only direct costs (including infrastructural establishment, intermittently occurring costs such as maintenance/repairs, as well as continuously/running costs, such as energy demand), but also potential changes that are either immediately triggered or established in a long-term manner by the introduction of AMIs, both on an individual and societal level. Each consumer could benefit through lower energy costs, innovative tariff schemes and the improved information on consumption, among others. On a societal level, the benefits include behavioral changes brought about by the compliance with energy savings and emission reduction targets, leading to reduced capacity demand, improved quality of supply, et cetera.

While this brief list is by far not exhaustive, it already shows that the estimation of the changes induced by AMIs exhaust multiple dimensions, therefore, not all effects are covered within the upcoming analysis. Instead, to be able to provide a reliable, quantitative cost analysis, only the direct costs mentioned in this paragraph are compared between SSMA and VNFA.

6.4.1 Deriving a Cost Function

To derive an appropriate cost function suitable for both non-virtualized SSMA AMIs as well as VNFA, the direct costs are analyzed. As already suggested in the AMI cost model developed by Wu *et al.*, the costs included and the assumptions made in the developed cost model are grouped by the continuity of their occurrence in the following list [173].

[23] Smart Meter werden teuer für Verbraucher - WELT, URL: https://www.welt.de/finanzen/immobilien/article164365881/Die-neuen-digitalen-Stromzaehler-Sparhilfe-oder-Kostenfalle.html, last accessed: 07/16/2019

1. **Static costs ($\mathcal{C}_s$):** Static costs only occur once in the mission time of a system. Prime examples are both hardware and software that needs to be developed/bought/deployed prior to an infrastructure's operation. It is assumed that the network infrastructure, i. e. the substrate network links and routers/switches, are already present and do not need to be installed beforehand. Also, *software* used on COTS components, such as OSs, router firmware, etc., is assumed to be already present free of charge. The static costs are calculated by:

$$\mathcal{C}_s(t,|U|) = \sum_{i=1}^{C_{hw}} (\| \frac{|U|}{cap_i} \| \cdot c_{hw_i}) + \sum_{j=1}^{C_s} c_{s_j}, \tag{6.10}$$

where t is the mission time, $|U|$ the number of users, C_{hw} a set of all hardware components, cap_i the number of users served by one component of type i, c_{hw_i} the costs of one hardware component of type i, C_s a set of all software components, and c_{s_j} the software development costs for component type j.

2. **Interval costs ($\mathcal{C}_i$):** These costs occur in regular or irregular intervals. Examples are the costs produced by maintenance operations to fix unplanned failures. On the one hand, the results of these failures are direct, as briefly discussed in Section 1.2, Niyato *et al.* [114], and Piaszeck *et al.* [123]: If no energy demand data is available to the service provider, this leads to the usage of estimations rather than current data, which in turn can incur costs due to power demand estimation errors resulting in power over- or under-supply. On the other hand, failed devices have to be repaired by the provider. The interval costs are calculated using:

$$\mathcal{C}_i(t,|U|) = \frac{t}{\text{MTBF}} \cdot (\alpha \overline{\text{MTTR}}_{hw} c_{rep_{hw}} + \beta \overline{\text{MTTR}}_{s} c_{rep_s}) + t \cdot |U| \cdot UA \cdot \overline{c}_{UA}, \tag{6.11}$$

where, in addition to the previously defined parameters, α and β are weighting factors ($\alpha, \beta \in [0,1] : \alpha + \beta = 1$), which determine the percentage of soft- and hardware entities in the AMI system. $c_{rep_{hw}}$ represents the hardware repair costs, c_{rep_s} the software repair costs, UA the unavailability of the AMI architecture, and $\overline{c}_{UA}$ the mean AMI communication failure cost per user and hour.

3. **Continuous costs ($\mathcal{C}_c$):** Continuous costs are constantly required while the infrastructure is active. These upkeep costs are mainly caused by the energy demand of the infrastructure's devices. It is assumed that devices demand energy both in failed and operational state.

$$\mathcal{C}_c(t,|U|) = t \sum_{k=1}^{C_{hw}} (\| \frac{|U|}{cap_k} \| \cdot d_{p_k}) \cdot c_{e_{Wh}}, \tag{6.12}$$

where, in addition to the previously defined parameters, d_{p_k} the power demand of one component of type k, and $c_{e_{Wh}}$ the energy costs per Wh.

To estimate the overall costs of an infrastructure design, it is necessary to first define a cost function that approximates the costs that need to be accounted in an AMI. As already suggested by Fawaz *et al.*, the dependability is an important, yet not the only factor that needs to be considered in AMIs [52]. To calculate the expected overall costs, the sum of Equations (6.10), (6.11) and (6.12) is taken, as shown in the proposed cost function Equation (6.13).

$$\mathcal{C}(t,|U|) = \mathcal{C}_s(t,|U|) + \mathcal{C}_i(t,|U|) + \mathcal{C}_c(t,|U|) \tag{6.13}$$

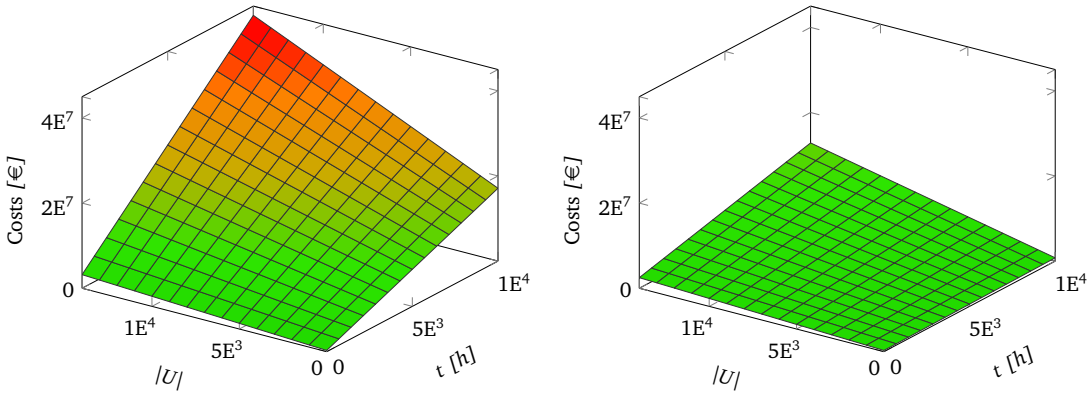
6.4.1.1 Example Scenario

To evaluate and compare the monetary impact of SSMA and VNFA, an example scenario is created. While the mission time and the number of users were left variable ($0\,\text{h} \leq t \leq 10\,000\,\text{h}$, $0 \leq |U| \leq 14\,000$), the following static parameter values are assumed, based on values given by the National Energy Technology Laboratory (NETL) in 2008 [107]:

		sm_{hw}	r_{hw}	gw_{hw}	dc_{hw}	s_{hw}	gw_v	dc_v
cap		1	1	1	100	100	1	1
c_{hw}	[€]	100	50	100	1000	5000	0	0
c_s	[€]	100 000	0	100 000	100 000	0	100 000	100 000
d_p	[W]	500	10	10	50	5	0	0
$c_{e_{Wh}}$	[€/Wh]	0.0002						
$c_{rep_{hw}}$	[€/h]	1000						
c_{rep_s}	[€/h]	100						
$\overline{c}_{UA}$	[€/h]	5						

6.4.2 Evaluation and Results

Using the results from Section 6.2.4, the unavailability (UA) of both architectures is derived as $UA = 1-A$, resulting in $UA_{SSMA} = 3.185\,\%$ and $UA_{VNFA} = 1.385\,\%$. The expected costs can then be derived using the parameters from the example scenario in Equation (6.13). The evaluation's results of both SSMA and VNFA are depicted in Figure 6.32.



(a) Cost development of SSMA

(b) Cost development of VNFA

Figure 6.32.: Cost development of SSMA compared to VNFA, based on Equation (6.13)

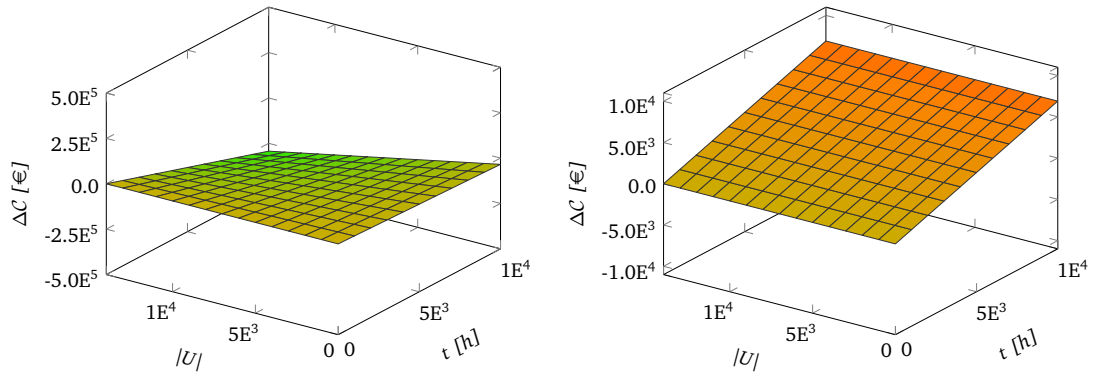
As clearly visible, the proposed VNFA approach significantly decreases the overall costs of the infrastructure. More precisely, the costs are lowered in both the time and per-user dimension. To give a quantitative measure of the expected monetary savings, the mean costs of both architectures

(i. e. the average of all scenarios with 0 h $\leq t \leq$ 10 000 h and $0 \leq |U| \leq$ 14 000) are calculated as given in Equation (6.14).

$$\overline{\mathcal{C}}\big(\max(t), \max(|U|)\big) = \frac{1}{\max(|U|) \cdot \max(t)} \int_0^{\max(|U|)} \int_0^{\max(t)} \mathcal{C}(t, |U|) dt d|U|, \tag{6.14}$$

where $\max(|U|)$ is the maximum number of users (in the given scenario 14 000), $\max(t)$ is the maximum time during the investigation (in the given scenario 10 000 h). The results show that VNFA costs 64.72 % less on average in the given scenario than SSMA. The main reason is on the one hand the reduction of unavailability times, leading to a more precise forecast of energy demand. On the other, the usage of VNFs offers the benefit of reduced hardware requirements per user.

To estimate the influence of replicas on the costs of VNFA, Equation (6.13) is employed again; the differences being the number of servers required (as additional replicas require more resources to be hosted) and the estimated downtime, which is reduced by having multiple replicas. In the first case, the cost difference between VNFA using no replicas and a VNFA variant using a single replica for the dc services are compared in Figure 6.33a; the cost difference of VNFA using a single and two replicas for the dc services are depicted in Figure 6.33b.



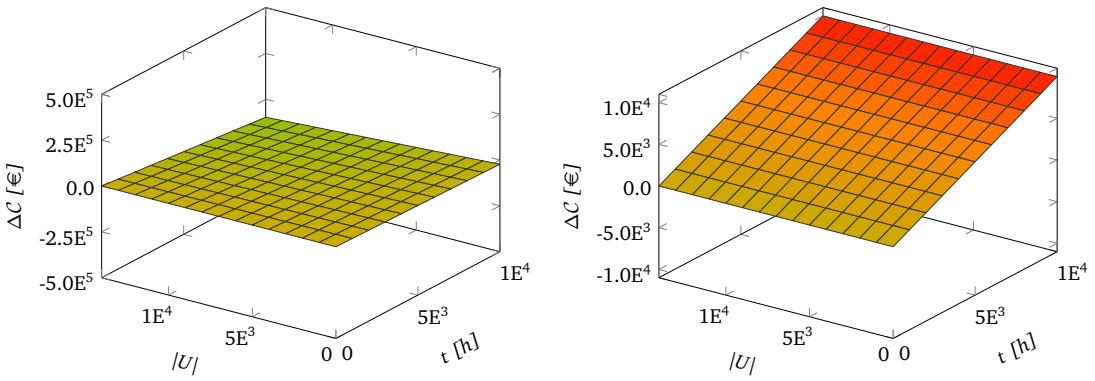
(a) VNFA $\Delta\mathcal{C}$ using no/one replica for dc

(b) VNFA $\Delta\mathcal{C}$ using one/two replica(s) for dc

Figure 6.33.: $\Delta\mathcal{C}$ in VNFA using replica(s) for dc

The results show that the usage of a single replica for thew dc service is beneficial regarding the VNFA costs. This is mainly due to the significant increase in availability and the only moderate increase in required hosting resources (as each dc service serves up to 100 users). This leads to a mean cost decrease of 2.453 %. On the contrary, there is no benefit in using two replicas, as the additional availability increase is only minute, while the additional resource requirements outweigh the benefits. This leads to a relative mean cost increase of 0.105 %.

Using a single replica for each dc and gw service, a lower benefit compared to only one dc service replica is achieved. The decreased availability gain is due to the lower failure rates encountered in gw services being of simpler nature than dc services, as described in Table 6.5. The benefit is a mean cost decrease of 0.920 % compared to using a single replica for each dc service. Similar to the effect of running two replicas for dc services, employing two replicas for each dc and gw service increases the costs. The relative mean cost increase comparing a single replica for dc services and two replicas for each dc and gw service is 0.152 %.



(a) VNFA $\Delta\mathcal{C}$ using one replica for dc/one replica for dc and gw each

(b) VNFA $\Delta\mathcal{C}$ using one/two replica(s) for dc and gw each

Figure 6.34.: $\Delta\mathcal{C}$ in VNFA using replica(s) for both dc and gw

6.5 Summary

In the analysis chapter, first, the assumptions for the evaluation and comparison of SSMA and VNFA are given and the three test scenarios (single service, single user, and Passau city) are described in detail. After that, the models developed in Section 5.3 are analyzed and results are drawn and compared. Thereafter, a validation of the approach is performed using an implementation of a vSMGW. Finally, a monetary cost analysis is performed to compare SSMA and VNFA.

Briefly speaking, the findings confirm that VNFA is superior to the currently suggested SSMA model in most aspects. The availability of services are increased and network delays are shortened by centralizing the hosting locations of services to COTS servers interlinked by a high-bandwidth network. Due to the extensive coverage of the dependability and performance analysis of SSMA and VNFA in Section 6.2.4, no extensive repetition is given here. Instead, it is referred to the aforementioned section.

From a monetary perspective, VNFA offers severe benefits in comparison to SSMA. Using VNFA without any replicas, a cost decease of 64.72 % is achieved. Using a single replica for the dc service, this benefit is further increased by 2.453 %. In contrast, the usage of two replicas for each dc service is monetary-wise not beneficial, as the additional resource requirements exceed the gained availability. This leads to a mean cost increase of 0.105 % in comparison to using a single replica for each dc service. A similar situation unfolds if replicas are used for both the dc and gw services. Here, the overall lowest cost is achieved, decreasing the mean costs by an additional 0.920 % in comparison to using a single replica for dc only. As before, employing two replicas for each dc and gw service increases the mean costs, in this case by 0.152 % compared to using a single replica for each dc and gw service.

Lastly, the evaluation of a prototype vSMGW implementation reveals that it is possible and feasible to run multiple instances of vSMGWs on a single substrate server. Depending on the performance metric assessed, the influence of co-hosting several instances of vSMGW varies. While e. g. the latency is nearly unimpaired by the number of co-hosted vSMGW instances, the throughput suffers severe degradation once $\approx$ 80 % of substrate resources are occupied. Therefore, generally speaking, as long as a certain hosting substrate resource threshold is not exceeded, the performance impairment incurred is minor. Concerning the dependability of vSMGWs, a Musa-Okumoto approach reveals that—even though only a proof-of-concept prototype was analyzed—the dependability requirements

of AMIs are fulfilled after a “burn-in” period. The availability of the prototype is expected to reach 99.968 % assuming an MTTR of 1.833 h.

7 Applicability, Conclusions and Future Work

The final chapter is split into two parts. The first deals with practical considerations regarding the realization of VNFA in Section 7.1, including the possible challenges that a real-life implementation of VNFA might face, which are covered in Section 7.1.1. This includes, among others, the re-usability of existing AMI resources, the additional hard- and software required by VNFA, and IT security considerations. In Section 7.1.2 legal and regulatory challenges relevant in the scope of virtualized AMIs are discussed.

After that, the second part of the chapter deals with the conclusion in Section 7.2, summarizing the current state of AMIs in Germany and the contributions of this thesis, and the future work, which is elaborated upon in Section 7.3. Main future research directions include the optimization of the *AMIgo* simulation framework regarding performance and adding specialized embedding algorithms for virtual AMIs, the development of a more elaborate and fine-granular communication sub-model to include in the AMI model, as well as the usage of advanced network virtualization features, such as SDN, to offer improved communication features, such as message prioritization.

Contents

7.1 Practical Applicability **172**
7.1.1 Usability Challenges 172
7.1.2 Regulatory Challenges 175
7.2 Conclusion and Outlook **177**
7.3 Future Work **179**

7.1 Practical Applicability

Apart from the long-standing criticism towards the operation of AMIs in general, which are usually based on concerns regarding their health and safety[24], possible privacy infringements[25], and the forced installation in new buildings[26], in this section, the change from an SSMA towards the novel VNFA is investigated and analyzed concerning its practical feasibility.

First, the general usability challenges regarding the realization of the VNFA are discussed in Section 7.1.1. This includes required changes in the AMI hardware architecture, resource considerations which deal with the re-usability of currently suggested AMI hardware in the novel architecture and additional resources required for the realization of VNFA, as well as the possible IT security implications. After that, in Section 7.1.2, the legal and regulatory challenges associated with a change to VNFA are elaborated. With a focus on the current regulatory requirements in the EU as well as Germany in particular, possible legal hurdles are identified and discussed.

7.1.1 Usability Challenges

The introduction of VNFA to replace the currently planned SSMA may introduce several usability challenges, which need to be elaborated to give an educated estimation of the incurred hardships while doing so. To give an overview regarding the usability challenges which might arrive when deploying VNFA, several contributing factors are discussed next.

7.1.1.1 AMI Hardware Architecture

Comparing Figures 3.1 and 4.1 in Sections 3.1.1 and 4.1, respectively, several differences in the hardware architecture of SSMA and VNFA are apparent. In the next two paragraphs, these differences are further elaborated upon, including the re-usability of resources of SSMA in VNFA, as well as an investigation of additional resources required during the implementation of VNFA, including substrate/hardware entities as well as human investments.

Re-Usability of Existing Resources. Depending on the degree of implementing VNFA, different subsets of entities are reusable. First off, all entities which cannot be virtualized due to either *type* or *location* restrictions (or both) are re-usable in VNFA, as their substrate is required to be present in an AMI, no matter if SSMA or VNFA (see Section 4.2.1). This includes, among others, smart meters, IHDs, CLSs, and cyber-physical provider-side systems. If VNFA is used in its entirety, an additional challenge during the re-usage of smart meters is implied in Figure 7.1 that depicts the challenges encountered during encryption due the the relocation of the vSMGW in comparison to the substrate-based SMGW. Here, it might be required to upgrade the currently used SMGW firm- and/or hardware to allow encryption operations. This challenge could be circumvented if vSMGWs are hosted on VMs running on home routers instead of COTSs located in the WAN. This idea is further discussed in Section 4.2.2 as well as by Steidele [153].

---

[24] Stop Smart Meters! (UK), URL: `http://stopsmartmeters.org.uk/`, last accessed: 01/03/2019

[25] How Smart Meters Invade Individual Privacy, URL: `https://smartgridawareness.org/privacy-and-data-security/how-smart-meters-invade-individual-privacy/`, last accessed: 01/03/2019

[26] Smart Meters: Forced Installation, Energy Usage and Higher Bills, URL: `http://emfsafetynetwork.org/smart-meters-forced-installation-energy-usage-and-your-higher-bills/`, last accessed: 01/03/2019

Additional Resources Required by VNFA. To host the virtualized counterparts of the SSMA substrate entities, COTS have to be acquired. The number of required server is thereby linked to three main factors: first, the number of users present in the VNFA AMI determines the amount of virtualized services (and their VMs), which—in turn—decide the amount of resources ($\Omega_{cpu} d^s_{cpu}$) necessary on the substrate entities. Apart from that, the second factor is the resource threshold (see Section 4.4.1) set for the servers. Last, the embedding process may lead to non-allocated resources being left unused on substrate servers due to either the amount of resources left on the server cannot fit another virtualized service ($\forall\, i \in V^s : \Omega_{cpu} d^s_{cpu_i} < r^{hw}_{cpu}$) (due to the relatively small demands, the unused resource amount here is expected to be low) or because of heuristics being employed during the embedding. To host the servers, depending on their location distribution (see Section 4.2.2), either a single data center (using "centralized hosting") or several tamper-proof server casings (using "distributed hosting") need to be acquired.

Using a centralized server location, another decision would be if TPSs are integrated into the same data center as service provider AMI functions. This would allow for a more efficient hardware usage through higher server utilization as well as better performability as potentially more backup servers are available. In addition, a lower communication overhead is to be expected as distances between servers are lowered (in-house communication only) or non-existent (VMs hosted on same server). In contrast, using a separate substrate would allow a better separation of responsibilities as well as an increased isolation of substrates, therefore it would be less likely that harmful events caused by either side spread throughout the AMI.

Besides the properties of distributed hosting pointed out in Section 4.2.2, such an approach induces an increased demand regarding physical protection. This can either be satisfied by re-using the existing, tamper-proof casings employed to house in-field hardware, such as local transformers etc., or by the acquisition of new, tamper- and weather-proof casings to harbor the required COTS servers. The later option would on the one hand increase the costs, on the other, however, the placement of servers is more flexible if legacy infrastructure may be disregarded.

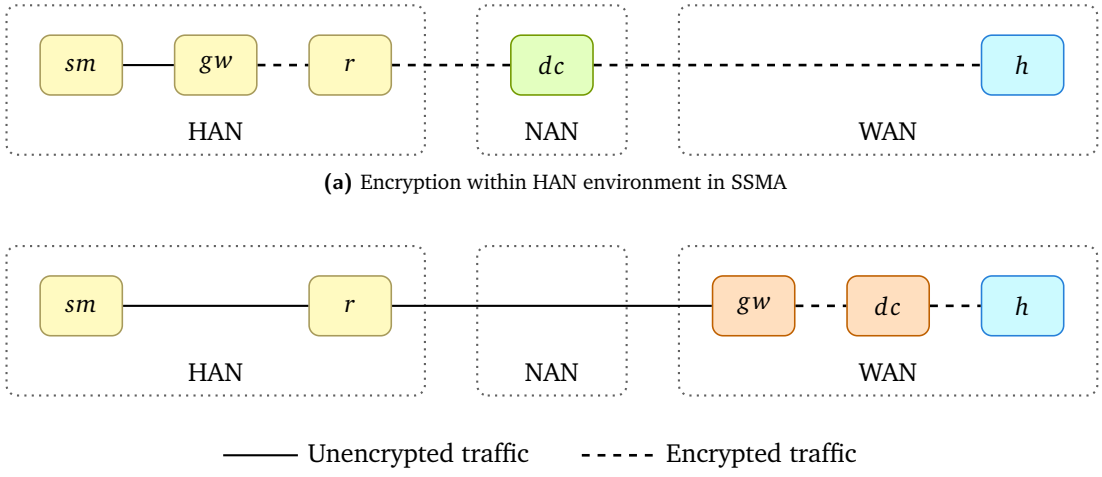
Apart from the physical resources required by VNFA, it shall be mentioned here that the author is aware of the possibly specialized human resources required to maintain the VNFA infrastructure. However, as the maintenance is also required in current legacy power grids as well as in SSMA architectures, giving a well-educated prediction of the difference in amount of human resources required and their level of competence needed to operate VNFA is challenging and is not done within the scope of this thesis.

7.1.1.2 IT Security Considerations

Although IT security is not regarded as a main topic within this thesis, the author still considers the need to address potential security challenges accompanying the introduction of VNFA. Due to the strict regulations of AMI and smart grid related ICT systems in the European Union (EU), and Germany in particular, special care is required when introducing modifications to such systems. Several possible challenges are discussed here.

The introduction of virtualization and the associated relocation of services onto servers is not solely advantageous, but comes with a caveat: The virtualization of the gateway, as discussed in Section 4.2.2, moves it from the HAN to the WAN. This is particularly problematic when security, more specifically the security goal *confidentiality*, is considered, because the data sent by smart meters may not be encrypted. When the gateway is not longer within the trusted HAN, unencrypted data might

be sent through public networks. This problem is illustrated in Figure 7.1. However, two possible solutions to remediate this challenge do exist.



(a) Encryption within HAN environment in SSMA

(b) Missing encryption in VNFA

Figure 7.1.: Comparison of encryption behavior in SSMA and VNFA

1. **Add encryption to the smart meter:** Since smart meters are handed out by the utility company, symmetric encryption routines and the respective keys could be added. In principle, symmetric encryption is feasible on resource constrained devices such as smart meters. This would eliminate the need for a gateway, from a strictly smart meter-centered, confidentiality-wise point-of-view.

2. **Relocation of virtualized gateways:** It is possible to host the virtualized gateway on commodity routers, which are present in user households. By doing so, the metering data do not leave the HAN before the encryption is performed. A prototype of this idea is implemented and tested by Steidele [153]. This approach, however, places the virtualized gateway inside the customer's premises and therefore dampens some advantages discussed in Section 1.3.

Throughout this thesis, the first option is assumed. As smart metering devices work at increasing computational performance levels (and can safely be expected to continue to do so in the future), a minor overhead due to symmetric encryption is anticipated to be acceptable.

Also, virtualization opens up several possible attack vectors by itself, which is e. g. discussed in detail by Sahoo *et al.* [140]. Due to detailed explanations given by Sahoo *et al.*, several security issues are only going to be listed here:

- Communication among VMs or between VMs and host,
- VM escape,
- VM monitoring from host/another VM,
- Denial of Service (DoS),
- guest-to-guest attacks,
- external modifications of VMs,
- external modification of the hypervisor.

Besides the possible negative impacts or challenges VNFA causes regarding IT security, several benefits are also possible: If properly configured and maintained, the isolation of virtual machines may help to separate compromised VMs. Moreover, using VNFA, a tighter security supervision is possible, as more parts of the infrastructure are hosted by a professional service provider. This reduces both the time for experts to detect a possible malfunction or failure as well as the repair times. Both in case of substrate failures, the required items for repairs are much more likely to be readily available at a service provider's location than at a user's household. Also, in case of software failures (which are increasing in likelihood using a virtualized AMI environment), remotely triggered recovery mechanisms would offer much shorter repair times than conventional restoration, as detailed in Section 3.2.2.

7.1.2 Regulatory Challenges

Apart from the technical challenges of introducing VNFA, several regulatory restrictions need to be discussed in this context, too. As briefly introduced in Section 1.2, depending on the location where AMIs are established, the regulatory bodies may enforce different requirements and constraints on smart metering infrastructures. Such regulations may affect data protection and privacy, IT security mechanisms to be employed in services as well as performability requirements of the AMI. To analyze the influence of regulatory requirements on the realization of VNFA, two sets of rules need to be distinguished:

1. **Abstract, goal-centered regulations:** Such rules are defined to make sure certain abstract goals are met, while the way these goals are achieved is not explicitly stated. Examples for such regulations are e. g. to provide certain levels of security, a certain maximum communication delay as well as a specified availability of the overall infrastructure. Abstract, goal-centered regulations therefore *specify a goal* (typically a minimum or maximum requirement), however *the implementation details leading to the realization of the stated goal are not specified*.

2. **Specific, realization-centered regulations:** In contrast to the abstract rules presented before, specific, realization-centric regulations provide exact rules regarding implementation details, especially regarding hardware components to be used in a system. Specific, realization-centered regulations *do or do not specify the goal* of a certain action, however, they *require to strictly abide to the implementation details specified within the regulation*.

The first regulations are of no further concern in the following, because virtualized services provide an increased performability compared to non-virtualized ones, as shown in Section 6.2.4; in addition, other requirements, such as security or privacy, are rather improved by virtualization, as e. g. Cleeff *et al.* states [167]. The second type of regulations, namely specific, realization-centered ones, may lead to challenges regarding the possible usage of VNFA. However, in the following paragraphs, it is shown that at least current rules in Germany may have viable alternatives which are feasible. To do so, first, several examples of currently active specific, realization-centered regulations in Germany are collected (an exhaustive collection and discussion of such regulations is beyond the scope of this thesis). Second, possible solutions satisfying the previously collected regulations in similar quality in virtualized environments are discussed.

7.1.2.1 Specific, Realization-Centered AMI Regulations in Germany

In Germany, the functional and non-functional rule sets for AMIs are set by several cooperating regulatory bodies and persons: Bundesministerium für Wirtschaft und Energie (BMWi), BSI, the

federal commissioner for data protection and freedom of information (Bundesbeauftragter für den Datenschutz und die Informationsfreiheit), the federal network agency, as well as the physico-technical federal institute (Physikalisch-Technische Bundesanstalt). In the upcoming paragraph, several AMI regulations currently active in Germany are briefly described, capturing the main ideas and challenges while realizing these rules within VNFA.

- **BSI TR-03109-2:** The BSI TR-03109-2 specifies the usage of a hardware security module that provides cryptographic core routines for signature creation and validation, key generation, key negotiation, and random number generation for the SMGW and serves as a secure storage for cryptographic key material. Both the functionality and its integration with the SMGW is standardized in the security profile BSI TR-03109-2. Due to the virtualization used in VNFA, a separate hardware security module might not be present for each vSMGW. As the virtual services get embedded and possibly migrated between different servers, it would add additional complexity, restrictions and monetary overhead to ensure that each server features at least a certain number of security modules for all hosted vSMGWs.
- **BSI CC-PP-0073/BSI TR-03109-1:** Both documents demand the usage of physically separate network ports for the connection of the SMGW with each of its interconnected network areas (LMN, HAN, and WAN). In VNFA, as a nearly arbitrary number (n) of vSMGW may be spawned on a server, it is likely that there are not $3n$ network ports available. While the standards do not explicitly demand that each physical port is a wired network port, even the additional usage of wireless interfaces is not likely to be able to satisfy the aforementioned requirement of $3n$ network ports altogether.
- **Law on metering point operation and data communication in intelligent energy networks (Messstellenbetriebsgesetz – MsbG):** Due to the possible manipulation of data sent via home routers, their usage is restricted. While home router were proclaimed as a feasible solution to forward smart meter data until 2014 even by the BSI, with the introduction of the revised Messstellenbetriebsgesetz, their usage within AMIs was prohibited. The main downside of home routers that led to their ban from AMI networks is the inability to ensure data integrity. With the usage of VNFA, this leads to issues as home routers are used to forward traffic from the HAN part of the network to the NAN and WAN parts of the AMI.

7.1.2.2 Solution Approaches

With the regulatory challenges discussed in Section 7.1.2, several solution approaches are investigated next.

- **BSI TR-03109-2:** Especially within virtualized systems, the challenge of supporting hardware-based security is a well-known issue. While the BSI TR-03109-2 in its current version strictly requires such a hardware-based solution for each SMGW, a possible solution would be the usage of virtualized hardware-based security modules, similar to the Virtual Trusted Platform Module (vTPM) approach used to virtualize TPMs. Such a solution could be feasible, since the tasks of a TPM and the hardware security module required by an SMGW are almost completely similar. To be able to virtualize a TPM, which uses a trust approach rooted in hardware, several important key features must be fulfilled:
  - Same usage model and command set
  - Strong association between VM and its vTPM instance

  - Strong association between real TPM and vTPM
  - Easy to distinguish between real and virtual TPM

  Currently, the vTPM approach is already supported by several virtualization solutions, such as VMware, HyperV, and XEN, and has even been extended to allow the migration of VMs employing a vTPM.

- **BSI CC-PP-0073/BSI TR-03109-1:** In a similar manner as the challenges encountered with the BSI TR-03109-2's demand for a physically present security module, virtualization may offer a solution to enforce the isolation of multiple networks on a single server. This has already been shown to be an effective and feasible solution by Steidele [153], where the concept was employed to separate the network connections of a vSMGW to the LMN, HAN, and NAN/WAN with a single hardware network port using internal network virtualization (offering several vNICs).

- **Law on metering point operation and data communication in intelligent energy networks (Messstellenbetriebsgesetz – MsbG):** To exclude common household routers from the communication in AMIs is theoretically possible in SSMA and VNFA—however, it would require the usage of dedicated communication hardware to do so, that would need to be embedded inside a device located in the LMN or HAN in a tamper-proof way, most probably the SMGW. Instead, several alternatives exist that circumvent the problems of data manipulation in home routers. One possibility, which is discussed in Section 4.2.2, is the introduction of encryption on smart meters, which can hinder targeted manipulation attacks. Secondly, the usage of isolated, virtualized network connections realized e. g. by VPNs could be a viable solution.

7.2 Conclusion and Outlook

Summarizing the results of this thesis, several main conclusions can be drawn. First, it is unquestionable that the usage of virtualization offers numerous benefits, especially in the areas of performability, adaptability, and cost efficiency of systems, which is once more briefly addressed in the following:

- **Performability:** Using the VNFA approach with its choices between deterministic and dynamic timings and three rejuvenation strategies (cold, warm, and migration), as well as optional replica usage an improved performability compared to SSMA is achievable on service level (0.313 %). In addition, the availability of an AMI also increases using VNFA compared to SSMA. On service level, the maximum increase is 0.669 %, on infrastructure level VNFA can achieve an availability of 98.657 %, SSMA, in comparison, 96.815 %.

- **Adaptability:** VNFA offers adaptability in several dimensions: For once, as described, modeled and finally evaluated in Chapters 4–6, VNFA enables the deployment of VMs on COTS servers. Using this approach, the hardware required for adding additional users to an AMI can be reduced from a smart meter, SMGW as well as potentially a new data concentrator to a single smart meter. All other components are realized as virtualized services, which improves the approach's scalability, as the performance analysis in Section 6.2.3.5 reveals. In addition, employing virtualized services enables a rapid response to changing operational requirements, as software updates to the VMs (which are under the central control of the service provider) are deployable with much lower effort than providing updates to multiple distributed devices,

each being vendor-specific, featuring its own update mechanism and—if an international usage scenario is envisioned—its own set of regulatory requirements.

- **Cost efficiency:** From a monetary perspective, VNFA offers severe benefits in comparison to SSMA. The maximum monetary benefit is achieved using a single replica for each dc and gw service, offering a cost decrease of $\approx 65\,\%$ in comparison to SSMA using the scenario given in Section 6.4.

However, despite these numerous advantages, to make an educated judgment on the outlook if the adoption of VNFA in Germany would be possible, it is important to discuss the current state of smart grid development. While there are concrete plans on the installation of smart meters and their usage, they are currently more theoretical than practically realized. Looking at the practical state of AMI and smart grid usage in Germany, it is apparent that the AMI roll-out is on a forced halt, which has several reasons[27]:

First off, the currently still **lacking level of certification** is hindering the adaption of AMI solutions in Germany, because of the requirement set by the Smart Meter Act that the roll-out can only start once three SMGWs by different manufacturers have been certified in accordance with the technical regulations. The first certification has been announced only recently. Second, the smart meter roll-out in Germany **lacks a sophisticated, independent cost-benefit analysis**. Commonly, such analyses are performed by multiple cooperating partners from various domains, such as legal, economic and ICT experts. Despite this fact, the German cost-benefit analysis has been carried out in its entirety by a single consultancy (Ernst & Young GmbH [48]), whose manager has a working history at the German heavyweight electric utilities company RWE[28]. In close relation to the second point, third, the **current smart metering fees are based on unrealistically high energy saving estimates**, which in turn leads to consumers required to pay their saving potential on top of the metering fee for old meters. These additional costs especially put the profitability of small-scale projects at risk. In addition, it is even questionable whether small-scale Photovoltaic (PV) is of any relevance for a grid operators' service portfolio, as there is most likely no need to shut down small-scale PV to guarantee grid security, while the obligation to include small-scale PV is heavily straining their staff and technological capabilities.

This leads to several simultaneous developments.

Service providers in Germany are hesitant to install smart meters and the AMI backbone infrastructure due to these missing certifications and the possibility of imminent changes to the certification process, which would in turn lead to high expenses to upgrade/change currently made investments. One example of such a change is the currently ongoing discussion if the previously required usage of the Companian Specification for Energy Metering (COSEM) protocol is even going to be supported in AMI environments in the future; from the side of industrial stakeholders, other standards, such as American National Standards Institute (ANSI), received tremendously more support, which is the reason for a possible change of the standard. The certification process itself as introduced by the BSI is criticized[29], and drastic changes are asked for. From the industry's side, it is demanded that in

---

[27] Smart Meter - The German 'Sonderweg' at a crossroads – Energy Democracy, URL: `https://energy-democracy.org/smart-meter-the-german-sonderweg-at-a-crossroads/`, last accessed: 05/03/2019

[28] Biography - Dr. Helmut Edelmann, URL: `https://www.ey.com/de/de/industries/power--utilities/biography--dr-helmut-edelmann`, last accessed: 05/03/2019

[29] According to the German federal association of the new energy industry (Bundesverband Neue Energiewirtschaft) the legally required certification process is "structurally problematic".

the future, the BSI should only determine which security requirements of smart metering systems have to be met. The manufacturers would then carry out a self-certification in combination with calibration regulations[30]. The installation of smart meters lead to a considerable investment and operating expenses, which are (at least partially) invoiced to end customers. Therefore, the decision whether the currently planned AMI realization will be widely accepted has still not been made; deciding factors will be the perceived smart meter benefits contesting against the incurred costs [57].

The conclusion to be drawn from these developments is that the AMI standardization and regulation is still in flux; also, customers are still not fully convinced of the currently proposed AMI strategy. Therefore, requirements regarding the concrete implementation and realization of certain features may be adjusted in the future. The author already introduced the idea of virtualized AMIs in discussions with the German regulatory bodies Verband der Elektrotechnik, Elektronik und Informationstechnik (VDE) and Deutsche Kommission Elektrotechnik Elektronik Informationstechnik (DKE) (e. g. by Benze *et al.* [14]), which realized the potential benefits of virtualization approaches. Despite that, a wide-scale adoption of VNFA is unlikely in the near future, however, certain features or concepts might be used as an inspiration for upcoming improvements to current ideas and regulations.

7.3 Future Work

Regarding concrete future work, there are several further optimizations possible with the proposed virtualized AMI approach given in this thesis, which are listed in the following.

1. ***AMIgo* optimizations:** During the validation of the VNFA virtual AMI embedding, the *AMIgo* software tool is used to evaluate the VNFA embedding onto a given substrate and estimate the performance and required resources for such an embedding. While the results obtained are promising, both the performance and the embedding strategy of the software tool still require further optimizations.

   The performance of *AMIgo* is currently not able to work feasibly in large embedding scenarios. An approach on how to improve the performance further exists already; a slight change in the structure of the queuing network could possibly enhance the tool's performance. First and foremost, the explicit modeling of routers leads to a severe performance bottleneck in the virtualized network embedding algorithm. If the model could implicitly model routers by including their properties in other entities and omit them, connecting the servers directly to the smart meters, the overall number of nodes in the queuing network could be drastically reduced and should therefore increase computation speed.

2. **More elaborate communication sub-model:** The communication inside the AMI model is currently modeled in a simplified way, featuring several assumptions that might not be able to represent the delays encountered in all scenarios accurately. Depending on the used protocols (e. g. OSGP, ANSI C-Series, etc.[31]) and current transmission stage (before or after data compression by a data concentrator), the induced delay may vary. In addition, the failure

[30] Handelsblatt: Regierung bremst intelligente Stromzähler aus, URL: `https://www.handelsblatt.com/politik/deutschland/smart-meter-regierung-bremst-intelligente-stromzaehler-aus/23688518.html`, last accessed: 10/01/2019

[31] For a full list of network protocols used in AMIs see Section 2.1.2.2

rates of communication channels were excluded during the analysis, which is—in general—a valid assumption, as in 1st world countries communication networks offer a high availability [99]; however, if this assumption is removed, more dependability and performance measures for links might be required to be included, e. g. channel bit errors, possibly required packet re-transmissions, and propagation delay, among others. This would require a more detailed communication channel model to be developed and implemented.

3. **Advanced usage of virtualization:** Network virtualization is introduced in VNFA briefly in Section 4.3 as a measure to link a user's VNFCs in an optimal way to create a VNF-FG. Several improvements might be considered when network virtualization techniques are employed: On the one hand, the resolution at which VNFCs are defined might be increased. Currently, a service level resolution is used to embed and organize VNFCs to a VNF-FG. In future extensions to VNFA, the components of each service might be modeled (adding a sub-service layer to the existing micro- and macro- model layers), making a single service comprising, e. g. data logic, compression, and encryption components. While the composition of the currently used "coarse" VNFs is already not a trivial task in all scenarios (as described in Section 4.3.2), a more fine-granular resolution would result in a higher complexity during the VNF-FG composition, but it might be possible to optimize the composition of VNFCs even further, leading to improved embedding results. Apart from that, virtual networking could be further used to adapt to channel properties and requirements of certain message types in the AMI environment by employing SDN. Research in this direction has already been started by Rehmani *et al.* [134], where several key benefits of SDN in AMIs are stated as isolation of different traffic types, traffic prioritization, increased dependability/fast failure recovery and enhanced interoperability. Especially the combination of traffic isolation and prioritization could be employed to reserve certain amounts of the channel to be able to better harness the performance, e. g. using more priority for low-delay tolerant messages, such as (i. e. critical measurement data and control commands), while billing or other optional messages might get sent via another virtual link offering less priority/speed/dependability.

4. **Dynamic services and prioritization:** Currently, the modeling of services is performed in a static manner, i. e. the properties of the service, such as its priority flag, its resource demand, etc. remain constant throughout its lifetime. While this assumption is reasonable for the gross of all services in currently planned AMIs, it could be worthwhile to consider a greater number of services in the future. This would allow to not only harness the benefits of virtualization in an AMI context, but rather throughout a smart grid. Noteworthy in this respect are especially monitoring and reporting services, which could receive varying priorities depending on the current state of a smart grid. An example could be the prioritization of measurement and sensing services as soon as unusual sensor data is detected for the first time to ensure timely reporting of potentially escalating deviations.

   This kind of prioritization would be an addition to the current used concept, where different VNFCs are ranked over others within a VNF. On the larger scale, whole VNFs may be prioritized over others, especially in the context of the previously described non-user services, such as monitoring or reporting. These services often have importance over individual users' AMI services due to their contribution to overall smart grid stability.

Appendix

A

A.1 *AMIgo* Configuration File

In the following an example *AMIgo* configuration file (using slight naming adjustments to improve readability) is given.

```
{
   "PROJECT_NAME":  "defaultproject",
   "SAVE_NETWORK_STACK":  false,
   "SAVE_FOR_JMT_CLASSICAL":  true,
   "SAVE_FOR_JMT_V":  true,
   "USE_MULTI_THREADING":  false,
   "PROCESS_NR":  true,
   "PROCESS_CB":  true,
   "PROCESS_AFA":  true,
   "PROCESS_NR_V":  true,
   "PRINT_DAMAGE_REPORT":  false,
   "PRINT_WRITE_TO_DISK":  true,
   "PRINT_JSON":  true,
   "PRINT_NEIGHBORHOOD_TEMPLATE":  true,
   "PRINT_SUMMARY":  true,
   "MAPPING_ALGORITHM":  "SubgraphIsomorphism",
   "SERVER_FAILURE_INCREMENTAL":  false,
   "SERVER_FAILURE_BUCKET_COUNT":  1,
   "ARRIVAL_UNIFORM_DC_MIN":  0.0,
   "ARRIVAL_UNIFORM_DC_MAX":  900.0,
   "ARRIVAL_UNIFORM_SM_MIN":  0.0,
   "ARRIVAL_UNIFORM_SM_MAX":  60.0,
   "PENALTY_MULTIPLIER_FOR_VIRTUALIZATION":  1.0,
   "PERFORMABILITY_V_GW":  1.0,
   "PERFORMABILITY_V_R":  1.0,
   "PERFORMABILITY_V_DC":  1.0,
   "PERFORMABILITY_V_A":  1.0,
   "PERFORMABILITY_V_EO":  1.0,
   "PERFORMABILITY_NV_GW":  1.0,
   "PERFORMABILITY_NV_R":  1.0,
   "PERFORMABILITY_NV_DC":  1.0,
   "PERFORMABILITY_NV_A":  1.0,
   "PERFORMABILITY_NV_EO":  1.0,
   "SERVICE_RATE_HES":  1000.0,
```

```
"SERVICE_RATE_DC":  60.0,
"SERVICE_RATE_SM":  20.0,
"SERVICE_RATE_GW":  40.0,
"SERVICE_RATE_R":  1000.0,
"SERVER_COUNT":  3,
"SERVER_MAX_LOAD":  0.8,
"NEIGHBORHOODS_COUNT":  3,
"HOUSEHOLDS_PER_NEIGHBORHOOD_MAX":  100,
"HOUSEHOLDS_PER_NEIGHBORHOOD_MIN":  100,
"RANDOM_POLICY":  "Random",
"RANDOM_POLICY_ADD":  1,
"CPU_HES":  0.0,
"CPU_DC":  200.5,
"CPU_GW":  3.0,
"CPU_R":  2.0,
"CPU_SM":  2.0,
"CPU_SERVER":  1000.0,
"LINK_MEDIUM_HAN":  "ZigBee",
"LINK_MEDIUM_WAN":  "OpticalFiber",
"LINK_MEDIUM_NAN":  "PLC",
"LINK_MEDIUM_PERFORMANCE":  "MIN",
"LINK_MEDIUM_SERVER":  "Gigabit",
"DEMAND_CPU_HES":  0.0,
"DEMAND_CPU_GW":  2.5,
"DEMAND_CPU_DC":  200.0,
"DEMAND_CPU_R":  1.5,
"DEMAND_CPU_SM":  1.5,
"DEMAND_BANDWIDTH_DC":  6000.0,
"DEMAND_BANDWIDTH_SM":  9000.0,
"TPS_SUBSTRATE_MAX_LOAD":  0.8,
"TPS_A_PERCENT":  0.7,
"TPS_EO_PERCENT":  0.4,
"TPS_A_HOUSEHOLDS_PER_SERVER":  1000,
"TPS_EO_HOUSEHOLDS_PER_SERVER":  1000,
"TPS_EO_CPU_OFFERED":  3750.0,
"TPS_A_CPU_OFFERED":  3750.0,
"TPS_EO_SERVICE_RATE":  30.0,
"TPS_A_SERVICE_RATE":  30.0,
"PATH_STACK_HEALTHY_PRE_MAPPING_NR":  "healthy_stack_pre_mapping_nr.txt",
"PATH_STACK_HEALTHY_POST_MAPPING_NR":  "healthy_stack_post_mapping_nr.txt",
"PATH_STACK_HEALTHY_PRE_MAPPING_CB":  "healthy_stack_pre_mapping_cb.txt",
"PATH_STACK_HEALTHY_POST_MAPPING_CB":  "healthy_stack_post_mapping_cb.txt",
"PATH_STACK_HEALTHY_PRE_MAPPING_AFA":  "healthy_stack_pre_mapping_afa.txt",
"PATH_STACK_HEALTHY_POST_MAPPING_AFA":  "healthy_stack_post_mapping_afa.txt",
"PATH_STACK_HEALTHY_PRE_MAPPING_NR_V":  "healthy_stack_pre_mapping_nr_v.txt",
"PATH_STACK_HEALTHY_POST_MAPPING_NR_V":  "healthy_stack_post_mapping_nr_v.txt",
"PATH_TGF_JMT_HEALTHY_NR":  "jmt_healthy_nr.tgf",
```

```
"PATH_JMT_HEALTHY_NR":  "jmt_healthy_nr.jsimg",
"PATH_TGF_JMT_HEALTHY_TO_HES_NR":  "jmt_healthy_to_headend_nr.tgf",
"PATH_JMT_HEALTHY_TO_HES_NR":  "jmt_healthy_to_headend_nr.jsimg",
"PATH_TGF_JMT_HEALTHY_CB":  "jmt_healthy_cb.tgf",
"PATH_JMT_HEALTHY_CB":  "jmt_healthy_cb.jsimg",
"PATH_TGF_JMT_HEALTHY_TO_HES_CB":  "jmt_healthy_to_headend_cb.tgf",
"PATH_JMT_HEALTHY_TO_HES_CB":  "jmt_healthy_to_headend_cb.jsimg",
"PATH_TGF_JMT_FIXED_CB":  "jmt_fixed_cb.tgf",
"PATH_JMT_FIXED_CB":  "jmt_fixed_cb.jsimg",
"PATH_TGF_JMT_FIXED_TO_HES_CB":  "jmt_fixed_to_headend_cb.tgf",
"PATH_JMT_FIXED_TO_HES_CB":  "jmt_fixed_to_headend_cb.jsimg",
"PATH_TGF_JMT_HEALTHY_AFA":  "jmt_healthy_afa.tgf",
"PATH_JMT_HEALTHY_AFA":  "jmt_healthy_afa.jsimg",
"PATH_TGF_JMT_HEALTHY_TO_HES_AFA":  "jmt_healthy_to_headend_afa.tgf",
"PATH_JMT_HEALTHY_TO_HES_AFA":  "jmt_healthy_to_headend_afa.jsimg",
"PATH_TGF_JMT_FIXED_AFA":  "jmt_fixed_afa.tgf",
"PATH_JMT_FIXED_AFA":  "jmt_fixed_afa.jsimg",
"PATH_TGF_JMT_FIXED_TO_HES_AFA":  "jmt_fixed_to_headend_afa.tgf",
"PATH_JMT_FIXED_TO_HES_AFA":  "jmt_fixed_to_headend_afa.jsimg",
"PATH_TGF_JMT_HEALTHY_NR_V":  "jmt_healthy_nr_v.tgf",
"PATH_JMT_HEALTHY_NR_V":  "jmt_healthy_nr_v.jsimg",
"PATH_TGF_JMT_HEALTHY_TO_HES_NR_V":  "jmt_healthy_to_headend_nr_v.tgf",
"PATH_JMT_HEALTHY_TO_HES_NR_V":  "jmt_healthy_to_headend_nr_v.jsimg",
"PATH_TGF_JMT_FIXED_NR_V":  "jmt_fixed_nr_v.tgf",
"PATH_JMT_FIXED_NR_V":  "jmt_fixed_nr_v.jsimg",
"PATH_TGF_JMT_FIXED_TO_HES_NR_V":  "jmt_fixed_to_headend_nr_v.tgf",
"PATH_JMT_FIXED_TO_HES_NR_V":  "jmt_fixed_to_headend_nr_v.jsimg",
"PATH_TGF_SUBSTRATE_HEALTHY_NR":  "healthy_substrate_nr.tgf",
"PATH_TGF_VIRTUAL_HEALTHY_NR":  "healthy_virtual_nr.tgf",
"PATH_TGF_MAPPING_NR":  "mapping_nr.tgf",
"PATH_TGF_SUBSTRATE_HEALTHY_CB":  "healthy_substrate_cb.tgf",
"PATH_TGF_SUBSTRATE_FIXED_CB":  "fixed_substrate_cb.tgf",
"PATH_TGF_VIRTUAL_HEALTHY_CB":  "healthy_virtual_cb.tgf",
"PATH_TGF_VIRTUAL_FIXED_CB":  "fixed_virtual_cb.tgf",
"PATH_TGF_MAPPING_CB":  "mapping_cb.tgf",
"PATH_TGF_MAPPING_SIMPLIFIED_CB":  "mapping_simplified_cb.tgf",
"PATH_TGF_MAPPING_FIXED_CB":  "mapping_fixed_cb.tgf",
"PATH_TGF_SUBSTRATE_HEALTHY_AFA":  "healthy_substrate_afa.tgf",
"PATH_TGF_SUBSTRATE_FIXED_AFA":  "fixed_substrate_afa.tgf",
"PATH_TGF_VIRTUAL_HEALTHY_AFA":  "healthy_virtual_afa.tgf",
"PATH_TGF_VIRTUAL_FIXED_AFA":  "fixed_virtual_afa.tgf",
"PATH_TGF_MAPPING_AFA":  "mapping_afa.tgf",
"PATH_TGF_MAPPING_SIMPLIFIED_AFA":  "mapping_simplified_afa.tgf",
"PATH_TGF_MAPPING_FIXED_AFA":  "mapping_fixed_afa.tgf",
"PATH_TGF_SUBSTRATE_HEALTHY_NR_V":  "healthy_substrate_nr_v.tgf",
"PATH_TGF_SUBSTRATE_FIXED_NR_V":  "fixed_substrate_nr_v.tgf",
"PATH_TGF_VIRTUAL_HEALTHY_NR_V":  "healthy_virtual_nr_v.tgf",
```

```
  "PATH_TGF_VIRTUAL_FIXED_NR_V":  "fixed_virtual_nr_v.tgf",
  "PATH_TGF_MAPPING_NR_V":  "mapping_nr_v.tgf",
  "PATH_TGF_MAPPING_SIMPLIFIED_NR_V":  "mapping_simplified_nr_v.tgf",
  "PATH_TGF_MAPPING_FIXED_NR_V":  "mapping_fixed_nr_v.tgf",
  "PATH_NETWORK_TEMPLATE_STATS":  "network_template_stats.txt",
  "PATH_NETWORK_LOAD_NR":  "network_load_nr.txt",
  "PATH_NETWORK_LOAD_CB":  "network_load_cb.txt",
  "PATH_NETWORK_LOAD_AFA":  "network_load_afa.txt",
  "PATH_NETWORK_LOAD_NR_V":  "network_load_nr_v.txt",
  "PATH_NETWORK_LOAD_FIXED_CB":  "network_load_fixed_cb.txt",
  "PATH_NETWORK_LOAD_FIXED_AFA":  "network_load_fixed_afa.txt",
  "PATH_NETWORK_LOAD_FIXED_NR_V":  "network_load_fixed_nr_v.txt",
  "PATH_CRASH_PLAN":  "crashplan.txt",
  "PATH_DAMAGE_REPORT_CB":  "damagereport_cb.txt",
  "PATH_DAMAGE_REPORT_AFA":  "damagereport_afa.txt",
  "PATH_DAMAGE_REPORT_NR_V":  "damagereport_nr_v.txt",
  "PATH_DAMAGE_REPORT_CSV_CB":  "damagereport_cb.tsv",
  "PATH_DAMAGE_REPORT_CSV_AFA":  "damagereport_afa.tsv",
  "PATH_DAMAGE_REPORT_CSV_NR_V":  "damagereport_nr_v.tsv",
  "PATH_BENCHMARK":  "benchmark.txt",
  "JMT_LOG_DIR":  "jmt_logs",
  "JMT_MAX_STEPS":  10000000
}
```

A.2 Virtualized Smart Meter Gateway Evaluation Data

In the following, the raw failure evaluation data of a proof-of-concept vSMGW implementation is listed.

t *[h]*	# of failures	Cumulative failures	MTTR *[h]*	$\lambda(t)$	$\mu(t)$
0	2	2	1.25	2.000E^{-1}	1.892E^{-1}
10	0	2	n/a	9.259E^{-2}	1.328E^{0}
20	1	3	0.75	6.024E^{-2}	2.069E^{0}
30	0	3	n/a	4.464E^{-2}	2.586E^{0}
40	0	3	n/a	3.546E^{-2}	2.983E^{0}
50	0	3	n/a	2.941E^{-2}	3.305E^{0}
60	1	4	0.50	2.513E^{-2}	3.577E^{0}
70	0	4	n/a	2.193E^{-2}	3.811E^{0}
80	0	4	n/a	1.946E^{-2}	4.018E^{0}
90	0	4	n/a	1.748E^{-2}	4.202E^{0}
100	0	4	n/a	1.587E^{-2}	4.368E^{0}
110	0	4	n/a	1.453E^{-2}	4.520E^{0}
120	0	4	n/a	1.340E^{-2}	4.660E^{0}
130	0	4	n/a	1.244E^{-2}	4.789E^{0}
140	0	4	n/a	1.160E^{-2}	4.909E^{0}
150	1	5	1.75	1.087E^{-2}	5.021E^{0}
160	0	5	n/a	1.022E^{-2}	5.127E^{0}
170	0	5	n/a	9.653E^{-3}	5.226E^{0}
180	0	5	n/a	9.141E^{-3}	5.320E^{0}
190	0	5	n/a	8.681E^{-3}	5.409E^{0}
200	0	5	n/a	8.264E^{-3}	5.494E^{0}
210	0	5	n/a	7.886E^{-3}	5.574E^{0}
220	0	5	n/a	7.541E^{-3}	5.652E^{0}
230	1	6	0.75	7.225E^{-3}	5.725E^{0}
240	0	6	n/a	6.935E^{-3}	5.796E^{0}
250	0	6	n/a	6.667E^{-3}	5.864E^{0}
260	0	6	n/a	6.418E^{-3}	5.930E^{0}
270	0	6	n/a	6.188E^{-3}	5.993E^{0}
280	0	6	n/a	5.974E^{-3}	6.053E^{0}
290	0	6	n/a	5.774E^{-3}	6.112E^{0}
300	0	6	n/a	5.587E^{-3}	6.169E^{0}
310	0	6	n/a	5.411E^{-3}	6.224E^{0}
320	0	6	n/a	5.247E^{-3}	6.277E^{0}
330	0	6	n/a	5.092E^{-3}	6.329E^{0}
340	0	6	n/a	4.946E^{-3}	6.379E^{0}
350	1	7	6.00	4.808E^{-3}	6.428E^{0}
360	0	7	n/a	4.677E^{-3}	6.475E^{0}
370	0	7	n/a	4.554E^{-3}	6.521E^{0}
380	0	7	n/a	4.437E^{-3}	6.566E^{0}

Table continues on the next page ⇒

t *[h]*	# of failures	Cumulative failures	MTTR *[h]*	$\lambda(t)$	$\mu(t)$
390	0	7	n/a	4.325E^{-3}	6.610E^{0}
400	0	7	n/a	4.219E^{-3}	6.653E^{0}
410	0	7	n/a	4.119E^{-3}	6.694E^{0}
420	0	7	n/a	4.023E^{-3}	6.735E^{0}
430	0	7	n/a	3.931E^{-3}	6.775E^{0}
440	0	7	n/a	3.843E^{-3}	6.814E^{0}
450	0	7	n/a	3.759E^{-3}	6.852E^{0}
460	0	7	n/a	3.679E^{-3}	6.889E^{0}
470	0	7	n/a	3.602E^{-3}	6.925E^{0}
480	0	7	n/a	3.529E^{-3}	6.961E^{0}
490	0	7	n/a	3.458E^{-3}	6.996E^{0}
500	0	7	n/a	3.390E^{-3}	7.030E^{0}
510	0	7	n/a	3.324E^{-3}	7.064E^{0}
520	0	7	n/a	3.262E^{-3}	7.097E^{0}
530	0	7	n/a	3.201E^{-3}	7.129E^{0}
540	0	7	n/a	3.143E^{-3}	7.161E^{0}
550	0	7	n/a	3.086E^{-3}	7.192E^{0}
560	0	7	n/a	3.032E^{-3}	7.222E^{0}
570	0	7	n/a	2.980E^{-3}	7.253E^{0}
580	0	7	n/a	2.929E^{-3}	7.282E^{0}
590	0	7	n/a	2.880E^{-3}	7.311E^{0}
600	0	7	n/a	2.833E^{-3}	7.340E^{0}

Table A.1.: Input and result data for vSMGW test ($\Theta = 0.58$, $\lambda_0 = 0.2$)

Bibliography

[1] **M. N. Albasrawi, N. Jarus, K. A. Joshi, and S. S. Sarvestani**. "Analysis of Reliability and Resilience for Smart Grids." In: *IEEE 38th Annual Computer Software and Applications Conference*. July 2014, pp. 529–534. DOI: 10.1109/COMPSAC.2014.75.

[2] **A. O. Allen**. *Introduction to Computer Performance Analysis with Mathematica*. San Diego, CA, USA: Academic Press Professional, Inc., 1994. ISBN: 978-0-12-051070-2.

[3] **T. Almeroth, O. Kühn, and G. Linß**. "Lifetime Prediction of Smart Meter - Estimation of Lifetime Paramters." In: *Proceedings of the 56. IWK: Innovation in Mechanical Engineering – Shaping the Future*. Vol. 56. 2011.

[4] **J. Alonso, M. Grottke, A. P. Nikora, and K. S. Trivedi**. "The Nature of the Times to Flight Software Failure during Space Missions." In: *IEEE 23rd International Symposium on Software Reliability Engineering*. Nov. 2012, pp. 331–340. DOI: 10.1109/ISSRE.2012.32.

[5] **G. Alves, D. Marques, I. Silva, L. A. Guedes, and M. Silva**. "A Methodology for Dependability Evaluation of Smart Grids." In: *Energies* 12 (2019), p. 1817. DOI: 10.3390/en12091817.

[6] **H. H. Ammar, B. Cukic, A. Mili, and C. Fuhrman**. "A Comparative Analysis of Hardware and Software Fault Tolerance: Impact on Software Reliability Engineering." In: *Annals of Software Engineering* 10.1-4 (Jan. 2000), pp. 103–150. ISSN: 1022-7091. DOI: 10.1023/A:1018987616443.

[7] **A. Avižienis and J.-C. Laprie**. "Dependable Computing: From Concepts to Design Diversity." In: *Proceedings of the IEEE* 74.5 (May 1986), pp. 629–638. ISSN: 0018-9219. DOI: 10.1109/PROC.1986.13527.

[8] **A. Avižienis, J.-C. Laprie, and B. Randell**. *Fundamental Concepts of Dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[9] **A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr**. "Basic Concepts and Taxonomy of Dependable and Secure Computing." In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (Jan. 2004), pp. 11–33. ISSN: 1545-5971. DOI: 10.1109/TDSC.2004.2.

[10] **A. Aydeger, K. Akkaya, and A. S. Uluagac**. "SDN-based Resilience for Smart Grid Communications." In: *IEEE Conference on Network Function Virtualization and Software Defined Network*. Nov. 2015, pp. 31–33. DOI: 10.1109/NFV-SDN.2015.7387401.

[11] **A. Battaglini, J. Lilliestam, and G. Knies**. "The SuperSmart Grid – Paving the Way for a Completely Renewable Power System." In: *Global Sustainability: A Nobel Cause*. Ed. by H. J. Schellnhuber, M. Molina, N. Stern, V. Huber, and S. Kadner. Global Sustainability – A Nobel Cause. Cambridge University Press, 2010. Chap. 25, pp. 289–305. ISBN: 978-0-521-76934-1. URL: `http://www.nobel-cause.de/potsdam-2007/book/NobelCauseBook_chapter25.pdf`.

[12] **M. T. Beck and J. F. Botero**. "Coordinated Allocation of Service Function Chains." In: *IEEE Global Communications Conference*. Dec. 2015, pp. 1–6. DOI: 10.1109/GLOCOM.2015.7417401.

[13] **F. Benevenuto, C. Fernandes, M. C. Santos, V. Almeida, and J. Almeida**. *A Quantitative Analysis of the Xen Virtualization Overhead*. Tech. rep. Federal University of Minas Gerais, 2018.

[14] **J. Benze, A. Berl, K. Daniel, et al.** "VDE-Positionspapier Energieinformationsnetze und -systeme (Smart Grid Security)." In: *VDE-Kongress Smart Cities*. VDE, 2014.

[15] **A. Berl, M. Niedermeier, and H. De Meer**. "Smart Grid Considerations – Energy Efficiency vs. Security." In: *Green and Sustainable Computing: Part II*. Ed. by Ali Hurson. Vol. 88. Advances in Computers. Elsevier B.V., 2013, pp. 159–198. DOI: 10.1016/B978-0-12-407725-6.00004-6.

[16] **A. Berl, M. Niedermeier, A. Fischer, H. De Meer, and D. Hutchison**. "Virtual Energy Information Network: A Resilience Perspective." In: *e & i Elektrotechnik und Informationstechnik* 130.4–5 (2013), pp. 121–126. ISSN: 0932-383X. DOI: 10.1007/s00502-013-0142-4.

[17] **M. Bertoli, G. Casale, and G. Serazzi**. "JMT: Performance Engineering Tools for System Modeling." In: *ACM SIGMETRICS Performance Evaluation Review* 36.4 (2009), pp. 10–15. ISSN: 0163-5999. DOI: http://doi.acm.org/10.1145/1530873.1530877.

[18] **N. Biggs, E. K. Lloyd, and R. J. Wilson**. *Graph Theory, 1736-1936*. New York, NY, USA: Clarendon Press, 1986. ISBN: 978-0-19-853916-2.

[19] **B. E. Biringer, E. D. Vugrin, and D. E. Warren**. *Critical Infrastructure System Security and Resiliency*. CRC Press, 2013.

[20] **A. Birolini**. "Reliability and Availability of Repairable Systems." In: *Reliability Engineering: Theory and Practice*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 162–276. ISBN: 978-3-540-49390-7. DOI: 10.1007/978-3-540-49390-7_6.

[21] **B. S. Blanchard, D. C. Verma, and E. L. Peterson**. *Maintainability: A Key to Effective Serviceability and Maintenance Management*. Wiley-Interscience, 1995.

[22] **G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi**. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience, 2001, pp. 263–309. ISBN: 978-0-471-19366-1.

[23] **S. Brand**. "On Performance and Dependability in Virtualized AMI Systems." MA thesis. University of Passau, 2018.

[24] **Z. Bronstein and E. Shraga**. "NFV Virtualisation of the Home Environment." In: *IEEE 11th Consumer Communications and Networking Conference*. Jan. 2014, pp. 899–904. DOI: 10.1109/CCNC.2014.6940493.

[25] **Bundesamt für Sicherheit in der Informationstechnik**. *Protection Profile for the Gateway of a Smart Metering System (Smart Meter Gateway PP)*. Bundesamt für Sicherheit in der Informationstechnik. Mar. 2013.

[26] **Bundesamt für Sicherheit in der Informationstechnik**. *Technische Richtlinie BSI-TR-03109-1: Anforderungen an die Interoperabilität der Kommunikationseinheit eines intelligenten Messsystem*. Bundesamt für Sicherheit in der Informationstechnik. 2013. URL: `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03109/TR03109-1.pdf?__blob=publicationFile&v=3`.

[27] **Bundesverband der Energie- und Wasserwirtschaft e.V.** *TAB 2007 (Ausgabe 2011)*. Tech. rep. Berlin: Bundesverband der Energie- und Wasserwirtschaft e.V., 2011.

[28] **C. G. Cassandras and S. Lafortune**. *Introduction to Discrete Event Systems*. 2nd ed. Springer US, 2008. 772 pp. ISBN: 978-0-387-33332-8. DOI: 10.1007/978-0-387-68612-7.

[29] **W. Cerroni and F. Callegati**. "Live Migration of Virtual Network Functions in Cloud-Based Edge Networks." In: *IEEE International Conference on Communications*. June 2014, pp. 2963–2968. DOI: 10.1109/ICC.2014.6883775.

[30] **B. Chandra and M. M. Halldórsson**. "Approximation Algorithms for Dispersion Problems." In: *Journal of Algorithms* 38.2 (Feb. 2001), pp. 438–465. ISSN: 0196-6774.

[31] **H. Cheng, W. Wang, and C. Rong**. "Privacy Protection beyond Encryption for Cloud Big Data." In: *2nd International Conference on Information Technology and Electronic Commerce*. Dec. 2014, pp. 188–191. DOI: 10.1109/ICITEC.2014.7105598.

[32] **P. Chopade and M. Bikdash**. "Structural and Functional Vulnerability Analysis for Survivability of Smart Grid and SCADA Network under Severe Emergencies and WMD Attacks." In: *IEEE International Conference on Technologies for Homeland Security*. Nov. 2013, pp. 99–105. DOI: 10.1109/THS.2013.6698983.

[33] **N. M. M. K. Chowdhury and R. Boutaba**. "A Survey of Network Virtualization." In: *Computer Networks* 54.5 (2010), pp. 862–876. ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/j.comnet.2009.10.017. URL: `http://www.sciencedirect.com/science/article/pii/S1389128609003387`.

[34] **G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi**. "Automated Generation and Analysis of Markov Reward Models using Stochastic Reward Nets." In: *Linear Algebra, Markov Chains, and Queueing Models*. Ed. by C. D. Meyer and Robert J. Plemmons. New York, NY: Springer New York, 1993, pp. 145–191. ISBN: 978-1-4613-8351-2. DOI: 10.1007/978-1-4613-8351-2_11.

[35] **G. Ciardo, J. Muppala, and K. S. Trivedi**. "SPNP: Stochastic Petri Net Package." In: *Proceedings of the Third International Workshop on Petri Nets and Performance Models*. Dec. 1989, pp. 142–151. DOI: 10.1109/PNPM.1989.68548.

[36] **F. M. Cleveland**. "Cyber Security Issues for Advanced Metering Infrastructure (AMI)." In: *IEEE Power and Energy Society General Meeting – Conversion and Delivery of Electrical Energy in the 21st Century*. July 2008, pp. 1–5. DOI: 10.1109/PES.2008.4596535.

[37] **F. M. Cleveland**. *List of Cybersecurity for Smart Grid Standards and Guideines*. Whitepaper. IEC, May 2013.

[38] **D. Cotroneo, L. De Simone, A. K. Iannillo, A. Lanzaro, and R. Natella**. "Dependability Evaluation and Benchmarking of Network Function Virtualization Infrastructures." In: *1st IEEE Conference on Network Softwarization*. Apr. 2015, pp. 1–9. DOI: 10.1109/NETSOFT.2015.7116123.

[39] **D. Cotroneo, L. De Simone, A. K. Iannillo, et al.** "Network Function Virtualization: Challenges and Directions for Reliability Assurance." In: *IEEE International Symposium on Software Reliability Engineering Workshops*. Nov. 2014, pp. 37–42. DOI: 10.1109/ISSREW.2014.48.

[40] **C. Develder, J. Buysse, B. Dhoedt, and B. Jaumard**. "Joint Dimensioning of Server and Network Infrastructure for Resilient Optical Grids/Clouds." In: *IEEE/ACM Transactions on Networking* 22.5 (Oct. 2014), pp. 1591–1606. ISSN: 1063-6692. DOI: 10.1109/TNET.2013.2283924.

[41] **M. Di Mauro, M. Longo, and F. Postiglione**. "Performability Evaluation of Software Defined Networking Infrastructures." In: *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools*. Taormina, Italy: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2017, pp. 88–95. ISBN: 978-1-63190-141-6. DOI: 10.4108/eai.25-10-2016.2266605.

[42] **D. V. Dollen**. *Report to NIST on the Smart Grid Interoperability Standards Roadmap*. Tech. rep. National Institute of Standards and Technology, 2009.

[43] **X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk**. "Software-Defined Networking for Smart Grid Resilience: Opportunities and Challenges." In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security - CPSS '15*. ACM Press, 2015. DOI: 10.1145/2732198.2732203.

[44] **N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, and C. Wietfeld**. "Software-Defined Networking for Smart Grid Communications: Applications, Challenges and Advantages." In: *IEEE International Conference on Smart Grid Communications*. Nov. 2014, pp. 422–427. DOI: 10.1109/SmartGridComm.2014.7007683.

[45] **C. Eckert, C. Krauß, and P. Schoo**. *Sicherheit im Smart Grid – Eckpunkte für ein Energieinformationsnetz*. Stiftung-Verbundkolleg / Projekt Newise Nr. 90. 2011. URL: `https://www.aisec.fraunhofer.de/content/dam/aisec/Dokumente/Publikationen/Studien_TechReports/deutsch/sr90_sicherheit_im_energieinformationsnetz_gesamt.pdf`.

[46] **B. Egger, Y. Cho, C. Jo, E. Park, and J. Lee**. "Efficient Checkpointing of Live Virtual Machines." In: *IEEE Transactions on Computers* 65.10 (Oct. 2016), pp. 3041–3054. ISSN: 0018-9340. DOI: 10.1109/TC.2016.2519890.

[47] **EnerNex Corporation**. *Advanced Security Acceleration Project for the Smart Grid (ASAP-SG)*. Report prepared for the SG Security Working Group (UCAIug) and the NIST Cyber Security Coordination Task Group. Version 1.0, p. 16. EnerNex Corporation, Dec. 2009.

[48] **Ernst & Young GmbH**. *Kosten-Nutzen-Analyse für einen flächendeckenden Einsatz intelligenter Zähler. Endbericht zur Studie im Auftrag des Bundesministeriums für Wirtschaft und Technologie*. Report. Bundesministerium für Wirtschaft und Technologie, 2013. 239 pp.

[49] **European Telecommunications Standards Institute**. *Network Functions Virtualisation (NFV); Management and Orchestration, ETSI GS NFV-MAN 001 V1.1.1 (2014-12)*. Tech. rep. European Telecommunications Standards Institute, 2014.

[50] **European Telecommunications Standards Institute**. *Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action*. Tech. rep. European Telecommunications Standards Institute, 2012. URL: `https://portal.etsi.org/nfv/nfv_white_paper.pdf`.

[51] **Z. Fan, P. Kulkarni, S. Gormus, et al.** "Smart Grid Communications: Overview of Research Challenges, Solutions, and Standardization Activities." In: *IEEE Communications Surveys Tutorials* 15.1 (2013), pp. 21–38. ISSN: 1553-877X. DOI: 10.1109/SURV.2011.122211.00021.

[52] **A. Fawaz, R. Berthier, and W. H. Sanders**. "A Response Cost Model for Advanced Metering Infrastructures." In: *IEEE Transactions on Smart Grid* 7.2 (Mar. 2016), pp. 543–553. ISSN: 1949-3053. DOI: 10.1109/TSG.2015.2418736.

[53] **Federal Energy Regulatory Commission**. *Assessment of Demand Response & Advanced Metering*. Tech. rep. pp. 17, 20-24, 31, Appendix C, Report required by the Energy Policy Act of 2005. Federal Energy Regulatory Commission, Dec. 2008.

[54] **A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach**. "Virtual Network Embedding: A Survey." In: *IEEE Communications Surveys & Tutorials* 15.4 (2013), pp. 1888–1906. DOI: 10.1109/SURV.2013.013013.00155.

[55] **J. C. Foreman and D. Gurugubelli**. *Cyber Attack Surface Analysis of Advanced Metering Infrastructure*. 2016. URL: `https://arxiv.org/abs/1607.04811`.

[56] **M. A. Friedman, P. Y. Tran, and P. L. Goddard**. *Reliability of Software Intensive Systems*. Advanced computing and telecommunications series. Noyes Data Corporation, 1995. ISBN: 978-0-8155-1361-2.

[57] **T. J. Gerpott and M. Paukert**. "Determinants of Willingness to Pay for Smart Meters: An Empirical Analysis of Household Customers in Germany." In: *Energy Policy* 61 (2013), pp. 483–495. ISSN: 0301-4215. DOI: 10.1016/j.enpol.2013.06.012. URL: `http://www.sciencedirect.com/science/article/pii/S0301421513004977`.

[58] **M. Grottke, A. P. Nikora, and K. S. Trivedi**. "An Empirical Investigation of Fault Types in Space Mission System Software." In: *IEEE/IFIP International Conference on Dependable Systems Networks*. June 2010, pp. 447–456. DOI: 10.1109/DSN.2010.5544284.

[59] **T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell**. "A Systematic Literature Review on Fault Prediction Performance in Software Engineering." In: *IEEE Transactions on Software Engineering* 38.6 (Nov. 2012), pp. 1276–1304. ISSN: 0098-5589. DOI: 10.1109/TSE.2011.103.

[60] **B. R. Haverkort**. *Performance of Computer Communication Systems: A Model-Based Approach*. New York, NY, USA: John Wiley & Sons, Inc., 1998. ISBN: 978-0-471-97228-0.

[61] **B. R. Haverkort, R. Marie, G. Rubino, and K. S. Trivedi**. *Performability Modelling Techniques and Tools*. John Wiley & Sons Academic Press Professional, Inc., 2001. 338 pp. ISBN: 978-0-471-49195-8.

[62] **Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton**. "Software Rejuvenation: Analysis, Module and Applications." In: *25th International Symposium on Fault-Tolerant Computing. Digest of Papers*. June 1995, pp. 381–390. DOI: 10.1109/FTCS.1995.466961.

[63] **Z. Huang, C. Wang, M. Stojmenovic, and A. Nayak**. "Balancing System Survivability and Cost of Smart Grid Via Modeling Cascading Failures." In: *IEEE Transactions on Emerging Topics in Computing* 1.1 (June 2013), pp. 45–56. ISSN: 2168-6750. DOI: 10.1109/TETC.2013.2273079.

[64] **J. Y. Hwang, S. B. Suh, S. K. Heo, et al.** "Xen on ARM: System Virtualization Using Xen Hypervisor for ARM-Based Secure Mobile Phones." In: *5th IEEE Consumer Communications and Networking Conference*. Jan. 2008, pp. 257–261. DOI: 10.1109/ccnc08.2007.64.

[65] **International Electrotechnical Commission**. *International Electrotechnical Vocabulary: Chapter 191: Dependability and Quality of Service*. Standard. Geneva, Switzerland: International Electrotechnical Commission, 2001.

[66] **IEC SC 65A**. *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*. Standard IEC 61508. International Electrotechnical Commission, 2010.

[67] **International Energy Agency**. *World Energy Outlook 2017*. 2017, p. 763. DOI: 10.1787/weo-2017-en. URL: `https://www.oecd-ilibrary.org/content/publication/weo-2017-en`.

[68] **P. Jalote and B. Murphy**. "Reliability Growth in Software Products." In: *15th International Symposium on Software Reliability Engineering*. Nov. 2004, pp. 47–53. DOI: 10.1109/ISSRE.2004.34.

[69] **B. Jaumard, A. Shaikh, and C. Develder**. "Selecting the Best Locations for Data Centers in Resilient Optical Grid/Cloud Dimensioning." In: *14th International Conference on Transparent Optical Networks*. July 2012, pp. 1–4. DOI: 10.1109/ICTON.2012.6253739.

[70] **I. Kaitovic, S. Lukovic, and M. Malek**. "Unifying Dependability of Critical Infrastructures: Electric Power System and ICT: Concepts, Figures of Merit and Taxonomy." In: *IEEE 21st Pacific Rim International Symposium on Dependable Computing*. Nov. 2015, pp. 50–59. DOI: 10.1109/PRDC.2015.38.

[71] **K. C. Kapur and M. Pecht**. *Reliability Engineering*. Wiley Series in Systems Engineering and Management. Wiley, 2014. ISBN: 978-1-118-14067-3.

[72] **D. S. Kim, F. Machida, and K. S. Trivedi**. "Availability Modeling and Analysis of a Virtualized System." In: *15th IEEE Pacific Rim International Symposium on Dependable Computing*. Nov. 2009, pp. 365–371. DOI: 10.1109/PRDC.2009.64.

[73] **H. Kim**. "Reliable p-Hub Location Problems and Protection Models for Hub Network Design." PhD Thesis. Ohio State University, 2008.

[74] **S.-H. Kim and B. L. Nelson**. "A Fully Sequential Procedure for Indifference-zone Selection in Simulation." In: *ACM Transactions on Modeling and Computer Simulation* 11.3 (July 2001), pp. 251–273. ISSN: 1049-3301. DOI: 10.1145/502109.502111.

[75] **A. Koziolek, A. Avritzer, S. Suresh, et al.** "Design of Distribution Automation Networks Using Survivability Modeling and Power Flow Equations." In: *IEEE 24th International Symposium on Software Reliability Engineering*. Nov. 2013, pp. 41–50. DOI: 10.1109/ISSRE.2013.6698903.

[76] **J. F. Kurose and K. W. Ross**. *Computer Networking: A Top-down Approach*. Pearson, 2013. ISBN: 978-0-13-285620-1.

[77] **M. Kuzlu, M. Pipattanasomporn, and S. Rahman**. "Communication Network Requirements for Major Smart Grid Applications in HAN, NAN and WAN." In: *Computer Networks* 67 (2014), pp. 74–88. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2014.03.029. URL: `http://www.sciencedirect.com/science/article/pii/S1389128614001431`.

[78] **J.-C. Laprie**. *Dependability: Basic Concepts and Terminology*. Ed. by Springer-Verlag Wien. Springer-Verlag Wien, 1992. DOI: 10.1007/978-3-7091-9170-5.

[79] **J.-C. Laprie**. "Dependable Computing and Fault Tolerance: Concepts and Terminology." In: *25th International Symposium on Fault-Tolerant Computing*. June 1995, pp. 2–11. DOI: 10.1109/FTCSH.1995.532603.

[80] **M. G. Lauby, J. Bian, A. Bennett, et al.** *2009 Long-Term Reliability Assessment. Annual Evaluation of the Reliability of the Bulk Power System*. Tech. rep. North American Electric Reliability Corporation, Oct. 2009.

[81] **Z. Li, M. Kihl, Q. Lu, and J. A. Andersson**. "Performance Overhead Comparison between Hypervisor and Container Based Virtualization." In: *IEEE 31st International Conference on Advanced Information Networking and Applications*. Mar. 2017, pp. 955–962. DOI: 10.1109/AINA.2017.79.

[82] **V. Liberatore and A. Al-Hammouri**. "Smart Grid Communication and Co-Simulation." In: *IEEE Energytech*. May 2011, pp. 1–5. DOI: 10.1109/EnergyTech.2011.5948542.

[83] **C. Lindemann**. "An Improved Numerical Algorithm for Calculating Steady-state Solutions of Deterministic and Stochastic Petri Net Models." In: *Proceedings of the 4th International Workshop on Petri Nets and Performance Models*. Dec. 1991, pp. 176–185. DOI: 10.1109/PNPM.1991.238803.

[84] **C. Lindemann**. "Employing the Randomization Technique for Solving Stochastic Petri Net Models." In: A. Lehmann and F. Lehmann. *Messung, Modellierung und Bewertung von Rechensystemen: 6. GI/ITG-Fachtagung, Neubiberg, 18.–20. September 1991*. Informatik-Fachberichte. Springer Berlin Heidelberg, 1991, p. 388. ISBN: 978-3-642-76934-4.

[85] **P. Lv, X. Wang, Y. Yang, and M. Xu**. "Network Virtualization for Smart Grid Communications." In: *IEEE Systems Journal* 8.2 (June 2014), pp. 471–482. ISSN: 1932-8184. DOI: 10.1109/JSYST.2013.2260695.

[86] **F. Machida, D. S. Kim, J. S. Park, and K. S. Trivedi**. "Toward Optimal Virtual Machine Placement and Rejuvenation Scheduling in a Virtualized Data Center." In: *IEEE International Conference on Software Reliability Engineering Workshops*. Nov. 2008, pp. 1–3. DOI: 10.1109/ISSREW.2008.5355515.

[87] **F. Machida, D. S. Kim, and K. S. Trivedi**. "Modeling and Analysis of Software Rejuvenation in a Server Virtualized System." In: *IEEE 2nd International Workshop on Software Aging and Rejuvenation*. Nov. 2010, pp. 1–6. DOI: 10.1109/WOSAR.2010.5722098.

[88] **F. Machida, J. Xiang, K. Tadano, and Y. Maeno**. "Combined Server Rejuvenation in a Virtualized Data Center." In: *9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*. Sept. 2012, pp. 486–493. DOI: 10.1109/UIC-ATC.2012.52.

[89] **K. Maheshwari, M. Lim, L. Wang, K. Birman, and R. van Renesse**. "Toward a Reliable, Secure and Fault Tolerant Smart Grid State Estimation in the Cloud." In: *IEEE PES Innovative Smart Grid Technologies*. Feb. 2013, pp. 1–6. DOI: 10.1109/ISGT.2013.6497831.

[90] **S. Manel, A. Ridha, and M. Alia**. "Optimised Migrate Virtual Machine Rejuvenation." In: *Journal of Computer and Communications* 3.8 (2015), pp. 33–40. DOI: 10.4236/jcc.2015.38004.

[91] **K. Marashi and S. S. Sarvestani**. "Towards Comprehensive Modeling of Reliability for Smart Grids: Requirements and Challenges." In: *IEEE 15th International Symposium on High-Assurance Systems Engineering*. Jan. 2014, pp. 105–112. DOI: 10.1109/HASE.2014.23.

[92] **Massachusetts Institute of Technology**. *The Future of the Electric Grid – an Interdisciplinary MIT Study*. Tech. rep. Massachusetts Institute of Technology, 2011.

[93] **V. Mathew, R. K. Sitaraman, and P. Shenoy**. "Energy-aware Load Balancing in Content Delivery Networks." In: *Proceedings IEEE INFOCOM*. Mar. 2012, pp. 954–962. DOI: 10.1109/INFCOM.2012.6195846.

[94] **R. D. S. Matos, P. R. M. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi**. "Sensitivity Analysis of Server Virtualized System Availability." In: *IEEE Transactions on Reliability* 61.4 (Dec. 2012), pp. 994–1006. DOI: 10.1109/tr.2012.2220711.

[95] **D. A. Menascé**. "Virtualization: Concepts, Applications, and Performance Modeling." In: *31th International Computer Measurement Group Conference, December 4-9, 2005, Orlando, Florida, USA, Proceedings*. 2005, pp. 407–414. URL: `http://www.cmg.org/?s2member%5C_file%5C_download=/proceedings/2005/5189.pdf`.

[96] **D. S. Menasché, A. Avritzer, S. Suresh, et al.** "Assessing Survivability of Smart Grid Distribution Network Designs Accounting for Multiple Failures." In: *Concurrency and Computation: Practice and Experience* 26.12 (Aug. 2014), pp. 1949–1974. ISSN: 1532-0626. DOI: 10.1002/cpe.3241.

[97] **W. Meng, R. Ma, and H. H. Chen**. "Smart Grid Neighborhood Area Networks: A Survey." In: *IEEE Network* 28.1 (Jan. 2014), pp. 24–32. ISSN: 0890-8044. DOI: 10.1109/MNET.2014.6724103.

[98] **J. F. Meyer**. "Performability: A Retrospective and some Pointers to the Future." In: *Performance Evaluation* 14.3 (1992), pp. 139–156. ISSN: 0166-5316. DOI: http://dx.doi.org/10.1016/0166-5316(92)90002-X. URL: http://www.sciencedirect.com/science/article/pii/016653169290002X.

[99] **L. Mirtskhulava, R. Kakubava, N. Ananiashvili, and G. Gugunashvili**. "Internet Reliability and Availability Analysis Using Markov Method." In: *AMSS 16th International Conference on Computer Modelling and Simulation*. Mar. 2014, pp. 423–427. DOI: 10.1109/UKSim.2014.14.

[100] **R. R. Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar**. "A Survey on Advanced Metering Infrastructure." In: *International Journal of Electrical Power & Energy Systems* 63 (2014), pp. 473–484. ISSN: 0142-0615. DOI: http://dx.doi.org/10.1016/j.ijepes.2014.06.025. URL: http://www.sciencedirect.com/science/article/pii/S0142061514003743.

[101] **R. Morabito, J. Kjällman, and M. Komu**. "Hypervisors vs. Lightweight Virtualization: A Performance Comparison." In: *IEEE International Conference on Cloud Engineering*. Mar. 2015, pp. 386–393. DOI: 10.1109/IC2E.2015.74.

[102] **K. Moslehi and R. Kumar**. "A Reliability Perspective of the Smart Grid." In: *IEEE Transactions on Smart Grid* 1.1 (June 2010), pp. 57–64. ISSN: 1949-3053. DOI: 10.1109/TSG.2010.2046346.

[103] **J. K. Muppala, G. Ciardo, and K. S. Trivedi**. "Stochastic Reward Nets for Reliability Prediction." In: *Communications in Reliability, Maintainability and Serviceability*. 1994, pp. 9–20.

[104] **J. D. Musa and K. Okumoto**. "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement." In: *Proceedings of the 7th International Conference on Software Engineering*. ICSE '84. IEEE Press, 1984, pp. 230–238. ISBN: 978-0-8186-0528-4.

[105] **M. T. H. Myint and T. Thein**. "Availability Improvement in Virtualized Multiple Servers with Software Rejuvenation and Virtualization." In: *4th International Conference on Secure Software Integration and Reliability Improvement*. June 2010, pp. 156–162. DOI: 10.1109/SSIRI.2010.19.

[106] **National Energy Technology Laboratory**. *A Systems View of the Modern Grid*. Tech. rep. Report conducted for the U.S. Department of Energy Office of Electricity Delivery and Energy Reliability. National Energy Technology Laboratory, Jan. 2007.

[107] **National Energy Technology Laboratory**. *The NETL Modern Grid Strategy Powering our 21st-Century Economy: Advanced Metering Infrastructure*. Tech. rep. National Energy Technology Laboratory, 2008. URL: https://www.smartgrid.gov/files/NIST_SG_Interop_Report_Postcommentperiod_version_200808.pdf.

[108] **National Institute of Standards and Technology**. *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0 (NIST Special Publication 1108)*. National Institute of Standards and Technology. Jan. 2014.

[109] **O. Neagu and W. Hamouda**. "Performance of Smart Grid Communication in the Presence of Impulsive Noise." In: *International Conference on Selected Topics in Mobile Wireless Networking*. Apr. 2016, pp. 1–5. DOI: 10.1109/MoWNet.2016.7496614.

[110] **C. P. Nguyen and A. J. Flueck**. "A Novel Strategy for Failure-tolerant Communication in Smart Grids." In: *IEEE PES General Meeting, Conference Exposition*. July 2014, pp. 1–5. DOI: 10.1109/PESGM.2014.6939516.

[111] **T. A. Nguyen, D. Min, and E. Choi**. "A Comprehensive Evaluation of Availability and Operational Cost for a Virtualized Server System Using Stochastic Reward Nets." In: *The Journal of Supercomputing* (Aug. 2017). ISSN: 1573-0484. DOI: 10.1007/s11227-017-2127-2.

[112] **M. Niedermeier and H. De Meer**. "Constructing Dependable Smart Grid Networks using Network Functions Virtualization." In: *Journal of Network and Systems Management* (2016), pp. 1–21. ISSN: 1573-7705. DOI: 10.1007/s10922-016-9380-1.

[113] **A. Nieße, M. Tröschel, and M. Sonnenschein**. "Designing Dependable and Sustainable Smart Grids – How to apply Algorithm Engineering to Distributed Control in Power Systems." In: *Environmental Modelling & Software* 56 (2014). Thematic Issue on Modelling and Evaluating the Sustainability of Smart Solutions, pp. 37–51. ISSN: 1364-8152. DOI: 10.1016/j.envsoft.2013.12.003. URL: http://www.sciencedirect.com/science/article/pii/S136481521300306X.

[114] **D. Niyato, P. Wang, and E. Hossain**. "Reliability Analysis and Redundancy Design of Smart Grid Wireless Communications System for Demand Side Management." In: *IEEE Wireless Communications* 19.3 (June 2012), pp. 38–46. ISSN: 1536-1284. DOI: 10.1109/MWC.2012.6231158.

[115] **A. F. Ocampo, J. Gil-Herrera, P. H. Isolani, et al.** "Optimal Service Function Chain Composition in Network Functions Virtualization." In: *Security of Networks and Services in an All-Connected World*. Ed. by D. Tuncer, R. Koch, R. Badonnel, and B. Stiller. Cham: Springer International Publishing, 2017, pp. 62–76. ISBN: 978-3-319-60774-0.

[116] **E. E. Ogheneovo**. "Software Maintenance and Evolution: The Implication for Software Development." In: *West African Journal of Industrial and Academic Research* 7.1 (2013), pp. 81–92.

[117] **V. H. Okabayashi, I. C. G. Ribeiro, D. M. Passos, and C. V. N. Albuquerque**. "A Resilient Dynamic Gateway Selection Algorithm Based on Quality Aware Metrics for Smart Grids." In: *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. MSWiM '15. Cancun, Mexico: ACM, 2015, pp. 91–98. ISBN: 978-1-4503-3762-5. DOI: 10.1145/2811587.2811613.

[118] **B. Parhami**. "From Defects to Failures: A View of Dependable Computing." In: *ACM SIGARCH Computer Architecture News* 16.4 (Sept. 1988), pp. 157–168. ISSN: 0163-5964. DOI: 10.1145/54331.54345.

[119] **D. L. Parnas**. "Software Aging." In: *Proceedings of 16th International Conference on Software Engineering*. May 1994, pp. 279–287. DOI: 10.1109/ICSE.1994.296790.

[120] **D. L. Parnas, A. J. van Schouwen, and S. P. Kwan**. "Evaluation of Safety-critical Software." In: *Communications of the ACM* 33.6 (June 1990), pp. 636–648. ISSN: 0001-0782. DOI: 10.1145/78973.78974.

[121] **T. Petermann, H. Bradke, A. Lüllmann, M. Poetzsch, and U. Riehm**. *Was bei einem Blackout geschieht. Folgen eines langandauernden und großflächigen Stromausfalls*. Studien des Büros für Technikfolgen-Abschätzung beim Deutschen Bundestag, Bd. 33. Berlin: Edition Sigma, 2011. ISBN: 978-3-8360-8133-7. URL: http://www.tab-beim-bundestag.de/de/pdf/publikationen/buecher/petermann-etal-2011-141.pdf.

[122] **T. Pfeiffenberger, J. L. Du, P. B. Arruda, and A. Anzaloni**. "Reliable and Flexible Communications for Power Systems: Fault-tolerant Multicast with SDN/OpenFlow." In: *7th International Conference on New Technologies, Mobility and Security*. July 2015, pp. 1–6. DOI: 10.1109/NTMS.2015.7266517.

[123] **S. Piaszeck, L. Wenzel, and A. Wolf**. *Regional Diversity in the Costs of Electricity Outages: Results for German Counties*. Tech. rep. Hamburg Institute of International Economics, Sept. 2013.

[124] **G. J. Popek and R. P. Goldberg**. "Formal Requirements for Virtualizable Third Generation Architectures." In: *SIGOPS Operating Systems Review* 7.4 (Jan. 1973), pp. 412–421. ISSN: 0163-5980. DOI: 10.1145/957195.808061.

[125] **P. P. S. Priya and V. Saminadan**. "Performance Analysis of WiMAX Based Smart Grid Communication Traffic Priority Model." In: *International Conference on Communication and Signal Processing*. Apr. 2014, pp. 778–782. DOI: 10.1109/ICCSP.2014.6949949.

[126] **D. N. Quang, O. H. See, L. L. Chee, C. Y. Xuen, and S. Karuppiah**. "Performance Testing Framework for Smart Grid Communication Network." In: *IOP Conference Series: Earth and Environmental Science* 16.1 (2013), pp. 1–4. URL: http://stacks.iop.org/1755-1315/16/i=1/a=012147.

[127] **H. V. Ramasamy and M. Schunter**. "Architecting Dependable Systems using Virtualization." In: *Workshop on Architecting Dependable Systems in conjunction with 2007 International Conference on Dependable Systems and Networks*. 2007.

[128] **R. Ramaswamy, N. Weng, and T. Wolf**. "Characterizing Network Processing Delay." In: *IEEE Global Telecommunications Conference*. Vol. 3. Nov. 2004, pp. 1629–1634. DOI: 10.1109/GLOCOM.2004.1378257.

[129] **D. F. Ramírez, S. Cespedes, C. H. R. Becerra, and C. Lazo**. "Performance Evaluation of Future Ami Applications in Smart Grid Neighborhood Area Networks." In: *IEEE Colombian Conference on Communication and Computing* (2015), pp. 1–6.

[130] **S. S. Rathore and S. Kumar**. "A Study on Software Fault Prediction Techniques." In: *Artificial Intelligence Review* (May 2017). ISSN: 1573-7462. DOI: 10.1007/s10462-017-9563-5.

[131] **S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi**. "Facility Dispersion Problems: Heuristics and Special Cases." In: *Algorithms and Data Structures*. Ed. by F. Dehne, J.-R. Sack, and N. Santoro. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 355–366. ISBN: 978-3-540-47566-8.

[132] **S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi**. "Heuristic and Special Case Algorithms for Dispersion Problems." In: *Operations Research* 42.2 (Apr. 1994), pp. 299–310. ISSN: 0030-364X.

[133] **S. Rawat, N. Goyal, and M. Ram**. "Software Reliability Growth Modeling for Agile Software Development." In: *International Journal of Applied Mathematics and Computer Science* 27.4 (Dec. 2017), pp. 777–783. DOI: 10.1515/amcs-2017-0054.

[134] **M. H. Rehmani, A. Davy, B. Jennings, and C. Assi**. *Software Defined Networks based Smart Grid Communication: A Comprehensive Survey*. 2018. URL: `http://arxiv.org/abs/1801.04613v3`.

[135] **C. Rehtanz**. *Autonomous Systems and Intelligent Agents in Power System Control and Operation (Power Systems)*. Springer Verlag, 2003. ISBN: 978-3-540-40202-2.

[136] **S. Rinaldi, P. Ferrari, D. Brandao, and S. Sulis**. "Software Defined Networking Applied to the Heterogeneous Infrastructure of Smart Grid." In: *IEEE World Conference on Factory Communication Systems*. May 2015, pp. 1–4. DOI: 10.1109/WFCS.2015.7160573.

[137] **P. Rook**. *Software Reliability Handbook*. New York, NY, USA: Elsevier Science Inc., 1990. ISBN: 978-1-85166-400-9.

[138] **L. Rosenberg, T. Hammer, and J. Shaw**. "Software Metrics and Reliability." In: *9th International Symposium on Software Reliability Engineering*. 1998.

[139] **M. Rost and S. Schmid**. "NP-Completeness and Inapproximability of the Virtual Network Embedding Problem and Its Variants." In: *CoRR* abs/1801.03162 (2018).

[140] **J. Sahoo, S. Mohapatra, and R. Lath**. "Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues." In: *2nd International Conference on Computer and Network Technology*. Apr. 2010, pp. 222–226. DOI: 10.1109/ICCNT.2010.49.

[141] **F. Salfner and M. Malek**. "Proactive Fault Handling for System Availability Enhancement." In: *19th IEEE International Parallel and Distributed Processing Symposium*. Apr. 2005, pp. 7–13. DOI: 10.1109/IPDPS.2005.360.

[142] **A. Sauhats, V. Chuvychin, N. Gurov, et al.** "The Latvian Experience and Problems of the Grid Integration of Renewable Energy Sources in the Power System." In: *IEEE Power Tech Russia*. June 2005, pp. 1–7. DOI: 10.1109/PTC.2005.4524794.

[143] **R. R. Schrieber, H. L. Willis, and E. Philips**. *Aging Power Delivery Infrastructures*. Ed. by CRC Press. 2nd ed. Power Engineering (Willis). CRC Press, 2013. ISBN: 978-0-203-91091-7.

[144] **B. Schroeder and G. Gibson**. "A Large-Scale Study of Failures in High-Performance Computing Systems." In: *IEEE Transactions on Dependable and Secure Computing* 7.4 (Oct. 2010), pp. 337–350. ISSN: 1545-5971. DOI: 10.1109/TDSC.2009.4.

[145] **S. M. Shariati, A. Abouzarjomehri, and M. H. Ahmadzadegan**. "Challenges and Security Issues in Cloud Computing from Two Perspectives: Data Security and Privacy Protection." In: *2nd International Conference on Knowledge-Based Engineering and Innovation*. Nov. 2015, pp. 1078–1082. DOI: 10.1109/KBEI.2015.7436196.

[146] **Y. Shi, X. Qiu, and S. Guo**. "Genetic Algorithm-based Redundancy Optimization Method for Smart Grid Communication Network." In: *China Communications* 12.8 (Aug. 2015), pp. 73–84. ISSN: 1673-5447. DOI: 10.1109/CC.2015.7224708.

[147] **T. Shiobara, P. Palensky, and H. Nishi**. "Effective Metering Data Aggregation for Smart Grid Communication Infrastructure." In: *41st Annual Conference of the IEEE Industrial Electronics Society*. Institute of Electrical and Electronics Engineers Inc., Jan. 2015, pp. 2136–2141. DOI: 10.1109/IECON.2015.7392417.

[148] **R. Shojaee, A. Latifi, and N. Yazdani**. "A Stochastic Reward Net Approach to Model Availability of Cloud Virtualization." In: *7th International Symposium on Telecommunications*. Sept. 2014, pp. 683–688. DOI: 10.1109/ISTEL.2014.7000790.

[149] **Silicon Laboratories, Inc.** *Smart Metering Brings Intelligence and Connectivity to Utilities, Green Energy and Natural Resource Management*. Tech. rep. Silicon Laboratories, Inc., 2012. URL: `http://www.silabs.com/Support%20Documents/TechnicalDocs/Designing-Low-Power-Metering-Applications.pdf`.

[150] **L. M. Silva, J. Alonso, and J. Torres**. "Using Virtualization to Improve Software Rejuvenation." In: *IEEE Transactions on Computers* 58.11 (Nov. 2009), pp. 1525–1538. ISSN: 0018-9340. DOI: 10.1109/TC.2009.119.

[151] **C. Sivapragash, S. R. Thilaga, and S. S. Kumar**. "Advanced Cloud Computing in Smart Power Grid." In: *IET Chennai 3rd International on Sustainable Energy and Intelligent Systems*. Dec. 2012, pp. 1–6. DOI: 10.1049/cp.2012.2238.

[152] **D. R. Skuce and A. Mili**. "Behavorial Specifications in Object-Oriented Programming." In: *Journal of Object Oriented Programming* 7.8 (1995), pp. 41–49.

[153] **M. J. Steidele**. "Virtualisierung von Smart Grid Gateways." MA thesis. University of Passau, 2015.

[154] **Z. Sui, M. Niedermeier, and H. De Meer**. "RESA: A Robust and Efficient Secure Aggregation Scheme in Smart Grids." In: *10th International Conference on Critical Information Infrastructures Security*. 2015, pp. 183–194.

[155] **A. T. Tai, J. F. Meyer, and A. Avizienis**. "Performability Enhancement of Fault-tolerant Software." In: *IEEE Transactions on Reliability* 42.2 (May 1993), pp. 227–237. ISSN: 0018-9529. DOI: 10.1109/24.229492.

[156] **T. Thein, M. Pokharel, S. D. Chi, and J. S. Park**. "A Recovery Model for Survivable Distributed Systems through the Use of Virtualization." In: *4th International Conference on Networked Computing and Advanced Information Management*. Vol. 1. Sept. 2008, pp. 79–84. DOI: 10.1109/NCM.2008.213.

[157] **O. Tickoo, R. Iyer, R. Illikkal, and D. Newell**. "Modeling Virtual Machine Performance: Challenges and Approaches." In: *SIGMETRICS Performance Evaluation Review* 37.3 (Jan. 2010), pp. 55–60. ISSN: 0163-5999. DOI: 10.1145/1710115.1710126.

[158] **G. Tong, H. Jin, X. Xie, W. Cao, and P. Yuan**. "Measuring and Analyzing CPU Overhead of Virtualization System." In: *IEEE Asia-Pacific Services Computing Conference*. Dec. 2011, pp. 243–250. DOI: 10.1109/APSCC.2011.40.

[159] **W. Torell and V. Avelar**. *Mean Time Between Failure: Explanation and Standards*. Tech. rep. Schneider Electric, Jan. 2004.

[160] **M. Treaster**. *A Survey of Fault-Tolerance and Fault-Recovery Techniques in Parallel Systems*. 2005. URL: `http://arxiv.org/abs/cs/0501002`.

[161] **U. Trick, M. Steinheimer, P. Ruhrig, et al.** "Herausforderungen an die Kommunikationstechnik im Smart Home/Grid." In: *VDE/ITG Fachtagung Mobilkommunikation*. May 2012. URL: `https://www.vde-verlag.de/proceedings-en/453438016.html`.

[162] **K. S. Trivedi**. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. 2nd ed. John Wiley and Sons Ltd., 2002. ISBN: 978-0-471-33341-8.

[163] **K. S. Trivedi and A. Bobbio**. *Reliability and Availability Engineering: Modeling, Analysis, and Applications*. Cambridge University Press, 2017. ISBN: 978-1-107-09950-0.

[164] **K. S. Trivedi, D. S. Kim, A. Roy, and D. Medhi**. "Dependability and Security Models." In: *7th International Workshop on Design of Reliable Communication Networks*. Oct. 2009, pp. 11–20. DOI: 10.1109/DRCN.2009.5340029.

[165] **M. Uslar, M. Specht, C. Dänekas, et al.** *Standardization in Smart Grids: Introduction to IT-Related Methodologies, Architectures and Standards*. Power Systems. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-34916-4.

[166] **H. Valchanov**. "Performance Study of Virtualization Platforms for Virtual Networking Laboratory." In: *Proceedings of the International Scientific Conference on Information, Communication and Energy Systems and Technologies*. June 2012, pp. 443–446.

[167] **A. van Cleeff, W. Pieters, and R. J. Wieringa**. "Security Implications of Virtualization: A Literature Study." In: *International Conference on Computational Science and Engineering*. Vol. 3. Aug. 2009, pp. 353–358. DOI: 10.1109/CSE.2009.267.

[168] **P. P. Varaiya, F. F. Wu, and J. W. Bialek**. "Smart Operation of Smart Grid: Risk-Limiting Dispatch." In: *Proceedings of the IEEE* 99.1 (Jan. 2011), pp. 40–57. ISSN: 0018-9219. DOI: 10.1109/JPROC.2010.2080250.

[169] **J. P. Walters, V. Chaudhary, M. Cha, S. Guercio Jr., and S. Gallo**. "A Comparison of Virtualization Technologies for HPC." In: *22nd International Conference on Advanced Information Networking and Applications*. Mar. 2008, pp. 861–868. DOI: 10.1109/AINA.2008.45.

[170] **B. Wei, C. Lin, and X. Kong**. "Dependability Modeling and Analysis for the Virtual Data Center of Cloud Computing." In: *IEEE International Conference on High Performance Computing and Communications*. Sept. 2011, pp. 784–789. DOI: 10.1109/HPCC.2011.111.

[171] **L. Wenting, L. Jiang, X. Zongyou, Q. Guangping, and Z. Hu**. "Step-stress Accelerated Life Testing to Predict Service Life for Space Vehicle Electrical System." In: *IEEE Chinese Guidance, Navigation and Control Conference*. Aug. 2016, pp. 1120–1125. DOI: 10.1109/CGNCC.2016.7828945.

[172] **K. Wolter, A. Avritzer, M. Vieira, and A. van Moorsel**. *Resilience Assessment and Evaluation of Computing Systems*. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-29032-9.

[173] **C. X. Wu, C. Y. Chung, F. S. Wen, and D. Y. Du**. "Reliability / Cost Evaluation With PEV and Wind Generation System." In: *IEEE Transactions on Sustainable Energy* 5.1 (Jan. 2014), pp. 273–281. ISSN: 1949-3029. DOI: 10.1109/TSTE.2013.2281515.

[174] **M. Xiang, S. Tauch, and W. Liu**. "Dependability and Resource Optimation Analysis for Smart Grid Communication Networks." In: *IEEE 4th International Conference on Big Data and Cloud Computing*. Dec. 2014, pp. 676–681. DOI: 10.1109/BDCloud.2014.115.

[175] **Y. Xiang, L. Wang, and T. Fu**. "A Preliminary Study of Power System Reliability Considering Cloud Service Reliability." In: *International Conference on Power System Technology*. Oct. 2014, pp. 2031–2036. DOI: 10.1109/POWERCON.2014.6993999.

[176] **Y. Xie, H. Wen, J. Wu, et al.** "Three-Layers Secure Access Control for Cloud-Based Smart Grids." In: *IEEE 82nd Vehicular Technology Conference*. Sept. 2015, pp. 1–5. DOI: 10.1109/VTCFall.2015.7391174.

[177] **Y. Xin, I. Baldine, J. Chase, et al.** "Virtual Smart Grid Architecture and Control Framework." In: *IEEE International Conference on Smart Grid Communications*. Oct. 2011, pp. 1–6.

[178] **S. Xu, Y. Qian, and R. Q. Hu**. "On Reliability of Smart Grid Neighborhood Area Networks." In: *IEEE Access* 3 (2015), pp. 2352–2365. DOI: 10.1109/ACCESS.2015.2502250.

[179] **W.-L. Yeow, C. Westphal, and U. Kozat**. "Designing and Embedding Reliable Virtual Infrastructures." In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*. VISA '10. New Delhi, India: ACM, 2010, pp. 33–40. ISBN: 978-1-4503-0199-2. DOI: 10.1145/1851399.1851406.

[180] **W. Youn and B. Yi**. "Software and Hardware Certification of Safety-critical Avionic Systems: A Comparison Study." In: *Computer Standards & Interfaces* 36.6 (2014), pp. 889–898. ISSN: 0920-5489. DOI: 10.1016/j.csi.2014.02.005. URL: http://www.sciencedirect.com/science/article/pii/S0920548914000415.

[181] **A. J. Younge, R. Henschel, J. T. Brown, et al.** "Analysis of Virtualization Technologies for High Performance Computing Environments." In: *IEEE 4th International Conference on Cloud Computing*. July 2011, pp. 9–16. DOI: 10.1109/CLOUD.2011.29.

[182] **R. Yu, G. Xue, V. T. Kilari, and X. Zhang**. "Network Function Virtualization in the Multi-Tenant Cloud." In: *IEEE Network* 29.3 (May 2015), pp. 42–47. ISSN: 0890-8044. DOI: 10.1109/MNET.2015.7113224.

[183] **W. D. Yu**. "A Software Fault Prevention Approach in Coding and Root Cause Analysis." In: *Bell Labs Technical Journal* 3.2 (Apr. 1998), pp. 3–21. ISSN: 1089-7089. DOI: 10.1002/bltj.2101.

[184] **D. Zhang and J. P. G. Sterbenz**. "Modelling Critical Node Attacks in MANETs." In: *Self-Organizing Systems*. Ed. by W. Elmenreich, F. Dressler, and V. Loreto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 127–138. ISBN: 978-3-642-54140-7.

[185] **G. Zhang, J. Li, G. Bao, and B. Zhang**. "A New Method for Product Field Reliability Assessment Based on Accelerated Life Test." In: *11th International Conference on Reliability, Maintainability and Safety*. Oct. 2016, pp. 1–5. DOI: 10.1109/ICRMS.2016.8050035.

[186] **Y. Zhong, J. Xu, J. Zhong, and F. Liu**. "Research on Rejuvenation Analytical Models for a Virtualized System with Live VM Migration." In: *Computer Modelling & New Technologies*. 12A 18 (2014), pp. 439–445.

[187] **J. Zhou, R. Q. Hu, and Y. Qian**. "Scalable Distributed Communication Architectures to Support Advanced Metering Infrastructure in Smart Grid." In: *IEEE Transactions on Parallel and Distributed Systems* 23.9 (Sept. 2012), pp. 1632–1642. ISSN: 1045-9219. DOI: 10.1109/TPDS.2012.53.