FRIMPONG ANSAH

# PERFORMANCE AND OPTIMIZATION TECHNOLOGIES FOR SOFTWARE DEFINED INDUSTRIAL NETWORKS


# PERFORMANCE AND OPTIMIZATION TECHNOLOGIES FOR SOFTWARE DEFINED INDUSTRIAL NETWORKS

FRIMPONG ANSAH

A Thesis submitted for Doctoral Degree

Faculty of Computer Science and Mathematics
University of Passau

Frimpong Ansah
*Performance and Optimization Technologies for Software Defined Industrial Networks*


REVIEWERS:

Prof. Dr.-Ing. Hermann de Meer
Professor of Computer Networks and Computer Communications
University of Passau
Innstraße 43
94032 Passau, Germany
hermann.deMeer@uni-passau.de
http://www.net.fim.uni-passau.de

Prof. em. Dr.-Ing. Dr.-Ing.E.h Dr.h.c. Paul J. Kühn
University of Stuttgart, IKR
Pfaffenwaldring 47
D 70569 Stuttgart, Germany
paul.j.kuehn@ikr.uni-stuttgart.de
http://www.ikr.uni-stuttgart.de/ kuehn

In dedication to the loving memories of:

S. K. Ansah Darkwah

1938 – 2011

Ellen Achiaa Brefo (Maa Ellen)

1954 – 2009

Attaa Agyeiwaah (Maame Asana)

1920 – 2017

Akua Nimo

1907 – 2014

Kweku Owusu Ansah Azaria

1989 – 2015

Ike Owoahene-Acheampong (Wofa Adamu)

1961 – 2021

## ABSTRACT

The concept of programmable networks is radically changing the way communication infrastructures are designed, integrated, and operated. Currently, the topic is spearheaded by concepts such as software-defined networking, forwarding and control element separation, and network function virtualization. Notably, software-defined networking has attracted significant attention in telecommunication and data centers and thus already in some production-grade networks. Despite the prevalence of software-defined networking in these domains, industrial networks are yet to see its benefits to encourage adoption. However, the misconceptions around the concept itself, the role of virtualization, and algorithms pose a significant obstacle.

Furthermore, the desire to accommodate new services in the automation industry results in a pattern of constantly increasing complexity of industrial networks, which is compounded by the requirement to provide stringent deterministic service guarantees considering characteristically different applications and thus posing a significant challenge for management, configuration, and maintenance as existing solutions are architecturally inflexible.

Therefore, the first contribution of this thesis addresses the misconceptions around software-defined networking by providing a comparative analysis of programmable network concepts, detailing where software-defined networks compare with other concepts and how its principles can be leveraged to evolve industrial networks.

Armed with the fundamental principles of programmable networks, the second contribution identifies virtualization technologies and proposes novel algorithms to provide varied quality of service guarantees on converged time-sensitive Ethernet networks using software-defined networking concepts.

Finally, a performance analysis of a software-defined hybrid deployment solution for control and management of time-sensitive Ethernet networks that integrates proposed novel algorithms is presented as an industrial use-case that enables industrial operators to harness the full potential of time-sensitive networks.

# ACKNOWLEDGMENTS

This thesis marks the end and the beginning of another chapter of my life that would not be possible without the help and support of many other people. Therefore, I would like to express the sincerest gratitude to all who have supported me through this journey.

First and foremost, I would like to thank my doctoral supervisor, Prof. Hermann de Meer for allowing me to write this thesis. His continued guidance, insights, and encouragement have enabled me to develop as a scientist. Also, my sincerest gratitude goes to Prof. em. Paul J. Kühn for readily agreeing to serve as the external reviewer of this thesis.

Furthermore, I would like to thank Dr. Johannes Riedl and the entire Siemens industrial network research and development team[1] for their support and valuable discussions during my research.

To the members of the chair of computer and communication networks, past and present, especially fellow Ph.Ds, I say thank you for your support and the cordial atmosphere. It was a pleasure working with all of you. Special thanks go to Michael Niedermeier and Dr. Amine Abid Mohamed for their invaluable discussions. To members of the FIND project, thank you for the fruitful discussions.

Last but not the least, I would like to thank Prof. Andreas Christ for being with me at the beginning of this journey, and Dr. Antwi Nimo for his support.

1 Dr. Andreas Zirkler, Dr. Joachim Walewski, Mr. Hans-Peter Huth, Dr. Vivek Kulkani, Mr. Reinhard Frank, Dr. Matthias Scheffel, Dr. Florian Zeiger, Mr. Ermin Sackic, Mr. Volkmar Dorricht, Dr. Johannes Riedl.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

## LISTINGS



## ACRONYMS

A-CPI Application-Control Plane Interface . . . . . 22
ALEVIN Algorithms for Embedding of Virtual Networks . . . . . 95
AP Application Plane . . . . . 22
API Application Programming Interface . . . . . 32
AR Application Relation . . . . . 63
ARFE Arbitrary-Root-Fixed-Endpoints . . . . . 38
ASIC Application Specific Integrated Circuit . . . . . 28
AtF Automation Function . . . . . 75
ATM Asychronous Transfer Module . . . . . 31
AVB Audio-Video Bridging . . . . . 54
CAPEX Capital Expenditure . . . . . 10
CB Credit Based . . . . . 60
CBS Credit Based Shaper . . . . . 30
CE Control Element . . . . . 19
CNC Centralized Network Configuration . . . . . 116
CP Control Plane . . . . . 15
CPS Cyber-Physical Systems . . . . . 1
CR Communication Relation . . . . . 63
CRC Cyclic Redundancy Check . . . . . 52
CRM Centralized Reservation Model . . . . . 116
CS Communication Service . . . . . 117

CUC Central User Configuration . . . 116
DARPA Defends Advanced Research Project Agency . . . 31
DB Data Base . . . 120
DBA Demand Based Approach . . . 5
DBvLEA Demand Based virtual Link Embedding Algorithm . . . 72
D-CPI Data-Control Plane Interface . . . 22
DDoS Distributed Denial of Service . . . 149
DP Data Plane . . . 16
DRM Decentralized Reservation Model . . . 115
e2e End-to-End . . . 2
EDF Earliest Deadline First . . . 21
EPT Engineering and Planing Tool . . . 16
ES End-Station . . . 81
FE Forwarding Element . . . 19
FIB Forwarding Information Base . . . 118
ForCES Forwarding and Control Element Separation . . . 2
FP Forwarding Plane . . . 15
FSFD Fixed-Source-Fixed-Destination . . . 38
GA Greedy Approach . . . 70
GCL Gate Control List . . . 78
HRM Hybrid Reservation Model . . . 6
ICN Industrial Control Networks . . . 13
IETF Internet Engineering Task Force . . . 31
ILP Integer Linear Programming . . . 43
IoT Internet of Things . . . 1
IT Information Technology . . . 1
JMT Java Modelling Tools . . . 133
LAN Local Area Network . . . 2
LCM Least Common Multiple . . . 80
LDP Label Distribution Protocols . . . 19
LLDP Link Layer Discovery Protocols . . . 19
LRB Listener Root Bridge . . . 77
LRP Link Registration Protocol . . . 19
M2M Machine-to-Machine . . . 1
MAC Media Access Control . . . 114
MIB Management Information Base . . . 86
MIP Mixed Integer Programming . . . 43
MMRP Multiple MAC Registration Protocol . . . 114
MP Managment Plane . . . 16
MRP Multiple Stream Registration Protocol . . . 21
MSE Mean Squared Error . . . 140
MSRP Multiple Stream Registration/Reservation Protocol . . . 19
MVRP Multiple VLAN Registration Protocol . . . 114
NBI North Bound Interface . . . 22
NC Network Calculus . . . 45
NE Network Element . . . 15

| | | |
|---|---|---|
| TE | Traffic Engineering | 44 |
| TLV | Type Length Value | 80 |
| TRB | Talker Root Bridge | 77 |
| TSN | Time-Sensitive Networking | 2 |
| TT | Time-Triggered | 56 |
| ULL | Utra-Low latency | 54 |
| UNI | User Network Interface | 117 |
| VLAN | Virtual Local Area Network | 52 |
| VLiM | Virtual Link Mapping | 5 |
| VN | Virtual Network | 4 |
| VNE | Virtual Network Embedding | 2 |
| VNoM | Virtual Node Mapping | 36 |
| VNR | Virtual Network Request | 4 |

# 1 INTRODUCTION

Despite the numerous research on industrial network architectures, adaptation of networks to changing requirements or constraints imposed by the emergence of technologies such as Machine-to-Machine (M2M) communication, Internet of Things (IoT), Haptic communication, and tactile internet, etc., still remains elusive.

There has been a lot of research on the adaptation of existing network systems to new functionalities such as demands for audiovisual in e. g., automotive industry, coupled with ever-changing customer requirements of industrial operators such as the remote operation of machinery, actuators and long-distance control of Cyber-Physical Systems (CPS) with deterministic service requirements.

These past and current trends provide sufficient data to support the assertion that the environment of networks (application and customer requirements) will continue to exhibit similar trends in the future. Regardless of these trends, current industrial networks are continually built on traditional architectural principles which introduce significant challenges w.r.t the ability and capacity to evolve with their changing environments and thus, the continuous development of new communication solutions that are often optimized for specific use-cases.

Moreover, with the emergence of new technologies such as IoT and edge computing, industrial network owners and operators want to take advantage of these new technologies to expand their portfolio of services to stay competitive. These new services can be categorized as either Information Technology (IT) or Operational Technology (OT) based.

For decades, industrial IT and OT networks have existed on different infrastructures [1]. The desire to serve these two contrasting areas of applications with a common network infrastructure has also increased with the introduction of time-sensitive Ethernet bridges purported to support any type of service. The new opportunities seen by the realization of IT and OT convergence on Ethernet is gradually coming to fruition and will enable industrial owners to cut down cost. However, there are still a lot of lingering questions around how these networks can be deployed and managed. Among the lingering questions are the quest for flexible resource reservation solutions, Quality of Service (QoS), effortless service provisioning, integration with brown- and green-field systems (e. g., Fieldbus and 5G systems), efficient and flexible deployments.

To address these issues, network programmability has been put forth by the research community as a way to ameliorate the current

blockade to industrial network evolution [2]. Although network programmability concepts and principles have seen significant adoption in domains such as telecommunication e. g., 5G, and data centers, industrial networks (e. g., factory automation) are yet to consider the possibility of adopting programmable networking concepts.

The transition to programmable networks is, however, non-trivial as recent accounts of programmable network concepts such as Software-Defined Networking (SDN) has become too slick and thus tailored to Service Provider (SP) networks where requirements are contrary to traditional expectations in the industrial domain. In the industrial domain, high-precision communication is the primary goal of network services. The capacity of programmable networking frameworks to provide the same level of high-precision delivery of packets as those ensured by legacy networks systems such as field-bus networks is also an open research question.

This thesis addresses these challenges, by giving a structured analysis of programmable network architectural frameworks such as SDN and Forwarding and Control Element Separation (ForCES) considering industrial requirements. As part of this, concepts and methodologies that can be applied in End-to-End (e2e) service QoS guarantees such as capacity and latency within programmable industrial networks will be identified, examined and analyzed.

Bridging network programmability concepts in SDN with enabling concepts/technologies such as network slicing and Virtual Network Embedding (VNE) in the industrial context, Software-Defined Industrial Networking (SDIN) is presented as a complementary albeit an alternative solution to the recently introduced stream-reservation solutions in Time-Sensitive Networking (TSN) ( see IEEE 802.1Qcc [3], also discussed in Chapter 7).

The viability of the proposed solution is demonstrated by way of design of novel resource orchestration algorithms, qualitative and quantitative performance analysis of how SDN can introduce significant flexibility as well as enable the integration of IT and OT requirements on a common bridge Local Area Network (LAN) infrastructure.

## 1.1 THESIS CONTRIBUTION

This thesis proposes the use of programmable network concepts based on SDN to address the challenges of industrial networks on control and management of IT and OT requirements. The contributions are summarized in three parts:— network programmability, QoS provisioning, and deployment architectures.

### 1.1.1 *Network programmability*

First, this thesis examines programmable network concepts and why they are beneficial for evolution of current industrial networks.
The main goal of the contribution is to provide a detailed account of some original inception of the concept, current state, and enabling innovations such as virtualization and how it can be leveraged for production-grade industrial networks. This includes review, analysis and comparison of architectural principles of the ForCES and SDN architectures.

The methodology used in addressing this contribution is one that is perhaps better summarized by a famous quote from Peter Sarnak in his advice to young researchers, captured in the Princeton Companion to Mathematics (by T. Gowers et al. [4]) under final perspective as:

> *" When learning an area, one should combine reading modern treatments with a study of the original papers, especially papers by the masters of the subject. One of the troubles with recent accounts of certain topics is that they can become too slick. As each new author finds cleverer proofs or treatments of a theory, the treatment evolves toward the one that contains the "shortest proofs." Unfortunately, these are often in a form that causes the new student to ponder, "How did anyone think of this?" By going back to the original sources one can usually see the subject evolving naturally and understand how it has reached its modern form."*

The concept of network programmability has evolved from several schools of thought since its inception. To understand programmable networks as known today with SDN, it is worth reviewing the fundamental or original account of seemingly similar concepts.

For example, in most educational material on communication and networks, to understand Transport Connection Protocol/Internet Protocol (TCP/IP), researchers reference the Open System Interconnect (OSI) model which gives an original account of layered communication. The trend within the networking research community is one that hinges on timing among all things important. For equally good concepts or solutions to gain the traction required to foster adoption, timing of release and enabling technologies are often important factors to consider. Like the OSI model and TCP/IP, the race for adoption of programmable networking concept is one that is currently between ForCES and SDN, with ForCES playing the role of OSI model while SDN takes the limelight like TCP/IP.

### 1.1.2 *QoS guarantee in SDIN*

Research innovation in programmable networks introduce a paradigm shift from traditional network management [5]. The new paradigm focuses on the synergy of network abstraction and virtualization, Open service interfaces, integration of computational and communication models [6] in what is termed intelligent service automation. This paradigm introduces new challenges such as service creation and QoS guarantees e. g., bandwidth and latency, mainly considering services with contrasting requirements on the same infrastructure.

The second contribution of the thesis proposes novel concepts and algorithms for resource orchestration for bandwidth and delay guarantees within the SDIN framework. This is achieved firstly by identifying and leveraging state-of-the-art concepts and methodologies like Network Slice (NS) and Virtual Network Embedding (VNE) as service creation and QoS building blocks within the SDN paradigm. Important highlights of this contribution are listed in the following:

MATHEMATICAL MODELS FOR DELAY GUARANTEE IN TSN Resource allocation in SDN-enabled infrastructures begins at the abstraction and virtualization layer. At this point, the applications' requirements are abstracted and represented as a Virtual Network Request (VNR). A VNR is instantiated as Virtual Network (VN) on a substrate infrastructure using VNE algorithms in what is term virtualization.

To guarantee QoS requirements such as bandwidth and latency, the VNR must be mapped to the logical resource and tested against particular service demands such as capacity, delay, reliability, etc., before it is committed to actual resources in the physical network. This consists of three main processes; the request modeling, infrastructure modeling, and resource mapping.

To ensure service guarantees in SDN-enabled industrial infrastructures, this thesis shows how formal mathematical models can be developed using a Trajectory Approach (TA) and integrated in Virtual Network Embedding (VNE) problem as components of Service Function Construct (SFC) to ensure deterministic guarantees in practical deployments.

ACCURACY AND EFFICIENCY OF VNE ALGORITHMS Within the SDN paradigm, simulation of the VNE problem is often confused with actual deployments of VNE algorithms. While the accuracy of solutions produced by VNE algorithms depend on the models of the system under study, the efficiency of the VNE algorithms depend on the efficacy of the algorithm to follow an optimization goal. As part of this contribution, this thesis shows how VNE algorithms can be designed and integrated as components of SFCs to ensure efficient use

of network resources within SDIN. This includes the design of novel Demand Based Approach (DBA) to Virtual Link Mapping (VLiM), topology-aware mapping and time-aware resource allocation for periodic time-sensitive applications.

### 1.1.3 *Deployment architectures*

The emergence of TSN standards has seen significant strides towards its adoption at every level of industrial communication networks. One of the leading research challenges is finding a deployment model that enables industrial network operators to reap the benefits of all proposed features.

The final contribution of this thesis proposes and examines an SDIN-enabled TSN hybrid deployment that integrates the concepts and algorithms discussed in the previous contributions. A comparative performance analysis of the proposed hybrid SDIN-enabled, and fully decentralized TSN deployments are examined, developed, and analyzed considering response time for resource allocation mechanisms. The quantitative analysis leverages a combination of open Queuing Network (QN) and the expressiveness of deterministic Petri Net (PN) formalism to estimate the response time of SDIN-enabled TSN control mechanisms for resource allocation.

## 1.2 STRUCTURE OF THESIS

At a glance, Figure 1.1 illustrates the outline of this thesis whilst the following describe how the thesis is organized:

Chapter 2— **Industrial Concerns and Network Programmability**
Examines requirements and challenges of industrial networks operators. With a clear understanding of challenges of current industrial network solutions, it proposes network programmability as a key to addressing these concerns. However, with the existence of several programmable network concepts and varied interpretations, it is essential to clarify the subject from the view of industrial operators. To do so, a detailed analysis of the concepts from ForCES and SDN perspective is provided. Through the analysis, key elements are identified and used to derive Software-Defined Industrial Networking (SDIN).

Chapter 3— **SDIN: Service Provisioning Enablers**
After identifying the challenges and concerns in Chapter 2, Chapter 3 discusses how SDIN can be used to provide service guarantees for industrial applications on a common network infrastructure using NS concept. To do so, the chapter describes how existing methodologies such as VNE and delay estimation methodologies can be leveraged to

create slices.

Chapter 4— **Modeling Delay in TSN-Enabled Infrastructure**
From the analysis of VNE in Chapter 3, it is revealed that QoS guarantees within the VNE problem lies primarily with the underlying infrastructure. Also, to leverage SDN for industrial communication, an SDN compatible underlay network is required. On that basis, a TSN-enabled infrastructure is adopted as a use-case for an SDN-enable industrial network where the underlying Quality of Service (QoS) features of the network are analyzed to develop delay computation models for the VNE problem.

Chapter 5— **Designing VNE Algorithms for SDIN-enabled TSN**
The analysis of VNE problem from Chapter 3 also reveals that aside from the need to ensure QoS which depends on the features of Substrate Network (SN), efficient resource usage depends on the underlying orchestration hypothesis used by a VNE algorithm designer. This chapter proposes the design of novel VNE algorithms within SDN context and provides a contrast to the orchestration approach used in literature.

Chapter 6— **Evaluation of VNE Algorithms in SDIN**
Chapter 6 evaluates the proposed VNE algorithms and constraint computation models developed in Chapters 4 and 5 to assert their efficiency and efficacy of underlying hypothesis as well as accuracy. It also describes evaluation guidelines on how VNE algorithms used in SDN must be evaluated for production-ready deployment.

Chapter 7— **Analysis of SDIN-enabled TSN Deployment**
One of the significant challenges identified in Chapter 2 was the lack of flexibility in existing industrial solutions, to showcase the flexibility and benefits of an SDN-enabled industrial network, Chapter 7 examines and proposes an SDIN-enabled Hybrid Reservation Model (HRM) as an alternative albeit a complementary solution that allows industrial operators to utilize the full potential of Time-Sensitive Networking (TSN). It further backs the proposed concept with a qualitative and quantitative performance analysis showcasing the advantages of an SDIN operated TSN infrastructure.

Chapter 8—**Conclusion**
Chapter 8 summarizes the main contributions of the thesis, a discussion of the lessons learned, and the impact of the results. It concludes with an outlook on future research directions.

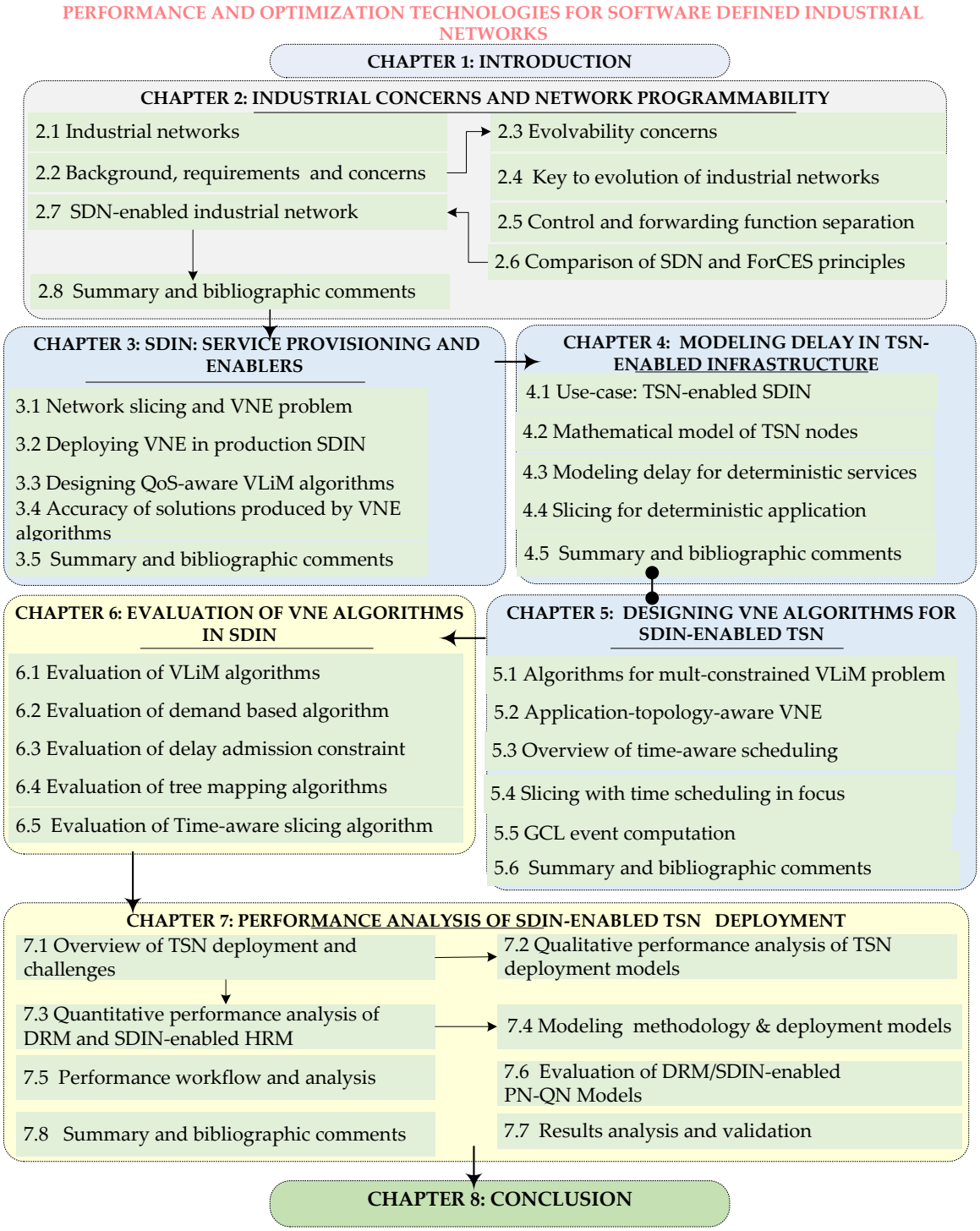



Figure 1.1: Structure of thesis at a glance

# 2

# INDUSTRIAL CONCERNS AND NETWORK PROGRAMMABILITY

The fundamental task in every communication network is to exchange data between different devices or participants. This exchange can be via direct Point-to-Point (P2P) line connections between two participants or over a network systems or devices [7]. Practically, it may require more than two participants to exchange data; also, the data exchange can span vast geographical areas. As an example, let us examine a real world communication scenario that entails a direct exchange of information between participant A and B.

Participant A says something, B listens, and performs an action. The action can be a simple response from participant B in acknowledgment of message received or pick an object, change position due to impending danger, etc. The urgency ascribed to the exchanged messages depends on the actions at hand.

In the case of an eminent danger, the message must get to participant B to enable a quick action in order to avoid a catastrophic situation. In human conversation, the urgency attached to such messages may be demonstrated by a high-pitched tone, however, in machines, it is characterized by speed.

Now, let us also consider a case where participant A and B are at least a kilometer apart, the message exchange might not be possible over this distance. Even if possible, participant B may not understand or comprehend the vociferous nature of the message to act quickly as a result of pitch distortion due to the distance between them. Communication networks are well suited to handle this kind of communication.

Over the years, communication networks have evolved from simple serial data transmission to more complex systems that perform different forms of data exchange (e.g., the internet). Despite the sole purpose of a communication network being data exchange, no two communication networks are the same. This problem is because data exchange varies in complexity and sometimes are very contradictory as illustrated in the above example. The contradictions lead to different ways of handling data exchange over communication networks and hence different solutions developed specifically to suit different applications or domains.

The authors in [7] provide extensive characterization and classification of networks based on attributes such as geographical coverage, mode of interaction of network participants or devices, type of service or applications' requirements. The authors further discuss the rationale behind the plethora of network solutions today and why such

contrasting differences exist as one moves from telecommunications and data-centers to the industrial domain. This thesis will use their characterization on industrial network to describe industrial network operators' concerns and requirements, showing how the application of programmable network concepts such as SDN is beneficial for the evolution of industrial networks.

## 2.1 INDUSTRIAL NETWORKS

In the industrial domain, the application (use-case) dictates communication networks, i.e., be it in manufacturing, automotive, process and factory automation, the task of data exchange vary in complexity with very contradictory requirements [7]. Networks can be required to exchange data for control of drives, motors, braking systems as in cars or trains, video display, and audio packets from a microphone to speakers.

By applying the example of data exchange between participant A and B to industrial networks, it can be realized that the requirements on data exchange are not inherently the same. This makes it *almost* impossible to optimally address all applications' needs with single or common communication infrastructure.

*customized solutions enable optimum performance*

In order to optimally address the multiple requirements, different solutions are required for each domain of application [7]. However, though optimum from applications or performance perspective, this approach often leads to many solutions. Ultimately, with a plethora of solutions for different applications, the rate of growth and evolution of industrial networks in the context of integrating emerging applications such IoT, M2M, Haptic communication [8, 9], etc., become slower as existing tailored network solutions are unable to support these emerging requirements. That is, every new use-case or application requires a customized network solution to accommodate them. This approach does not scale well and often leads to substantial long-term Capital Expenditure (CAPEX) due to device purchases and Operational Expenses (OPEX) which leads to cost[1] of performing network management tasks (e. g., engineering, configuration, monitoring, maintenance).

To address this concern, industrial networks need to evolve towards more flexible and adaptable network systems capable of accommodating different requirements. However, these goals are not achievable with traditional industrial network architectures (e. g., Field-bus) as these network systems are architecturally inflexible to accommodate new requirements.

Therefore, this section of the thesis examines communication in the industrial network domain to identify challenges and requirements, as well as propose solutions.

1 cost here refers to the time and effort needed to plan, engineer, and configure NEs

The subsequent sections of this thesis will briefly discuss and examine the different aspects and requirements of industrial networks and identify concerns from the perspective of different stakeholders such as the industrial network owner, operators and engineers. Subsequently, it will highlight the critical industrial concerns that are addressed in this thesis.

Finally, the chapter will conclude with architectural concepts that show great potential to tackle the framed concerns in order to achieve flexible and convergent industrial network systems. This will include a thorough analysis of two competing architectural concepts for programmable networks and their industrial implications. As a result, we will compare and contrast the two architectures to highlight where such ideas have been considered.

## 2.2 BACKGROUND, REQUIREMENTS AND CONCERNS

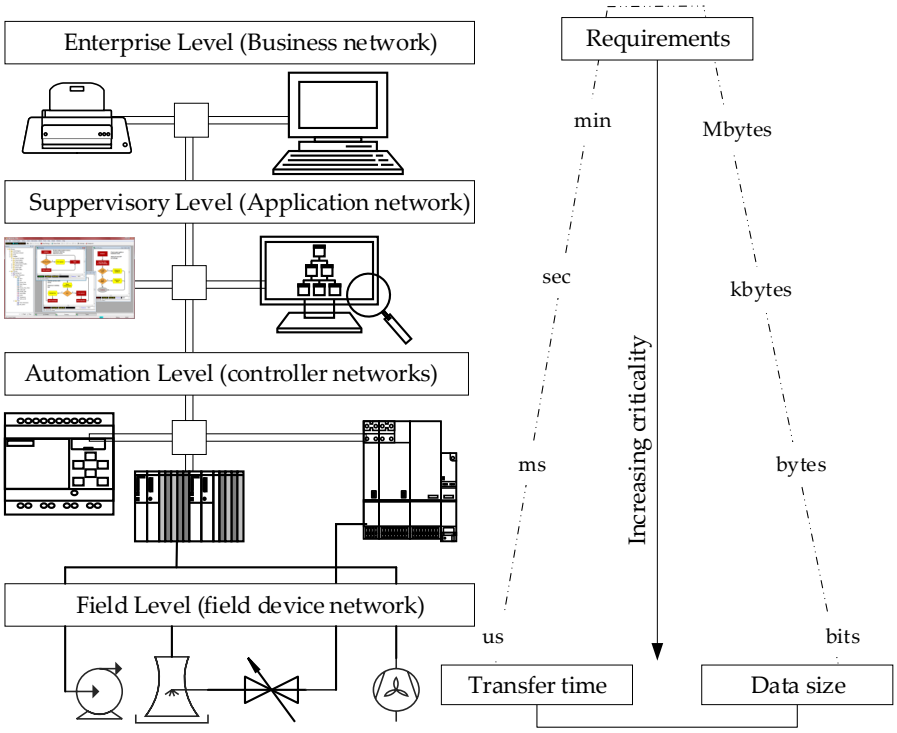



Figure 2.1: Hierarchical levels of industrial communication systems [7]

The topics of communication networks are generally very complicated especially in the industrial domain. This is because of the several aspects to industrial communication. Often, it is very difficult to make clear distinctions between industrial communication networks based alone on coverage or whether circuit or packet switching is required.

Rather, a much practice-oriented classification is achieved by assign-

ing communication requirements to different hierarchical or applications levels (as examined in [7]). These levels form what is termed the industrial communication pyramid.

Figure 2.1 shows the main application levels illustrating the data size and order of time required for data exchange in the respective levels. From the Enterprise to Supervisory/Automation levels, large volumes of data in the Giga/Mega/Kilobytes orders are required to be transported over the network often without time bounds. In a situation where time bounds are required, the response time for data transfer can be in the order of minutes if not seconds. Essentially, the safe transfer of huge data is more important than the duration of transfer of the data [7].

The supervisory level is responsible for the supervision of an entire factory process which includes several automation processes. Each of these factory processes may require different networks tailored specifically to the application's requirements.

The Automation level controls the lower part of the pyramid[1, 7] referred to as the Field level. On the Supervisory and Automation levels, data size ranges from a few kilo-bytes to an average of 10 to 500 bytes respectively with delays between 5 to 20 ms [7].

The lower part of the pyramid shows the Field levels. In these levels data exchange is in the orders of bytes to bits. At the Field level, delay requirements are in the orders of micro to a few milliseconds. With communication between the automation and field levels often in a few millisecond duration. Most important is the fact that all requirements on the data size and delay have strict deterministic bounds as shown in the lower part of the pyramid.

The reason for the huge differences lies in the different goals of communication. On the one hand, communication in the automation and field levels is addressed from the monitoring and control of physical systems (e. g., valves, sensors, robots, actuators, motors.) which require real-time operations. On the contrary, the Enterprise and Supervisory levels behave more like conventional IT, data-center and telecommunication networks where real-time requirements are not very stringent compared to the lower parts of the pyramid. That is, in networks at the base of the pyramid, although packet losses or delayed packets may lead to catastrophic scenarios such as endangering human life or crashing of a production line, it does not compare to a late delivery or arrival of an email or a chat message on a social platform such as Whatsapp.

It is difficult to find a common denominator for networks designed for the different levels as solutions and terminologies differ from company to company within the industrial domain. However, they are often captured under Information Technology (IT) and Operational Technology (OT).

IT subsumes all industrial systems that employ enterprise appli-

cations, information processing and business process communication. OT subsumes all systems used in the industrial environment for the sole purpose of monitoring and control of Cyber-Physical Systems (CPS). Communication networks designed for carrying monitoring and control data of physical devices are referred to as Industrial Control Networks (ICN).

At their core, ICNs are entirely different from conventional IT networks in several aspects such as functionality, architecture and even ways of management. They are often deployed in some form in any environment that requires machine monitoring and control. Often, the communication infrastructures differ based on the specific requirements of the individual sectors such as manufacturing, transportation, chemical refinery, power generation, food and beverage processing, etc (cf. [1]).

In recent times, the difference in ICNs compared to commercial networks fades as a result of current integration of Ethernet [1]. However, as discussed by Hancke and Galloway [1], the recent advancements of Ethernet do not necessarily make the functionalities of the industrial networks and conventional IT networks the same. In fact, the core functionality which is monitoring and control of CPS remains the same. The only change comes with an improvement in some underlying infrastructures such as Ethernet, previously used in the IT sector to accommodate as well OT related data exchange on a single infrastructure. This brings up the topic of convergence of industrial networks (i.e. IT and OT), new mechanisms for control and management, algorithms and enablers for QoS performance guarantees in such networks.

Before addressing the various industrial concerns, it is best to give a formal definition of what is referred to as a concern.

**Definition 2.2.1.** Concern (co): A concern identifies a problem or desire that needs to be addressed w.r.t to a system or application.

A major challenge envisaged with the influx of emerging technologies and services created around them is whether industrial networks can be adaptable to rapid changes in requirements. This is summarized in what is termed evolvability concerns:

**Definition 2.2.2.** Evolvability concern address the ability or capacity of a network to adapt itself to requirements imposed on it by applications that use the service it provides.

The subsequent section examines the evolvability concern within industrial networks.

## 2.3 EVOLVABILITY CONCERNS

With consideration for IT and OT convergence [2], future industrial networks are still required to address the following concerns:

- co.1 Same QoS guarantees as those obtained on tailored solutions such as Field-buses e. g. PROFIBUS, CANopen, SERCOS, etc.
- co.2 Accommodate deterministic as well as stochastic requirements on the same infrastructure.
- co.3 Accommodate new as well as currently undiscovered services.
- co.4 Enable flexible management and control of different forwarding mechanisms.
- co.5 Reduce initial and long term capital investment by enabling the integration of new services with minimum required changes.
- co.6 Integration with traditional (brown-field) and non-traditional (green-field) industrial networks e. g. 5G.

These concerns can be framed under adaptability and/or evolvability concerns. Adaptability refers to industrial communication networks' ability or capacity to support green- and brown-field use-cases often during the operation period of the system. Green-field use-cases include currently untapped systems or applications while brown-field covers legacy systems or applications.

Adaptability is in fact a concern that deals with operation and service provisioning, often considered from a network operator/service provider or engineer's perspective. On the other hand, network owners or operators turn to look at these concerns from a business point of view which mostly centers on the longevity of their assets and the capacity of these systems to continually provide a competitive edge over time. In this context, a more befitting functional term will be evolvability.

Evolvability describes the capacity of a system to adapt to a changing environment. To put it in a better context, if we consider the requirements of applications or use-cases as the environment within which a network must exist, then the ability of the network to change in order to meet the requirements imposed by the applications can be termed as evolvability.

Mostly, it entails designing systems that provide the optimum benefits in terms of projected CAPEX over a period of time. This concern is illustrated in an article by Dovrollis et al. [10], where network evolvability concern is analyzed from a biological, engineering and business perspective.

2 Convergence refers to applications with different bandwidth and delay requirements sharing the same network infrastructure.

After discussing the focus of industrial networks w.r.t IT and OT convergence and the concerns to be addressed, now we identify specific technologies and architectural concepts that can ameliorate these concerns but first, let us examine the fundamental constituent of a network from a functional point of view. The functional view gives a better appreciation of how network architectures have been defined over the last decades and why concerns identified in this thesis must first be tackled from an architectural perspective in order to gain the requisite understanding for the development of functions and algorithms that addressed these concerns.

## 2.4 KEY TO EVOLUTION OF INDUSTRIAL NETWORKS

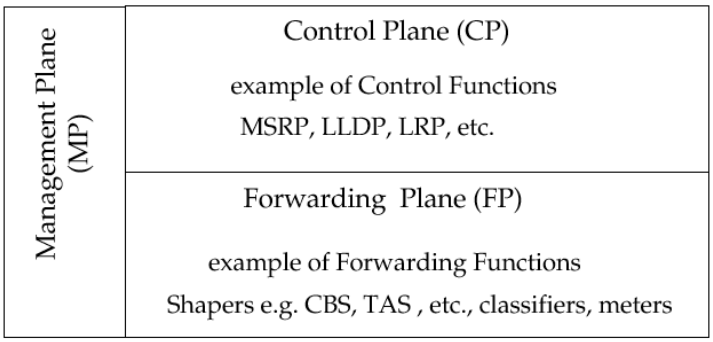



Figure 2.2: Internal view of Network Element architecture

A network is composed of connected devices (Network Element (NE)) e. g., switches/bridges, routers, etc. These NEs appear from an end-user perspective as single monolithic entities, however, a microscopic view into the internal structure of NEs reveals that they are composed of logical entities that corporate together to provide common functionalities e. g., moving a packet of data from one end of a switch to the other. These logical entities can be categorized into two sets of components namely; the control and forwarding components.

Architecturally, these components are classified by planes i. e., the Control, Forwarding (sometimes used interchangeably with data) and Management planes. Figure 2.2 provides basic architecture of the internal view of an NE. As shown, the control components are assigned architecturally to the Control Plane (CP) whilst the forwarding components are assigned to the Forwarding Plane (FP).

The CP handles all the intelligence displayed by an NE. Intelligence in this context refers to the algorithms or control protocols used for example signaling, path computation and resource reservation, address resolution, authentication, registration, collection of statistical information, etc. In effect, the CP functions decide the behavior of the FP, however, the CP functions are defined based on the capabilities of the FP. The FP handles the actual movement of data or packet from

one end of the network or NE to the other, often referred to as the Data Plane (DP) or muscle of the network. Examples of the functionalities that can typically be found in this plane are traffic shapers and/or schedulers, meters, classifiers, etc. The Managment Plane (MP) as implied by its name, allows management of the CP and FP.

Generally, the three different planes are combined in one physical system or NE. This presupposes that the intelligence of the network elements is developed with strict adherence to the mechanisms available in the FP hence, forms a tightly coupled system.

There are situations where due to the restrictions of the physical device capabilities (i. e., hardware capabilities such as processors, ASICs, and memory), the level of intelligence is limited to exactly what is required for a specific type of application. Thus, it hinders the ability to adapt the network to new requirements without changing the entire NEs.

In legacy industrial networks like Fieldbuses, limitation in computationally involving functions such as bootstrapping and service creation are usually compensated for by an Engineering and Planing Tool (EPT). An EPT allows network engineers to perform the most computationally intensive functions in an offline system and integrate the offline solutions into the control plane via interfaces provided by the MP.

Sometimes, even when this is possible, restrictions of the FP may not allow new applications or requirements. This paradigm of networking introduces a 1-dimensional system that poses the following challenges to the concerns ( co.3 → co.6) defined in subsection 2.3.

ch.1 Services for new applications (co.3) cannot be accommodated in the legacy networks because their architectural designs are tightly coupled to application-specific requirements. Hence, they fail for new applications. To enable the network for new applications that do not share similar characteristics, the entire NE will have to be replaced with new ones that have the requisite intelligence to support the said service. This, of course, does not do well for long term CAPEX, co.5. It can also take a very long time through lengthy standardization processes to enable such features even when capital is not an issue.

ch.2 A flexible control and management system may not be possible as several solutions may impose customized management and control functions, co.4. For example, CANopen and TSN [11].

ch.3 Integration with other network domains and technology becomes increasingly difficult and inflexible because of the plethora solutions available, co.6.

Considering the challenges identified, one simple solution is developing new technologies but it does not solve the problem entirely. In

fact, the challenge with current network systems lies deep within the fundamental architecture of network systems as illustrated. To enable networks to evolve to accommodate different services, they need to be adaptable.

Forwarding and control function separation, is a concept that enables networks to be adaptable. By allowing the control and forwarding planes to evolve separate of each other, one can develop different forwarding functions to handle different types of services. The forwarding functions can include general as well as specialized functions and can be put together in several combinations to fulfill different service requirements and thus making the NE programmable [12].

The subsequent section examines the basic principles of programmable networks.

### 2.4.1 *Basic principles of programmable networks*

Network programmability according to Galis et al. [6] is the next step of network evolution. The primary goal of this evolution is to enable rapid deployment and customization of services, which can be achieved via a two-dimensional model that allows the integration of computational and communications models.

This two-dimensional model describes the interaction of components of communication models such as packet header processing and forwarding (QoS), and computational models such as programming languages, operating systems, algorithms, etc.

Though the idea of programmable networks is not a new one, recent accounts and implementations has led to frameworks that address specific computer network domains. For example, SDN can be considered as a service-defined network which is more tailored to telecommunication and data-center service providers. However, a fundamental principle that cuts across all frameworks is the concept of " separation of control and forwarding plane".

The subsequent sections discuss the requirements of forwarding and control separation, notable architectural concepts, their key strength and differences and how they can be leveraged to address the industrial concerns discussed in section 2.3.

## 2.5 CONTROL AND FORWARDING FUNCTION SEPARATION

Figure 2.3 illustrates the basic principle for control and forwarding function separation. As shown, it is a slight derivative of Figure 2.2 with the exception that the different planes are separated by the introduction of a protocol (control messages) between the respective planes. This separation enables the development and combination of different control and forwarding functions. As a result, it enables a

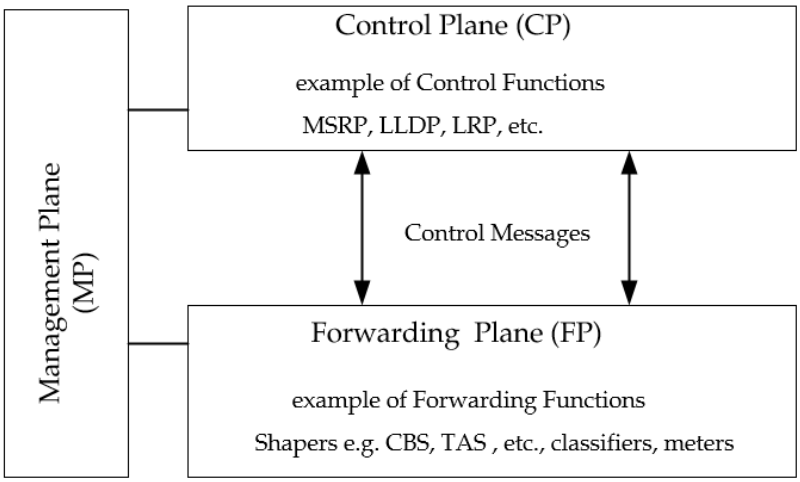



Figure 2.3: Proposed NE function separation enabling the evolvability of networks to support varied services.

network operator to readily adapt the network to different service requirements by combining different network functions.

Two notable architectural concepts that proposes this radical approach to networking are the ForCES and SDN architectures. The key differences in the two architectures are discussed next.

### 2.5.1 *Principles of ForCES*

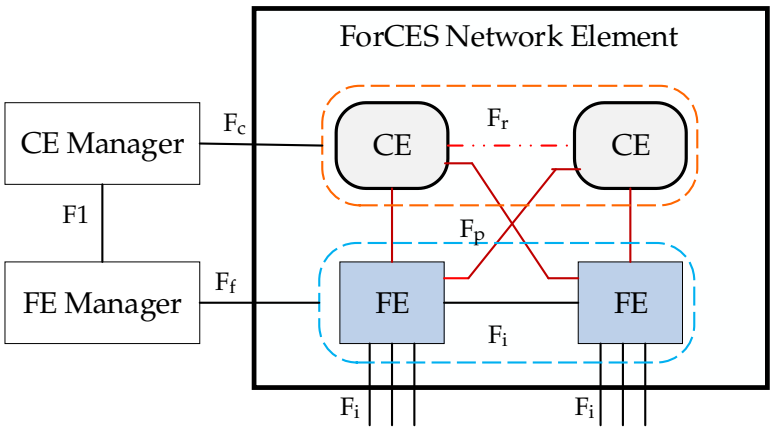



Figure 2.4: Forwarding and control function separation: ForCES Architecture [6]

ForCES is one of the fore notable architectural concepts in today's networking paradigm. Fundamentally, it proposes the separation of the Control and Forwarding plane of NEs to enable them evolve independent of each other.

To illustrate the fundamental principles of the ForCES architectural

concept, Figure 2.4 provides a ForCES representation of an NE. According to the architectural representation, the NE consists of two sets of elements;— CEs and FEs. Architecturally, the Control Element (CE) operates in the CP while the CE operates in the FP.

The CE consist of network control functions that perform tasks such as routing and signaling. Example of signaling protocols that can be implemented in this block are;—Label Distribution Protocols (LDP), Link Layer Discovery Protocols (LLDP), Link Registration Protocol (LRP), and Resource Reservation Protocols (RSVP) such as Resource Allocation Protocol (RAP), Multiple Stream Registration/Reservation Protocol (MSRP), etc.

The FEs perform packet operations such as metering, shaping/scheduling and classification. Figure 2.4 also shows the relationship between the logical components where the CE and Forwarding Element (FE) can be interconnected in every possible combination, i. e., CE-FE, CE-CE, FE-FE. Each possible combination defines a reference point which can consist of a collection of protocols that enables the transfer of messages between them.

The Fp reference point implements the ForCES protocol which is intended to facilitate the CE with the ability to control the behavior of the FE via abstraction of FE capabilities. This abstraction allows network operators to define CE functions that adapt the FE to different application requirements.

As a requirement of FE within the ForCES architecture, FEs can manifest varying functionalities which implies, the CE can only make minimal assumptions w.r.t the capabilities of FEs [6]. This in fact posses a major problem for CE function definition. That is, in order for the CE to effectively control an FE, the CE must have a vivid understanding of how an FE processes and forwards packets. With this knowledge, the CE can dynamically control how packets are processed by the FE even at network run-time. This, therefore, makes the network highly programmable, hence adaptable to different uses.

The detailed description of the ForCES architecture ( [12]), requirements on CEs, FEs ([13]) and Fp are defined in RFC 3746, 3654 and 3756 respectively.

The following are some key feature requirements proposed by the ForCES concept in RFC 3654 [13]:

1. The CE and FE must be able to connect by a variety of technologies ( RFC 3654 [13]). This implies any protocol or mechanism that conforms to the requirements of ForCES Fp can be used. It allows for multiple of such protocols to be supported by a ForCES NE.

2 The FE must support a minimal set of capabilities; however, the number of capabilities of the FEs are not restricted.

3. Packets must arrive at an NE via one FE and leave the NE via different FEs.

4. An FE must asynchronously inform the CE of a failure or increase/decrease in available resources or capabilities on the FE due to usage. That is, the FE should be able to give statistical information on its resource usage and events. This enables the CE to monitor and adjust the FE behavior with a specific application when necessary.

5. The combination of CEs and FEs within an NE, must appear as a single device to an end-user.

6. FEs must have the ability to redirect packets addressed to their interface to the CE.

7 The CE must be able to learn the topology by which FEs are connected within an NE and change the topology when necessary.

8. The ForCES architecture must allow FEs and CEs to join and leave NE dynamically.

9. The CE should be capable of off-loading certain functions onto the FE

RFC 3654 provides extensive details of these requirements, features stated here are important for the subsequent discussions. For further readings, refer to [13]. Other requirements of the ForCES elements and protocols are also summarized in Appendix A.1.

PROGRAMMABILITY IN THE CONTEXT OF FORCES The ForCES architecture supports three levels of programmability. The first level of programmability addresses the control and configuration of statically defined FEs. This covers a scenario where the structure of atomic functions such as classifiers, shapers or schedulers that constitute the FE functional block remain fixed. This includes a fixed topology of the FE functions.

At this level of programmability, a network operator or engineer can define the constituent CEs deterministically. That is, CE function can be changed or augmented; however, the set of FE functions and their topology must remain the same for the life-cycle of an NE.

The second level of programmability addresses a scenario where the CE can discover the capabilities of the FE as well as change the topology of the FE functions. This implies, the CE can choose from a pool of FE functions, define how the functions are chained (topology) in order to meet the desired way of handling packets.

To put this into perspective, consider a scenario where FE constitutes three different scheduling functions e. g., priority scheduler,

Earliest Deadline First (EDF) and Time-Aware Scheduler (TAS). From a deployment point of view, an NE may implement only one of the above functions which will dictate the kind of application the NE can support. However, what the second level ForCES programmability proposes is that, the CE discovers the fact that the FE functional block has the three different scheduling capabilities and chooses which capability should be included in the topology of the FE block to support the application. One important thing to note here is that, FE must constitute a fixed pool of forwarding functions.

The third and final level of ForCES programmability is similar to the second level but with an added functionality that enables the CE to download new FE functional blocks into an NE. This implies, the CE can remove and add new FE functional blocks, define new FE topology during the life-cycle of the NE or even at run-time.

### 2.5.2 *Principles of SDN*

From a generic viewpoint, the SDN concept is similar to or an instance the first level of ForCES programmability. Conceptually, SDN follows the same line of thought considered in the ForCES concepts. However, the SDN architecture emphasizes control and forwarding plane separation where connectivity between the CP and the FP is governed by open communication interfaces. More so, it advocates for physical instantiation of the CP element (CE) as a centralized entity outside the NE.

The SDN concept allows all NEs within a network to share a centralized intelligence by providing open interfaces to enable the development of software that can control the connectivity of a network of resources and the flow of traffic through the resources.

Even though the SDN concept advocates for control plane separation and centralized instantiation, some aspect of control inevitably resides within the DP which is referred to as NE in the SDN terminology [14]. That is, even though the centralized controller has oversight of forwarding resources, it allows for the possibility of delegating control functions, e. g., LLDP, Multiple Stream Registration Protocol (MRP) to the DP which it refers to as the network infrastructure or NEs.

Figure 2.5 shows the basic architectural components and terminologies as described in [14]. Within the SDN concept, the DP also known as the FE in the ForCES architecture, consist of a set of one or more NEs, each of which constitutes a set of forwarding functions or traffic processing functions. DP resources are abstract representations of the capabilities of the traffic forwarding functions.

The CP comprises the SDN controller(s) which have exclusive control over the set of resources exposed by the DP (NEs). According to the Open Networking Foundation (ONF) [14], a common but non-essential function of an SDN controller is to act as the CE in feedback

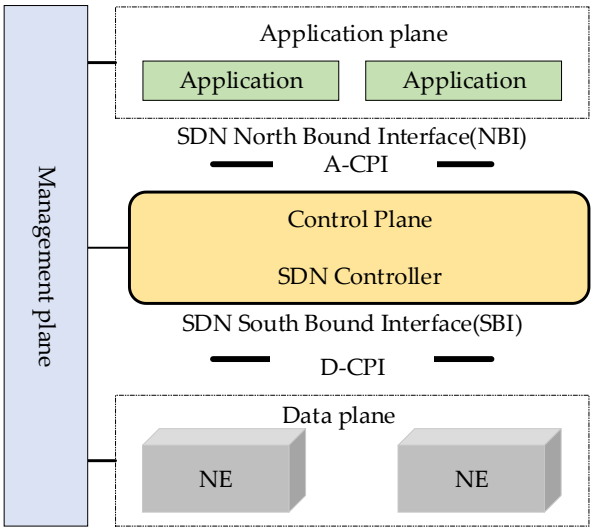



Figure 2.5: Forwarding and control function separation: SDN Architecture

control, responding to network events in order to recover from failure, allocate or re-optimize network resources and as well provide applications with the semblance of logically separate networks.

In this regard, Clause 4.3.5 of [14] explains how resources can be shared among applications on a first-come-first-served or Best-effort basis. As one contribution of this thesis, we propose concepts that enable resources to be shared among applications in a varied manner (e. g., prioritized) in addition to what the SDN traditionally proposes. This is discussed in Chapter 4.

The Application Plane (AP) consists of a set of applications, each of which has exclusive control of a set of resources exposed by the SDN controller(s). Though the ONF's definition presents the SDN controller as a black-box, its functions can be defined so that it provides isolation between applications and thus, portrays as sense that each application exists on a network tailored specifically to its needs. In the next Chapter of this thesis, we will identify, examine and discuss some enabling technologies that can be used to achieve this goal.

The elements of the AP exchange messages with the SDN controller via a non-standardized Application-Control Plane Interface (A-CPI) also referred to as the North Bound Interface (NBI). Communication between the SDN controller and the NEs in the DP occurs via one or more standardized open Data-Control Plane Interface (D-CPI) also referred to as South Bound Interface (SBI). The nature and conceptual definition of the SBI have been a bone of contention, often leading to great misunderstanding of the SDN concept. This misunderstanding is addressed in this thesis by analyzing SDN programmability through the lenses of the of the ForCES framework to help us in deriving the

term Software-Defined Industrial Networking (SDIN) as a solution for industrial networks.

PROGRAMMABILITY IN THE CONTEXT OF SDN Under the ForCES framework, the concept of programmability is applied to the adaptation of forwarding functions of the NEs and subsequent control function modification within a network device during its operation and/or entire life-cycle. Under SDN, programmability is examined from a slightly different perspective.

Regardless of the subtle differences, one will not be wrong to allude that the entire concept of SDN programmability fits to at least one of the three levels of programmability proposed by the ForCES framework. Upon critical analysis of both frameworks, it can be observed that the SDN concepts fits to the first level of programmability proposed by the ForCES framework. That is, the ability to manipulate and augment the forwarding behavior of an NE during network run-time. However, what makes the SDN concept slightly different from the ForCES concept for level one programmability is that, all NEs are seen as FEs that share common intelligence via the centralized SDN controller.

The similarity is that the FE capabilities cannot be changed during run-time. The SDN controller abstracts the capabilities of the NEs and represents them as single shared logical resource and exposes individual quotas of resources to the application as if they were separate resource (logical isolation). However, in the ForCES framework, FE resources are seen as dedicated directly to applications which can lead to pre- and over-provisioning hence may limit resource usability.

## 2.6 COMPARISON OF SDN AND FORCES PRINCIPLES

Though both concepts (ForCES and SDN) propose the programmability via control and forwarding function separation, there exist some differences behind the governing principles of the respective concepts. While ForCES address network programmability from a generic computer network perspective, SDN evolves towards principles that address more specifically the requirements of service providers. Next, we will compare and contrast the two concepts based on five unique principles.

### 2.6.1 *Principle of control/forwarding function Separation*

Although both frameworks propose the decoupling of CP and FP/DP to enable each evolve separately, SDN defines its separation slightly different from the ForCES framework. That is, the FEs in the ForCES framework are considered in the perspective of an NE while in the SDN framework, an NE is regarded as an FE.

The similarity between the two principles is that although CP and FP are separated, the SDN controller can delegate some control functions to the DP as long as these control functions operate in a manner acceptable to the SDN controller. Feature 9 of the ForCES framework (see Section 2.5.1), stipulates that the CE can also offload some control functionalities to the FEs. Therefore, the two frameworks are similar on this point. However, a subtle distinction of the control and forwarding separation in the context SDN is that, all NEs share common intelligent entities due to the centralized instance of the SDN controller. On the other hand, FEs in ForCES may share common intelligent entities only within an NE .

Also, considering the protocol used for information exchange between the CE and FE/DP, SDN proposes the use of open interfaces where as ForCES does not restrict nor advocate for the use of open interfaces. However, the characteristic requirements of the ForCES protocol (Fp) can be achieved as a non-proprietary protocol.

Comparing the two frameworks from the true essence of evolution as studied by Dovrolis et al. [10], from a biology point of view, the true meaning of evolution is the ability of an organism (in this case a Network) to change its features in order to adapt to a changing environment (application requirements).

Considering the analysis of the two architectural frameworks from a biological evolution of an organism, it can be seen that ForCES meets the evolvability criteria described by Dovrolis et al. [10]. This is because the second and third level ForCES programmability proposes changing of device forwarding functions and behaviors to meet requirements of its environment while SDN adapts its environment to the network.

Although both approaches can be seen as an evolution in some sense, SDN's claim of independent evolution of the different planes as a result of decoupling CP and FP/DP is not entirely true in the real sense of evolution as defined by Dovrolis et al. [10]. It is the opinion that the current SDN principles, though still viable for industrial usage, if not made to adopt some ForCES principles may fail to deliver fully on its promise of independent evolution of control and forwarding planes.

### 2.6.2 *Concept of shared resource*

The notion of network resource is that all applications on a network have dedicated resources. This notion of resource is similar in the context of ForCES framework. It implies that applications hold on to the physical resources of the FP even at inactive periods. This dedicated resource model is often inflexible as it provides limited opportunity to increase resource utilization efficiency due to over-provisioning.

SDN provides a concept whereby applications only hold onto resources at an abstract level, that is, the resources exposed by the

controller to applications are logical and are only committed as and when they are required on the DP. In this regard, applications can be over subscribed to the network resource as long as the applications can live with the resulting guarantees when the logical resources are committed (instantiated) as physical resources in the DP.

To ensure prioritization, SDN proposes to serve applications with stringent requirements first. However, this often does not work well as arrival, departure, and time of resource usage can be very unpredictable. Even in deterministic networks like ICNs, there still exist the possibility that the network operator may want to add a new service, planned or unplanned, to the system at run-time and therefore the unpredictability must somehow be curtailed to provide deterministic bounds.

This thesis proposes alternative ways to the first-come-first-served and best-effort approaches by adopting worst-case concept to guarantee application QoS requirements by using a Demand Based Approach (DBA) for efficient resource orchestration.

### 2.6.3 *Abstraction and virtualization*

The SDN controller operates on globally abstracted capabilities of the physical resources of NEs which are represented or instantiated as virtualized resources. This means applications have access to the physical resources only when all requirements have been tested and contractually established in the virtualization layer.

This can be achieved by the design and development of common information models that represent the functionalities of physical hardware (NEs) as well as the applications using the resource. In Chapter 4 of this thesis, we show how this can be achieved as a component of the service functions using enabling concepts and technologies described in Chapter 3.

On the contrary, the ForCES CE operates on a locally abstracted resource of the FEs within an NE. Therefore, it is not easy to achieve the same level of resource usability in the level one programmability of the ForCES framework.

### 2.6.4 *Service creation*

One of the most important service routines is the deployment of configuration data to the NEs in the FP/DP. Before actual resource reservations can be achieved across the network, all NEs must have some default configurations. This allows them to communicate and establish connections before applications can begin to use the service they provide. This aspect of networks persist in both ForCES and SDN frameworks.

As shown in Figures 2.4 and 2.5, the two architectures underline the

need for management components at all planes of the architectural frameworks including the FP/DP. In this regard, the SDN framework describes the possibility of fusing traditional network management systems as a part of the SDN controller. This enables network operators to appropriate traditional network management protocols such as Simple Network Management Protocol (SNMP), Network Configuration Protocol (NETCONF), etc. as south-bound protocols between the SDN controller(s) and the NEs in the DP if they meet the requirement of an SBI.

With the SDN controller serving as a feedback control entity to the NEs in the DP as well providing oversight of resources, it enables network engineers to integrate service functions to orchestrate different services on the network in an automated manner.

The service functions can consist of carefully curated chains of atomic or basic network functions. The network functions may also leverage the power of high-level abstraction and virtualization which are traditionally done as a separate task or routine. The centrally instantiated SDN controller, as will be discussed later in Chapter 7, provides some possibilities for different deployment scenarios of the same network infrastructure.

On the other hand, ForCES framework does not restrict the possibility of having these features that make SDN so attractive. It supports these features but its specifications are a bit silent. This in my opinion was intended so as not to dictate deployment aspects of networks. The evidence of this can be inferred from RFC 364 (Architecture).

Another difference between the ForCES and SDN framework is the terminology and role of the architectural components in the different planes w.r.t the protocol operating between the CP and FP. While the SDN framework identifies the whole system as a hierarchically recursive client-server relation with an $N$-level agent system, the ForCES framework proposes a non-hierarchical, non-recursive master-slave relation.

In the hierarchical-recursive client-server relation, an entity at a level N+1 perceives another entity at level N as a resource that exposes and/or has a set of actions. SDN applies this concept through out the entire system. This implies all elements of the AP perceives elements of the CP (Controller mainly) as servers that provide set of actions on resources that can be used to request services. Likewise, elements in the CP perceive elements of the DP (mainly NEs) as servers as well. This makes SDN a desirable programmable framework for service providers and the internet.

Additionally, unlike the traditional client-server relation where the client receives information or data by proactively requesting it, the SDN framework allows servers to notify clients of changes occurring on them. This feature is especially included between the level N;—the CP where Controller resides and level N-1;— the FP where NEs reside.

This feature allows the network controller to stay abreast with resource availability and faults occurring within the DP.

The ForCES' Master/Slave relation unlike the hierarchical recursive client-server model, defines a strict relation where the roles of the CE and the FE are specific. That is, the CE always plays the role of a master while the FEs play the role of slaves. Thereby making the ForCES protocol a strictly master-slave protocol. What happens in this kind of relationship is that the CE always instructs on what to do and the FEs must comply and report back on the actions.

It must be emphasized that the Client-Server model proposed in the SDN framework especially between level N and N-1 is a Master/Slave model. A typical Client-Server model describes a relation between systems in which one or more systems;—the client(s) requests service from another system;— the server which fulfills the request.

Generally, a Server can serve several clients in a Client/Serve models. In this relation, clients should only be able to access information but not modify data on the server. Often this is a requirement that ensures data integrity as errors can be introduced by unsynchronized or unpredictable modification of Server data. This is not the case in SDN i.e; the specifications describe the role of controllers as entities that manipulate NE (Server) data in order to change the behavior of the network. This completely contradicts the principles of the client-server model.

Contrarily, the Master/Slave model proposes a system of communication where an entity; the Master has unilateral control over one or more entities; the slaves e. g., NEs. In this model, a Master oversees all slaves and can change the data of Slaves as and when required. This in my opinion describes better the principles of SDN protocol as rightly captured by the ForCES framework.

### 2.6.5 *Evolvability/ adaptability*

In the context of adaptability and evolvability concerns described in Section 2.3, the two frameworks provide a good set of features that present the potential to address the adaptability/evolvability concerns in different ways.

The SDN framework provides the potential to integrate legacy and currently unknown application requirements on existing infrastructures by leveraging the power of logical abstraction and virtualization. With a good use of virtualization concepts (abstraction and instantiation), it can be used to meet the QoS requirements of very contrasting applications on the same network infrastructure. However, a fundamental thing that must be noted is that SDN provides the ability to adapt an application's requirements to the network rather than changing the network to suit applications' requirements. This is because the SDN's DP maintains a specific set of capabilities that do not change

during deployment. Therefore, service requirements can only be supported if the DP possess adequate FE (QoS) mechanisms to do so. Hence, the independent evolution of control and forwarding plane promised by the SDN framework is strictly dependent on the capabilities of the DP and thus evolvability only applies to the adaptability of CP functions rather than DP. That is, SDN only seeks to augment the control functionality of network devices by integrating computational models e. g., programming languages, operating systems, object orientation together with communication models e. g., QoS and packet forwarding [6]. Unlike SDN, ForCES provides specification that allows the CP and FP planes to evolve separately as illustrated by the 3 levels of programmability discussed in Section 2.5.1.

The current implementation of SDN can also be made to adapt the packet processing functions of NEs but cannot be achieved in real-time or at run-time of the network. That is, to do so, the Application Specific Integrated Circuit (ASIC) of the NEs need to be editable. This aspect is, however, not covered under SDN. The ability to edit control functions on ASIC is covered under Programmable Protocol-Independent Packet Processors (P4) which deals with defining packet processing functions in the NE.

P4 is developed to target some components of the SDN architecture that enables it to reach ForCES level two programmability to a certain degree. However, forwarding functionalities like schedulers and shapers cannot be changed or bypassed. Silicon-independent switch operating system (Stratum) is another project currently being developed by the ONF to enable SDN reach full level 2 ForCES programmability by building minimal production-ready distribution for white-box switches where the packet processing and forwarding functions can be changed to make networks evolvable under the SDN framework.

On the other hand, the ForCES framework promises different kinds of adaptability and evolvability in comparison to SDN. While the first level of programmability can be extended to behave in the same way as the SDN abstraction and virtualization concepts with a centrally instantiated controller, the second and third level programmability emphasizes the adaptation of the network infrastructure to its environment (requirements of application). Simply put, the SDN framework enables the adaptation of requirements of a network's environment to the network whilst the ForCES framework in its fullest realization adapts the network to its environment. A good study of the adaptation of requirements of heterogeneous industrial application by leveraging SDN is examined in the FIND project in [11].

### 2.6.6 *Long-term capital expenditure*

From a cost perspective, the initial capital and effort required to realize the ForCES framework to its fullest are capital intensive but very rewarding over a long period. This is because a common NE can be reused in any environment regardless of new trends that may arise in future industrial production systems.

Comparably, SDN will involve less capital as the only component of the network that is required to change over time is the network controller which might not even require any dedicated hardware for deployment.

The combination of P4, and SDN can enable SDN to reach level 2 ForCES programmability but forwarding functions such as schedulers and shapers which determine some QoS capabilities of the network cannot be changed. If ever, the ability to change these forwarding functionalities comes into fruition in future, possibly with the Stratum [15][3] project, both frameworks may be indistinguishable in terms of cost. In their current state of implementation in the network community, SDN offers more functionalities for money based on requirements of legacy and current applications, however, from a technical and architectural view point, ForCES provides all the ingredients required to make networks evolvable.

## 2.7 SDN-ENABLED INDUSTRIAL NETWORKS

This section discusses the potential of SDN for industrial networks illustrating the core concerns and how SDN concepts can help in achieving these concerns. The discussions here provides a context for subsequent chapters where service guarantee (QoS) and performance analysis are discussed.

To illustrate the synergy between SDN and the ForCES framework, after establishing that SDN programmability can be categorized as a level one ForCES programmability, the FE and NE will be used interchangeably in the subsequent discussions within the SDN context when required.

Some significant takeaways from the analysis of SDN and ForCES architecture are as follows:

1. Control functions must be developed with a detailed understanding of FP mechanisms or functions. Even though the decoupling of the planes enables each plane to evolve, hence making network adaptable, the control plane cannot evolve independently of FP. The forwarding plane provides the baseline upon which CP functions are designed. Unlike the CP, the FP can evolve

3 Stratum is a next-generation of SDN based silicon-independent switching operating system that runs on white-box switching platforms [15]

with zero dependence on the control. However, SDN framework does not support this feature.

2 The concept of shared resources in the SDN framework should enable the integration of abstract reservation directly into the network service creation during run-time. However, these models must emulate the QoS capabilities of the network which is only provided by the NEs in the FP.

3 SDN framework provides a feedback control system (SDN controller) which can integrate intelligent functions and models capable of mapping new application requirements by leveraging the optimum topology of the DP. This feature and point 2, are very interesting for industrial networks as service creation can be automated. This comes from the recent integration of time-sensitive FE mechanisms such as the Time-Aware Scheduler (TAS) in standard Ethernet bridges. Furthermore, the centrally instantiated network controller provides significant flexibility for deployment purposes. This is discussed in Chapter 7.

It is important to note here that in order to leverage SDN for industrial networks, there must exist an SDN compatible data plane for industrial networks. Ethernet has already been established as an SDN compatible DP for telecom and data-center networks. However, with the integration of forwarding functions such Credit Based Shaper (CBS) and TAS in Ethernet, TSN provides a perfect SDN compatible DP for industrial networks due to the varied ways packets can be forwarded to meet different delay guarantees. We will build on the above takeaways to derive what is termed in this thesis as the SDIN.

**Definition 2.7.1.** Software-Defined Industrial Networking (SDIN) is an industrial network constructed on SDN guidelines to address industrial network evolvability concerns whilst still providing service performance similar to legacy tailored solutions.

Therefore, the evolvability concern addressed in this thesis has to do with the mapping of industrial application requirements to SDN-capable industrial infrastructure rather than the adaptation of the infrastructure to the application requirements. This, therefore, requires the definition and design of control functions to augment existing SDN controllers to support some industrial applications (e.g., Programmable Logic Controller (PLC) to IO devices) which otherwise could not co-exist on the same bridge infrastructure with video, audio and other best-effort traffic due to their widely contrasting traffic characteristics.

## 2.8 SUMMARY AND BIBLIOGRAPHIC COMMENTS

### 2.8.1 *Bibliographic comments*

Several research papers have been published within the context of programmable networks. Among them, Campbell and de Meer et al. [5] provide a survey highlighting several innovations leading to network programmability levels. The innovations identified include the separations between transmission hardware and control software, accelerated creation and deployment of new network services through virtualization of network infrastructure, and availability of open interfaces. Also, Galis et al. [6] provide a concise account of the genesis of programmable networking concepts in which the authors identify the Open Singnaling (Opensig) community [4] and Defends Advanced Research Project Agency (DARPA) Active networks as the two school of thought that began the topic of programmable networks.

The basic idea of the Opensig community advocates for a service composition framework that is accessible via open interfaces. In this framework, the Opensig community proposes a restructuring of monolithic and complex control architectures in a minimal set of layers making the services within each layer accessible via open interfaces. These ideas were later formalized in what became the IEEE P1520 reference model [6, 16].

Contrary to the Opensig idea, DARPA Active networks [17, 18] propose a much more radical approach that allows applications to directly define how networks process their packets by injecting programmable code snippets in IP packet headers. One fundamental principle that both communities are convergent on, is the clear separation of control and transport planes [6]. The ForCES architectural framework proposed by the Internet Engineering Task Force (IETF) also advocates for the clear separation of control and transport planes in what is referred to as FP discussed in Section 2.5.1. The analysis by Galis et al. [6] show that the ForCES FE model uses a similar approach to the building block approach of the IEEE P1520.3 working group where they encapsulate distinct logical functions within FE blocks.

Based on the fundamental principle of open service interfaces, separation of planes and the integration of computational models with communication models, the ONF proposed SDN. SDN has seen significant adoption compared to the ForCES framework though, in the true sense of evolution, the ForCES principle show great promise.

The reason for the widespread adoption of SDN in today's network research vocabulary is how its fundamental principles can easily be demonstrated with the implementation of the SBI protocol

4 OPENSIG was established in 1995 with the main objective of making the internet, mobile and Asychronous Transfer Module (ATM) networks open, extensible and programmable [6]

OpenFlow [19], and the readily available OpenFlow programmable switches [20]. Recent advances [21–23] has seen network management protocols such as SNMP (for backward compatibility) and NETCONF[5] [24] appropriated as SDN south-bound protocols. Furthermore, there are several open-source SDN controllers [25, 26] which have promoted its widespread adoption.

Based on analysis of Galis et al. [6] and facts presented, we can confidently conclude that SDN is partly developed on similar principles as IEEE P1520.30.

### 2.8.2 *Chapter summary*

This chapter examined the fundamental concepts behind the development of industrial networks, highlighting the different requirements and concerns that have contributed to the plethora of network solutions today. More so, it identifies and discusses why current industrial networks are in general not flexible enough to support emerging services. It pinpoints the architectural inflexibility of NE as a major contributing factor that has halted industrial networks' ability to evolve with changing requirements. It further examined programmable networks as a framework that can ameliorate industrial network evolvability concerns. Among existing programmable network concepts, the SDN and ForCES frameworks were identified, examined and compared, highlighting similarities and differences in their governing principles to derive solutions that can be used to address industrial evolvability concerns.

Based on the analysis and consideration for current requirements of the automation industry, SDIN is derived as a solution that integrates SDN principles to manage industrial network infrastructures. The analysis illustrates that although SDN does not fulfill requirements of network evolvability in all sense and purposes as defined by Dovrolis et al. [10], the centralization of CP does enable the integration of computational and communication models for control and management of industrial networks. Thereby allowing the adaptation of widely contrasting production environments to a common network.

While SDN framework enables the management of heterogeneous requirements, a fundamental question that remains unclear is how its promises can be achieved in industrial communication networks given the several concerns and contrasting characteristics of applications.

Chapter 3 identifies and examines enabling technologies to provide differentiated service guarantees and how they can be leveraged in production-grade SDIN. These enablers are extensively presented with novel functions and algorithms for SDIN in Chapters 4 and 5.

5 NETCONF is a management protocol which allows NEs to expose an Application Programming Interface (API) via which extensible configuration data can be sent/retrieved and/or modified.

# 3

# SDIN:SERVICE PROVISIONING ENABLERS

In a typical production network, computational models are used in what is termed engineering phase. In this phase, the network is designed to fit the operational characteristics of applications it supports and solutions used as an operational model on which the network runs. This chapter leverages the features of SDN controller as a feedback control to define algorithms that enable the integration of computational models as part of service functions to automate offline service engineering as an online process.

First, we identify existing methodologies used to simulate the study of systems and their operation. Second, these methodologies are carefully examined to provide a clear understanding of how they can be leveraged in SDIN to develop analytical models that integrate in resource allocation algorithms to enable an SDIN controller engineer and create services based on defined policies.

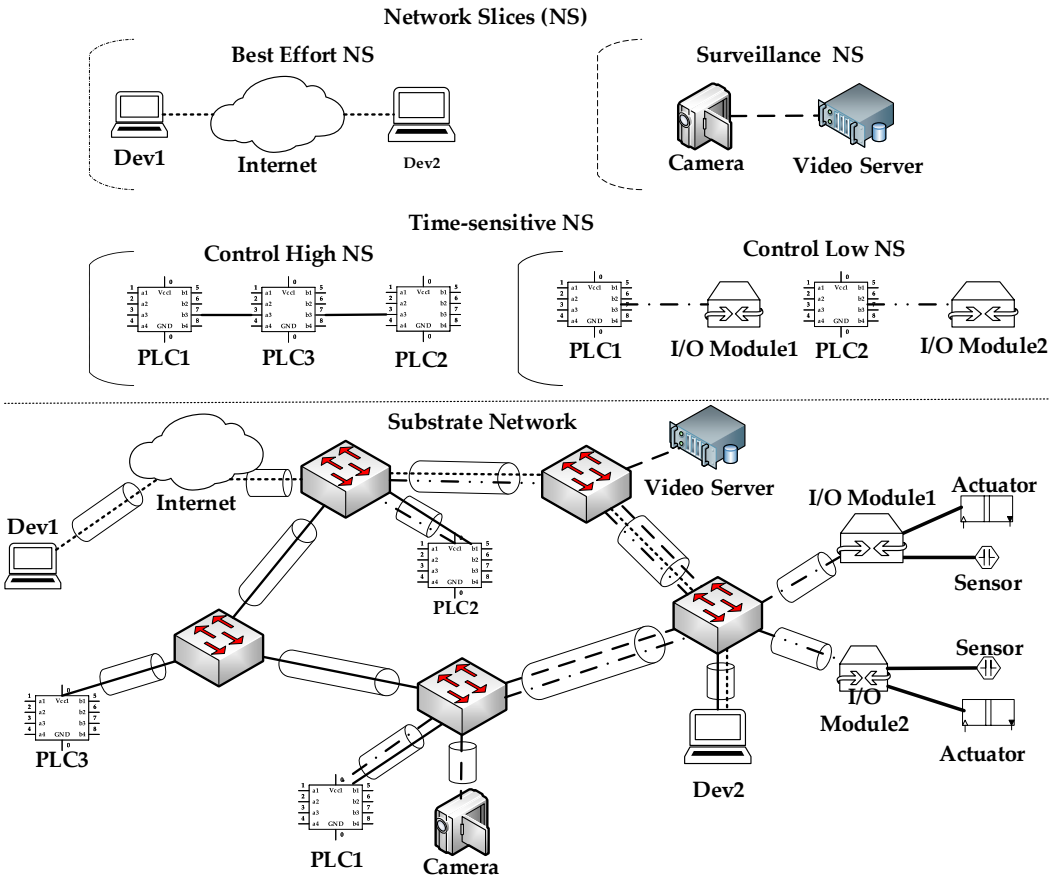



Figure 3.1: Network slicing in converged industrial networks

## 3.1 NETWORK SLICING AND THE VNE PROBLEM

Network Slice (NS) is a general term used to describe a set of resources used to provide service guarantees within SDN. The concept of NSs represents a specific type of network virtualization that enables multiple virtual networks to co-exist on shared network infrastructure. This thesis provides a concise definition of an NS within SDN as follows:

**Definition 3.1.1.** Network Slice (NS) is an abstract representation of a virtualized set of network resources that ensures the e2e network requirements of applications are guaranteed on a shared network infrastructure.

The guarantee provided by an NS to an application serves as a binding contract between the application and the network. This implies whenever resources held by an NS are committed to the DP, the required guarantee can be assured by the QoS features within the DP. The reverse also implies if the QoS features of the DP cannot adequately ensure a certain level of guarantee then the NS cannot fulfill the contract with the application.

The definition of an NS reveals a primary dependence of the concept on virtualization which is a widely researched topic. According to Fischer [27] and others [28–31], virtualization of resources enables flexible design and management of networked systems such that even the network itself can be virtualized. This enables network engineers to create highly flexible, and tailored network structures for arbitrary communication (cf. [27–30]). It also implies virtualization is an indispensable tool for the realization of NSs on shared network infrastructure.

Figure 3.1 illustrates an example of the concept of NSs on a converged industrial network where applications with contrasting characteristics can share a physical network. In this example, network requirements of applications from different levels of the industrial automation pyramid (see Figure 2.1) are accommodated on a shared network. The resources that provide the network guarantees are defined by NSs within the CP provided as SDIN controller. However, the logical resources held by the individual NSs can occupy the same physical resource in the DP.

The creation of NSs for arbitrary communication using virtualization concepts is a popular resource assignment problem known as the Virtual Network Embedding (VNE) problem. The topic's importance has seen numerous VNE algorithms proposed and discussed in literature, focusing on varied areas of application. In recent times, it has been the primary methodology for the creation of NSs due to the flexibility of integrating QoS features.

A formal description of the VNE problem helps to clarify the underlying principles of the methodology. It will also provide an understanding of how the VNE can be applied in SDIN.

Formal descriptions of the VNE problem have been provided by many authors in literature. These descriptions are often specialized for particular use-cases; therefore, no single description captures the entirety of problem [27]. However, Fischer [27] provides a mathematical description which is very useful to build upon. This reduces the effort required to treat the entirety of the VNE problem. Therefore, this thesis leverages the contributions of Fischer [27] by highlighting the guidelines provided for *online* and *offline* treatment of the VNE problem discussed in Chapter 3 section 1 to 2 in [27].

The reason for leveraging Fischer's work is in part, the usability of mathematical definition, and flexibility to adapt and integrate computational models as part of VNE problem in production-grade SDIN controllers. However, a deeper look at requirements for the design of VNE algorithms, modeling of network infrastructures and application requirements that enable the creation of dynamic industrial NSs within the SDIN is needed.

A thorough analysis and mathematical definition of the VNE problem is given in the thesis of Fischer [27]. The definition given below is provided to highlight aspects of the problem that can be improved to design and develop computational models for NSs in the context of SDN.

### 3.1.1 *Basic concept of the VNE problem*

Given a Substrate Network (SN) and a set VNRs, the VNE problem describes a system of solutions to map VNRs on SN according to the inequation (3.1)

$\mathcal{D} = \{demands\}$
$\mathcal{R} = \{resources\}$

$$\sum_{g=1}^{G} \mathcal{D}(VN_g) \leq \mathcal{R}(SN) \tag{3.1}$$

where Virtual Network (VN) is the number of VNRs mapped to an SN.

The VNE problem represents a resource and demand pair where demands imposed by the Virtual Network Request (VNR) are matched with the type of resource provided by the SN. In Fischer's analysis, demands refer to the virtual nodes' and links' capacities which can be guaranteed by SN node and link resources respectively. For simplicity, Fischer assumes resources and demands are numbers in $\mathcal{R}_0^+$ such that the value of a demand indicates the amount of corresponding resource it occupies. This does not necessarily apply in all scenarios. Demands may sometimes represent requirements such as the application's communication pattern or application characteristics that are not in $\mathbb{R}_0^+$. This, therefore, implies a demand may or may not be in $\mathbb{R}_0^+$ [1].

1 a demand can also be a Boolean operation on a resource

### 3.1.2 *Formulation of VNE problem*

The VNE problem is mathematically formulated using graph theory. The SN and VNR can each be represented as a graph G, that consists of a tuple $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represents a finite set of vertices. $\mathcal{E}$ represents a finite set of edges such that each $e \in \mathcal{E}$ is a set consisting of two elements $\{u, v\} : u, v \in \mathcal{V}$. Solution to VNE problem consist of three processes namely:

- the modeling of Substrate Network (SN) as graph that captures abstraction and representation of the underlying network infrastructure.
- the modeling of VNR as a graph which captures abstraction and representation of application/service requirements.
- the instantiation of VNRs on SN as VNs.

The instantiation of VNs on SN is a process or task handled by VNE algorithms. VNE algorithms are generally use-case specific as no single algorithm is best in all scenarios. The VNE problem can be split into two sub problems;— the Virtual Node Mapping (VNoM) which deals with mapping virtual nodes on SN— the VLiM deals with the mapping virtual links on SN. Though the problem can be split into two sub problems, the VLiM remains an NP-hard problem due to its combinatorial nature.

Solution to the VNoM problem is to find a node in SN that matches the demands of a node in VNR. Demand can be a real number in $\mathcal{R}_0^+$ e. g., memory, processors, etc. or a label of a node in VNR that matches a node in SN. On the other hand, a solution to the VLiM problem is to find a set of paths within a SN that matches the demand of a link in VNR. In this sense, a path is defined in the instance of a SN such that a path between node $u$ and $v$ denoted as $P_{u,v}$ is defined as a consecutive sequence of edges between $u$ and $v$.

Solving the VNE problem requires solutions that must address two main objectives:

Objective 1 map demands of a VNR on the SN such that all demands of the VNR are satisfied.

Objective 2 make efficient use of the SN resource in order to accommodate more VNRs.

Objective 1 deals with modeling of the SN infrastructure as resources, demands of the VNR as constraints, and constraint verification models.

The basic operation here is to compare the solutions on a set of resources to the constraint imposed by the demands. For demands in $\mathbb{R}_0^+$, if a solution on a set of resources, $\mathcal{R}$, is algebraically less than

or equal to the demand, D, (i. e., $\mathcal{R} \leq D$), then the set of resources in question is said to be a solution that satisfies the demand; otherwise the demand on the resource cannot be satisfied. For demands not in $\mathbb{R}_0^+$, (i. e., $D \notin \mathbb{R}_0^+$), simple equality or binary operation is a sufficient verification.

For a VNE problem, there can be more than one solution or multiple combinations of a set of resources that satisfy a demand. Therefore, an algorithm designer can choose solutions such that they fulfill an optimization goal. The decisions on which solutions fit the optimization goal for a large set of demands makes the problem NP-complete and thus, strongly NP-hard. This is where the second Objective 2 comes into focus.

Achieving the second objective (Objective 2) of the VNE problem requires a discretionary use of the resources on a SN by the algorithm designer. Often, an approach may perform well in certain areas of application and perform abysmally in other scenarios due to changing requirements of the problem space. In literature, e. g., authors in [32–34], researchers often go for an approach that tries to minimize the SN bandwidth usage in order to maximize the number of VNR that can be instantiated as VNs in the SN.

Contrary to this approach, one contribution of this thesis proposes a Demand Based Approach (DBA) which emphasizes on the steering demand in comparison to the set of feasible solutions when deciding on the resources that fit the optimization goal. The design and development of DBA algorithms are discussed in Chapter 5.

### 3.1.3 *VNE application domains*

VNE is applied in several areas of network and service management. Particularly in cloud and data-center networks, solutions to the VNE problem are required to instantiate VNs that can be operated as separate networks on a physical infrastructure. In this domain, it is required that the topology of the instantiated VN represents an exact match of the VNR in the SN.

Demands in this area of application requires finding sub-graph isomorphic representation of the VNR topology on the SN as illustrated by Ashahin and Lischka et al. [35, 36]. This implies VNoM can be achieved on any substrate node as long as numerical attributes of the demand can be met by an SN node. This area of application falls under sub-graph isomorphism.

Contrary, within the SDN context, nodes of a VNR are unambiguously defined. This is because they serve as the service endpoints within a network. Therefore, solution of the VNE problem within the SDN context reduces to a VLiM problem, where the VNoM process is assumed to complete successfully [33, 37, 38].

From this point onward, the VNE problem will be conceived as

a VLiM problem, where the VNoM process has already completed successfully.

### 3.1.4 *FSFD and ARFE VLiM use-case in SDN*

Within the context of SDN, the VLiM problem can take one of two forms; Fixed-Source-Fixed-Destination (FSFD) and Arbitrary-Root-Fixed-Endpoints (ARFE). We define the two forms as follows:

**Definition 3.1.2.** FSFD is a VLiM problem where the mapping of the source and destination nodes in VNR are unambiguously defined as two different nodes within a SN whose labels/identity has a one-to-one mapping to nodes in a VNR.

The solutions to the FSFD VLiM problem are applied in the SDN context for slicing communication services between applications on the network where streams are exchanged between only two applications as shown in Figure 3.2.

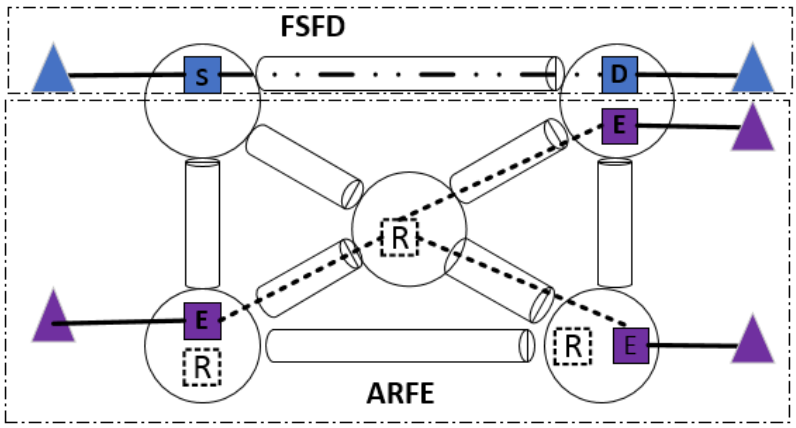



Figure 3.2: FSFD and ARFE service topology problem

**Definition 3.1.3.** ARFE is a VLiM problem where the mapping of the communication endpoint nodes in VNR are unambiguously defined but the anchor (Root) node can be any node within a SN.

The solutions to the ARFE VLiM problem are applied in the SDN context for slicing communication services between applications on the network where streams are exchanged between more than two applications as shown in Figure 3.2.

The mapping process for the FSFD and ARFE requires different mapping strategies just as uni-cast and multi-cast communications are treated differently in traditional networks.

## 3.2 DEPLOYING VNE IN PRODUCTION-GRADE SDIN

For the same VNE problem, the results produced by a VLiM algorithm can vary widely. Fischer's [27] investigation into the problem reveals a compelling condition under which such variations can result; the processing order of VNRs by a VNE algorithm.

Especially in a multiple constraint system, where VNRs impose different constraints, the acceptance of a $VNR_i$ influences the decision on a $VNR_{i+1}$ where $i = 1, 2, \cdots$. That is, if the order of VNRs arrival changes, the results produced by a VLiM algorithm may subsequently change as well. Therefore, this variation introduces two scenarios under which VLiM algorithms are evaluated:— offline scenario, in an offline scenario, the VLiM algorithm designer assumes complete knowledge of VNRs. Therefore, can decide the processing order that allows the VLiM algorithm to yield maximum acceptance. This implies the algorithm can run through several iterations of the problem to find the order which ensures that a maximum number of requests are instantiated as VNs on a SN. — online scenarios, unlike the former, assumes no knowledge of VNR characteristics nor their order of arrival and thus VNRs are processed based on the temporal state of the VNE problem.

The two scenarios stated above suggest a need for different ways of deploying VLiM algorithms.

OFFLINE DEPLOYMENT IN SDIN In legacy industrial networks like Profibus/Profinet, the network is fully pre-engineered before commissioning into operation. Once in operation, no new services can be added dynamically. To add new services, the entire system has to be halted and the process repeated all over again in a manual reconfiguration process.

An SDIN controller can proactively and reactively control and manage the DP. Proactive management is more suited to the legacy systems where several iterations of the VNE problems can be performed and results integrated directly by the SDIN controller in the DP. This, therefore, implies that offline scenarios are better suited for these kinds of deployments.

However, the offline algorithm is not required to be in an SDN controller since the entire VNE problem generated is static and parameters do not change. Therefore, the individual embedding solutions are the only relevant results of the problem required for integration. This notwithstanding, whether the VLiM algorithm can embed more or fewer requests is inconsequential to how services are provisioned by the SDIN since embedding solutions are the only required input.

ONLINE DEPLOYMENT IN SDIN Unlike the offline scenario, the online scenario considers the temporal state of the VNE problem.

This implies as VNRs arrive and depart, an online VLiM algorithm should be able to compute solutions based on the temporal state of the SN as well as the random and sporadic arrival of requests. This makes the online scenario an interesting use-case where services can be dynamically added or removed from an operational network. Therefore, online VLiM algorithms can be integrated into the SDIN controller for resources orchestration and service provisioning.

To do this, the algorithm must be designed to follow in general a hypothesis for resource usage that is based on the desired behavior of the system under study. This is discussed in Chapters 5 and 6.

### 3.2.1 *Designing online QoS-aware VLiM problem*

The term QoS-aware VNE algorithm is frequently used in literature (e. g., authors in [34, 39, 40]). However, to the best of my knowledge the term is not defined but implied. To better understand and discuss QoS-aware VNE, a formal definition is provided as follows:

**Definition 3.2.1.** QoS-aware VNE algorithm is a VN embedding algorithm that computes solutions on a SN to meet network demands of a VNR using computational models that captures one or more QoS capabilities of the SN.

A VNE algorithm can ensure service guarantees if and only if it is QoS-aware. From definition 3.2.1, for a VNE algorithm to be QoS-aware, three things must be considered:

- the algorithm must have computational models that can estimate accurately one or more QoS parameters e. g., bandwidth, delay, reliability, etc.
- the QoS capabilities of the SN must be modeled as an attribute of a resource in the VNE problem formulation in order for computation functions to compute precise solutions.
- the VNE formulation should be able to adequately characterize the demand in the problem formulation in order to enable QoS computation functions to match demands to SN QoS capabilities.

Depending on the level of abstraction, SN resource and/or an applications' demands (in $\mathbb{R}_0^+$) can be characterized in such a way that the solutions computed in relation to system(s) under study are not exact. Regardless of how demand or resources are characterized, a VNR algorithm does not give any inclination whether the models are correctly characterized or not. This can lead to solutions that are unusable in real-world or production networks.

To deploy QoS-aware VNE algorithms in production-grade networks, the production network and the applications they support must

be precisely modeled and represented using graphs. This involves abstraction and representation of applications' demands as VNR graph and the underlying network, specifically the FP/DP components (e. g., queue, shapers, schedulers, port or interfaces) that describe the QoS features of the production network as an SN graph.

VNR MODELING Often in literature (e. g., [29, 30, 32, 34, 39, 41]), VNRs are modeled simply as a graph that represent a service. Demands such as bandwidth, delay, jitter are randomly generated for each link in VNR. While this may be enough in certain application areas such as cloud and data-center networks, in other networks where high precision estimations are required (e. g., TSN), it is realistic to consider other aspects such as whether the application is sporadic/deterministic, periodic or non-periodic. These characteristics are very relevant if precise solutions are to be computed by the demand estimation function of the VNE algorithm. This is because, the solutions are only derived based on the interaction of flows or streams of the application with QoS capabilities of the supporting SN.

SN MODELING Likewise, the model of the Substrate Network (SN) graph must capture the QoS capabilities of the underlying network accurately. This implies FEs such as queues, shapers/schedulers in addition to port or interface functionality (e. g., speed of interface) must as well be characterized as accurately as possible in the SN model. The characterization involves the topology of the FEs within each of the NE in the underlay network. This aspect is discussed in the paper [42] and treated further in Chapter 4 where an SDIN-enabled TSN is examined as an industrial use-case.

### 3.2.2 *Resource, attributes and constraint relation in VNE problem*

In modeling a VNE problem, certain terms and notation are used sometimes interchangeably. These terms often present a source of misunderstanding. In order to ensure consistency throughout this thesis, we will clarify the terms used in the thesis by highlighting the relationship between them as follows:

CONSTRAINT-DEMAND RELATION A demand is an attribute of a VNR that describes the service requirements to be deployed on a network. These requirements from a mapping and embedding perspective of VNE define the solution space of a VNE algorithm. Therefore, from the perspective of a VNE algorithm, a demand imposes constraints which guide the outcome of solutions. Without constraints, all solutions returned by a VNE algorithm will satisfy all demands, which is undesirable. In the context of QoS-aware algorithms, demands cannot be satisfied without constraints. Though QoS-aware algorithms are

often characterized by the same parameters, it is worth to make this distinction.

Proceeding further, constraint is used in relation to solutions computed by a VNE algorithm while demand is used in relation to requirements of a VNR.

RESOURCE AND ATTRIBUTES Fischer [27] and Chowdhury et al. [43, 44] define resource as the capacity of a link or node in SN that can be used to fulfill demands of VNR. Generally, resources can take the form of numerals in $\mathbb{R}_0^+$ which describe the characteristics or attribute of the resource.

Example of resources considered in generic VNE problem include the number of processors or memory capacity of a node. Bandwidth is always considered as a resource provided by an SN link. Which resource belongs to a node or link in the VNE problem depends on how the designer represents the problem. Bandwidth can equally be attributed to a node if a node is modeled such that it constitutes interfaces. Since an interface can be characterized by its speed, it can also be represented as bandwidth. However, it is simpler to assign bandwidth as a resource provided by a link that connects two nodes of an SN. This representation provides easy identification of resources and their relationships.

Contrary to the definition provided by Fischer and Chowdhury et al. [27, 43, 44], resources can be referred to as the nodes and links within an SN, used to fulfill demands of VNRs. The reason for this, is that whenever VNE is considered in the context of QoS, the definition by Fischer and Chowdhury makes it difficult to distinguish between bandwidth as resource and parameters such as delay computed on a link in the SN.

Also, bandwidth and delay of a link can be characterized as attributes of the link. In this sense of the definition, it becomes simpler to relate the demands of a VNR (imposed constraints) to the solutions computed on the SN by VNE algorithm.

## 3.3 DESIGNING QOS-AWARE VLIM ALGORITHMS

No single VLiM algorithm performs well in all scenarios where they are deployed. The performance of the algorithm depends largely on the use-case or the problem area. Whereas a VLiM algorithm may perform exceptionally well in problem A, it may perform abysmally in problem B. Therefore, before designing a VLiM algorithm, it is important to examine the requirements to be fulfilled (e. g., time, efficiency and accuracy of the solutions) of the problem under study. The subsequent sections describe requirements which are relevant to consider when designing VLiM algorithms.

### 3.3.1 *Efficiency or efficacy*

VNE algorithms are generally evaluated based on their ability to efficiently utilize the resources of SN in order to instantiate more VNs. In this regard, efficiency is an evaluation metric often attributed to the optimization of resource/demand pairs (see Fischer [27]).

While efficiency is presented as the sole problem of the algorithm, it is more related to the designers approach to the problem than the optimization methodology. The use of Mixed Integer Programming (MIP) or Integer Linear Programming (ILP) methodologies as argued in some literature [32, 33] does not guarantee that a VNE algorithm will perform better in terms of acceptance ratio than their heuristic counterparts within a problem space. For this reason, there is a need to consider a qualitative metric known as efficacy as an additional metric that addresses the designers approach to the problem.

**Definition 3.3.1.** Efficacy of VLiM algorithm is a qualitative metric that examines the ability of VLiM algorithms to compute solutions that illustrates the desired hypothesis for resource usage intended by the algorithm designer in relation to the problem under study.

The definition of the efficacy of the VLiM algorithms enables the approach of the algorithm's designer to be evaluated as well. In Chapter 5 and 6 two design choices are discussed, and evaluated on efficiency and efficacy in relation to FSFD problem together with other algorithms proposed in this thesis for management of SDN-enable TSN.

### 3.3.2 *Time Complexities of VNE algorithms*

VNE is a combinatorial decision problem where generally, algorithms must decide among several solutions. Finding the right solution that efficiently addresses the problem among multiple solutions can be considered NP-complete, making the problem strongly NP-hard (see [45, 46]). This is even the case in the reduced problem where VNoM and VLiM are addressed as separate problems (see Fischer [27]). This makes the problem computationally intractable even in the offline scenario where the problem is relaxed by the assumption that characteristics of the VNRs are known before processing.

Especially in the offline scenario, the order of VNRs can be changed to find the optimum processing order that achieves the best result in unsolvable scenarios (see Fischer[27]). This makes the offline problem strictly concave and/or convex optimization if the objective is to maximize and/or minimize respectively and thus, are much easier to solve than the online case.

The online scenario as examined in this thesis is more challenging because the characteristics of the incoming VNRs are generally un-

predictable [47] and the search space for every VNR differs due to the different and multiple constraints imposed by the demands of VNRs [48]. The online algorithms can be used to dynamically create NSs due to the characteristics of the problem space.

Therefore, designing online VLiM algorithms, the choice of methodology becomes important as algorithm designers find ways to reduce the convergence time hence, the choice of methodology and the art of modeling can change from one algorithm designer to the other. Especially in this thesis, VLiM algorithms are integrated as network functions of SFCs in the SDIN controller where worst-case response time for the overall service provisioning process is expected to be within the same time order as traditional network deployments. This puts a strict time requirement on the algorithms as solutions must be computed in the shortest time possible (usually less than 1 second).

Therefore, online VLiM algorithms must be designed such that they converge in pseudo-polynomial time if not in polynomial time. This is more related to the methodology used to model the problem and algorithm.

In chapter 5, an analysis of the commonly used methodologies are examined to design MIP and heuristic versions of the VLiM algorithms for the FSFD for TSN-enable SDIN.

## 3.4 ACCURACY OF SOLUTIONS PRODUCED BY VNE ALGORITHMS

For practical application of VNE, the solution provided by VNE algorithms must adhere accurately to the service requirements before it can be used in production networks. It, therefore, warrants that online VLiM algorithms must be equipped with QoS computation functions that can derive precise solutions for imposed constraints.

*contribution from [42]*

The VLiM problem is considered within the context of Traffic Engineering (TE) which focuses on QoS guarantees such as bandwidth and delay [34, 39, 49, 50]. Often, the models considered for the problem focus on the instantaneous delay of streams, i.e., delay due to network traffic in an instance of time.

Nonetheless, the delay models are not detailed enough to include delays introduced by the underlying forwarding mechanisms; hence they are inadequate to be considered for deterministic, periodic and real-time requirements in industrial networks. This is because delay due to FEs (such as schedulers and shapers) contribute very little to the magnitude of delay requirement imposed by the services (e.g. video streaming) which are considered to be more tailored to the IT networks.

On the contrary, delay requirements of ICNs are much stringent and are significantly affected by shaping and scheduling mechanisms, aside from being periodic. Also, the worst-case delay scenarios are important as they provide the necessary information for safety engineers

to design countermeasures to guide against failure scenarios that may be caused by packets missing their delivery deadlines. In order to precisely verify VLiM solutions in relation to constraints imposed by VNR demands (e. g., constraints such as bandwidth and delay), the verification functions have to take into account how FEs interact with flows of the application allocated to an NS.

Furthermore, in the creation of dynamic NSs, the aspect of the delay required to be guaranteed by a NS is of a worst-case in principle rather than an instantaneous delay of the individual packets. This is because regardless of the customization of an NS, the physical resources used to enforce the guarantees are shared with other NSs which can influence each other in terms of the performance of the physical resource. For this reason, the overall load to be accommodated by an NS and characteristic of the application using the slice must be used to compute a guaranteed worst-case delay bound by which requirements of flows admitted into the NSs can be used as admission constraints in the VNE process.

In the subsequent section, we examine and leverage existing methodologies for delay analysis. These methodologies will be used in Chapter 4 to formulate computational models which are integrated as constraint verification functions in VLiM algorithms.

### 3.4.1 *Worst-case delay computation for VLiM problem*

Worst-case delay analysis is widely used in deterministic real-time industrial networks to evaluate the performance of configured flows rather than their creation. For industrial switched Ethernet networks, three groups of approaches are examined (*c.f.* [51]): 1) Simulation [52–54], 2) Model-checking [55–57] and 3) Guaranteed (exact) upper-bound. Characteristic of each approach are discussed in the Ph.D thesis of Xiaoting Li [51].

This thesis emphasizes guaranteed upper-bound because it focuses on computing delay by considering worst-case scenarios that a flow can encounter while it traverses a network. The algebraic nature of the methodologies used to characterize delay makes it flexible to integrate into online VLiM algorithms as computational models that can verify whether a VNR will be able to guarantee its delay demand if instantiated as a VN on a shared SN.

In this approach, two important methodologies are considered analytically in literature for exact worst-case delay estimation— TA [58–60] and —Network Calculus (NC). The NC approach is extensively treated in [61–63]. The deterministic and stochastic nature of NC makes it an interesting theory for computing deterministic and stochastic delay bounds especially for packet switched networks. However, the results have been proven to be very pessimistic in its fundamental application due to lack of conciseness to traffic and network component models.

Unlike NC, TA is less pessimistic primarily because of the simplicity of its modeling and mathematical operations. However, the general concept of a trajectory approach is still applicable to NC.

### 3.4.2 *Trajectory approach to Worst-case delay analysis*

To understand the Trajectory Approach (TA), firstly a formal definition of the approach is given as follows:

**Definition 3.4.1.** Trajectory Approach (TA) describes the sum of delay encountered by a packet at each hop in a network from its origin to its destination along a specified path.

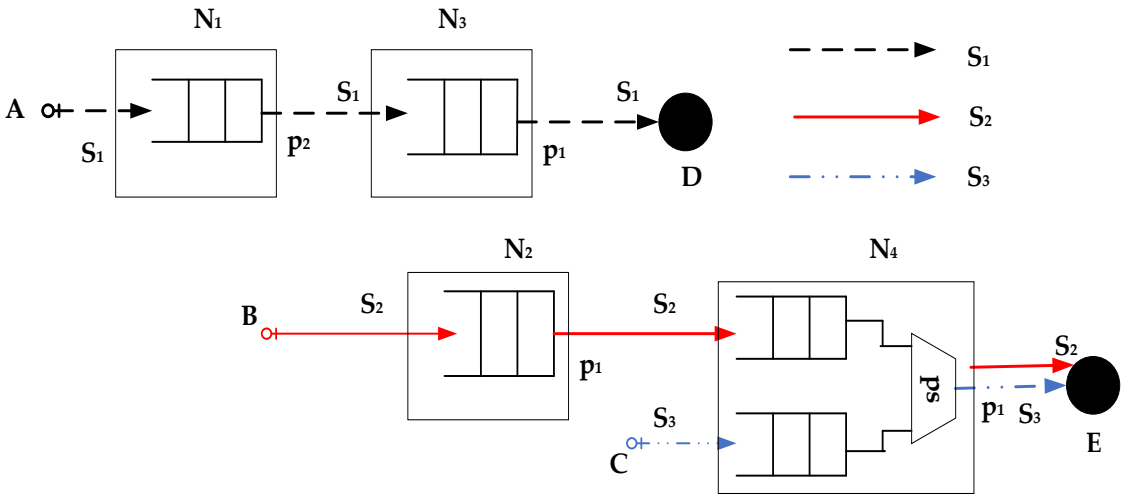



Figure 3.3: Trajectory approach and worst-case delay analysis for NS creation

To illustrate the delay estimation using the TA, consider a network of Ethernet bridges ($N_1 \rightarrow N_3$, $N_2 \rightarrow N_4$) connected as shown in Figure 3.3. Also, let us assume the link between each Ethernet bridge can forward a total of $10^8$ bits per second. Then, the total bandwidth at each bridge transmit interface or port ($p_n, n = 1, 2, \cdots$) is 100 Megabits per second (100 Mbps). Let us also consider three NSs ($S_1 = 512$ bits per 0.5ms, $S_2 = 2000$ bits per 1 ms, $S_3 = 12000$ bits per 2 ms) carrying flows with total burst size of 512 bits, 2 kbits, 12 kbits respectively, then the bandwidth that needs to be guaranteed for each of these flows is 1024 kbps, 2 Mbps, 6 Mbps.

Considering the TA, the trajectory of flows admitted to the respective NSs are as follows: $Tr(S_1)$ starting at point A and terminating at point D is given as $A \rightarrow N_1.P_2 \rightarrow N_3.P_1 \rightarrow D$. Likewise, $Tr(S_2)$ and $Tr(S_3)$ are given as $B \rightarrow N_2.P_1 \rightarrow N_4.P_1 \rightarrow E$ and $C \rightarrow N_4.P_1 \rightarrow E$ respectively. Where $Tr$ denotes the trajectory (path) of flows in an NS.

Considering the $Tr(S_1)$, the delay that can be guaranteed by a NS

for any flow admitted to it, is given by the time taken to forward the last bit of the flow during a *busy-period*. Where a *busy-period* is defined in [58–60] as a duration of time for which the forwarding function has packets to send. This means, the delay of a flow which is ready to send right at the beginning of the maximum possible *busy-period* is the worst-case delay. Any other delay less than the worst-case delay is instantaneous and cannot be guaranteed by the NS.

The maximum possible *busy-period* occurs when packets of all competing flows arrive at the same time, and when additionally further packets of one or more of other competing flows arrive before packets from the first burst of the flow in focus are forwarded.

In the case of $S_1$, the worst-case e2e delay guarantee for all flows in the slice as its flows enters and exit $N_1.p_2$ and $N_3.p_1$ is $\frac{(2\times 0.512)Kb}{100Mbps} = 10.24\ \mu$s. Unlike $S_1$, $S_2$ and $S_3$ interfere with each other at bridge $N_4, p_1$. Moreover, the respective bursts of each slice are generated at different cycles (periodic).

For this computation it is important to define a measurement interval in which the *busy-period* occurs, i. e., how long the *busy-period* lasts. A network cycle (or path cycle) needs to be defined to do this computation. In this example, one can take the network cycle as the largest cycle of all NSs within the network and consider all possible scenarios of packets that can be received at the egress port within the network cycle or consider an arbitrary measurement interval in which case, the computed results become pessimistic with long measurement intervals for application with longer cycles or non-cyclic applications.

The following example explains the concept of network cycle;— consider a video stream with a data rate of 6 Mbit/s. As the maximum Protocol Data Unit (PDU) size for Ethernet networks is limited to 1500 bytes (=12000 bits), this stream will appear at the egress line as one packet of 1500 bytes every 2 ms.

*network cycle can be considered more precisely as path cycle*

For the sake of clarity, let us ignore video compression algorithms' details leading to variable packet sizes. With this, the video stream appears as a cyclic application from the perspective of the network, where the network cycle time is 2 ms, which matches $S_3$ in the example above.

To compute the worst-case delay for a packet of this video stream, one has to sum up all other flows that compete for the same resources (transmit interface) and might arrive within one network cycle (2ms). This will result in a worst-case port delay at $N_4.p_1$ $\frac{(2*2+12)kb}{100Mbps} = 160\ \mu$s during a *busy-period*. This is the worst-case delay that will be experienced by $S_2$ and $S_3$ if they are treated equally without any form of shaping or prioritization, which implies the worst-case e2e delay for $S_2$ over the two hops is $(\frac{2\ kb}{100Mbps} + 160)\mu$s$= 180\mu$s while that of $S_3$ remains $160\mu$s over a single hop.

Here, it makes sense that $S_3$ has the least e2e worst-case delay compared to $S_2$ given that it traverses a single hop. It can also be observed

that based on the load (burst) requirements of the respective slices, the bandwidth reservation at $N_4.p_1$ if expressed as a percentage of the total bandwidth at the port will be $S_2 = \frac{2Mbs}{100Mbps} = 2\%$ likewise $s_3 = 6\%$. However, regardless of their guaranteed bandwidth reservation, the guarantee on delay is not straight forward as in the case of $S_1$.

Assuming that at $N_4.p1$, $S_2$ is prioritized ahead of $S_3$ then, the e2e worst-case delay guaranteed for $S_2$ occurs at a *busy-period* where a non-preemptive packet ($\tau$ bits) of a flow admitted to $S_3$ and the last bit of flows admitted to $S_2$ exit $N_4.p_1$, thus $\frac{2\times 2kb+\tau}{100Mbps} = (40\mu s + \frac{\tau}{100Mbps}s) \approx$ 40 $\mu$s over the two hops.

For simplicity of illustration, $\frac{\tau}{100Mbps}$ is neglected for now. Here, we see for the first time the effect of priority scheduling mechanism in the delay analysis. It can be observed that regardless of the percentage of bandwidth reservation, $S_3$ has at $N_4.p1$ and also just traversing a single hop, the e2e worst-case delay of $S_2$ is much smaller than $S_3$ (=160 $\mu$s).

Now, let us consider the scenario where $\frac{\tau}{100Mbps}$ is not neglected. In Ethernet networks, the largest non-preemptive packet size on a transmission line is equivalent to 1500 bytes (i.e., $\tau$ =1500 bytes). In this case the delay introduced by $\tau$ is $\frac{12kb}{100Mbps} = 120\mu$s. Here, it can be seen that priority scheduling is not even enough. That is, the only way to guarantee an e2e worst-case delay of 40$\mu$s for $S_2$ is to use time-schedules.

In Chapter 4, the operational characteristics of different forwarding mechanisms in TSN are examined individually, and in a tandem composition as an SDIN use-case. Delay computation functions will be derived based on the methodology described here, and integrated as computational models for online QoS-aware VLiM algorithms.

## 3.5 SUMMARY OF CHAPTER AND BIBLIOGRAPHIC COMMENTS

### 3.5.1 *Bibliographic comments*

The concept of slicing is widespread in today's communication network vernaculars as there exist tons of literature describing the concept in general [64]. In [65–68], the authors describe various elements and functional building blocks on network slicing e. g., challenges, requirements, orchestrations, enablers, etc. Afolabi et al. [69] in their survey article, provide an extensive research summary regarding network slicing from a 5G perspective.

In next-generation 5G networks, the concept of slicing is also proposed to be the bedrock upon which several services emanating from the different domains such as industrial factory automation, datacenters and telecommunication can be supported. An article by Zhou et al. [70], proposes a hierarchical e2e NS as a service (NSaaS), and a means to facilitate the building of customized and dedicated services

where the integration of quality assurance by mapping Service Level Agreement (SLA) is discussed. Trivisonno et al. [71] also propose the use of an ad-hoc e2e NS in both 5G access and core network for a massive IoT connectivity use-case which covers a massive deployment of devices requiring sporadic connectivity and small data transmission with stringent QoS requirements. These articles discuss the challenges of slicing pertaining to QoS. However, their proposed solutions are generally tailored to sporadic applications.

Although majority articles focus on NS in the context of 5G, only a few consider industrial networks in particular nor provide any detailed descriptions of how NSs can be created in an industrial network. Kalør et al. [72], identifies abstraction methods and provide e2e delay analysis for industry 4.0 applications where mathematical models based on deterministic NC were generated for NS analysis. An NS can be seen from its users' perspective as an independent private network offering services tailored to their requirements. Contrary, from network operators' perspective, an NS may also exist alongside several other NSs on the same infrastructure such that each NS is customized for different services. This customization is realized by logical isolation of physically allocated resources, abstracted and represented as virtual resources using solutions to the VNE problem [27, 33, 47].

VNE deals with the research problem of realizing a given number of VNs on single network infrastructure by mapping requirements of the individual NSs onto the VN. It has been used in several use-cases such as TE [39, 49, 50], energy [73, 74] and cloud systems to mention a few. Particularly those focused on QoS seek to provide bandwidth and delay guarantees [39, 49, 50, 72]. Often, the models considered for the problem focus on the instantaneous delay of streams, i.e., delay due to network traffic in an instance of time. Nonetheless, the delay models are not detailed enough to include delay introduced by the underlying forwarding mechanisms, hence, makes them inadequate for deterministic periodic real-time requirements for some industrial applications. This is because delay due to these mechanisms (e. g., schedulers/shapers) contribute very little to the magnitude of delay requirement imposed by the services (e.g. video streaming) considered in their use-cases in terms of the order of magnitude. Contrary, this is not the case in ICNs as they are much stringent and are significantly affected by shaping and scheduling mechanisms. Aside from being periodic, some application have strict release times and hard deadlines; thus, any form of packet interference on the network must be avoided.

Furthermore, in the creation of NSs, the aspect of the delay required to be guaranteed by an NS is one that is of a worst-case in nature rather than an instantaneous delay of the individual packets. This is what the next chapter of this thesis addresses which is contrary to current literature.

### 3.5.2 *Chapter summary*

This chapter shows how solutions to existing problems in VNE and worst-case delay methodologies can be leveraged in SDN-enabled industrial communication networks for service guarantees.

Although, VNE is used in several areas and use-cases in literature, none emphasizes the solutions produced by VNE algorithms. Rather, existing experimental evaluations focus on the statistical data such as acceptance ratio and convergence time of algorithms, making the applicability of VNE in production-grade networks unclear.

Examination of the VNE methodology as a tool for the realization of Network Slice (NS) is provided to illustrate the various facets of the methodology that can be used in SDIN. Based on the analysis, guidelines for the usage of VNE in SDIN were outlined. It is shown that within the SDIN context, the VNE problem reduces to a VLiM problem where the Virtual Node Mapping (VNoM) sub-problem is always successful due to the unambiguous nodes of the Virtual Network Request (VNR).

Furthermore, the analysis of the VNE problems reveals that although VNE algorithms are reported in literature to provide QoS guarantees, the algorithms themselves cannot give any indications to whether solutions accurately capture the guarantees. Rather, ensuring QoS guarantee hinges on constraint verification functions which are largely dependent on the forwarding mechanism of the underlying Data Plane (DP).

Based on the outcome of VNE analysis, the chapter provides guidelines and recommendations for designing VNE algorithms and different types of constraints and how existing methodologies can be leveraged in constraint verification functions. Particularly, exact worst-case e2e delay concepts such as Trajectory Approach (TA) was examined showing exactly how delay constraint verification functions can be designed and integrated in VNE problems.

Chapter 4 gives an in-depth step-by-step use of the Trajectory Approach (TA) in the derivation of delay constraint verification models on Time-Sensitive Networking (TSN) as a converged network for industrial applications. Chapter 5 proposes novel VLiM algorithms for resource orchestration and QoS guarantees in Software-Defined Industrial Networking (SDIN). Finally, the algorithms are evaluated in Chapter 6.

# 4

# MODELING DELAY IN TSN-ENABLED INFRASTRUCTURE

One of the core principles enabling programmability in SDN is the flexibility of creating services at run-time. Furthermore, it enables the requirements of different applications to be adapted to the underlying network technology. Also, with appropriate virtualization mechanism, one can use SDN features to accommodate applications with very contrasting requirements as long as the underlying DP possesses the requisite QoS mechanisms that can enforce the service guarantee promised or negotiated in the CP.

In Chapter 2, the management and control of TSN using SDN principles termed SDIN was proposed. However, clarity on why TSN is presented as an industrial use-case for SDN was only examined briefly. The subsequent sections of this chapter will provide an overview of TSN and why it is the DP technology of choice by examining its features in relation to industrial concerns discussed in Chapter 2.

Furthermore, computational models will be derived using worst-case delay methodologies discussed in Chapter 3.4.1. These computational models are then used as verification functions by VLiM algorithms (see Chapter 5) for constraint verification as an admission control mechanism in SDIN controller(s).

## 4.1 USE-CASE: TSN-ENABLED SDIN

Time-Sensitive Networking (TSN) describes a group of IEEE standards with the intended goal of introducing deterministic real-time communication within standard Ethernet-based networks. See Appendix A.2 for list of TSN specifications. TSN enables Ethernet-based networks to enforce deterministic guarantees that make them predictable and ensure guaranteed e2e latency, low jitter and almost zero packet loss; this allows the convergence of critical and non-critical requirements on the same Ethernet network.

The standards propose several features which are as of now partly integrated into the IEEE 802.1Q standard extension. The features address topics such as synchronization, queuing and forwarding, stream reservation, seamless redundancy.

Time-scheduling and synchronization are the key TSN features which enable deterministically guaranteed packet delivery. These features are proposed in the IEEE 802.1AS and 802.1Qbv [75]. IEEE 802.1AS ensures all network devices are synchronized to a common network clock that enables all devices to be aware of when scheduled

critical traffic arrives and departs. IEEE 802.1Qbv adds gates to queue and ensures packets in each queue are held till a schedule permits their opening. The IEEE 802.1Qbv defines specifications for cyclically scheduled traffic using Time-Aware Scheduler (TAS). A detailed examination of the TAS and its relevance are provided in Section 4.2.1.

## 4.2 MATHEMATICAL MODEL OF TSN NODES

This section introduces the Queue-Scheduler-Link (QSL) model for TSN nodes. Here, the use of worst-case delay concepts and analysis are illustrated. This study will discuss the trajectory approach and how it can be integrated with TSN for deterministic delay guarantees but first, let's examine the sources of delay in switched Ethernet networks.

### *Sources of Delay in Switched Ethernet Networks*

A key element to consider in the switched Ethernet network model is the sources of delay within the network. Those considered in literature for worst-case performance analysis are introduced below:

FORWARDING OR TRANSMISSION DELAY is the time to forward a packet on the transmission line. It is expressed as the ratio of packet size and line speed.

QUEUING OR BUFFERING DELAY is experienced as a result of how schedulers/shapers handle packets in the same, higher or lower priority queues. It depends on the given priority with which packets from queues are processed by a scheduler/shapers in relation to others queues.

PROCESSING OR SWITCHING DELAY is caused by packet processing operations in a network device. It comprises e. g., Cyclic Redundancy Check (CRC), address lookup, Virtual Local Area Network (VLAN) table mappings, traffic classification, and ingress policing. Since lookup table sizes are bounded and the packet processing time of the device is known, a device-specific upper- bound is often specified by the device manufacturer. Therefore, switching delay does not contribute any additional information w.r.t how flows interact with the forwarding mechanisms.

PROPAGATION OR WIRE-LINE DELAY is experienced on the transmission medium as a result of signal propagation. It is often specified by the manufacturer, therefore it is not influenced by packet size. Its significance depends on the length of the transmission medium. For short cable lengths in the range of a few meters as used in most ICNs,

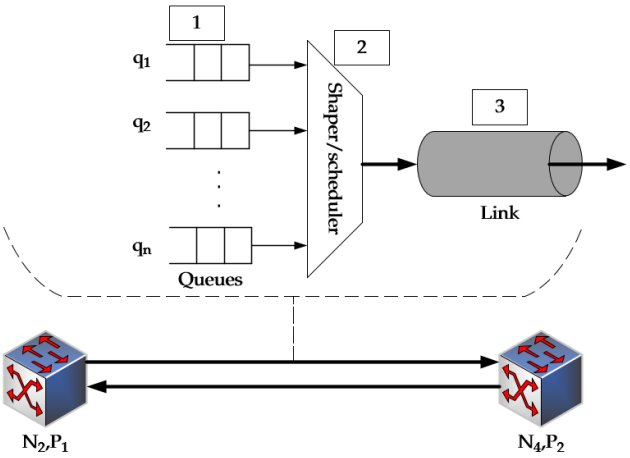



Figure 4.1: Abstraction of network link into QSL model.

they are in the orders of nanoseconds and therefore can be neglected in performance analysis but not in production-grade networks.

Considering these sources of delay, an abstraction that captures the effect of the identified components of delay is derived. This is termed the Queue-Scheduler-Link (QSL) model. Figure 4.1 illustrates the QSL model abstracted from a direct connection between two Ethernet bridge ports. The abstraction focuses on the transmit interface of the port. As shown in Figure 4.1, the most significant components of delay are the points ***1*** and ***2*** as they capture the queuing, scheduling and processing delay while point 3 captures the propagation delay.

The queuing delay depends on;— queuing discipline used within the queues and scheduler/shaper type, which selects packets from queues for transmission. The worst-case delay models are based on the DP characterized by the QSL model. The QSL model is integrated into the SN model in Section 4.4.2. A link-queue model was proposed by Guck et al. [76]. However, the difference between the QSL model and the link-queue model is the inclusion of schedulers/shapers. The examination of the delay components shows how the handling of packets in a queue by scheduler/shapers affect the queuing component of delay significantly. Which implies that a link-queue model does not provide an adequate abstraction to characterize delay in TSN for guaranteed deterministic delays.

### 4.2.1 *Behavioral analysis of TSN forwarding mechanisms*

The analysis of delay sources within Ethernet reveals queuing delay as the only source of delay that introduces significant variation in delay at the transmit port/interface. This variation is caused by how shapers/schedulers process packets from the respective queues. Among the features of TSN highlighted in Section 4.1, the basic oper-

ation of Stricit Priority Scheduler (SPS), CBS, and TAS are examined next.

STRICT PRIORITY SCHEDULER (SPS) The SPS operates on a simple principle: to provide preferential packet forwarding for packets within queues in the order of their priority. Based on this principle, an SPS will continuously forward packets from a higher priority queue until it is empty before moving to lower-priority queues. This behavior can lead to significant delays in lower queues and an unfair queuing system where lower priority queues are overly starved. Detailed operation of SPS in Ethernet-based networks are discussed in [58–60]

A distinct observation with SPS is, once a packet in a lower priority queue is put on the transmission line, a packet in a higher priority queue that just arrived at the transmission port has to wait till the lower priority packet has been forwarded. In real-time systems, this behavior is undesirable for Utra-Low latency (ULL) real-time systems. Therefore, preemptive SPS is used in real-time Ethernet networks. However, preemptive SPS cannot preempt the forwarding of a minimum PDU size as described in Worst-case delay example in Section 3.4.2. This, therefore, makes SPS inadequate for interference-free packet forwarding.

CREDIT-BASED SHAPER (CBS) SPS has two significant deficiencies; starvation of lower priority queues and delay of higher-priority packet by minimum PDU size. To address unfair queuing, the Audio-Video Bridging (AVB) group proposed CBS for audio and video traffic as part of IEEE 802.1Q standard. The main enhancement compared to the strict-priority scheduler is that it implements a fair scheduling discipline. Higher-priority queues do not overly starve packets in lower priority queues. Packets in higher-priority queues are forwarded based on credit accumulation; that is, a higher-priority queue is allowed to access the transmission line only if it has a non-negative credit accumulation.

The AVB specification defines two stream classes (Stream class A and B) which are mapped to two queues. Generally, the number of CBS-shaped queues is not limited. All queues access the transmission line on a strict priority basis if all queues have non-negative credit accumulation.

For credit-shaped stream/flow, four important parameters must be defined:

1. ***High-Credit***: this is the maximum credit that a queue can accumulate. This credit provides an upper bound on the number of bits than can be transmitted in one class measurement interval.

2. ***Low-credit***: defines the minimum credit of a queue. This credit represents the minimum level to which a credit may be depleted.

The time taken for credit to accumulate to a non-negative value represents the queuing delay, a higher priority packet which arrived just before credit reached the minimum level.

3. ***Send-Slope***: defines the rate at which credit decrease while packets are offloaded onto the transmission line. A packet can only be selected for forwarding if the credit of the queue is zero or greater. Credits can decrease to the limit defined by *Low-Credit*

4. ***Idle-Slope***: defines the rate at which credit increases whilst packets are not being forwarded from a credit-shaped queue. Credits can accumulate to the limit defined by *High-Credit*.

The parameters described above are derived from the bandwidth requirements (policy) for credit shaped streams. AVB stream classes A and B have standardized class measurement intervals, but they are freely adjustable in general.

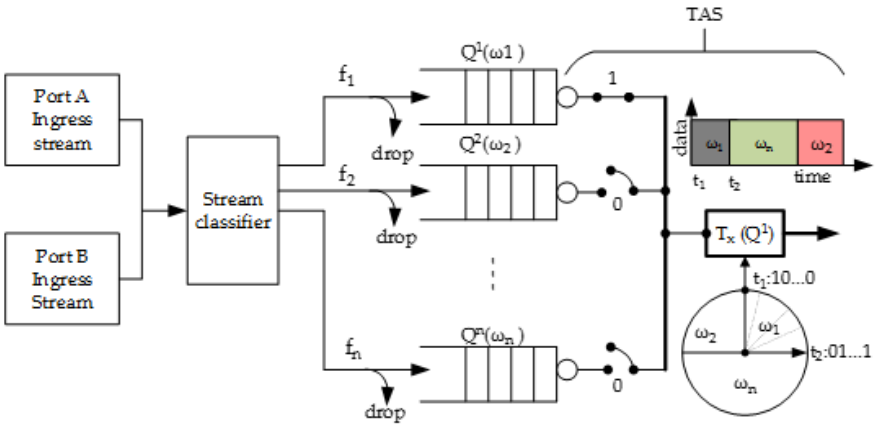



Figure 4.2: Time-triggered enhancement of Ethernet bridges for ULL traffics where packets assorted to priority queues are forwarded based on predefined time schedules. To enable frame forwarding from specific queue(s), the gate(s) of the queue(s) must be in an `open-state`= 1.

TIME-AWARE SHAPER (TAS) The Time-Aware Scheduler (TAS) was introduced by the TSN group to address the second deficiency of the SPS discipline; delay of higher priority packet by minimum PDU size from lower-priority traffic. TAS adds a gate to every queue. Packets cannot be selected from a queue when its gate is closed. At any point of time, zero, one, or more gates may be opened. If multiple gates are opened simultaneously, strict priority scheduling may be enforced between queues with opened gates. Figure 4.2, illustrates the operation of TAS on the egress/transmit port of an Ethernet bridge.

Assuming a perfect synchronization of bridges and end-stations (Talker and listener applications), end-to-end time schedules can be

calculated to ensure no queuing delay occurs for Time-Triggered (TT) packets. This is because all TT packets have exclusive, pre-planned time slots to forward packets on every bridge in its trajectory (path).

Note that this also means the ESs send the TT streams of packets precisely at a pre-planned point in time, otherwise the packets will miss their epoch and will have to wait for the next epoch. This causes undesirable interference with the next packet, if the latter is on time. A detailed work on how the time schedules are computed in SDIN is discussed in Chapter 5.

## 4.3 MODELING THE DELAY OF DETERMINISTIC SERVICES

In Chapter 3.4.1, the worst-case delay concept was introduced. TA and NC were identified as methodologies that can be applied to compute the exact worst-case delay.

In this section, the concept and methodologies are applied to derive delay models for dynamic slicing in the QoS-aware VLiM problem. First, a generic delay model is derived, it is then adapted based on the SPS, CBS and TAS forwarding mechanisms. Additionally, a model for correlated operation consisting of a tandem topology of TAS, CBS and SPS as an IT and OT convergence use-case is derived.

### 4.3.1 *Trajectory approach*

The TA is used to estimate a worst-case upper-bound on the delay at every hop (egress/transmit port) along a path taken by a packet from its origin to destination and summed up. Therefore, the e2e worst-case delay experienced by a packet using an NS (denoted $S_n$) along a path $\Upsilon$ is given by Equation (4.1):

$w_d^{e2e}$ =worst-case e2e delay
$p_d$ = worst-case port delay
$\Upsilon$ = number of hops in the path
$f_d$ = transmission delay
$\aleph_p$ = switching delay
$q_p$ =queuing delay
$\gamma_p$ =propagation delay

$$w_d^{e2e}(S_n) = \sum^{\Upsilon} p_d(S_n) \tag{4.1}$$

Where $p_d$ is the worst-case port delay of NS $n$ ($S_n$) at a *busy-period*. the total delay that occurs at the egress port is defined by Equation (4.2).

$$p_d(S_n) = f_d(S_n) + q_d(S_n) + \aleph_p + \gamma_p \tag{4.2}$$

Given the already bounded nature of the propagation ($\gamma_p$) and switching ($\aleph_p$) delays, it is important that the delay model centers on load dependent components such as queuing and forwarding delay (see Section 4.2).

Intuitively, a NS not experiencing any interference from another NS does not incur queuing delay as a result of packets admitted to other NSs. This means that a packet admitted to such an NS only experiences queuing delay from packets admitted to the same NS. However, since the worst-case delay guarantee is considered from a slice perspective, we can safely state that the delay guarantee of such

an NS at an egress port is the same as the transmission delay given by Equation (4.3):

$\sigma$ = *burst of NS*
$R$ = *transmission rate of egress port*

$$f_d(S_n) = \frac{\sigma(S_n)}{R} \tag{4.3}$$

Similarly, for an NS experiencing interference from other NSs admitted to the same queue, the queuing delay experienced at the port is given by Equation (4.4):

$S_i$ = *all interfering NSs of* $S_n$

$$q_d(S_n) = \frac{\sum_{i=1}^{K} \sigma(S_i)}{R} \tag{4.4}$$

APPLICATION ABSTRACTION The applications which use network services are often characteristically different, that is, some application may generate bursts (or burst of packets) periodically while others in a sporadic manner. Due to this basic characteristic differences, packets may be processed differently. This also implies that certain forwarding mechanisms are better suited for some application if QoS e. g., delay must be guaranteed.

In the case of non-sporadic periodic applications [1], TSN provides specialized forwarding mechanism in order to meet delay guarantees. For periodic applications with ULL requirements, TSN uses time-shaped forwarding which is covered under TAS (see Section 4.2.1), application with this characteristics are referred to as Time-Triggered (TT) applications in this thesis. For periodic applications with real-time requirements, non-TT periodic applications:— these are processed with credit-shaping mechanism; — it is covered under CBS. For sporadic non-periodic applications with real-time requirements, a simple priority forwarding mechanism is applied; this is covered under SPS.

Having discussed the generic computational models (i.e, Equations (4.2) and (4.1)), the derived Equations will be extended w.r.t to the operations of the individual TSN forwarding mechanisms described in Section 4.2.1. The following conditions must apply:

Condition (1) If one or more NSs are mapped to the same queue, packets of these NSs will be forwarded using FIFO approach.

Condition (2) Packets of NSs mapped to different queues are forwarded based on the priority of the respective queues.

#### 4.3.1.1 *Strict Priority Scheduler*

The basic principle of SPS is to expedite forwarding of packets in higher-priority queues ahead of lower-priority queues as discussed in Section 4.2.1. This means as long as packets are en-queued in a higher-priority queue, all packets in lower-priority queues will be delayed.

1 Applications with non-deterministic packet release time can generally be classified as sporadic regardless of whether they exhibit cyclic or non-cyclic packet repetitions [77].

Also, per Condition (1) and Condition (2), packets of different NSs will delay each other. Therefore, the worst-case port delay of a slice, $S_n$, assigned to the lowest priority queue, $p_d^l(S_n)$, is equivalent to Equation (4.2). However, there exists the possibility that the largest non-preemptive packet in any lower-priority queues either than the lowest priority queue can delay packets of any other queue.

To capture this effect, Equation (4.2) is extended to Equation (4.5) by introducing $\delta$, where $\delta = \frac{\tau}{R}$: $\tau$ is the number of bits of the largest non-preemptive packet in a lower priority queue. This often constitutes a packet data unit size of 1500 bytes.

$\delta$ = *largest non-preemptive packet*

$$p_d^u(S_n) = f_d(S_n) + q_d(S_n) + \aleph_p + \gamma_p + \delta \tag{4.5}$$

$p_d^u$ = *worst-case port delay for high priority queues*

$p_d^u$ is the worst-case port delay for a packet in any queue but the lowest priority queue. That of the lowest priority queue is obtained by neglecting $\delta$. Subsequently, the worst-case e2e delay of a packet in the lowest priority queue is given by Equation (4.1), (i.e. $w_d^{e2e(l)}(S_n) \equiv$ Equation (4.1)) whilst that of a packet in any other queue is obtained by substituting Equation (4.5) into Equation (4.1) to get Equation (4.6):

$w_d^{e2e(u)}$ = *e2e worst-case delay of NS in an upper queue*

$$w_d^{e2e(u)}(S_n) = \sum^{\Upsilon} p_d^u(S_n) \tag{4.6}$$

#### 4.3.1.2 *Credit-Based Shaper*

CBS maintains credits for each credit-shaped queue. While sending, credits are decreased at a rate defined by the parameter, send-slope. While not sending, credits are increased according to the idle-slope. Thus, send- and idle-slopes indirectly define the throughput or reserved bandwidth for a queue. The epoch for this measurement is called class measurement interval.

Hi-credit and low-credit parameters limit the committed burst size of the class measurement interval. If a queue has negative credit, no packet is selected from this queue, thus packets in lower priority queues are not overly starved by those in credit-shaped higher priority queues. In accordance with the AVB standard, two CBS queues for classes A (the highest priority) and B (lower credit-shaped priority queue) are defined.

The worst-case port delay CBS adds to a packet of a certain class occurs when a packet arrives at a time such that all credits for the class are depleted to the limit of its low-credit thus no frame will be selected for transmission until credit has increased to zero or higher.

Furthermore, just as the credits accumulate to zero, the packet may not be forwarded immediately if a non-preemptive packet ($\tau$) from a lower priority queue is on the transmission line. This means $\delta$ is also introduced just as for SPS. Therefore, the worst-case port delay for class measurement interval A is derived by Equation (4.7):

$p_d^A$ = *worst-case delay for measurement class A*

$H_l^A$ = *time to zero*

$$p_d^A(S_n) = f_d(S_n) + q_d^A(S_n) + H_l^A + \aleph_p + \gamma_p + \delta \tag{4.7}$$

where $q_d^A(S_n)$ is queuing delay due to packets of class A, which arrived while credit was negative and $H_l$ is the time taken to accumulate credit to a non-zero value. Similarly, for class B, $q_d^B(S_n)$ includes the queue delay of all packets in class B which arrived when credit was negative. If the credit of B rises to a non-negative value, and class A has a non-negative credit, packet of NSs in A will also delay those in B. Therefore, the guaranteed port delay for NSs in class B is given by Equation (4.8):

$q_d^B$ = *queuing delay due NSs in A and B*

$$p_d^B(S_n) = f_d(S_n) + q_d^B(S_n) + q_d^A(S_n) + H_l^B + \aleph_p + \gamma_p + \delta \tag{4.8}$$

$p_d^B$ = *worst-case port delay class B*

Subsequently, the e2e-worse-case delay guarantee of the respective classes A and B is obtained by substituting Equation (4.7) or (4.8) into Equation (4.1) to obtain Equations (4.9) or (4.10) respectively.

$w_d^{e2e(A)}$ = *worst-case e2e delay for NS in CBS class A*

$$w_d^{e2e(A)}(S_n) = \sum^{Y} p_d^A(S_n) \tag{4.9}$$

$w_d^{e2e(B)}$ = *worst-case e2e delay for NS in CBS class B*

$$w_d^{e2e(B)}(S_n) = \sum^{Y} p_d^B(S_n) \tag{4.10}$$

#### 4.3.1.3 *Time-Aware shaper*

Based on the operational characteristic of TAS, an unsynchronized system can lead to far more delays than SPS and CBS. TAS was introduced basically to ensure interference-free packet transmission where exchange of packet between a talker application (e. g., PLC) and Listener application(s) (e. g., IO-device like a drive) occurs as if there is direct circuit between them.

TT applications cannot operate well in worst-case scenario, therefore, worst-case delay computation is not necessary. However, computing the start-time and gate-events is an important research problem; this will be examined in Chapter 5. At the start of any gate open window, if only one queue has exclusive access to the transmission line, there is no queuing delay. Therefore, the only delay experienced at the egress port is the transmission delay. This is the case where the NS hosts TT applications. Therefore, port delay assuming packets arrive within their gate-open window is given by Equation (4.11).

$p_d^{tt}$ = *worst-case port delay of TT NS with ideal sych*

$$p_d^{TT}(S_n) = f_d(S_n) + \aleph_p + \gamma_p \tag{4.11}$$

Subsequently, the e2e delay is derived from Equation (4.11) and (4.1) as Equation (4.12).

$w_d^{e2e(tt)}$ = *worst-case e2e delay of TT NS with ideal synchronization*

$$w_d^{e2e(tt)}(S_n) = \sum^{Y} p_d^{tt}(S_n) \tag{4.12}$$

For non-TT NSs, multiple gates may be opened at the same time in which case, packets are forwarded based on queue priority. However,

all non-TT NSs experience queuing delay during gate-open windows of the TT NS which may send packets during the *busy-period*, therefore non-TT in upper priority queues experience a worst-case port delay given by equation (4.13).

$p_d^{nl}$ = *worst-case port delay for non-TT NSs in upper priority queues*

$$p_d^{nu}(S_n) = f_d(S_n) + q_d(S_n) + \aleph_p + \gamma_p + \delta + \chi^{tt} \tag{4.13}$$

The lowest priority does not experience delay due to $\delta$ since it emanates from the same queue, it is already considered in the intra-class queuing delay, therefore, the worst-case port delay is given by Equation (4.14).

$p_d^{nl}$ = *worst-case port delay for non-TT NSs in lowest priority queue*

$$p_d^{nl}(S_n) = f_d(S_n) + q_d(S_n) + \aleph_p + \gamma_p + \chi^{tt} \tag{4.14}$$

Subsequently, the e2e worst-case delay for non-TT NSs are obtained by substituting Equation (4.13) or (4.14) for upper or lowest priority respectively in Equation (4.1).

$\chi^{tt}$ = *sum of gate-open duration for TT within a busy-period*

### 4.3.2 *Tandem composition for converged OT and IT*

IT and OT convergence can be achieved in two ways;— using specific TSN features based on specific requirements of the communication pyramid (see Chapter 2), or — combining all TSN forwarding features at an egress port. For the former, the derived worst-case computation models are sufficient for slicing. However, the later requires correlating the operation of individual shaper/schedulers in a tandem composition as shown in Figure 4.3.

In the Figure 4.3, it can be observed that not all the shapers apply directly to all queues at the egress port of the switch. Nevertheless, depending on the composition of these shapers with respect to the corresponding queues, queues reserved solely for TT, or Credit Based (CB) or normal priority traffic are referred in the following; TT-queues refer to queues reserved specifically for periodic time-triggered flows with ULL requirements, CB-queues refer to queues reserved specifically for periodic flows with critical delay requirements but are not TT, and Priority Based (PB)-queues refer to queues that hold non-periodic flows with relaxed delay requirements.

It can also be observed that the tandem composition of shaper/scheduler introduces an inherent precedence to packets of NSs admitted to the respective queues. TAS affects all queues directly. Each TT-queue (Q1 & Q2) has an exclusive time window, configured so that packets from each of the queues experience no interference of any form. All other queues share the remaining time of the scheduling cycle.

CBS affects Q1 to Q7 directly; however, only Q3 and Q4 are managed by credits, all queues not managed by credits have to be configured with infinite *idleSlope* and a zero *sendSlope* as shown in Figure 4.3. CB queues Q3 and Q4 are configured to have finite *idleSlope* and *sendSlope* based on specific allocation for CB packets. Packets in all other queues

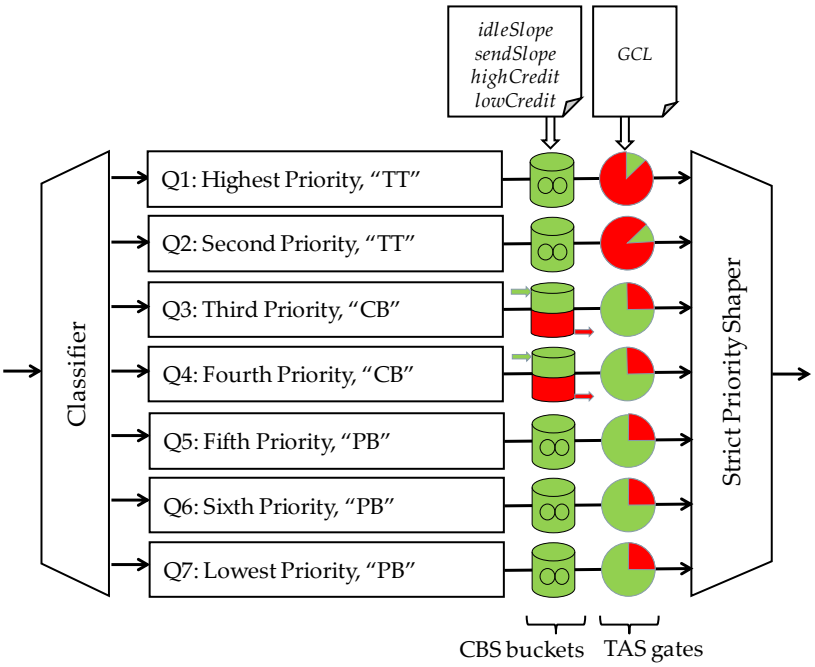



Figure 4.3: Correlation of shapers and queues.

(i.e. Q5→Q7) are scheduled directly by SPS within a non-TT window.

With this composition, Q1 and Q2 are prioritized ahead of all the other queues within a busy-period, i.e. they get the pre-planned time window required to forward all planned packets without any queuing delay. Based on the correlated operation of the tandem shaper/schedulers, the derived Equations for the individual schedulers are revised as follows:

- TT-NSs maintain the same Equations as defined in Equations (4.11) and (4.12).
- for CB-NSs, Equations (4.7) and (4.8) must take into account the delay introduced by the TT traffic during a busy-period, this results in Equations (4.15) and (4.16):

$$p_d^A(S_n) = f_d(S_n) + q_d^A(S_n) + H_l^A + \aleph_p + \gamma_p + \delta + \chi^{tt} \tag{4.15}$$

$$p_d^B(S_n) = f_d(S_n) + q_d^B(S_n) + q_d^A(S_n) + H_l^B + \aleph_p + \gamma_p + \delta + \chi^{tt} \tag{4.16}$$

- non-TT/non-CB NS must as well take into account the queuing delay introduced by the TT during a busy-period as well as CB. Therefore, Equations (4.5) and (4.2) are modified to derive Equations (4.17) for high priority non-TT/non-CB and (4.18) for the lowest priority queue as follows:

$$p_d^u(S_n) = f_d(S_n) + q_d(S_n) + \aleph_p + \gamma_p + \delta + \chi^{tt} \tag{4.17}$$

$p_d^l$=worst-case port delay for lowest priority queue

$$p_d^l(S_n) = f_d(S_n) + q_d(S_n) + \aleph_p + \gamma_p + \delta + \chi^{tt} \tag{4.18}$$

$q_d(S_n)$ inherently captures the delay introduced by the CB-NSs. The e2e worst-case delay for the respective NSs are obtained by substituting Equations (4.15), (4.16), (4.17), and (4.18) into Equation (4.1).

## 4.4 SLICING FOR DETERMINISTIC APPLICATIONS

This section describes and discusses how the mathematical models derived from the analysis above can be used to create NSs for guaranteed service deployment. The analysis is done in the context of deterministic applications. This implies, the committed burst of the application does not exceed a known threshold.

The NS creation process represents a demand-resource relation as discussed in Section 3.2.1, where a set of demands are verified against resources abstracted from the substrate network. The demands impose constraints on the solutions computed with the resources. The mapping process ensures that demands are verifiable through a preceding verification process before an NS is embedded. During the embedding process, resources are updated based on the mapping solutions that fulfill all demands for a given NS.

Resources are occupied when demands are embedded. This implies, if a demand (NS requirements on, e.g. bandwidth, delay.) is embedded in a substrate entity, the relevant resource is exclusively reserved, resulting in residual resource that is used in the subsequent embedding of other NSs. All the processes described above are undertaken by a VNE algorithm which can be hosted in a network controller in an online scenario, or offline in a planning tool. The algorithm uses a model of the NS which is represented as VNRs with demands and the SN model composed of abstracted network resources. [2]

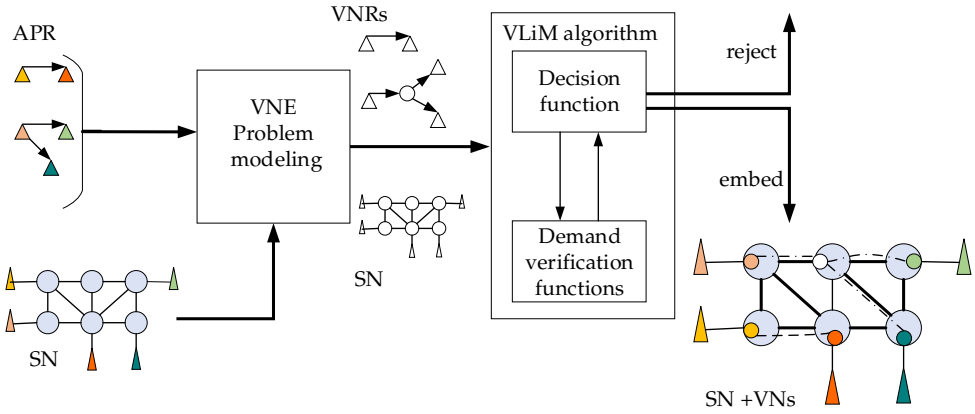



Figure 4.4: NS creation and VNR embedding process

### 4.4.1 *Network slice creation process*

As discussed in Chapter 3, the accuracy of solutions computed by the VNE algorithm is dependent on how precise the underlying substrate network is modeled to reflect the QoS mechanism within the network. Likewise, the accuracy of the VNR model to capture the relevant characteristic details of applications using the NS.

Figure 4.4 illustrates the entire process for NS creation. As shown, the process begins with the abstraction of the Substrate Network (SN) and characteristics of the applications for which the NS is created. This process is termed VNE problem formulation or modeling. In the problem formulation, attributes such as AR, which describes the interaction between the application as well as the Communication Relation (CR) which describes the QoS requirements are modeled into VNR graphs ($G^v$).

Similarly, the SN network entities and connectivity are represented as an SN graph ($G^s$). The SN graph must be represented in a manner that integrates all the necessary features of the SN that influence the problem space. That is, without an precise representation, solution computed becomes inaccurate for deployment in production networks.

Furthermore, the VNE algorithms use demand verification functions which can be formal models that estimate delay, reliability, bandwidth, etc., to compute solutions as discussed in Chapter 3. These verification functions may also include simple operations such as matching identifiers of the VNR node to SN nodes in scenarios where the VNoMs are uniquely defined. If one or more demand verification asserts false, the decision function of the VLiM algorithm rejects the VNR in question. If all demand verification asserts true, the set of resources used becomes a possible solution. If more than one possible solutions exist for a specific VNR, the underlying orchestration approach of the decision function determines which resources are allocated to the VNR. The decision logic embodies the resource orchestration hypothesis of the VLiM algorithm designer. The different approaches are discussed in Section 4.4.3.

In the context of the embedding process, the QSL model (see Figure 4.1) must be integrated in the SN model whilst the e2e worst-case requirements of an NS and the type of applications admitted to the NS are abstracted in the VNR model. This enables VLiM algorithms to use the worst-case delay computation functions described in Section 3.4.1 according to the QoS forwarding capabilities of the network.

2 Application Relation (APR) is sometimes used interchangeably with with Application Relation (AR)

### 4.4.2 *TSN substrate network model*

The substrate network is modeled as a non-reflexive directed graph $G^s = (\mathcal{V}^s, \mathcal{E}^s)$, where $\mathcal{V}^s = \{v_1^s, v_2^s, \cdots, v_N^s\}$ is a set of $N$ substrate nodes and $\mathcal{E}^s = \{e_{ij}^s\}$, $i, j = 1, 2, \cdots, N$ is a set of substrate edges such that $e_{ij}^s$ represents the connectivity of $n_i^s$ to node $n_j^s$. Considering the non-reflexive property of $G^s$, $e_{ij}^s = 0$ if $i = j$, which implies loop-back edges are not considered as resources. For any edge to be considered as a resource, $\forall e_{ij}^s : \ i \neq j$). The superscript $s$ denotes that the resource belongs in a physical plane (DP).

EXTENSION TO THE EDGE MODEL USING QSL MODELS The substrate edge notation is further extended as $e_{p_u^{n_i}, p_v^{n_j}}$ where $p_u$ and $p_v$ denote the interfaces that comprise the source and destination of the edge on nodes: $u, v = 1, 2, \cdots$. The use of directed graphs implies the QSL model must integrate directly in the source, i.e. $p_u$, as the source of the edge represents the transmit interface of the bridge. Additionally, the capabilities and attributes of an interface such as transmission rate or bandwidth ($R \equiv b$), propagation delay ($\aleph_p(e_{p_u^{n_i}, p_v^{n_j}})$), switching delay ($\gamma_s(e_{p_u^{n_i}, p_v^{n_j}})$), the forwarding and queuing delay of the respective forwarding mechanisms discussed in Section 3.4.1 (i.e $f_d(e_{p_u^{n_i}, p_v^{n_j}})$. $q_d(e_{p_u^{n_i}, p_v^{n_j}})$), must be associated to the edge. This enables a VLiM algorithm to compute accurate results. An example of an SN definition file which integrate the QSL model is shown in Appendix A.3 (Listing A.1).

**Virtual Network Request Model:** The NS is modeled as a non-reflexive directed graph $G^V = (V^v, L^v)$, where $V^v = \{v_1^v, v_2^v, \cdots, v_N^v\}$ is a set of $N$ virtual vertices representing the endpoints of the NS. Each NS endpoint maps unambiguously to a substrate node in the substrate network described earlier. The superscript $v$ denotes a virtual plane. $L^v = \{l_{ij}^v\}$, $i, j = 1, 2, \cdots, N$ is a set of virtual links where $l_{ij}^v$ shows that NS endpoint $v_i$ is connected to NS endpoint $v_j$. The definition of the VNR requires additionally a definition of requirements associated with vertices and their connectivity termed demands (see section 3.1).

For each link in the NS, a set of link demands are defined. This includes maximum load or burst (bandwidth) and delay requirements of the NS, denoted by $B_{max}^{l_{ij}^k}$, and $D_{max}^{l_{ij}^k}$ respectively. $B_{max}^{l_{ij}^{v_k}}$ is expressed in bits per second. Therefore, in other to define an operation on the bandwidth in $G^s$, the periodic burst of the NS must also be expressed in bits per second. This implies, for NS carrying periodic streams, the bandwidth $= \frac{\sigma}{x}$ where $x$ is the period of the NS.

Though this simplifies the model, it also introduces pessimism in the results. This is because some applications do not generate a single burst every second. As illustrated in the worst-case delay analysis in Chapter 3, there can be several busy-periods in a second. For the

delay computation, an accurate result can only be obtained if burst per period is used instead of burst per second.

The up-side of this representation, on the other hand is, since bandwidth is expressed per second, it provides a straight forward addition or subtraction operation to verify bandwidth constraints. Appendix A.3 provides an example of NS request file A.2.

### 4.4.3 *Role of decision functions in embedding process*

As discussed previously in Chapter 3, VNE problems are designed with two main objectives (see section 3.2.1). Objective 1 is particularly important for NS creation. That is, QoS-aware algorithms are required to firstly verify constraints imposed by the demands of VNRs on the resources of the SN before the resource(s) are committed to the network.

For the QoS-aware VLiM problem, two important steps are required; Path computation and Path selection.

PATH COMPUTATION is an important step in the VLiM problem. In this step, the QoS demands of VNRs are verified as constraints on the resources of the SN. Once all constraint verification assert true on a set of resources, the resources become possible solutions to the VLiM problem. This implies for each constraint, a verification function is required. In section 4.3, delay computation models were derived. These models are used by the VLiM algorithms as delay-constraint verification functions for path computation. The path computation process is discussed together with the path selection process in Chapter 5.

PATH SELECTION before a virtual link can be mapped, it is mandatory to find a path between substrate nodes where the endpoints are mapped and verify if the demands can be satisfied along the path (see path computation 4.4.3). Path computation can result in several possible solutions to a single mapping processes. However, for every VNR, a single solution must be selected among the several solutions available. From the perspective of a VNR, the path selection is not so relevant but, from a resource utilization view point, it is essential how paths are selected from the solution space. This aspect of the problem deals with the second objective of the VNE problem (see Objective 2), which is discussed Chapter 5.

## 4.5 SUMMARY OF CHAPTER AND BIBLIOGRAPHIC COMMENTS

### 4.5.1 *Bibliographic comments*

The realization of virtualized networks on physical infrastructures depend on achieving two major goals;— embed virtual networks in

an efficient manner such that the SN is efficiently utilized; — ensure that embedded VNs can guarantee the QoS requirements of the applications that use them. While the former depends on effective techniques and algorithmic approach for resource orchestration within the problem space, the latter depends solely on the characteristics of the applications that use the instantiated VNs and the QoS features of the SN.

In literature, several researchers have investigated the QoS guarantees of virtualized infrastructures under the theme "QoS-aware VNE" where bandwidth and latency (delay) are the predominant guarantees considered [78–84]. A significant occurring problem with these literature is the homogeneous treatment of delay which seemingly asserts that latency is mainly dependent on traffic intensity [49]. However, the analysis provided in this thesis proves contrary to this assertion, especially for ULL applications in TSN-enabled LANs.

One may argue that at a higher layer such as network layer (L3), the effect of shapers and schedulers do not significant impact some applications' delay requirements; hence, these assertions are valid. However, to cover a broader spectrum of applications, delay guarantees must reflect the effects of shapers and schedulers.

The estimation of delay especially in Ethernet switched networks is broadly studied in [52–60]. Although there exist several methodologies in literature as discussed in the Ph.D thesis of Xiaoting Li [51], not all methodologies can be applied in the QoS-aware VNE problem within the context of SDN. This because they are mostly simulations of the real systems [52–54]. The application of VNE for NS creation requires accurate estimations to determine whether an NS can surely guarantee delay demands before they are instantiated, therefore, it calls for exact approaches.

Two methodologies known for exact worst-case delay estimations are the Trajectory Approach (TA) and Network Calculus (NC). Notably, TA which provides a simplistic algebraic analysis, has been applied to real-time switched Ethernet networks where the effect of schedulers and shapers such as the SPS and CBS have been studied [58–60].

On the other hand, NC is based on Min-Max-algebra and has also been used in similar analysis like the TA [61–63]. Though results have been reported to be very pessimistic, the theory's pessimism has not be proven. Though this thesis does not claim to prove it nor analyze NC, an analysis performed in comparison to TA shows that the pessimism does not originate from the theory itself but the representation of arrival curves in relation to the busy-period introduces the pessimism. Aside from this pessimism, the mathematical operations of NC are complex in comparison to TA hence requires more effort to compute solutions.

### 4.5.2 *Summary of chapter*

Service guarantees of an instantiated Virtual Network (VN) depend on the Quality of Service (QoS) features of the underlying Substrate Network (SN) as well as the characteristic of applications that use them. Therefore, the chapter examines in closer detail, components of the SN which contribute to delay guarantees.

Based on the analysis of Time-Sensitive Networking (TSN)-enabled Local Area Network (LAN), a Queue-Scheduler-Link (QSL) model is derived to characterize the effect of the various components of delay emanating from the effect of queues, shapers/schedulers and links.

Furthermore, a Trajectory Approach (TA) is leveraged to develop computational models using the QSL model by considering the characteristics of applications and the interaction of their flows within TSN-enabled LANs. The QSL models are subsequently integrated into an SN model as part Virtual Network Embedding (VNE) problem formulation which enables Virtual Link Mapping (VLiM) algorithms to use the derived computational functions to verify exact delay guarantees due to embedded and yet to be embedded VNRs. This, in effect, ensures QoS-aware VLiM algorithms can provide delay guarantees for all kinds of applications that use an VN.

The delay models address the effect of schedulers and shapers such as Stricit Priority Scheduler (SPS), Credit Based Shaper (CBS) and Time-Aware Scheduler (TAS) when used individually or as the tandem configuration. The delay models can be used to ensure different kinds of services regardless of the delay requirements of an application.

# 5

# DESIGNING VNE ALGORITHMS FOR SDIN-ENABLED TSN

*contributions in this chapter emanates from [48, 85]*

In this chapter, the methodologies for the design of VNE algorithms that address Objective 2 of the VNE problem (see 3.1.2) are discussed. This begins with the examination of state-of-the-art approaches followed by the proposed Demand Based Approach (DBA), designed specifically for the multi-constraint online VLiM problem. Under the DBA, demands are examined from two perspectives; numerical attributes of the demand ( $\in \mathcal{R}_0^+$), which address QoS,—and non-numerical attributes, which address the communication pattern (e. g., multi-cast or uni-cast communication pattern).

## 5.1 ALGORITHMS FOR MULTI-CONSTRAINED VLIM PROBLEM

In chapter 3, fundamental principles for designing VNE algorithms were examined. In this section, we will apply these principles to design algorithms for multi-constrained QoS online VLiM problems within the FSFD use-case. A Demand Based Approach (DBA) which is proposed in this thesis will be compared to published approaches in literature.

As discussed in section 4.4.3, path selection is a vital process that ensures efficient utilization of the Substrate Network (SN) resources within the VLiM problem. It is often misconstrued to be a requirement addressed by optimization methodologies such as Mixed Integer Programming (MIP) or Integer Linear Programming (ILP). This perception is based on the assertion that MIP/ILP algorithms yield the most efficient results. However, critical analysis of the problem will reveal that it is the underlying resource orchestration approach that determines the efficiency of a VLiM algorithm.

A review of related work [37, 38] reveals a common resource orchestration approach used in literature for path selection in the VLiM problem. This approach is categorized as Shortes Path First Approach (SPFA) or bandwidth-minimization.

SPFA selects paths for the embedding process based on the richness of resources in the SN, i. e., whether the path provides the least cost among several feasible solutions. The subsequent subsections examine in detail the SPFA.

### 5.1.1 *Shortest path first approach*

**Definition 5.1.1.** Shortest Path First Approach: is an approach that selects the least-cost path among several possible options that fulfills a demand.

From definition 5.1.1, the Shortes Path First Approach (SPFA) is in fact a Greedy Approach (GA) which is synonymous with the shortest path routing algorithms. The use of this approach in the VLiM problems can be observed in MIP-based VLiM algorithms which are defined mainly to minimize resource usage in order to increase the acceptance of VNRs.

To proof this assertion, let's examine VLiM algorithms proposed and discussed in [32, 33, 86, 87]. An objective function that summarizes concisely the core approach used in these related work is expressed by Equation (5.1):

$y^g$ = *request decision variable*
$x_{ij}$ = *is a link decision variable*

$$\max\{\sum_{g=1}^{G} y^g - \epsilon \cdot \sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij} \cdot x_{ij}\} \tag{5.1}$$

$b_{ij}$ = *bandwidth attribute of links in SN*
$\epsilon$ = *normalization coefficient*

The objective function consists of two parts —a maximization part which guides the general output of the algorithms, — and a minimization part that determines the approach used by the algorithm's designer.

The minimization part ($\sum_{i=1}^{N}\sum_{j=1}^{N} b_{ij} \cdot x_{ij}$), minimizes the bandwidth, ($b_{ij}$), used for a demand, in other words minimizes the number of links. However, considering constraints imposed by the demands, demands can either be fulfilled or not. This implies if a VNR has a demand of 2Mb/s, the whole demand must be fulfilled on every link within the path. This means the approach of bandwidth minimization is in fact link minimization since bandwidth is concave (non-additive) attribute that must be satisfied by every link within the path.

This minimization aspect of the MIP based optimization algorithms will produce the same result as an algorithm that applies a heuristic shortest path algorithm multiple times within the VLiM problem. This is because both have an inherent design objective to find the shortest path first for every VNR. The SPFA selection design objective whether MIP optimization or heuristic based, comes at a cost of greater rejections in future requests due to its greedy nature. Particularly, demands with stringent requirements when considered in the scope of multi-constrained VLiM [34, 39].

#### 5.1.1.1 *Challenges with SPFA in the multi-constrained VLiM*

To illustrate deficiencies of the SPFA for multi-constrained VLiM problem within the FSFD use-case, Figure 5.1 describes a simple example of the problem. The problem requires mapping of VNRs with the

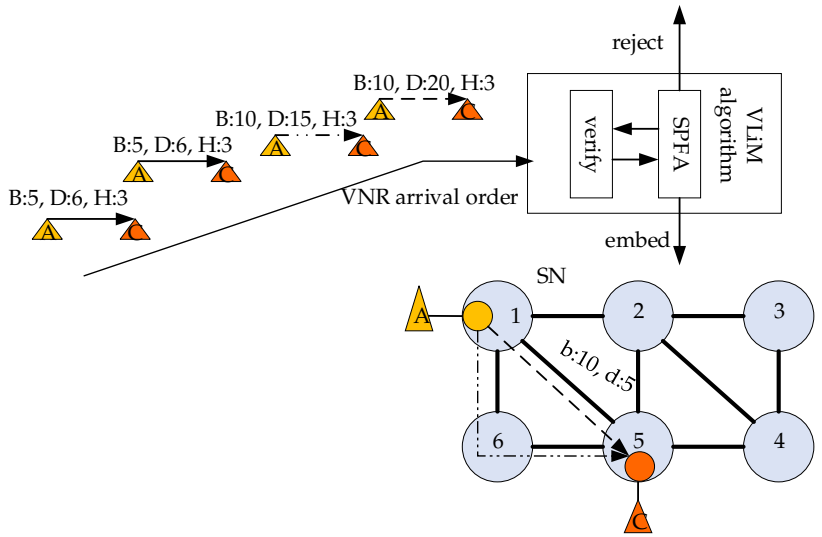



Figure 5.1: Deficiency of SPFA decisions in the bandwidth-delay constrained on-line VLiM problem

demands; bandwidth (B), delay (D), hop-count (H) on the SN using an online VLiM algorithm hosted as a network function in a network controller. The controller provides up-to-date substrate network resource and attributes to enable the verification functions to compute mapping solutions for each VNR. The SN has links with 10 units capacity and 5 units delay.

An SPFA VLiM design objective (decision logic) results in the mapping of $VNR_1$ on substrate link $1 \rightarrow 5$ and $VNR_2$ on substrate links $1 \rightarrow 6$ and $6 \rightarrow 5$. In the event of $VNR_3$ and $VNR_4$ arriving, there are no suitable resources with the attributes to meet their demands. Therefore, $VNR_3$ and $VNR_4$ are rejected despite available bandwidth on links ($1 \rightarrow 2$, $2 \rightarrow 5$), ($2 \rightarrow 4$, $4 \rightarrow 5$), and ($2 \rightarrow 3$, $3 \rightarrow 4$, $4 \rightarrow 5$). This results in low substrate bandwidth utilization which subsequently leads to low embedding ratio of requests with tighter demands and the overall embedding ratios of the algorithm.

In this simple illustration, it can be observed that the path selection decision leads to such an outcome. It further asserts that efficiency of the VLiM algorithm does not depend on formal methodologies such MIPs/ILP but fundamentally the algorithm designer's approach to path selection. This has been duly noted by Trivisonno and Guerzoni et. al. [34, 39]. This path selection approach leads to the following

problems:

- Rejection of future delay critical requests and subsequently lower VNR embedding ratios [34];
- Under utilization of the physical infrastructure due to saturation of critical substrate links (delay critical bandwidth) [34, 39, 86];
- High computational time due to the need for re-mapping of requests.

To address this problem, a Demand Based Approach (DBA) to path selection is proposed in this thesis. The description and design of DBA algorithms for FSFD use-case in the online VLiM problem is discussed next.

### 5.1.2 *Demand-based approach*

**Definition 5.1.2.** DBA is an approach that selects the highest-cost path among several possible paths that fulfills a demand.

The DBA is derived from a careful analysis of the multi-constrained online VLiM problem. The online nature of the problem present two fundamental challenges considering Objective 2, where a decision on path selection must focus on increasing the overall acceptance ratio as well as efficient utilization of the SN.

The underlying hypothesis for the design is, if path selection is considered in the context of the best solutions alone as done in routing algorithms, critical demands may be declined as illustrated in Figure 5.1. Given that the characteristics of the request are unknown before processing begins, it is better to make path selection decisions that serves a demand based on its requirements.

From definition 5.1.2, the goal of DBA is to find a solution that serves a demand but not necessarily the best solution in the problem space. Therefore, ensures that solutions are selected appropriately per demand, leaving enough room for future critical demands. DBA algorithm (Demand Based virtual Link Embedding Algorithm (DBvLEA)) computes all possible solutions that can satisfy the demands of a VNR.

Given that non-additive attributes of demands must be fulfilled on every link on a path, DBA focuses on the additive attributes of the demand such as delay, hop-count and jitter. For each set of additive demand, it selects one that is most relevant to the applications that will use the services of the VN if instantiated. Then, it selects the path which results in the minimum deviation to the additive attribute of the solution.

Still referring to Figure. 5.1, upon the arrival of $VNR_1$, a DBA algorithm computes all feasible paths between node A and C that satisfy

the hop-count, delay, and bandwidth attribute of the demand, i. e., ([B: 10, D: 20, H: 5]). It then chooses, for example, if delay is the focus of applications that will use the VN, the path which minimizes the difference of the actual path delay and the maximum delay of the demand. In the case of $VNR_1$, path ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$) is chosen.

Applying the same logic upon arrival of $VNR_2$, $VNR_3$ and $VNR_4$, will result in path ($1 \rightarrow 2 \rightarrow 4 \rightarrow 5$) and ($1 \rightarrow 2$) chosen respectively for $VNR_2$, $VNR_3$. This allows $VNR_4$ to be mapped to path ($1 \rightarrow 2$) as well. In this regard, a DBA algorithm achieves 100% acceptance ratio compared to 50% acceptance ratio of any SPFA algorithm.

In effect, considering a multi-constrained online VLiM problem in the FSFD use-case, a DBA algorithm will always perform better than an SPFA algorithm w.r.t Objective 2. The next section describes the design of DBA algorithm within the online VLiM problem using MIP/ILP and heuristic methodologies.

### 5.1.3 *Designing DBA algorithm for online VLiM problem*

Following the general methodologies used in literature, a MIP version and heuristic versions of the DBvLEA are designed. The use of a combination of heuristic and MIP are emphasized to reduce the number of decision variables that make any MIP problem *NP-complete*. This is to enable the computation of solutions in a pseudo-polynomial time if not polynomial time. That is, reducing the solution space to only feasible paths before optimizing for the embedding path. This is done by using loop-free K-path algorithms to compute a set of paths that fulfill the constraints and then perform a minimization of the deviation of the paths that fit the solution space of the VNR demands.

In effect, the number of equations (which is quadratic) that are required to solve the problem is reduced, resulting in a pseudo-polynomial time complexity.

DBA: MIP FORMULATION The general MIP objective function is shown in Equation (5.2). Contrary to [33, 34, 39, 86], the variable $d_{ij}$ is used in the objective function instead of $b_{ij}$. This is because bandwidth requirements impose non-additive constraints and must be fulfilled on every link along a path. Therefore, it is logical to use an additive (convex) constraint such as delay or hop-count since additive attributes of the demand impose constraints that are aggregated for the whole path rather than a single link in the path.

$d_{ij}$ = *delay on a link from node i to j*

$b_{ij}$ = *bandwidth on link from node i to j*

$D_{max}$ = *maximum delay constraint*

$$\max\{\sum_{g=1}^{G} y^g - \epsilon \cdot ((\sum_{i=0}^{N}\sum_{j=1}^{N} d_{ij} \cdot x_{ij}) - D^g_{max})\} \tag{5.2}$$

$i = j = 0 : i, j = 1, 2, \cdots, N$ where $N$ is the number of nodes, $G$ is the

$\epsilon$ = *normalization coefficient*

total number of VNRs, $y^g$ and $x_{ij}$ are decisions variables such that:

$$y^g = \begin{cases} 1 & g^{th} \text{ VNR is mapped} \\ 0 & \text{otherwise} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1 & y^g \xrightarrow{is\ mapped\ in} e^s_{ij} \\ 0 & \text{otherwise} \end{cases}$$

Based on the reduction technique discussed, Equation (5.2) can be reformulated as Equation (5.3), where all paths are in the feasible solution space for the $g^{th}$ VNR.

$$\max\{\sum_{g=1}^{G} y^g - \epsilon \cdot \sum_{j=1, p_j \in \mathcal{P}}^{K} (d(p_j) - D^g_{max})\} \tag{5.3}$$

where $\mathcal{P}$ is the set of paths within the solution space of the $g^{th}$ VNR, $K$ being the total number of candidate paths and $j = 1, 2, \cdots, K$.
The objective function for VNR mapping for Equation (5.2) and (5.3) are subject to the following constraints:

$$e^s_{ij} \cdot x_{i,j} \neq 0 \tag{5.4}$$

$$B^k_{max} \leq b_{e^s_{ij}} \cdot x_{ij} \tag{5.5}$$

$$\sum_{i=0}^{N} \sum_{j=1}^{N} d_{e^s_{ij}} \cdot x_{ij} \leq D^g_{max} \tag{5.6}$$

$$\sum_{i=0}^{N} \sum_{j=1}^{N} e^s_{ij} \cdot x_{ij} \leq H^g_{max} \tag{5.7}$$

$\forall\ e^s_{ij}, d_{ij}, \in \mathcal{E}^s, \mathcal{A}^s_{e^s_{ij}}$ and $D^g_{max}, H^g_{max} \in D^g$.

Equation (5.4) is a topology constraint which imposes that a VNR can be mapped to a substrate resource (link) only if it exists.
Inequations (5.5) to (5.7) are constraints imposed by the QoS demands $(D^g_{max}, H^g_{max})$ of the $g^{th}$ VNR respectively.

$\delta^g$=*deviation variable of the* $g^{th}$ *VNR*

Algorithm 5.1: DBvLEA; Computing ODP

```
Input: g ∈ G^v, G^s
Output: y^g, p_odp
begin
    initialize: y^g ← 0, p_odp ← 0, δ^g ← inf.,P ← ∅;
    P ← ComputePaths(g, G^s, B^g, D^g, K);//Yen's K path algorithm
    for p ∈ P: do
        if D^g − d(p) ≤ δ^g: do
            δ^g ← (D^g − d(p)); p_odp ← p;
    endfor;
    y^g ← embed(g, p_odp);
```

```
11      if y^g == 1: do
            updateResource(G^s, B^g);
            return y^g, p_odp;
    end
```

Considering Equation (5.2), the time complexity for finding a least-cost path in the minimization part of the objective function is $O(N^2)$, since there are $\approx N^2$ decision variables. This is a quadratic function with $\approx G \cdot N^2$ decision variables that must be exhaustively analyzed by the branch and bounds algorithms used in MIP solvers. For very large topology, this complexity may be reduced by introducing some heuristics before selecting the Optimal Demand Path (ODP).

DBA: HEURISTIC ALGORITHM The heuristic version of the DBvLEA algorithm that uses an objective function similar to Equation (5.3) is illustrated in Algorithm 5.1. In line 5, Yen's *k-path* algorithm is used to compute all paths that meet the demands of the $g^{th}$ VNR. This ensures that all paths are within the solution space of the $g^{th}$ VNR before demand based optimization is performed. Line 6 to 9 computes the optimal demand embedding solution as described in Equation (5.3). The path that sufficiently fulfills the demand is referred to as Optimal Demand Path (ODP), ($p_{odp}$). Lines 10 to 12 embed the $g^{th}$ VNR and updates the resources and attributes of the SN respectively.

The idea of reduction is to use an algorithm that converges in at least pseudo-polynomial time as a filter to compute all possible paths that fall within the solution space of the $g^{th}$ VNR before selecting an ODP. To achieve this, the DBvLEA algorithm starts by leveraging the Yen's loop-free k-path algorithm, which has a time complexity of $\approx O(K \log N(M+N \log N))$ .

*M= number of links*
*K= number of paths*

The combination of Equations (5.3) and Yen's loop-free K-path algorithm yields a time complexity of $O(K \log N(M+N \log N)+ K)$ for a single VNR. For a total of G VNRs, a time complexity of $O(K (G+\log N(M+N \log N))$ is achieved.

Evaluation of the SPFA and the DBA algorithms are discussed in Chapter 6.

## 5.2 APPLICATION TOPOLOGY-AWARE VNE

### 5.2.1 *Application and Communication Relation*

At the core of distributed automation systems is the necessity for reliable exchange of information. Any attempt to steer processes independently of continuous human interaction requires the flow of information between Automation Function (AtF) such as sensors, controllers, and actuators of some sort [2, 88].

On a functional level, Olaya et al. [89] describe AtFs as entities

within automation systems which represent identifiable actors of the productive environment where the relationship between AtFs are characterized by AR. Per the definition from IEC 61158 [90], ARs may represent material, energy or information exchange between AtFs. Whenever such exchange refer to information, it is characterized by Communication Relation (CR). Therefore, CRs describe the communication requirements of information exchange within an AR.

Although ARs and corresponding CRs are specified as abstract functional requirements of the productive environment, industrial networks are required to allocate resources that realize the CRs of an AR within a productive environment on a physical network. This requires the mapping of the functional viewpoint on the physical network.

Often, ARs manifest certain attributes that impose strict topologies on the communication pattern described by their CRs. A common occurrence which often determines how legacy industrial networks are constructed especially in factory automation results in network topologies such as trees, rings and lines (bus) [91]. These kinds of topologies are optimized to serve the productive environment.

Within TSN-enabled LANs, such specialized topologies do not exist. However, they can be constructed by VLANs derived by decentralized tree algorithms specified in IEEE802.1Q. Whilst VLANs can often be very difficult to deal with in terms managing individual CRs, this challenge can be ameliorated by the use of Network Slice (NS) that are instantiated as VNs within the SDN controller, where resources for individual CRs can be managed in a virtualization plane before they are committed to the SN.

Achieving resource allocation especially for tree based topologies requires specialized VLiM algorithms. However, due to the FSFD nature of slicing within the SDIN context, varied solutions may manifest due to the root node selection [91]. Root nodes do not occur in ARs nor their CRs hence an easier or faster way to define the VNR is to use network endpoints which peer with ARs or CR endpoints.

However, this approach does not necessarily lead to efficient use of the SN and often requires significant configuration effort. The subsequent sections describe the design of algorithms for mapping NSs that exhibit tree topologies as a result of the communication pattern imposed by the relation between applications that use them.

### 5.2.2 *Tree mapping algorithm*

Contrary to the previous approach, tree mapping algorithms take into account the definition of the VNR before applying VLiM for respective links in the VNR. This is because the root node is unknown from an AR/CR perspective. However, depending on the root node, solutions produced can be very different.

For a given CR emanating from an AR, a root node is selected for

Talker/Listener(s) tuple to form a tree. The root nodes then serves as the anchor node from which each link within the VNR is mapped using a VLiM algorithm. The root nodes can be selected based on 3 criteria:

- Listener Root Bridge (LRB) selects a common bridge within the SN to all Listeners in the tuple.
- Talker Root Bridge (TRB) selects a bridge within the SN connected directly to the Talker within the tuple.
- Rendezvous Root Bridge (RRB) selects a bridge within the SN that is central to Talker and all listeners within the tuple.

Solutions produced by the line/tree mapping VLiM algorithms based on the selected root bridge are used as scheduling topologies whenever the CR requires the use of TT schedules. The subsequent sections address TT schedule computation from a VNE perspective.

## 5.3 OVERVIEW OF TIME-AWARE SCHEDULING IN TSN

In TSN, the goal is to configure NEs in order to send time-critical packets to meet arrival deadlines. Time critical applications can be periodic/aperiodic with sporadic or deterministic packet release times. Transmission of time-critical streams can be served by priority scheduling (preemptive SPS or CBS) as discussed in Section 3.4.1. However, a periodic application with deterministic release times and hard deadlines requires network bridges to support enhancements that enable packet forwarding from an egress queue at specific epochs of time.

*contributions in this section are from [85]*

This enhancement for scheduled traffic (periodic/deterministic release time) is enabled by TAS, (specified in IEEE 802.1Qbv). TAS serves as a time enforcement mechanism positioned at the egress of priority queues on the transmit interfaces of Ethernet bridges. The main functions of TAS is to dynamically enable and disable selection of packets from respective priority queues onto the transmission line based on predefined epochs (time schedules).

TAS uses clock/time-driven schedulers to control queue access to the transmission line by associating a transmission gate with each queue. The state of a gate determines whether packets from a queue can be selected for forwarding. The gate of a queue can only be in one of two states; *open* or *close*. A gate event is triggered at the end or beginning of a time interval. This causes a gate transition from an *open* state to a *close* state or vice-versa;— A gate-close event disconnects the transmission selection function of the forwarding process (forwarding function) from the queue(s) hence prevents packets of that queue from being selected.

On the other hand, a gate-open event allows the forwarding function to select packets from the queue. The sequence of gate operations

can be scheduled to allow packets of periodic Time-Triggered (TT) applications to be forwarded in a timely manner without any form of interference. To ensure gate(s) of the queue(s) are opened or closed in

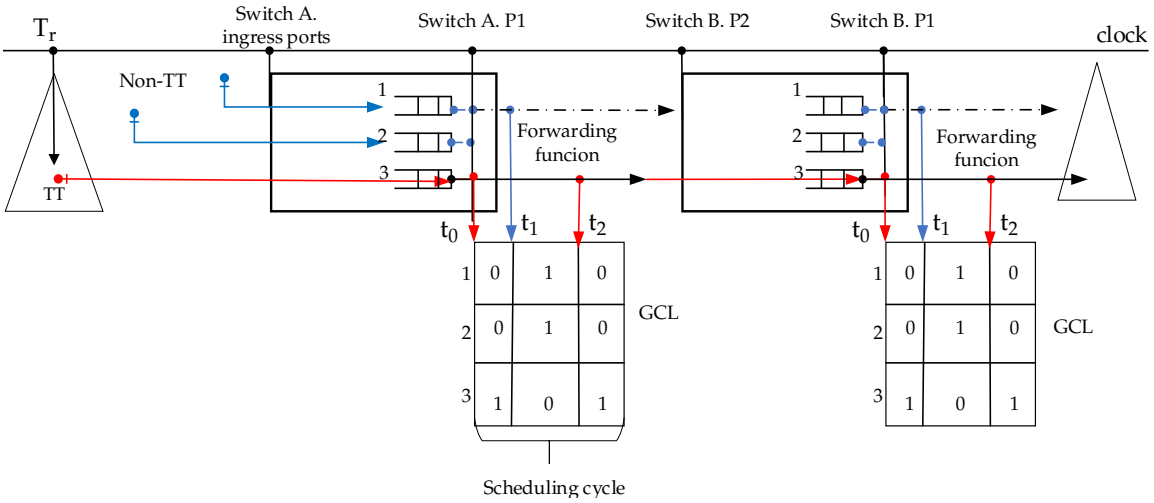



Figure 5.2: Operation of TAS in relation to periodic TT packet forwarding

the exact times, the bridges maintain notion of a working clock which synchronizes all switches and ESs. Since TAS allows each egress port to run its own schedules, the release time of the packet as well as the gate event at each egress port must be synchronized to a common clock ( specified in IEEE 802.1AS).

Figure 5.2 shows a TAS positioned at the egress of priority queues as well as illustrates the operation of network of TAS. As shown, packets from different ingress ports are assorted into the priority queues. For each egress port, the sequence of gate-events are specified by a Gate Control List (GCL). The GCLs consist of several epochs (time schedules), which are used by a time scheduler to change the state of the gate of each queue. See Appendix A.3 for an example of time-scheduler algorithm.

In Figure 5.2, a periodic TT application releases its packet at $T_r$ which is assorted to queue 3. At time $t_0$ till $t_1$ of switch A.P1 and switch B.P1 relative to $T_r$, the gates of queue(s) 2 & 1 are closed whilst queue 3 is opened. This allows packets sent by the TT application to be forwarded without any interference from non-TT packets. At time $t_1$ the gates of queues 2 & 1 are opened for non-TT packets while queue 3 is closed. At $t_2$ the gates change state per the GCL. The cycle of gate operations will continue in a periodic manner until a new GCL is installed at each egress port.

To create NSs for multiple independent periodic-TT applications, two challenges must be addressed:

- Determining whether a group of independent periodic TT application can be scheduled.

- If they are schedulable, can a feasible schedule be computed.

The former deals with determining the feasibility of a scheduling system[1] termed "schedulability" [2] and the later deals with the allocation of resources at several epochs of the scheduling cycle considering a scheduling system. The next section describes how this problem is addressed.

## 5.4 SLICING WITH TIME-SCHEDULING IN FOCUS

Aside bandwidth, delay and topological constraints identified previously, time scheduling of bandwidth cannot be fully considered in the same frame. Regardless, allocating bandwidth for periodic TT applications does not occur on arbitrary topology. This is because, in addition to finding a resource graph that matches the communication pattern of the applications, the set of periodic applications ought to be compatible within the scheduling system (i. e., can be scheduled together). Therefore, after finding a topology that matches the source and destination(s) as well as bandwidth and delay as per Equations (4.11) and (4.12), one has to determine if the network cycle or periods of applications to be scheduled are feasible. To address this issue, the following terminologies and constraints are discussed next.

### 5.4.1 *TAS terminologies and formal scheduling constraints*

**TIME-INTERVAL PER QUEUE OR APPLICATION** In the context of a queue, time-interval presents the duration between gate-open and gate-close events. However, in the context of a periodic TT application, it represents the time taken to forward a single packet on the transmission line or as deadline for completion of periodic task on the transmission line.

$\epsilon_i = \frac{\sigma_n}{R_{ij}}$

$\sigma_n$= *packet size of the* $n^{th}$ *TT application*

Given that the packet release time of each TT periodic application is predetermined, the gate of a queue can be opened for contiguous amount of packets till the next gate-close event. In this sense, the time-interval between gate-events from the perspective of a queue can be considered as the sum of execution times of multiple periodic TT applications. In the subsequent sections of this thesis, the time-interval is referred to as the execution time ($\epsilon$) or sum of several $\epsilon$ that exist between gate events.

**APPLICATION/SCHEDULING CYCLE** The application cycle refers to the time interval between consecutive jobs (packets) sent by each periodic application. The scheduling cycle on the other hand referred

1 Scheduling systems is a set of periodic applications to be scheduled.
2 Schedulability asserts whether feasible schedules exist or can be computed for a set of periodic applications within a scheduling system.

to as *Cycle-Time* is the overall duration for repetition of consecutive scheduling regime.

Mostly, time-triggered scheduling applies to deterministic periodic systems [92]. Even though aperiodic or sporadic tasks can be accommodated, they must be done within a periodic frame [77, 92]. For a set of periodic applications, the Least Common Multiple (LCM) of all cycles of the individual applications can be considered as the scheduling cycle. However, the most important requirement is that all cycles in the scheduling system must be a harmonic of the scheduling cycle. For a simply periodic system, the scheduling cycle is equal to the largest cycle in the system which is also equal to the LCM. Therefore, the analysis of the scheduling systems are restricted to simply periodic systems.[3]

#### 5.4.1.1 *Scheduling computation constraints*

**SIZE CONSTRAINT OF GCL** The sequence of gate-events in a GCL is defined as ordered entries of octet-string values with each octet encoded as a Type Length Value (TLV) as follows (*c.f.* [75]):

- The first octet of each TLV is an unsigned integer representing a gate operation;
- The second octet of the TLV is the length field which indicates the number of octets of the value that follows the length. A length of zero indicates that there is no value (i.e., the gate operation has no parameters); This also defines the number of gate events.
- The third through to (length -1) (i.e. $3^{rd}, 4^{th}, \cdots, (length-1)^{th}$) octets encode the parameters of the gate operation, in the order that they appear in the definition of the operation in IEEE802.1Qbv Table 8-6.

From the definition above, the maximum size of GCL is attained when all 8 bits of the length (L) octet are ones (i.e. $2^8 = 256$). With this bound, there can be a maximum of 256 gate-events (123 each of *gate-open* and *gate-close* events). The upper bound on gate-events results in scheduling constraints when considering different application periods. This problem is further discussed in the next paragraph.

**APPLICATION PERIODICITY CONSTRAINT** Consider $N$ periodic independent talkers generating a maximum packet burst-size $\sigma_n$ within a period $p_n$ where $n = 1, 2, \cdots, N$ represents the number of talkers to be scheduled. Let also assume the gating-cycle to be at least $H$, the LCM of periods of all talkers to be scheduled. Then the number of

3 A set of periodic applications are said to be simply periodic if the LCM of periods within the system is equal to the largest period.

packets generated by the $n^{th}$ talker in the hyper-period $H$ is expressed as $f_n = \frac{H}{p_n}$. Thus, it follows that for a group of periodic TT talkers, the total number of gate events (*close* and *open*) required in a GCL, denoted $F_s$, is given by Equation (5.8).

$$F_s = 2 \cdot \sum_{n=1}^{N} f_n = 2 \cdot \sum_{n=1}^{N} \frac{H}{p_n} \tag{5.8}$$

On the other hand, let $G^{oe}_{min}$ be the minimum number of *gate-open* events. Intuitively, every gate-open event may correspond to a gate-close event of the other queues and vice-versa. Then from the GCL definition $F_s \leq 123$. This also impose a constraint on the periods of the applications that can be scheduled. Therefore, from the analysis, a periodic system that consists of periods that are not harmonics or integer multiples of each other will result in very large number of gate-events.

However, as discussed in the previous paragraph, the definition of GCL limits configuration to a maximum of 123 events per port. That is if a single queue is considered for periodic TT packets and all other queues for sporadic packets[4]. Hence, another reason to consider TT applications that have periods which are harmonics of each other (or precisely simply periodic).

### 5.4.2 *Scheduling problem definition and network models*

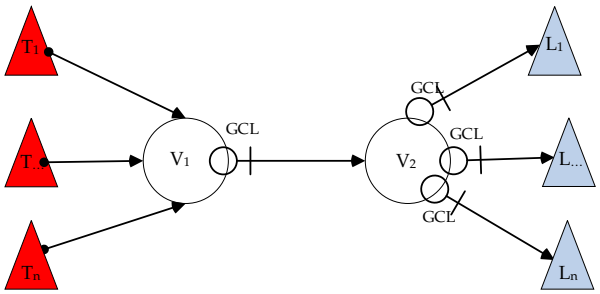



Figure 5.3: Example of a network model with GCL forwarding from Talker End-Station (s) $T_{1,\cdots,n}$ to Listener End-Station (s) $L_{1,\cdots,n}$

NETWORK MODEL Time-sensitive network consists of bridges, links and End-Station (ES). An ES may consist of a producer and/or consumer applications termed Talker and Listener. As illustrated in Figure 5.3, TAS exists on the transmit interface and packets are forwarded towards the listeners. For simplicity, the network is modeled in a unidirectional view from the Talker to Listener(s) by extending the graph

4 Note that, sporadic applications can also be periodic but with arbitrary packet/job release times.

model from Section 4.4.2. This is represented as a non-reflexive directed graph $G = (\mathcal{E}, \mathcal{V})$, where $\mathcal{E} = \{e_{ij}\}$ is a set of edges and $\mathcal{V} = \{v_i\}$ is a set of nodes. Each $e_{ij} = v_i \rightarrow v_j \in \mathcal{E}$ is characterized by the following attributes; $\langle W_{ij}^K, R_{ij} \rangle$ where $W_{ij}^K = {}^t(w_{ij}^1, \cdots, w_{ij}^k, \cdots, w_{ij}^K)$ denotes the total time window for scheduled traffic on queue $k$ ($k = 1, 2, \cdots, K$: $K \leq 8$) of node $v_i$, and $R_{ij}$ is the maximum line speed of $e_{ij}$.

EXTENDING VNR MODEL FOR TT APPLICATIONS Listener applications do not generate any packets and thus they do not put any load on the network. Therefore, the focus must be on packet generating applications (i. e., Talkers). The VNR model introduced in Section 4.4.2 for TT applications must also be extended as follows:—
Let a Talker be a TT periodic application $App_n$ that generates streams of data packets of size $\sigma_n$ at a frequency $f_n$. This application is described by a triplet $App_n = \langle \phi_n, p_n, \epsilon_n \rangle$ where $\phi_n, p_n, \epsilon_n$ denote the phase (*startTime*), the period (i. e., the time interval between two consecutive packets), and the maximum transmission time or packet duration on the forwarding line (*execTime*) respectively. The execution time ($\epsilon_n = \frac{\sigma_n}{R_{ij}}$) on a link $e_{ij}$.

PROBLEM DESCRIPTION Given a networked of TAS, $G$ and $Apps = \langle App_n = (\Phi_n, p_n, \epsilon_n),\ n = 1, 2, \cdots, N \rangle$, a set of $N$ periodic TT applications. The problem consists of two parts; firstly, verify the feasibility of scheduling a group of periodic time-triggered applications. Secondly, determine the optimal phases $\phi_n$s and subsequently the GCL on each TSN bridge along the trajectory of their flows such that they arrive exactly within their planned transmission window on each bridge. Thus, eliminates queuing delays at the bridges. This also implies an ideal clock synchronization is required and, therefore, effect of clock drifts are omitted from the analysis. Regardless, clock drift can easily be factored in as a constant within each window as part of a guard-band.

### 5.4.3 *Schedulability analysis and verification algorithms*

For the problem described in Section 5.4.2, the solution is decomposed in two parts. The first part consists of determining whether flows of a set of Talkers can be scheduled. The second, deals with using the "schedulability" information to determine the *start-time* and subsequently the gate opening times along the path.

PRELIMINARY CONDITIONS FOR SCHEDULABILITY The scheduling system consist of a set of periodic independent applications (Talkers). The periodicity of the applications imposes constraints which make it difficult to determine "schedulability". For this reason, the following conditions are used:—

- The scheduling system is simply periodic. That is, for a set of independent Talker applications, the scheduling system is simply periodic if for each pair of Talker applications, say $App_i$ and $App_j$ are such that $p_i \leq p_j$, $\exists\ n \in \mathbb{N}^*$ an integer verifying $p_j = n \times p_i$.
- The *periods* of all applications are natural: $p_i \in \mathbb{N}^*$.
- The *exectime*, $\epsilon_i$, of the $i^{th}$ application is less than its period $p_i$ (i.e $\epsilon_i \leq p_i$).
- The network uses an ideal clock synchronization. Therefore, all nodes in the network are perfectly synchronized. However, without this, one has to respect the clock jitter which can be factored into the time-intervals as guard-bands.
- The network topology consists of $M$ bridges with a single rendezvous point: $v_1 \xrightarrow{l_{12}} v_2, \cdots, v_{M-1} \xrightarrow{l_{M-1,M}} v_M$. Thus, it can safely be stated that $W^k_{1,2} = W^k_{2,3} = \cdots = W^k_{M-1,M}\ \forall k$ as the packets transit from one bridge to the next.

### 5.4.4 *Utilization condition*

The utilization of a system $S$ of $N$ periodic independent applications is defined as:

$$U_S = \sum_{n=1}^{N} U_i = \sum_{n=1}^{N} \frac{\epsilon_i}{p_i} \tag{5.9}$$

Following the proof by Liu et al. [77], a system $S$ of $N$ independent periodic tasks may be schedulable if $U_S \leq 1$. Regardless, a feasible schedule may not exist even if the above condition is met. If jobs are preemptive and deadlines ($d$) are at least equal to periods (i.e. $p_i \leq d_i$) and the system is simply periodic, then a Rate Monotonic Scheduling (RMS)[5] algorithm is optimal. In other words, whenever $U_S \leq 1$, it can be assured that $S$ is schedulable using RMS [77].

Since periodic independent tasks are considered in the context of a network where the generated jobs are simply the packets that need to be forwarded, one can safely state that it is not required to immediately send a packet once generated. Rather, it can be stored, if needed, until the next packet is generated, before being forwarded at the Talkers, i.e., $p_i \leq d_i$ is always respected. However, it must be emphasized that though this condition is valid for preemptive tasks, it does not hold true for non-preemptive tasks. In the TSN, a TT application cannot be preempted within its own transmission window.

Also, given that no packet must interfere with any other packet of any TT application, preemption must be ignored entirely in the scope

5 RMS is a priority scheduling algorithm that assigns the highest priority to shorter periods, hence schedules applications with shorter periods ahead of longer ones.

of TT flows, thus optimality might be missed and can result in the rejection of some TT applications.[6]

### 5.4.5 *Verifying the schedulability of the scheduling system*

In this subsection, the analysis considers a single queue for periodic time-triggered applications due to constraint on the number of gate events, however, the analysis is easily applicable to multiple queues. As explained earlier, the scheduling system consists of $N$ periodic talker applications. Firstly, the schedulability of the scheduling system is verified using Algorithm 5.2. This is done by considering a single rendezvous point. In this case, starting with the bridge directly connected to a Talker ES as shown in Figure 5.3 is the same as starting with a rendezvous bridge (node). The rendezvous node (specifically port) is any port where all TT applications contend for time-slots.

Lines 1 $\rightarrow$ 10 initializes the parameters where $H$ is the LCM of the scheduling system ($H = w^k$). It is also referred to as the hyper-period or the scheduling/network cycle.

$T_p$ (Tab Possibilities) is a data structure that saves in cell $i$ the maximum number of possible talker applications of period $p = i$ and execution time $\epsilon = 1$ unit that can possibly be scheduled in the system. $T_p$ is of a size equal to the biggest period in the scheduling system (i. e., for a simply periodic system, this corresponds to the hyper-period $H \equiv$LCM).

$S_{free}$ =*number of consecutive free slots*

Algorithm 5.2: Schedulability and Relative Phase computation

```
1  Output: A_s , Apps.Φ_i
   Input: {Apps = app_i = (Φ_i = ?, p_i, ε_i), i = 1, ··· , N} , G^s
   Data: T_p , H , b_w
   begin
      initialize:
6        H ← LCM(app.p_i; app ∈ Apps); A_s ← zeros; b_w ← H;
         for app ∈ Apps: do
            app.φ_i ← NaN;
         for i from 1 to size(T_p): do
            T_p[i] ← i;
11       Apps ← sortPeriods{Apps.p};
      Schedulability:
         for app ∈ Apps: do
            if app.ε ≤ T_p[app.p] & ≤ b_w: do
               i ← 1;
16             while i < H: do
                  if A_s[i] == 0: do
                     S_free ← getConsecutiveFreeSlots(i, H, A_s);
                     if app.ε ≤ S_free: do
                        app.Φ ← i − 1; break;
21                   else i ← S_free + 1;
                  else i ← i + 1;
```

6 Optimality in the sense of number of TT applications that can be scheduled not in terms resources (bandwidth). This is because non-TT windows can be used to forward sporadic flows.

```
                endwhile
                for i from 1 to sizeOf(T_p): do
                    T_p[i] ← T_p[i] − app.ε × i/app.p ;
26              updateTimeSlots(app.[Φ, p, ε], H, b_w, A_s) ;
              else reject;
    end
```

Note that $T_p[i] = x$ can be interpreted in many ways:— either one can schedule *" x applications of period $p = i$ with execution time $\epsilon = 1$ unit"*, or, *" $\frac{x}{2}$ applications of period $p = i$ with execution time $\epsilon = 2$ units (given that $\frac{x}{2} \in \mathbb{N}$)"*, ..., or *" 1 application of period $p = i$ with execution time $\epsilon = x$" units*.

$A_s$ = *available time-slots*

$0$ = *free*
$1$ = *used*

Initially, $T_p[i]$ is obviously equal to $i$. Also, $A_s$ (Available Slots) is a binary table of $H$ elements (time units) showing free slots on each link. Where, 0 means a free time-slot, 1 indicates a used time-slot. Initially, all the time-slots are available (all set to 0) and every time a new application is accepted for scheduling, its respective time-slots in $A_s$, occupied by its generated packets ($\epsilon$ time-slots every $p$ slots, starting at $\phi$, the phase to be determined) is turned to 1.

Lines 12 $\rightarrow$ 23 verifies whether a Talker application is schedulable by assigning them time-slots within the scheduling window. Besides, their relative phases with respect to the scheduling/network cycle of the first bridge to which the Talker ESs are connected must be computed. If an application $app_i$ is not schedulable then it is ejected from the scheduling system (i. e., would not be scheduled in the current scheduling system), if it is schedulable then the algorithm sets its relative phase $\phi_i$ which is the index of the first time-slots it occupies within the scheduling cycle. Afterwards, the $T_p$ and $A_s$ are updated accordingly.

$b_w$ = *longest consecutive empty time-slot in* $A_s$

The algorithm keeps track of the biggest window ($b_w$, line 6) of consecutive empty slots in $A_s$. This enables the algorithm to run a quick test on the schedulability of the current application $app_i$. Basically, if the size of the widest empty window, $b_w$, is greater than execution time $\epsilon_i$ of $app_i$, then $app_i$ can be scheduled (Line 14). If $app_i$ passes this test, the algorithm starts looking for the first empty window that can host the application. The beginning of this window indicates the initial phase, $\Phi_i$, to be assigned to $app_i$. For this, the algorithm iterates over $A_s$ until the first $b_w$ is found. This indicates that the application is accepted for scheduling (Lines 16 $\rightarrow$ 23). In fact, this line can be simplified or removed entirely by noting as well the index zero of $b_w$ in $A_s$.

Next, $T_p$, $A_s$ and $b_w$ are updated. It is obvious to see here that a newly accepted application will affect the number of possible applications (i.e. all cells in $T_p$) that can be scheduled. For cell $i$ (storing the number of possible applications of period $i$ that can still be scheduled in the system), $T_p[i]$ should be decreased as indicated in Line 25 depending on the execution time $\epsilon_i$ and the period $p_i$ of the newly accepted application.

The output of Algorithm 5.2 are the relative phases ($Apps.\Phi$) of all schedulable TT applications, and available time-slots table, $A_s$, which represents a mask of occupied-slots and free-slots over the entire scheduling cycle. This does not represent the actual schedules that can be configured on a bridge. This will be discussed in section 5.5.

### 5.4.6 *Schedule-aware VN mapping*

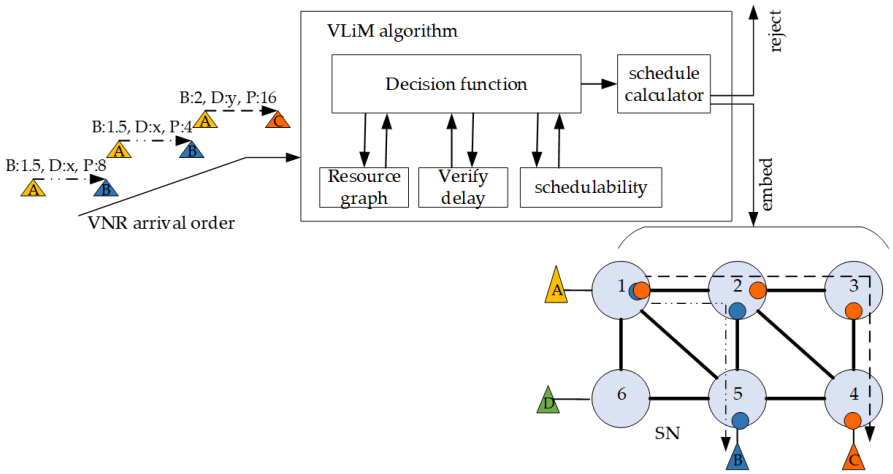



Figure 5.4: Admission control for online schedule-aware VLiM for slicing periodic TT applications

Traditionally, TAS schedule computation and configuration occur as an offline system, where schedules are pre-computed and fed directly into the Management Information Base (MIB) of the respective switches. Using the SDN controller as a feedback control provides opportunity to have a plug-and-play or event-driven resource reservation and schedule computation. To do this, the schedulability algorithm integrates into the VLiM algorithm. Figure 5.4 illustrates the correlation of VNE and TAS schedule computation for TT NSs. That is, before the actual schedules can be computed, a VLiM algorithm that is schedule-aware is able to assert whether the constraints imposed by the TT application can be fulfilled or otherwise, for a given topology.

#### 5.4.6.1 *Online schedule-aware VLiM algorithm*

Algorithm 5.3 shows the integration of schedulability algorithm and traditional VLiM algorithms in what is termed schedule-ware VLiM.

The idea of the combination is to provide an online system capable of asserting whether an application can fit to an existing VN instantiated for TT applications or if not, can one be created for such an application. This implies, dynamic NS creation algorithms described previous using the online VLiM should have the capability do so.

Schedulability is not a VNE problem, therefore, it must be integrated as as a demand verification function just as the topology and delay verification algorithms.

To accomplish this, the TT application is converted to VNR with all the characterization for TT applications described in Section 5.4.2 (see Algorithm 5.3, line 4). Before schedulability can be asserted, a scheduling resource graph is required. This means a set of links between the Talker and its listeners must first be found. Using topology-aware VLiM algorithms, a set of virtual topologies that illustrate the communication pattern of the TT application (see Algorithm 5.3, line 6 ) is computed. For each VN, if there exists a TT schedule for an existing NS, the rendezvous scheduling port must be identified and checked if the scheduling cycle is harmonic of the TT application period to be scheduled. If this check asserts true, then the delay and schedulability of the TT application is verified. However, if no TT schedules exist on any of the links in the VN, then any node within the VN can be selected as the rendezvous point, (see Algorithm 5.3, line 14).

Algorithm 5.3: Schedule-Aware VLiM

```
   Input: app_n, G^s
2  Output: y^g, P^s ⊂ G^s, A_s(e^s_ij), app_n.Φ
   begin
      initialize: g^v ← convertToVNR(app_n); P^s ← ∅; y^g ← 0
      VN graph
      VN ← computeVNs(g^v, G^s);// using algorithm 2
7     for VN ∈ VN: do
         if g^v.p harmonic of VN.p: do
            x^g ← verifyDelay(VN, g^s);//using equation (4.12)
            s^g ← Schedulability(VN, g^s);
            if x^g & s^g == 1: do
12             P^s ← VN;
               y^g ← 1;
               return y^g, A_s, app_n.Φ
      endfor
   end
```

If the delay verification and schedulability test assert true, the VN in focus becomes the scheduling graph, the algorithm returns this graph, the rendezvous node (port) with the scheduling mask $A_s$ and relative scheduling phase, $\Phi$ of the application. The result of this algorithm can then be fed to a GCL computation algorithm as illustrated in Figure 5.4. The schedule computation algorithm is described next.

## 5.5 GCL EVENT COMPUTATION

Computing the Gate Control List (GCL) on each bridge along the trajectory of the packets from a Talker ES requires a reference start-time $T_r$ on the talker ES, and the gate events (*gate-close* and *gate-open times*) on each bridge.

This is integrated into Algorithm 5.2 by adopting a RMS approach

where the scheduling system is sorted in increasing order of periods in line 11, thus gives priority to applications with shorter periods. However, note that scheduling applications in the decreasing order of their periods (reverse RMS) results in scheduling more applications within a hyper-period than RMS as shown in Equation (5.8). This of-course is dependent on the scheduling system and the goal of the service creator, therefore, it can be left as a scheduling preference of the user of schedulability algorithms. In either case, the number of gate-events occurring in very large $H$ can be reduced only if gates are opened for contiguous amount of packets belonging to independent TT applications. This prevents the use of multiple queues for TT applications.

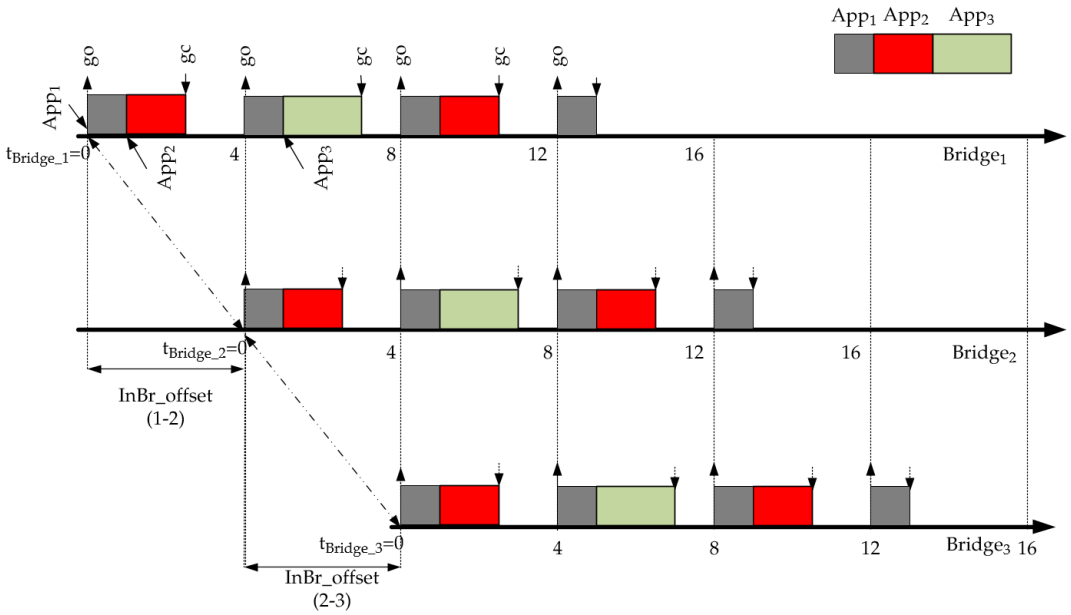



Figure 5.5: Output solution illustrating the schedule computation of three periodic Talker applications ($App_1 = \langle P_i = 4,\ \epsilon_1 = 0.5\ unit \rangle$, $App_2 = \langle P_i = 8,\ \epsilon_1 = 1.0\ units \rangle$, $App_3 = \langle P_i = 16,\ \epsilon_1 = 1.5\ unit \rangle$ using Algorithm 5.2

Figure. 5.5 shows the results obtained from Algorithm 5.2 considering a simple scheduling system that consists of 3 Talker applications ($App_1 = \langle P_i = 4,\ \epsilon_1 = 0.5\ unit \rangle$, $APP_2 = \langle P_i = 8,\ \epsilon_1 = 1.0\ units \rangle$, $App_3 = \langle P_i = 16,\ \epsilon_1 = 1.5\ unit \rangle$). In this example, the first bridge connected to the Talkers is the rendezvous bridge.

In this bridge, one simply needs to open/close gates according to the generated mask on the egress port where all TT application contend for resources, $A_s$ (i.e. Table 5.1), of Algorithm 5.2, where a 0 means a slot with a closed gate and a 1 means a slot with an open gate.

The same schedule mask is applied to the second and third bridges by taking into account the inter-bridge delay due to propagation, switching and transmission (inter-bridge Offset) as described by Equa-

tion (4.12).

Till this point, the computed schedules are made as if the talkers

Table 5.1: Time-slots ($A_s$) used as scheduling mask

| Index (i) | 1 | 2 | 3 | 4 | 5 | ··· | H |
|---|---|---|---|---|---|---|---|
| $A_s$ | 1 | 1 | 1 | 0 | 1 | ··· | 1 |

are running at the bridge (i.e. delay, called initial off-set ($\gamma$), from the talker to the bridge is 0) in Algorithm 5.2. Thus, to compute the reference start-time $T_r$ for each Talker application, the initial phase obtained from the schedulability analysis must be rectified to coincide exactly with the closing and opening of the scheduling window of each bridge along the path. Rectifying these phases is more complex than a mere shift. This is because the physical connection between the Talker ESs and the edge (rendezvous) bridge may not be the same for all talkers due to different cable lengths or the fact that they might not all be emanating from a common source.

Algorithm 5.4: Phase Rectification Algorithm

```
Input : Apps[.] = ⟨App_i = (Φ_i , σ_i, p_i)⟩
Output : AppStartTime(T_r[.])
Data : initialoffset : γ[.]
begin
    for i ∈ Apps[.] : do
        T_r[i] ← Φ_i − γ[i] ;
    endfor
    maxAbShift ← min(T_r[Apps])
    for i ∈ Apps : do
        T_r[i] ← T^r[i] − maxAbShift ;
    endfor
    return T_r[.] ;
end
```

This situation is illustrated in Figure. 5.6, where the scheduling order of the Talker applications calculated by Algorithm 5.2, does not necessary imply the same order of first packet firing. The most important thing that needs to be respected is to ensure that the first packet of the different TT applications reach the rendezvous bridge exactly as specified in the computed schedule. For this reason, the initial assumption that the talkers are running at the bridges must be rectified using Algorithm 5.4. Here, the reference start-time (effective release time) of each application must be calculated in a way that respects the schedule mask computed by Algorithm 5.2. This can be done by taking into account the delays due to the variable cable lengths to the edge bridge ( or rendezvous bridge), denoted $\gamma$ (initial-offsets).

In algorithm 5.4, every application $App_i$, should start firing $\gamma[i]$ units of time before its computed phase $\Phi_i$. This is done in the first for-loop of the algorithm (Lines $5-7$). Once these shifts are performed, all the new starting times need to be put in the same time reference

(i. e., absolute time referential), thus the need to shift all of them with $maxAbshift$ (Lines $9-11$).

In the example depicted in Figure 5.6, the maximum shift is given by $App_2$, and thus its release time can be regarded as the time origin: $T_r[i] \leftarrow T_r[i] - maxAbShift = maxAbShift - maxAbShift = 0$. Whilst this shifting of scheduling mask suffices for line, ring and tree topology with single resource contention point (rendezvous interface), a much generic shifting algorithm is required for tree topology with multiple resource contention interfaces. This part is currently reserved for future work. The number of gate-events for queue, $k$, triggered

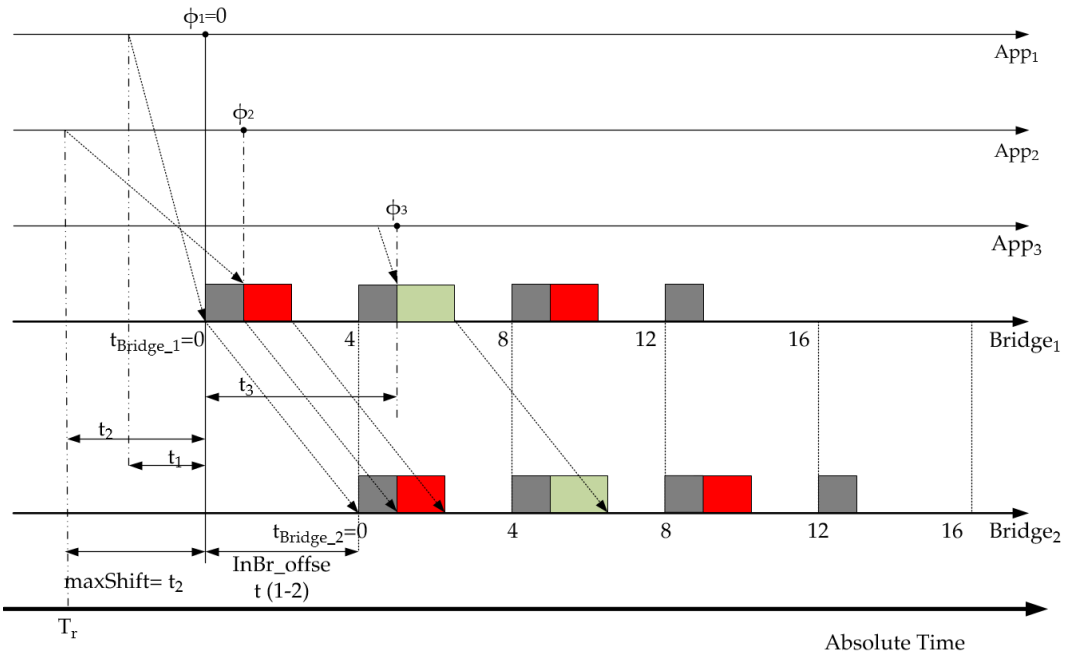



Figure 5.6: Rectification of relative schedule phases for GCL computation considering absolute reference clock. The delay between Talkers and the edge bridge as well as inter-bridge delay are used as offsets for calculating absolute time for gate-events on the bridges using Algorithm 5.4 and 5.5 respectively

by application, $App_i$, on each bridge is determined by the number of generated (informed by the application cycle $p_i$) within the scheduling window ($w^k = H$) of each egress interface. This means there are a total number of $2 \times \frac{w^k}{p_i}$ gate events generated solely by $App_i$. This is because for every gate open event, there is a corresponding close event.

In accordance with Equation (5.8), the total number of events can be obtained by simply summing over all applications in the system. However, the number of gate-events can be reduced significantly if gates are opened for an uninterrupted transmission of contiguous packets from independent TT applications, in which case, Algorithm 5.4 can be ignored entirely and replaced by using Equations spec-

ified in IEEE 802.1Qbv for start-time computation (see [75] 8.6.9.1.1). For this, one needs only to rely on the output of Algorithm 5.2, i. e., $A_s$.

Recall that Table 5.1 is an example of such an output $A_s$, the time-slot occupancy state of a bridge as calculated by Algorithm 5.2.
To reduce the total number of generated events, a GCL illustrated by Table 5.2 can be derived from $A_s$. Suppose that the number of events are reduced to $n$, then, row $k$ in Table 5.2 represents the events indices ($k = 0..n-1$), $t[k]$ is the starting time of the $k^{th}$ event in the relative time referential of the considered bridge; and $A[t[k]]$ represents the corresponding gate-events: open (i.e. op) or close (i.e. cl).
The duration of an event, say $i$, is simply given by the difference between the starting time $t[i]$ and its next event $t[i+1]$. Information from Table 5.2 is fed as input into Algorithm 5.5 to compute the GCL for each bridge along the trajectory of the packets on the network.

Again, shifts that rectifies applications' phases w.r.t their inter-bridge offsets need to be accounted. Whilst the former shifts are rectified by Algorithm 5.4, the latter is obtained by iteratively computing the time origin for bridge $br_i$ from the time origin of its predecessor along the path, $br_{i-1}$, and the transmission delay between them, which is given by $interBrOffset[i][i+1]$.

Table 5.2: Transformed scheduling Mask ($A_s$)

| $k$ | 0 | 1 | 2 | 3 | 4 | $\cdots$ | $n-1$ |
|---|---|---|---|---|---|---|---|
| $t[k]$ | 1.0 | 3.0 | 1.0 | 3 | 1 | $\cdots$ | 10 |
| $A[t[k]]$ | op | op | cl | op | cl | $\cdots$ | cl |

Algorithm 5.5: Per Bridge GCL Computation

```
Input: interBrOffset[][], initialRefStartTime, path[]
Output: baseTime[.]
Data:
begin
    baseStartTime[1] ← initialRefStartTime; //source bridge
        for i from 2 to size(path): do
            baseTime[i] ← baseTime[i − 1] + interBrOffset[i − 1][i];
        endfor
end
```

Note here that $initialRefStartTime$ represents the first event in the absolute time referential, corresponding to the release of the first packet in the system by a TT application. Path denotes the ordered sequence of switches/bridges relaying packets from the talker ESs to the listener ESs.

Finally, it must be emphasized that not all Talker ESs may be connected to a single bridge ($V_1$) as shown in Figure 5.7. There may exist scenarios where the Talker ESs start at different bridges but

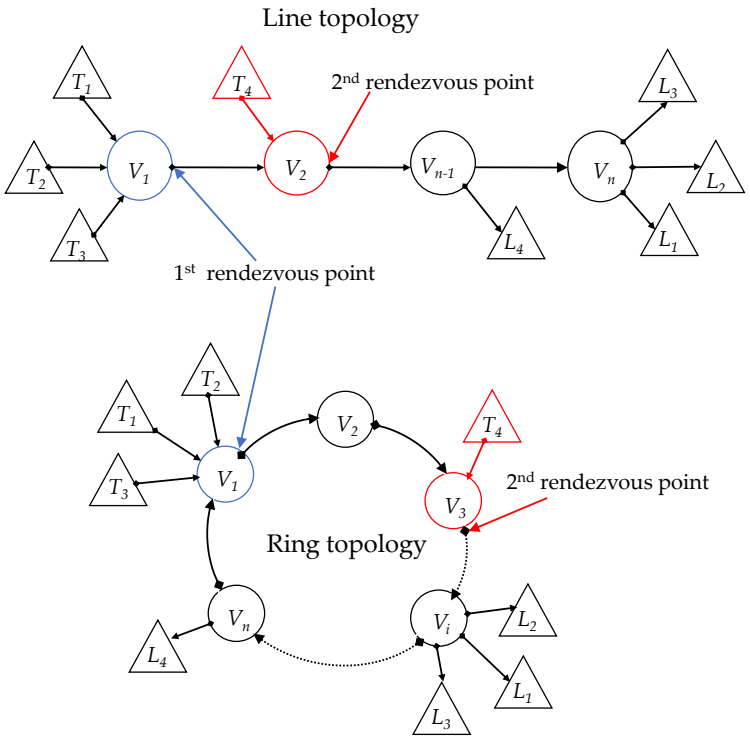



Figure 5.7: Scheduling mask $A_s$ computation takes into account the point of contention (rendezvous point) where TT applications meet. Complex Tree topology results when different application converge at more than one point. A Ring topology can be treated as a line topology because of the position of TAS

may require time-slot allocation at a common egress interface, termed the rendezvous point. The best-case topology scenario is when all Talkers are connected to single bridge and require time-slots on a single interface. In this case, the scheduling mask ($A_s$) is computed at the first rendezvous point and shifted along the path. This represents a simple line topology which suffices for a tree topology as well. That is if $T_4$ is added, $A_s$ must be computed at the second rendezvous point. However, for multi-hop trees, the scheduling mask may start from several rendezvous point(s) such that there can exist primary and several secondary points where one or more TT applications contend for time-slots and thus making it difficult to obtain a globally efficient time-slot allocation. This may require several iterative use of extended tree algorithms, hence, can result in huge convergence time and thus may not be adequate for online deployment.

Another approach may consider superimposing the scheduling mask for multiple rendezvous point to find a unified mask that works for all or most applications. These aspects are, however, reserved for further investigation.

Considering the positioning of TAS on each port of a TSN bridge, a

Ring topology can be treated in the same manner as a Line topology. This is because TSN bridges use duplex links and given that TAS only operate on transmit interfaces, there are no time-slot contention in the opposite direction of the same port.

## 5.6 SUMMARY OF CHAPTER AND BIBLIOGRAPHIC COMMENTS

### 5.6.1 *Bibliographic comments*

Resource assignment can be performed in several ways using VNE. However, the general approach to any VNE problem is to embed a set of virtual nodes and/or links on a SN. Hence, the design of VNE algorithms depend solely on the problem space and the outcome an algorithm designer may want to illustrate. As a result, there exist tons of VNE algorithms in literature as presented in the surveys of Fisher and Belbekouche et al. [93, 94]. As VNE is employed in new areas or use-cases, this number is expected to grow.

Within the context of SDN, several VNE algorithms have been developed to ameliorate the resource allocation and QoS problem [33, 39, 82, 84]. While most of these algorithms are able to achieve their design goals, often the lack of specificity to the underlying system makes it difficult to use results of the algorithm in production systems and thus, remains an academic conquest.

In this thesis, the integration of such specificity has been achieved due to the flexibility of the VNE problem formulation. This has allowed aspects of schedulability analysis and the mapping of functional requirements in the VNE problem.

Although VNE formulations are flexible, concepts or problems such as schedulability that occupy different problem space cannot be integrated if the approaches in such space are not flexible themselves. Several aspects of schedulability analysis have been treated in literature for real-time communication in the automotive and automation networks [85, 92, 95–99]. Some of which are based on Satisfiability Modulo Theory (SMT) [100]. Specific to TSN and the TAS schedule computation, Craciunas et al. [97] presented formal constraints for creating window based IEEE 802.Qbv schedules, some of which were highlighted and discussed together with additional constraints in this thesis. Contrary to these related work where synthesis of schedules are obtained using SMT solvers to validate scheduling formulations, this thesis develops a heuristic solution for the verification and computation of time-triggered schedules as part of the VNE problem.

### 5.6.2 *Summary of chapter*

Often, VLiM algorithms are designed with the intended goal of minimizing SN utilization in order to instantiate more VNs. In the

online application of VNE, such an approach to multiple constrained-QoS VLiM problem results in sub-optimal results. This chapter provided a thorough analysis of existing greedy approach and proposed the design of a VLiM algorithm which employs a Demand Based Approach (DBA). The DBA is developed on the hypothesis that for multi-constrained VLiM problem, a demand-centered approach solves challenges of greedy approach, therefore, uses the SN efficiently.

Furthermore, functional requirements such as Application Relation (AR) and corresponding Communication Relation (CR) involves the use of tree topologies which require iterative use of VLiM algorithms. However, mapping tree topologies can yield varied results due the selection of an anchor node (root node). Therefore, the proposed VLiM algorithm is enhanced with root selection features which enable the VLiMs to efficiently use SN resources in the case of ARs that impose tree CRs.

The final contribution of the chapter considers resource allocation for Time-Triggered (TT) applications, where VLiM algorithm is enhanced with schedulability algorithm within the VNE problem. The subsequent chapter provides analysis and performance evaluation of the respective features of the VLiM algorithm.

# 6 EVALUATION OF VNE ALGORITHMS IN SDIN

To ascertain that algorithms perform according to design objectives, they need to be evaluated via experimentation before deployment in a production environment. This enables the algorithm designer to verify the underlying hypothesis of the design. Important information derived from such experimentation gives the designer an idea of how the algorithms will operate under certain conditions.

Particularly for online scenarios, the focus lies more on how the algorithms handle demands rather than the solutions produced. This is because the requests and resources used during a simulation are not necessarily the same as those in the production network. As explained in section 3.2.1, the statistical results of a VNE algorithm does not provide any insights to whether solutions computed are accurate or not. This aspect has been established in this thesis to depend solely on the precision of demand verification functions and the details of the VNR and SN models. These models depend on the underlying network technology. On the other hand, inferences can be made from the statistical data provided by the respective VLiM algorithms to pinpoint precisely where one algorithm performs better than the other. This, in effect, helps the algorithm designer to improve or change the design hypothesis where necessary.

For such evaluations, it is logical to test VNE algorithms and demand verification functions under different conditions to examine its influence on the results. In this context, simulation is a preferred method for VNE algorithms and demand verification functions. Extensive simulation experiments for VNE algorithms require specialized codes which sometimes have to be developed from scratch. However, it is much flexible and practical to extend existing simulators as bugs in the simulation code becomes less likely ( cf. Fischer [27]).

This chapter presents the performance evaluation of VLiM algorithms and demands verification functions using the Algorithms for Embedding of Virtual Networks (ALEVIN) VNE simulator [47].

## 6.1 EVALUATION OF VLIM ALGORITHMS

The ALEVIN framework is an extensible VNE simulator that enables the integration of new metrics and models for VNE evaluation. In its current state, it is only designed for offline evaluations where characteristics of VNRs and their order of arrival are predetermined. This implies in unsolvable scenarios (see Fischer [27]), acceptance ratio of algorithms depend largely on VNR arrivals.

In the multi-constraint QoS use-cases, this is undesirable as QoS demands are often very contrasting and the result can be challenging to interpret.

Furthermore, the SN model needs to be modified to integrate the QSL model as discussed in section 4.2, as well NS model in the VNR model. This enables the VLiM algorithms to apply the demand verification functions effectively. The subsequent sections describe the simulation environment and parameter settings for the evaluation of the VLiM algorithms.

### 6.1.1 *Evaluation environment and ALEVIN extension*

The simulations are carried out using a set up of the ALEVIN framework on a 64 bit windows machine with 4x2.6GHz processor cores and 16GB Random Access Memory (RAM) capacity.

To maintain consistency of the underlying topology and NS requests with classes and structure of ALEVIN, it is necessary the there exist a consistent format for data representation between production SDIN controller and the simulator. This therefore requires codes that convert the slice request and substrate data structure to models that are consistent with the simulator data structures. An example of the SN and NS formats which are parsed in the ALEVIN simulator can be found in Appendix A.3. Furthermore, extensive description of the ALEVIN framework is provided in the thesis of Fischer (see [27, 47]).

## 6.2 EVALUATION OF DEMAND-BASED MAPPING ALGORITHMS

This section presents evaluation and analysis of demand context algorithm in comparison with the least-cost selection algorithm in an online multiple constrained link mapping problem. Here, we examine the underlying hypothesis that informs link mapping decisions leading the two path selection approaches.

### 6.2.1 *FSFD use-case for VLiM*

As discussed previously in Section 3.3, the use-case analysis focuses on a fixed source and destination pair. This is because the node mappings are always unambiguously defined in the VLiM TE use-case (see [32–34, 39, 101]). The use-case considered for our simulation is similar to the use-cases proposed for SDN-enabled 5G virtual core network in [34] and critical utility networks in [102] with the integration of TSN as proposed by 5G-ACIA ([103]), targeted at industrial communication.

Communication between applications such as motion-control systems and Input/Output device (IODs) or actuators can be achieved on the same network together with video surveillance, sensors, and

Table 6.1: Traffic class and VNR QoS Profile

| QoS Class | Traffic Class | burst (bytes) | Delay ($\mu$s) |
|---|---|---|---|
| 1 | Real-time high (RT-H) | $64 - 68$ | $1000 - 2500$ |
| 2 | Real-time low (RT-L) | $68 - 128$ | $2500 - 5000$ |
| 3 | Video surveillance | $512 - 1500$ | $\geq 50000$ |

alarm monitoring communication services. The use-case considered for the evaluation is depicted in Figure 6.1 where the realization of communication between field applications and control applications over a backbone TSN network are examined.

Here, communication services for real-time control traffic such as between a PLC and robot arm require very stringent e2e delay. Video surveillance and alarm monitoring services, which are also delay and bandwidth centric, are realized on the same network. For the use-case

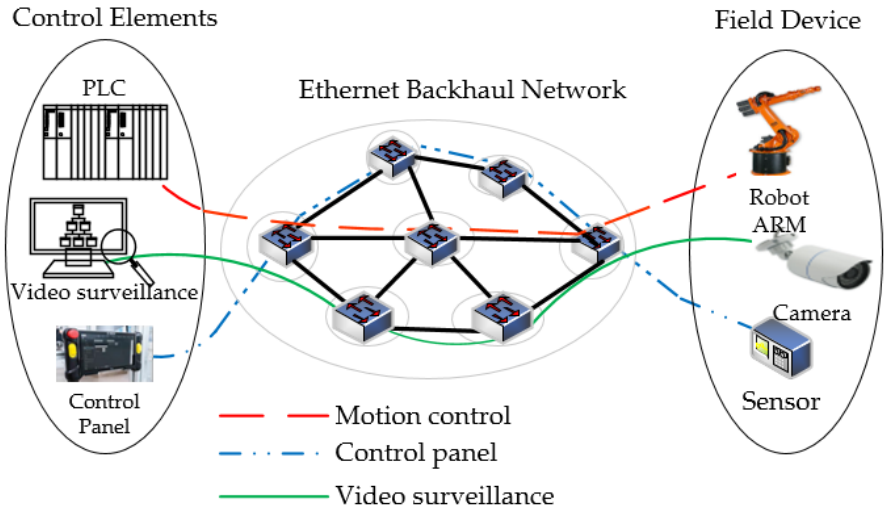



Figure 6.1: Multi-service communication over backbone Ethernet network

examined, the communication services are categorized into three traffic classes requiring very different demands on bandwidth, delay and hop-count as shown in Table 6.1. Two *real-time* traffic classes representing QoS class 1 and QoS class 2 for high and low cyclic (low and high transmitting) applications. For example sensor/alarm monitoring applications. The third traffic class is for *video surveillance or monitoring* designated for QoS class 3.

For each class, minimum and maximum demands e. g., bytes and delay are defined. VNRs are generated to represent traffics emanating from each class. The set of demands are defined by a random variable

uniformly generated between the range of minimum and maximum demand limits as shown in Table 6.1.

The substrate topology is generated randomly with an average node degree set to three and node connectivity probability set to 0.5 for core nodes. Two edge nodes are defined and connected at two or more core nodes with the highest node degree in the network. This is done to ensure that the bottleneck in the network occurs within the core of the network instead of the points at which the control and field device connect to the network.

Also, we define a traffic-mix ratio to determine the percentages of total VNRs that are generated per QoS class by the virtual network generator of ALEVIN. For simulation purposes, the traffic-mix ratio is set to 35% for QoS class 1 and 2 each, 30% for QoS class 3. The traffic-mix ratios of QoS class 1 and 2 are intentionally defined to be slightly more than QoS classes 3 to compensate for the low bandwidth demand of the VNRs in the class. This allows the analysis to focus more on the additive constraints such as delay rather than bandwidth.

To ensure a fair comparison, a mandatory condition for the online simulation requires that all algorithms process the VNRs in the same order of arrival.[1] Hence, the traffic-ratio of the QoS class 1 and 2 does not affect the efficiency of the process as all algorithms receive the VNRs in the same order.

A total of 10 simulation runs are taken per network size and the average results are evaluated at a confidence level of 95%. The goal of the analysis is to compare SPFA and DBA based embedding approaches on the following metrics as often investigated in literature [32–34, 39, 101]:

1. Overall acceptance ratio: is the ratio of successfully embedded VNRs and the total number of requests that arrived during the simulation.

2. Substrate links utilization: is a measure of the ratio of load accepted and the overall bandwidth ( and effective bandwidth) within the network.

3. Rejection or Blocking ratio per class: measures the number of VNRs per class that were rejected, expressed as a ratio of the number VNRs that arrived per class during the simulation.

4. Computational time: is the time required for an algorithm to process a VNR.

The DBvLEA is compared to a constrained SPF VLiM algorithm which epitomizes the SPFA design goal used in literature. It must be emphasized that SPFA MIP versions used in [32–34] from the analysis

1 If requests are not processed in the same order of arrival, the interpretation of embedding results can be erroneous as algorithms may selectively choose requests that improve their outputs

presented in section 5.1.1 produce the same results as constrained shortest path algorithms. This is also noted by Trivisonno et al. [32–34], where the gains in the case of their MIP version are as a result of the bulk processing of VNRs which can lead to situations where more VNRs belonging to a critical demand class may exist in a bulk submission of VNRs. To show this effect, a version of the VLiM algorithm is included, where an embedding solution is selected randomly from the feasible set of paths computed by the Yen's *k-paths* in Algorithm 5.1 line 7 tagged as Random Path Approach (RPA).

### 6.2.2 *Analysis of results*

Figure 6.2 shows the results for the overall average acceptance ratio and link utilization for a network size of 22 nodes and 300 VNRs over 10 simulation runs with a confidence interval of (2.26, 2.24), (2.32, 2.25), (2.21, 2.16) for DBA, RPA and SPFA respectively. Figure 6.5 shows similar results for increasing network size plotted as a bar chart.

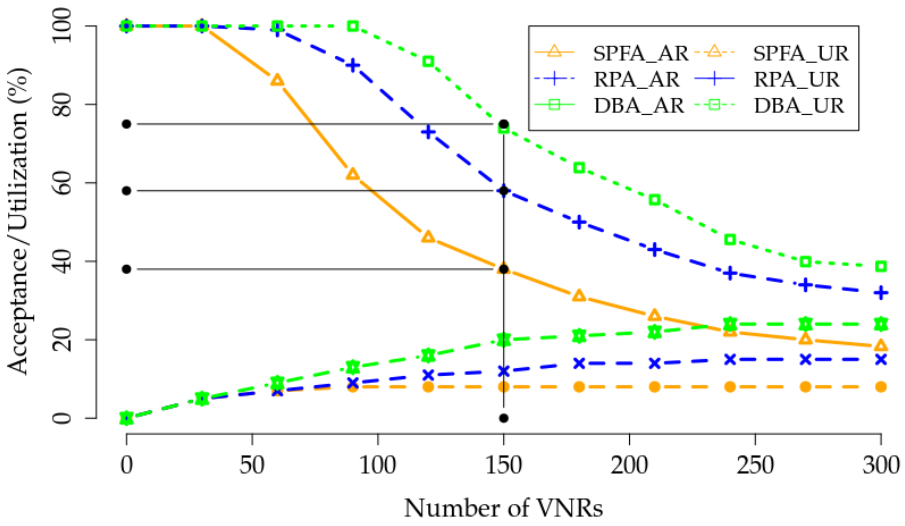



Figure 6.2: Acceptance on network size of 20 nodes

Referring to Figure 6.2, it can be observed that due a high performance substrate network conditions, all algorithms show good acceptance ratio in the beginning of the embedding process but gradually dwindles as the substrate network begins to saturate. This is made evident when the acceptance ratio is considered midway at which point the $150^{th}$ VNR has arrived for processing for all algorithms. It can be observed that SPFA acceptance ratio declines sharply compared to that of DBA. DBA and RPA shows approximately 20% and 36% gains in acceptance respectively compared to SPFA.

Also comparing DBA to RPA, significant gains in acceptance of about 16% is observed midway through the embedding process. The

reason for such gains in the acceptance ratio is further explained by examining Figure 6.3 which shows the VNR rejection ratio per class for respective algorithms. From the figure, it can be observed that class

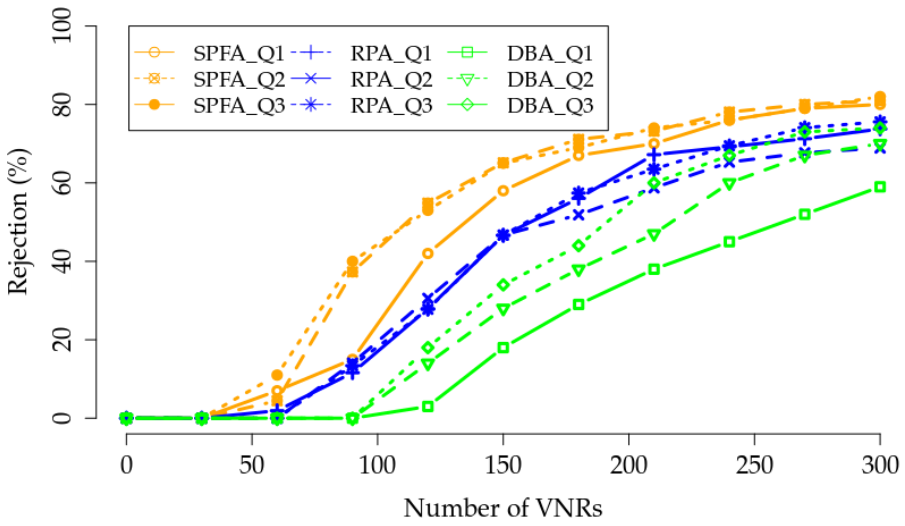



Figure 6.3: Acceptance per QoS class

1 experiences less blocking compared to the other classes for DBA. With class 3 being the most rejected class. However, comparing RPA and SPFA, it can be observed that class 1 and 2 are the most rejected classes for RPA and SPFA.

The reason for such results can be explained as:— SPFA and RPA reach early saturation for some classes as resources capable of embedding VNRs from class 1 and 2 are used up in the early stages by class 3 VNRs and thus leading to high blocking ratios. This is confirmed by examining how the substrate links are used by each of the algorithms in Figure 6.2. It can be observed that SPFA reaches early saturation consequently resulting in lower acceptance ratio likewise RPA.

Furthermore, very low substrate utilization is observed by close examination of Figure 6.2. The reason for the low utilization is solely due to the fact that link utilization metric is expressed as a ratio of total accepted capacity and the total substrate capacity as presented in [33, 34]. However, when considering VLiM problem in the SDN-TE context with unambiguously mapped source and destination, some substrate links be default are not considered in the solution space and thus may never be used. The bandwidth of these links are still considered in calculating the utilization ratio.

To efficiently evaluate the link utilization metric, *max-flow* utilization metric provides a better characterization. The metric defines the link utilization as a ratio of the accepted capacity and the maximum flow capacity that is required to separate the unambiguously mapped source and destination endpoints into two disjoint graphs. This defines the theoretical maximum capacity that can be attained by an ideal

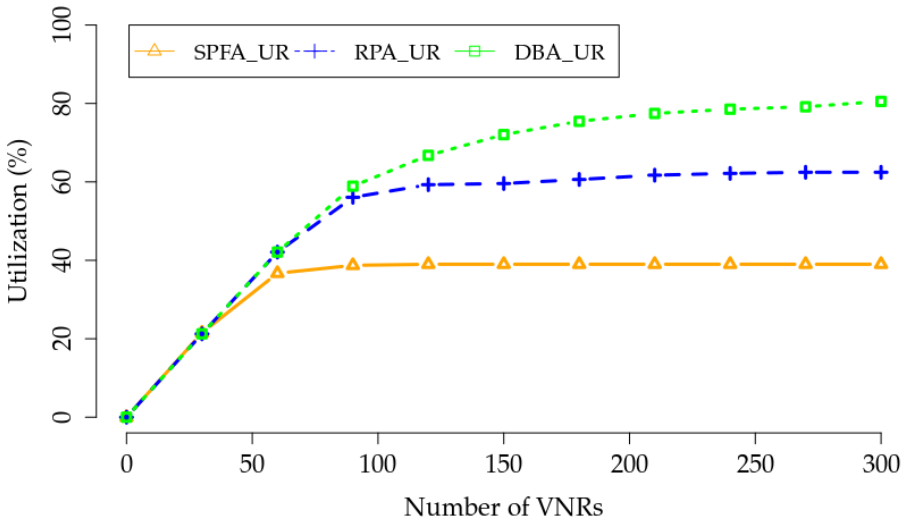



Figure 6.4: Accepted load per maximum effective capacity

VLiM algorithm. This new metric is shown in Figure 6.4 where the cumulative capacity of VNRs accepted in the network is expressed as a ratio of the *max-flow* capacity within the network. The figure shows how the different algorithms use the SN resources and how they contribute to their global acceptance ratios.

Also, a critical examination of Figure 6.4 shows a steep rise in

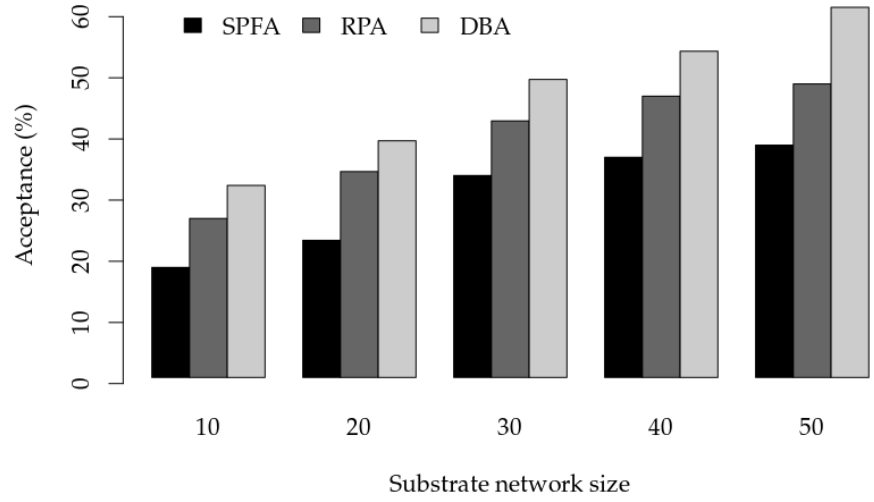



Figure 6.5: Acceptance per algorithm with increasing network size

capacity utilization of all algorithms at the beginning of simulation process but begin to decline slowly as the process reaches mid-point. It can be seen that at the arrival of the $100^{th}$ VNR, the SPFA can barely accept additional VNRs as it reaches saturation due to the greedy path selection, leading to sub-optimal utilization. However, DBA continues

to accept more VNRs due to its purposeful selection of paths and thus confirming the efficacy of the algorithm to the DBA hypothesis. RPA results can be attributed to the fact that some randomly selected paths from the feasible paths may sometimes coincide with similar solutions as the DBA. This phenomena is also the case when MIP versions embed VNRs in bulk as noted by Riccardo et al. [34]. Further examination of Figure 6.4 shows that even at the arrival of the $150^{th}$ VNR, DBA shows about 11% and 32% utilization gains over RPA and SPFA respectively while RPA also shows a gain of about 20%. This further highlights the efficiency of the DBA in the online multi-service VLiM use-case.

The overall acceptance ratio for the path selection approaches are shown in Figure 6.5, where a general increase with increasing network size is observed. However, this results can only be attributed to the fact that additional substrate nodes and links only enriches the solution space, therefore, more VNRs are accepted on the substrate network.

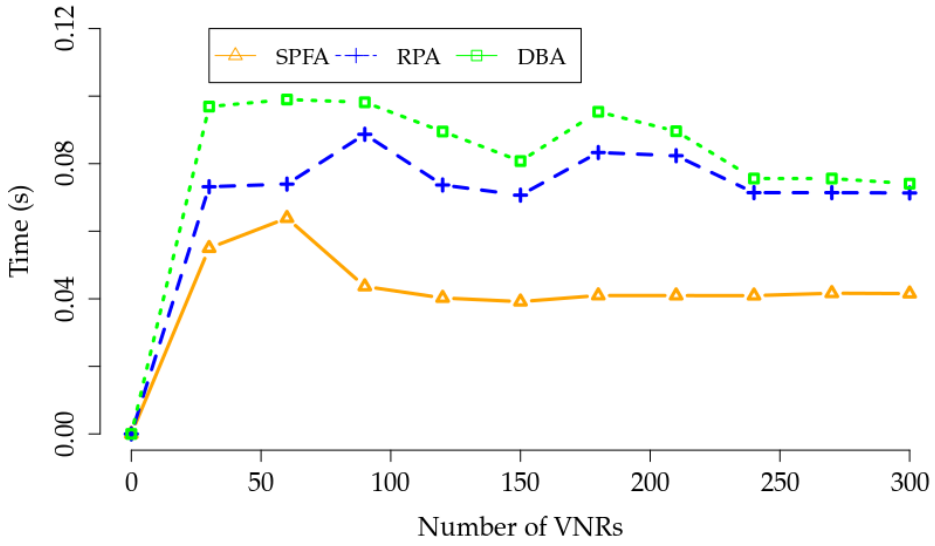



Figure 6.6: Computational time

Finally, a comparison of the algorithms in terms of computation time is examined. Figure 6.6 shows the average computation time for embedding a single VNR considering all the constraints involved. It can be seen that SPFA shows lower computation time than the DBA and RPA which is expected, given that the DBA and RPA find several paths from which one is selected as a solution while the SPFA only computes a single least-cost path for every VNR.

It can also be observed that the time gap between DBA and RPA reduces gradually to almost the same time as the simulation process approaches completion. This is because the number of candidate paths that have to be traversed to find the ODP at the beginning of the simu-

lation decreases as the substrate network approaches saturation. This explains the closeness of the computational time at the end. Regardless of the high computation time of the DBA approach, the computational time for DBvLEA is well below acceptable ranges compared to those in literature [34, 39].

## 6.3 EVALUATION OF DELAY ADMISSION CONSTRAINT

In section 6.2, performance focused on Objective 2 of the VNE problem where efficiency and efficacy of the algorithms are evaluated. This required the generation of unsolvable scenarios [27] and randomization of the SN to ensure that results are not dependent on the topological characteristics of the network nor the order in which VNRs are processed by the algorithms. However, this approach does not help us to understand whether the demand verification functions perform as desired in a production environment. This is particularly important in the QoS-aware problem because these functions ensure that the promised QoS can actually be guaranteed on the SN when resources are committed on the SN (VNE Objective 1).

For Objective 1 evaluation, a deterministic SN is a must. This allows the problem designer to track the solutions computed and compare them with the integrated models. The next section provides the evaluation of delay verification models.

### 6.3.1 *Simulation and analysis of delay verification models*

In this section, the ability and efficacy of VNE algorithms to apply the worst-case delay models to guarantee stringent demands of industrial applications is evaluated. The metrics considered are average worst-case delay guarantees due to reserved allocations of the various traffic classes as well as the delay due to the actual utilization of embedded NSs in the respective traffic classes.

A total number of 10 simulation runs are performed on a benchmark substrate network depicted in Figure.6.7 using the ALEVIN framework [47]. The SN from a real case-case deployment are bootstrapped, modeled in a Json format (see Appendix A.3) and fed to the VNE simulator.

The arrival of VNRs are randomized in accordance with the online mapping process to avoid the dependence of results on arrivals order of NSs. Furthermore, the same network parameters in the benchmark substrate topology is used in order to track the solution produced by the simulation platform so that they can be validated with numerical solutions as well. Multiple simulation runs are done only because the arrival order of VNR are randomized, however, the demands of each VNR remains the same in each run.

To the best of my knowledge, this is done contrary to VNE evalua-

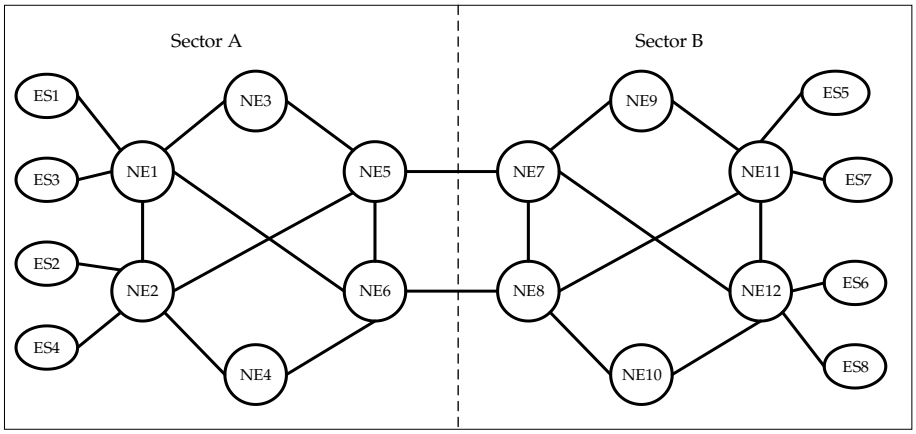



Figure 6.7: Benchmark topology showing End-Stations (ES) and TSN Forwarding Nodes of a production floor

tions in literature where the VNRs and SN generation are randomized at each simulation run. However, it is worth to note that the guarantees must be deterministic which means, it is important to keep a controlled simulation environment for validatable results. This also implies only solvable scenarios are generated in order to track solutions.

The simulations are performed on an SN that captures the tandem shaper/scheduler composition illustrated in Figure 4.3. As described, time-sensitive NSs consisting of *control-High* and *-Low* NSs are considered. The *control-High* NSs are instantiated solely for rapidly transmitting cyclic streams while the *control-Low* NSs are for less frequently transmitting applications. Also, *Surveillance* NSs for real-time video monitoring in the shop-floor and *Best Effort* NS for traffic with less stringent requirements on delay are considered.

The maximum demands, budget allocations per queue, and NS

Table 6.2: NS categorization w.r.t to demands, Queue assignment, and allocation per traffic class.

| NS class | Demands | | | Budget | Classification |
|---|---|---|---|---|---|
| | Burst (byte) | cycle [$\mu$s] | $w_d^{e2e}$ [ms] | (%) | |
| Control-high | 68-256 | 250 | 5 | 30 | TT |
| Control-low | 256-512 | 500 | 15 | 40 | CB/TT |
| Best effort | 512-1500 | 2000 | - | 30 | PB |

class are shown in Table 6.2. The *control-High* traffic are pre-allocated to time-triggered traffic class (TT) due to their tight requirement on delay. The *control-Low* and real-time video surveillance NSs are clas-

sified to credit-shaped traffic under CB classes and best effort traffic under PB class respectively. The embedding algorithm relies on the classifications to reduce the number of decisions that are required to map an NS to a queue.

It must be emphasized that this classification is a guess work that is done by the network engineer based primarily on the tightness of the delay demands. For this simulation equal traffic-mix ratios are generated for the different classes since it is a completely solvable scenario and there is no need to reject any VNR.

In order to ascertain the correctness of the results when comparing the models, the same substrate network is used for all simulation runs. A substrate network with link speed of 100 Mbps is considered for the simulation. The propagation and switching delays are neglected for simulation purposes. Given that they are bounded, they do not contribute additional information to the results.

Since deterministic periodic traffic are considered together with non-deterministic ones, the reservation per class can only be done within a network cycle where a busy-period occurs. It is therefore necessary to deterministically define a percentage of the network cycle allocated specifically for TT. Though a window for the credit-shaped classes is not entirely necessary, it must be defined in order not to starve non-credit-shaped priority queues controlled directly by the SPS as described in Figure 4.3. For this, a reservation of 30%, 40% and 30% each is defined for TT, CB and PB traffic, respectively.

Following the Trajectory Approach (TA) and the reservation per traffic class, the worst-case port delay contributed by the respective classes per the tandem scheduler is computed as per the delay models described in section 4.3.2 as follows:

- $q_d(CB) = \delta + \omega^{TT} \equiv \frac{12000}{10^8} + (0.3 \times 2.0\text{ms}) \approx 0.60\text{ms}$, where $\omega^{TT}$ is the queuing delay due to an entire budget for time-triggered traffic during a busy-period within the network cycle of 2.0ms.
- $q_d(PB) = w^{TT} + \omega^{CB} + \delta \equiv \frac{12000}{10^8} + (0.7 \times 2.0\text{ms}) \approx 1.4$ ms, where $\omega^{TT} + \omega^{CB}$ is the delay due to TT and CB budget within the network cycle in a busy period. Note here that $\delta$ is not considered for the lowest priority queue under PB.

### 6.3.2 *Analysis of results*

First, the worst-case port delay ($p_d$) for the respective NSs are examined as they are embedded (see Figure 6.8). Due to the FIFO intra-queue scheduling, it can be observed that there is a general increase in delay of the NSs as they are embedded. Particularly, it can be observed in Figure 6.8 how significantly it affects NSs in the CB and PB queues, however, the delay due to the TT queue remains fairly constant as they are unaffected by any form of queuing delay.

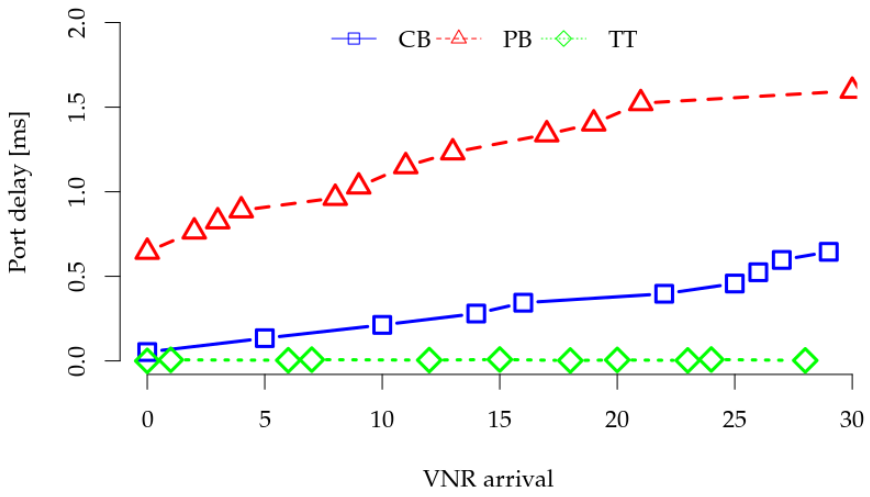



Figure 6.8: Actual delay guarantees per VN per QoS class

Also, still referring to Figure 6.8, considering the effect of the tandem composition of shapers described previously, the total delay due to NSs embedded in the TT queue are factored directly into the delay guarantees of NSs embedded in the CB queue, which also affects the NSs in the PB queue.

Notably, it can be observed that, the VNE algorithm considers the delay due to embedded NSs in the CB queue in that of the PB queue thus, even before the first NS is embedded in the PB queue, the port delay of PB due to CB and TT is exactly the same as the port delay of the last embedded NS in the CB queue which is $\approx 0.645$ ms. This is the actual queuing delay experienced by the first NS admitted to the PB queue. However, it must be noted that this is the best-case delay to be experienced by the NSs in their respective queues considering that the maximum allocated budget, $\omega$, of the respective classes are still under utilized.

Figure 6.9 further explains Figure 6.8 when the worst-case port delay due to unused budgeted allocation of the classes within the network cycle is examined. Here, a clear isolation of the NSs admitted to the respective classes by their worst-case delay bounds is observed. From the figure, it can be observed that the approach ensures clear isolation of NSs thereby ensuring delay guarantees regardless of the traffic class in which an NS is embedded. Also, the effect of embedding in the higher, same, as well as lower priority classes are shown as the VNE algorithm embeds the NS in an online manner. This outcome affirms that dynamic NS creation performs as expected.

This also implies the worst-case delay model are used accurately by the VNE algorithms. It is further validated by numerical analysis of the delay equations described for the respective queues. Also, comparing Figure 6.8 and 6.9, it can be seen that the instantaneous

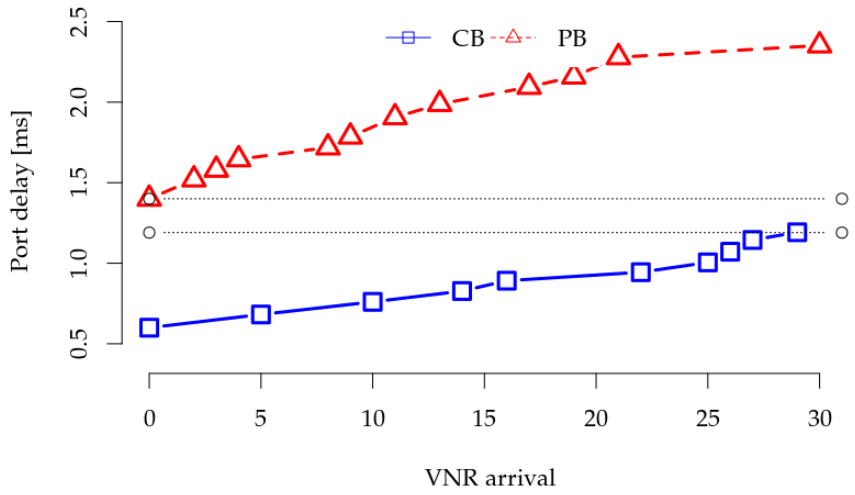



Figure 6.9: Worst-case delay guarantees per VN per QoS class

Table 6.3: Actual and worst-case e2e delays per path length

| Path length | Actual delay (ms) | | worst-case delay $w_d^{e2e}$(ms) | |
|---|---|---|---|---|
| | CB | PB | CB | PB |
| 5 | 3.22 | 8.0 | 5.96 | 11.76 |
| 6 | 3.87 | 9.58 | 7.15 | 14.11 |
| 7 | 4.51 | 11.17 | 8.35 | 16.46 |

(best-case) delay due to embedded NS cannot be guaranteed if NSs are continuously embedded till they exhaust their allocated budget within the network cycle. Some already embedded NSs may have their delays violated. This is prevented by the use of their worst-case delay bounds as the admission control hence emphasizes why the worst-case delay must be considered when creating NSs in TSN. Table 6.3 shows the actual e2e delay and e2e worst-case delays when all traffic classes exhaust their allocated link budgets over 5 to 7 hops. Note here that packets in the strict priority queues do not necessary need to have deterministic delays. Therefore, stochastic guarantees can be provided instead. This can be achieved by considering stochastic models that characterize the percentage-wise use of resource allocations by the deterministic streams to provide probabilistic guarantees. Stochastic Network Calculus (NC) can be leveraged together with the approach used in this thesis.

## 6.4 EVALUATION OF TREE MAPPING ALGORITHMS

As discussed in Section 5.2.2, there are scenarios where the NS to be created may have topology restrictions. A common topology restriction in factory automation is one where an AR/CR describes the transfer of information from a single application to more than one application or vice-versa e. g., PLC to multiple IO applications. The topology restriction on the AR/CR can be characterized by a tree topology. In this case, VLiM algorithms must be applied iteratively to realize the described AR. The iterative use of the VLiM algorithm can lead to different solutions for an NS based on the connectedness of the SN and selection of an anchor (root) node for the tree.

Once a root node is selected from the SN, a VNR illustrating a tree topology is created to represent the AR/CR. The VLiM algorithm is then used to map individual links in the VNR.

In this evaluation, the effect of root node selection on acceptance

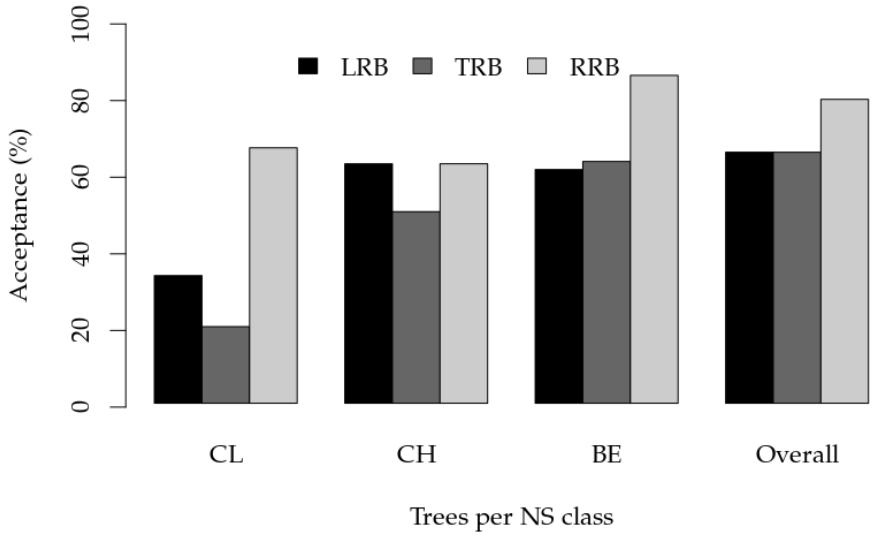



Figure 6.10: Acceptance ratio of VLiM algorithm based on Root node selection per traffic class

ratio and computational time is examined. Here, it is interesting to know which features provide the optimum results and time within which such results are achieved. This is particularly important since the algorithms are integrated as part of SFCs which provide an e2e service. Therefore, in as much as a particular feature provides better acceptance than another, the computation time is equally important.

Figure 6.10 shows the acceptance ratio of the respective root-node selection features per NS class and overall number of NS over 10 simulation runs.

Overall, the Rendezvous Root Bridge (RRB) selection yields the highest percentage of created NSs in comparison to Listener Root Bridge (LRB) and Talker Root Bridge (TRB) under the same simulation

settings. While LRB and TRB show equal performance, further analysis of how the VLiM algorithm performed in the respective traffic classes is illustrative.

Within the tight delay-constrained traffic classes, control-low and -high (CL, CH), the TRB shows very low acceptance while it performs slightly better than the LRB in the best-effort class. This indicates that the farther listeners are from the selected root bridge, the greater the chances that requests are declined due to delay and bandwidth demands. This analysis of the result is firmly established considering the RRB and LRB which selects a root node common to one or more participant in the tree.

Figure 6.11 shows the computation time, which includes the time

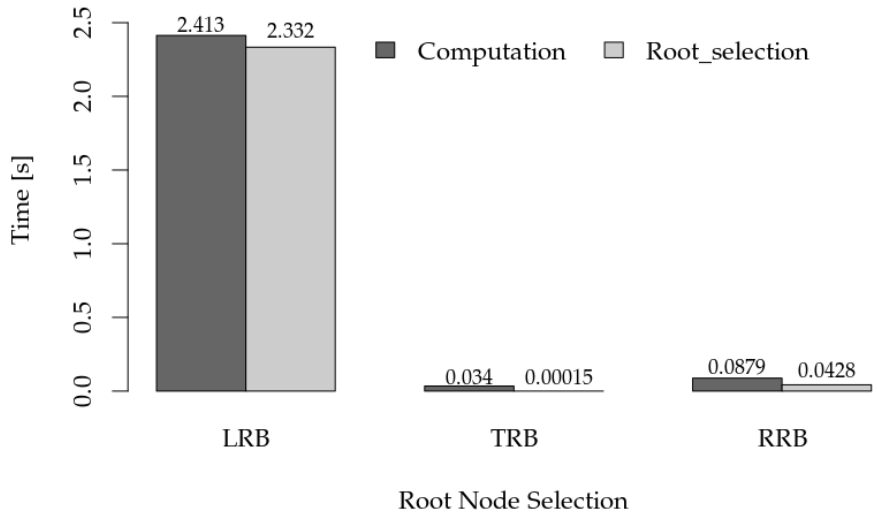



Figure 6.11: Computational and Root node selection Time

required for selecting a root node and time to perform resource allocations. From the figure, LRB shows the highest computation in comparison to TRB and RRB. Due to the wide time difference, LRB is not suitable for SDN deployment. On the other hand, RRB shows a good match between acceptance ratio and computation time and thus making it an efficient approach.

The TRB is largely employed in distributed tree algorithms for TSN which is specified in IEEE 802.1Q. The RRB provides are an efficient approach that can be realized centrally in an SDIN controller. Therefore, faster computational and resource efficiency provides a better alternative to distributed tree algorithms currently employed in Ethernet LANs.

## 6.5 EVALUATION OF TIME-AWARE SLICING ALGORITHMS

Whenever a valid schedule is feasible within a scheduling system, it can be said that the set of TT application is schedulable according to a scheduling algorithm (TAS). In this case, the scheduling algorithm can be evaluated on its effectiveness in following the defined schedules. Here, evaluation metrics such as the efficiency of utilization of resources lies primarily with the scheduling system.

The performance criteria of scheduling computation algorithms for hard real-time applications is one that is based on their ability to find feasible schedules w.r.t a given scheduling system whenever such schedules exist [77]. Often this may depend on the constraints of the scheduling system. Therefore, one can infer that a time-aware schedule computation algorithm for TT periodic applications is optimal if it always produces a valid schedule for a given job within a scheduling system, if and only if the schedule is feasible.

On the contrary, if the algorithm fails to find a feasible schedule, it could depend on whether the TT-periodic application is even feasible within the system at all. In this case, it can be concluded that clock-driven schedule computation algorithm cannot feasibly schedule the given TT application.

Therefore, in the subsequent section, the schedule verification algorithm (schedulability algorithm) is evaluated by analyzing its optimality in terms of finding a feasible schedule whenever one exists.

### 6.5.1 *Evaluating schedulability algorithm*

Schedulability algorithm can be evaluated based on its ability to assign exclusive time-slots to a system of independent periodic applications and/or based on the number of TT applications within the system, and the time required to compute a feasible schedule.

Considering the Algorithm 5.2, in line 11, the scheduling system is sorted in order of their periods. The scheduling system's order affects the number of TT periodic applications that can be scheduled within the system. This can be verified by algebraic analysis of equation (5.8).

A non-preemptive RMS algorithm which is known to be optimal [77] is expected to schedule all low rate applications (shorter periods) before high rate ones (longer periods). However, a careful analysis of equation (5.8) will show that a RMS algorithm does not result in a case where more TT application are schedulable.

Recall from equation (5.8) that the frequency of occurrence of the period of an application, $i$, within a scheduling cycle is $f_i = \frac{H}{p_i}$. Let $u_i = \frac{\epsilon_i}{p_i}$ be the utilization of the application within its own period. That is, the burst offloaded to the network within a period $p_i$ at an instant. If equation (5.8) is combined with the utilization $u_i$ of the application within its own period, the utilization of the application within the

scheduling cycle is obtained as $U_s(p_i) = \sum_{n=1}^{N} u_i$, where $N$ represents the number times $u_i$ occurs in the system. This can be represented as equation (6.1):

$$U_s(p_i) = \sum_{n=1}^{N} u_i = f_i \times u_i \tag{6.1}$$

In effect, equation (6.1) indicates how many time-slots units can be occupied by TT applications of a specific period within the hyper-period $H$ or the scheduling cycle.

From equation (6.1), it can be asserted that a reverse RMS ordering (i. e., decreasing periods of TT applications) may result in scheduling more TT applications with large periods than those with smaller periods. Due to the simplicity and lack of statistical variation, a performance via simulation for optimal mapping is not necessary since the system remains deterministic for a scheduling system.

Therefore, note that the utilization of the scheduling system is different from the acceptance ratios of independent periodic applications in the scheduling system achieved by VLiM algorithms.

If one is to go by the acceptance metric, which is based on the number of individual TT applications that the schedulability algorithm can schedule, then this metric is significantly affected by the order in which TT application are processed. Thus, scheduling applications with longer cycles first will achieve a higher number of scheduled applications than an scheduling applications with shorter cycles first. This is because high rate applications utilize more time-slots within the hyper-period than lower ones. However, this is flexible in the schedulability algorithm proposed in this thesis as applications can be processed in any order that suits the user.

Finally, the computational time of the scheduling algorithm is evaluated by providing its complexity. It must be emphasized that the convergence time is mostly irrelevant in the offline scenario; however, it is very relevant in the online case where schedules are computed dynamically within an SDIN controller. Given that the goal of the schedulability algorithm is to provide verification and dynamic resource allocation for TT applications in an SDIN-enabled TSN infrastructure, the convergence time must be within an acceptable service provisioning response time which is discussed in Chapter 7.

### 6.5.2 *Complexity analysis of time schedule computation algorithm*

To compute the schedules of all the bridges along a path with single rendezvous bridge, the algorithms are executed in order Algorithm 5.2, Algorithm 5.4 and then Algorithm 5.5. The temporal complexity of Algorithm 5.2 is $O(3 \times H \times N) \cong O(N)$, where $N$ is the number of applications to be scheduled, and $H$ is the hyper-period (i.e. $H = LCM(P_i, i = 1..N)$). Algorithm 5.4 is also polynomial as it has a

temporal complexity of $O(2 \times N) \approxeq O(N)$.

Finally, the time complexity of Algorithm 5.5 grows linearly with the number of bridges $M$ in the network (i.e. $O(M)$. In other words, the temporal complexity of the proposed system is linear to the size of the problem ($O(2 \times N + M) \approxeq O(N)$), which makes it a good candidate for an online schedule synthesis algorithm.

# 7 PERFORMANCE ANALYSIS OF SDIN-ENABLED TSN DEPLOYMENT

This chapter examines the performance of SDIN-enabled TSN in comparison to existing deployment models specified in IEEE 802.1Qcc.

## 7.1 OVERVIEW OF TSN DEPLOYMENT AND CHALLENGES

TSN/AVB standards specify a set of solutions which focus on resource allocation within Ethernet LANs. The solutions vary widely mostly due to advanced queuing and transmission techniques used on the FP as discussed in Chapter 4. These queuing techniques significantly impact how TSN infrastructures are deployed in industrial environments, e. g., process or factory automation industry. This is because of various QoS requirements within these markets [104]. In this regard, two solutions are proposed; "reserved-streams" and "scheduled-transmissions".

*contribution from [105]*

The concept of reserved-streams leverages a decentralized resource allocation mechanism enabled by Stream Reservation Protocol (SRP). Reserved-streams rely on predefined stream classes (e. g., Stream Class A and B ) and therefore, uses the CBS as an advanced queuing technique, thereby, enabling dynamic (plug-and-play) resource allocations across Ethernet based LANs.

The concept of e2e "scheduled-transmissions" uses TAS as an advanced shaping and queuing technique to provide ultra-low latency as discussed in Chapter 4. Therefore, scheduled-transmissions require detailed timing of streams at their sources (Talker ES) and on the network. Computation of time schedules for both talkers and the network as shown in Section 5.5 is generally a complex task, which requires complete knowledge of stream transmissions on specific network topologies.

The solutions described above impose three main deployment models which are specified in IEEE 802.1Qcc. These deployment models are discussed next.

### 7.1.1 *Fully decentralized deployment*

The fully decentralized deployment model is based on the concept of "reserved-streams" which emanates from AVB. It provides mechanisms which allow ESs to reserve bandwidth across a compliant LAN based on their willingness to "talk" or "listen". SRP enables this mechanism and it is built on a network management protocol called Multiple

Stream Registration Protocol (MRP). The basic function of MRP is to allow participants (Talker/Listeners) to register attributes which may be propagated through the network. Variants of MRP such as Multiple VLAN Registration Protocol (MVRP)–, VLANs and —Multiple MAC Registration Protocol (MMRP) exist.

The SRP standard provides a new MRP application called Multiple Stream Registration Protocol MSRP to manage attributes relating to the intent of applications and bandwidth reservation. In this regard, MSRP, MVRP, and MMRP provide all the network signaling for SRP (CP functions) within Ethernet bridges (cf. [106]).
The following provides an overview of these protocols. (See [106] for details on operations and implementations)

MRP is responsible for participant registration. A participant can make a declaration or withdrawal. Making a declaration results in the registration of the participant at its peer. This registration is persistent until a participant drops or issues a withdrawal. On the other hand, making a withdrawal results in the removal of pertinent attributes of the participant from its peer. Once a participant is registered, its declaration is propagated to all other participants.

MMRP Once a participant is registered by MRP, its declaration is broadcasted throughout the network. This is often inefficient as broadcasting declarations can create congestion within the network. In this regard, MMRP is used instead of the generic MRP. The fundamental difference is that instead of declarations being broadcasted, they are rather forwarded directly to specific destination Media Access Control (MAC) addresses, which makes it much more efficient. Therefore, the declaring participant provides the set of destination MAC addresses as an attribute within the declaration. This mechanism is referred to as talker pruning (see Appendix A.4 for talker data model).

MVRP works in a similar faction like MMRP, it allows participants to register to specific VLAN thereby allowing network bridges to tag declarations with their VLAN identifications (IDs). This implies propagation of declarations occurs within the VLAN where talkers and Listeners are members.

MSRP manages the intent of talker and Listener registrations. This intent can be a Talker wanting to transmit data or Listeners announcing their readiness to listen to publications if they exist.

SRP using the above MSRP/MRP, SRP maintains a set of tables in each bridge. These tables keep records of relevant operational information such as per-stream reservation and bridge-wide data which

enables it to keep track of ports that are SRP-enabled. Additional information includes all registered Talkers and Listeners, their rank, latency between peers, bandwidth, etc. See AppendixA.4 for MSRP data models.

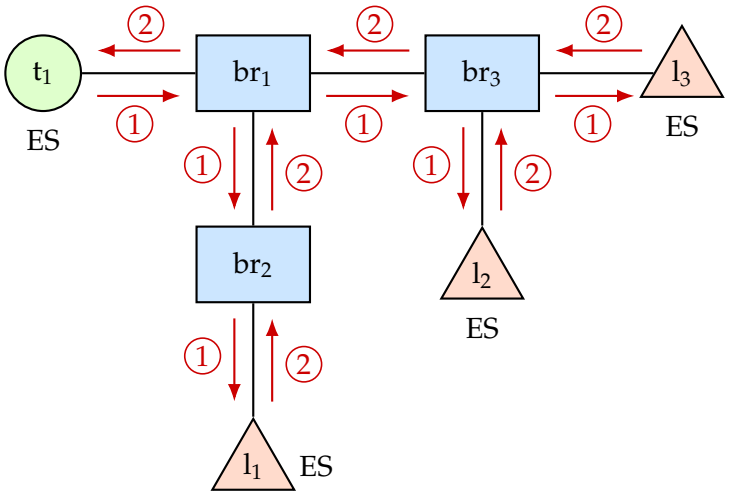



Figure 7.1: DRM based on the fully distributed model of existing MSRP

Figure 7.1 provides a simplified-non-formalized stream reservation (see IEEE 802.1Qcc for detailed description) as an example of the MSRP/MRP flow for Decentralized Reservation Model (DRM).

As shown, a Talker application $t_1$ (e.g., PLC, microphone, camera) ES sends a stream advertisement (*TAdvert*) to a TSN bridge $br_1$, the stream identification (*StreamID*) is resolved, registered and then propagated through the network using the destination multi-cast or uni-cast MAC address specified in the MSRP packet. This is repeated by all TSN bridges between the Talker and Listener(s) (i. e., $br_2$ and $br_3$) till the *TAdvert* packet reaches the intended Listener(s) (e.g. actuator, robot arm, speaker) ESs. This process is depicted by arrows ①.

Upon receiving the *TAdvert* packet, Listener applications (i. e., $l_1$, $l_2$, $l_3$) on the ES generate and send Listener-ready messages (*LReady*) if it has the credential and desire to receive packets from the Talker stream.

Such credentials are determined by the stream identifier (*StreamID*) which is composed of the Talker ES MAC address and a unique application ID (*AppID*). The combination of the *AppID* and the MAC address of the ES makes a *StreamID* unique in the LAN even if streams are transmitted from different Talkers on the same ES.

The *LReady* packet triggers the bridges ($br_3$, $br_2$, $br_1$) to reserve resources for the forwarding of packets between the Talker and the Listener(s). The *LReady* packet(s) are propagated to the Talker by the bridges in addition to information on the status of the stream reservation process. Upon receiving the *LReady* packet, if the status of the

reservation is successful, the Talker application begins to transmit packets with payload, which are then forwarded by the bridges to the Listeners using the reserved resources. The advertisement and registration processes are represented by the arrows ① and the subscription/reservation processes by ②.

### 7.1.2 *Fully centralized deployment*

The fully centralized deployment model unlike the DRM is mainly driven by the need for ultra-low latency, which as discussed extensively in chapters 4, 5 and 6 is achievable only through time scheduling. This approach, therefore, employs the use of a Centralized Network Configuration (CNC) for path and schedule computations as well as resource allocation.

Due to the difference in the forwarding mechanisms, operators are forced to either adopt DRM or Centralized Reservation Model (CRM) depending on the latency guarantees they want to achieve.

Within the CRM, the CNC achieves resource allocation by configuring bridges and ESs. The specification of configurations is defined in the IEEE802.1Qcc whilst parameters for scheduled traffics are specified in IEEE802.1Qbv.

An enhanced version of MSRP (MSRPv1) is provided to serve as a transfer protocol for carrying stream configuration over links between each ES and the peering bridge, which is also accessible to the CNC.

MSRPv1 operates in a slightly different way from its original version that propagates information across the network and performs resource reservation hop-by-hop on each bridge. If MSRPv1 is applied in a fully decentralized manner, it behaves essentially the same as the MSRPv0 with support only for CBS (cf. [104]).

The main advantage of CRM is its capacity to allow centrally configurable time-schedules to achieve ultra-low latency. This also makes it rigid and inflexible.

Furthermore, CRM is not event-based. Therefore, it does not incorporate any form of service dynamism (plug-and-play) as the whole reservation process is preceded by an offline pre-engineering phase. The reservation itself is proactively triggered by a Central User Configuration (CUC) and requires a human-in-the-loop.

### 7.1.3 *Hybrid deployment*

The TSN working group also proposed a hybrid deployment model where fully DRM and CRM could coexist on the same network. In the HRM, each reservation solution operates separately as described. This can, however, lead to several issues which have currently not been addressed. Some issues identified are described as follows:

- Definition of common information models for MSRPv1 and MSRPv0 such as domain enhancement to include scheduled traffics.

- Inconsistency introduced by resource allocations created by MSRPv0 and CNC via User Network Interface (UNI). The fundamental operation of MSRP enforces that resources are deallocated once network requirements such as maximum latency are exceeded. This generates a failure code to the Talkers and Listeners. On the other hand, allocations deployed by the CRM via the UNI may persist even when maximum e2e latency is violated. This can create resource fragmentation as well as a false representation of resource states.

In light of these deficiencies, this Thesis proposes an extended hybrid model that leverages on SDN concepts discussed in Chapter 2 and treated extensively in preceding chapters. This proposed solution is discussed next.

### 7.1.4 *SDIN: hybrid deployment model*

The underlying principle behind SDIN-enabled TSN is the use of an SDN controller as a feedback control that can proactively and reactively allocate resources as an alternative to existing solutions, ("scheduled-transmissions" and "Reserved-streams"). [1]

The controller integrates among other functions, the online slicing and schedule computation algorithms discussed in chapter 5 and 6. Figure 7.2 illustrates the proposed SDIN-enabled HRM. The controller consist of 3 functional blocks:

ORCHESTRATOR FUNCTIONAL BLOCK is responsible for orchestrating service functions based on defined set of SFCs. Here, SFCs represent a chain of network functions and computational models in the SDIN controller required to achieve a service objective. Such objectives include the creation, deletion, modification and monitoring of Communication Service (CS) and Network Slice (NS).

A core function of the orchestrator is to handle network events that are propagated from the DP or Remote Procedure Call (RPC) via North Bound Interface (NBI) to the controller. Based on the label of the event published via notifications or RPCs, the orchestrator executes an SFC that is designed to establish an e2e service. Within this functional block are components such as topology management and policy engines:—define the set of policies that determine how specific events should be handled.

1 SDIN is developed on the premise that the most efficient distributed systems are those orchestrated by a centralized coordinator

RECORDS DATA BASE is responsible for keeping statistical records of the network which includes state of virtual resources such as NSs in relation to physical resources in the DP. A core function of this functional block relevant for HRM is the ability to keep track of all MSRP declarations and withdrawals. It serves as a resolution and registration component for the HRM. Records such as AR and CR, VLAN membership are also stored in this functional block.

COMPUTATION AND ENFORCEMENT FUNCTIONAL BLOCK is responsible for network slicing, path computation and functions used to configure or deploy configurations to the Forwarding Information Base (FIB) and MIB subsystem of the DP. It integrates all the algorithms and computational models discussed in this Thesis. Additionally, it provides mechanisms for translating VNE and scheduling outputs into the specific data structures of the FIB/MIB subsystems.

For TSN, the NETCONF protocol is used for configuring TSN bridges, therefore the functional blocks rely on the flexible configuration system and the YANG subsystem to wrap parameters to specific bridges. Configuration can be delivered by a CUC as done in the CRM or can be event-driven via the orchestrator functional block with parameter via UNI.

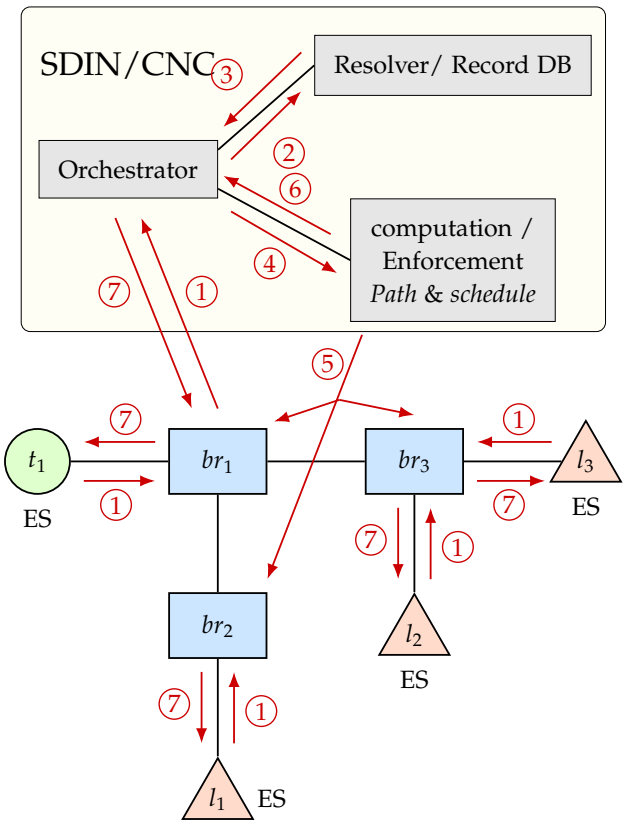



Figure 7.2: SDIN: HRM based on SDN concepts

#### 7.1.4.1 *Flow of MSRP packet in SDIN-enabled HRM*

Figure 7.2 illustrates the sequence of events flow through the SDIN-enabled HRM.

In this deployment, two mechanisms can be defined;— 1. Listener(s) send requests via the edge bridge to the SDIN controller, expressing their desire to join a stream (i. e., whether the stream is advertised or not) by issuing an *LJoin* message. *LJoin* message represents the label (or topic) of the notification event published by packet propagation of the MSRP at the edge bridge to the SDIN controller;— 2. Listener(s) express their readiness to join a stream by sending an *LReady* message to the SDIN controller only after receiving notification (*TAdvert*) from the SDIN controller about an advertised stream.

In the case of the former, the Listener(s) actively send *LJoin* requests which are forwarded to the SDIN-CNC as notifications or *packetIns* by the edge bridges. Likewise, a *TAdvert* request from a Talker ES is forwarded to the controller whenever it is published to the edge bridge. The advertisement and Listener join procedures are illustrated in Figure 7.2 with ①.

If a Listener tries to subscribe to a stream that is not advertised to the SDIN-CNC, the SDIN-CNC either replies with an error code, *StreamNOTFound* message or may decide to ignore the subscription request entirely. That is, Listener(s) can be registered in the SDIN-CNC stream DB and tagged with the stream identification it wants to subscribe to. Subsequent *LJoin* requests bearing the same *StreamIDs* from the Listener(s) are then ignored.

Alternatively, the SDIN-CNC may ignore all *LJoin* requests from a Listener if a Talker has not advertised the stream to which the Listener wants to subscribe. Note that the two procedures are possible design choices which can affect the SDIN-CNC processing time in general. In the case of the later procedure, Listener(s) send *LReady* message to the SDIN-CNC only as a reply to the Talker advertisement sent to them by the SDIN-CNC.

In either mechanism, receiving an *LJoin* or *LReady* notification at the SDIN-CNC starts a chain of processes defined by an SFC (e. g., path computation, configuration over NETCONF agents) in the controller which subsequently leads to reservations on the TSN bridges. This is illustrated by ② to ⑥ as shown in Figure 7.2. Once the Talker receives (LReady) notification from the SDIN-CNC (⑦), the Talker can send data to the Listener(s) via the DP of the TSN-enabled LAN.

The *TAdvert* requests are also propagated to the SDIN-CNC just like the *LJoin* and *LReady* requests from the Listener(s). However, unlike the *LJoin* requests, all *TAdverts* bearing different *streamIDs* are processed by the CNC since they represent entirely different stream requests.

Upon receiving a *TAdvert* notification, the SDIN-CNC resolves it by firstly checking whether a *TAdvert StreamID* is not already registered.

If this check is valid (true), the *TAdvert StreamID* and related data fields (see Appendix A.2) which includes network requirements and traffic specifications of the Talker application are entered into a stream configuration registry as a registered Talker.

On the other hand, if a stream with the same *StreamID* exists, then it implies there is an active Talker bearing this *StreamID* in which case, the edge bridge sends a *StreamExist* message to the Talker. *StreamExist* is a keep-alive message which is also present in the DRM. However, it must be noted that this aspect of the process does not directly affect the actual processing time for stream reservation in both deployment models. This is because *StreamIDs* registered on the edge bridge have been processed already; hence, there is no need to reserve resources for such streams. Such packets are effectively dropped at the network edge. This is important because it ensures that, no redundant resources are reserved for a specific talker application.

The keep-alive messages are necessary for ensuring that resources are reserved for only active Talker(s) and Listener(s). The stream resolution and registration process are illustrated in Figure 7.2 by arrows ② and ③. Once a Talker stream is registered, and there is Listener subscription(s) to the registered stream (determined by the *StreamID*), the CNC computes paths and necessary network requirements and enforces the reservation by configuring the bridges using NETCONF agent (client). The path computation includes consideration of network requirements such as bandwidth, delay (e. g., from the analysis in Chapter 3), online schedule synthesis discussed in chapters 5 and 6.

A Listener or Talker may withdraw from participating in a stream. In this regard, either participant may send a leave message via MSRP to the SDIN-CNC. In the case of a Listener withdrawal request, if the Listener is the last listening participant in the stream, the SDIN-CNC de-allocates all reserved resources, pushes the Talker to a configuration Data Base (DB) and informs the Talker about Listener(s) departures. Otherwise, the SDIN-CNC de-allocates only resources used by the departing Listener. This ensures efficient resource utilization and seamless fault resolutions.

A Talker withdrawal on the other hand results in de-allocation of resources for the entire stream from the DP. Furthermore, all information relating to the stream is deleted from the configuration and operational DB of the SDIN-CNC. See Figure A.2 in Appendix A.4 for proposed Talker/ Listener notification model for SDIN-enabled HRM.

## 7.2 QUALITATIVE PERFORMANCE ANALYSIS OF TSN DEPLOYMENT MODELS

### 7.2.1 *Benefits and deficiencies of DRM*

A significant benefit of DRM is its plug-and-play mechanisms which allows ESs to dynamically reserve resources across LANs. However, it is inflexible due to statically defined classes. These classes restrict the free propagation of packets, making it difficult to allocate stream for an application that does not belong to any of the AVB stream classes. That is, attributes of a stream are only propagated to ports that support AVB stream classes.

Additionally, this inflexibility does not allow for e2e scheduled transmission aside from the inherent difficulty in time-schedule computation. This limits the potential for utilizing TSN features such as the Time-Aware Scheduler (TAS).

### 7.2.2 *Merits and deficiencies of SDIN-enabled HRM*

SDIN-HRM proposed in this thesis combines the dynamism of DRM with CRM in an automated manner. That is, the stream reservation process in the TSN-enabled LAN can be reactively triggered by the Talkers and/or Listener(s) via notifications from the DP to the orchestrator functional block or proactively by the CUC through the controller's exposed service NBIs.

*zero to low touch service provisioning*

Aside from the dynamism it introduces, the centralized controller has complete oversight of events within the DP, and can react appropriately to changes. This prevents the short comings of the generic HRM approach where fragmentation of resources can arise due to resource allocation by two different mechanisms that are unsynchronized (i. e., via UNI ("Scheduled-transmissions") and SRP ("Reserved-streams")).

Furthermore, with the SDIN-HRM, both "reserved streams" and "scheduled transmissions" can be achieved using the online schedule computation algorithms described in the previous chapters of this thesis. Thus enabling industrial operators to use all TSN features without the restrictions imposed by the existing reservation solutions.

Chen [104] and Nasrallah [107] et al. argue that centralized control, in general, may be superfluous or add unnecessary complexity to a small-scale TSN network and therefore, increase CAPEX/OPEX. Contrarily, it can be argued that centralized control such as SDIN-CNC proposed in this thesis, can be easily evolved by integrating new network functions and SFCs to augment control functions of the TSN DP and will therefore reduce long-term CAPEX as well as OPEX.

A disadvantage of the SDIN-HRM is the Single Point of Failure (SPoF) problem which questions the reliability and availability of centrally managed systems in general. However, the unavailability of the

SDIN controller does not affect the operation of the existing streams on the DP. This becomes a problem only when faults occur in the DP or when new streams are to be allocated resources in the DP at a time the SDIN-CNC is unavailable.

The SPoF problem can be ameliorated using multi-controller clusters. Though it is not the focus of this thesis, issues regarding the problem are discussed by Sakic et al. [108–110].

Additionally, the centralized SDIN controller is the only means by which resources can be provisioned or de-allocated in the SDIN-enabled HRM. In as much as the benefits outweigh the stated deficiencies, it also introduces a huge attack surface which if not tackled properly can be very detrimental to industrial operators as attackers can use them to their advantage to circumvent the production system as argued in the following publications: [111–113].

After presenting the qualitative performance advantages of the SDIN-HRM deployment, the subsequent sections provide a quantitative performance analysis of the response time of the different infrastructural deployments from a resource reservation context.

## 7.3 QUANTITATIVE PERFORMANCE ANALYSIS OF DRM AND SDIN-HRM

In order to develop a quantitative performance model for the different reservation models, an underlying performance theory is required. The subsequent section examines performance theory based on the setup defined in Figure 7.1 and 7.2.

### 7.3.1 *Performance theory construction*

The performance theory begins with a generic construction of the system for a single Talker and Listener interaction consisting of $n$ bridges and a controller representing an SDIN-CNC. The modeling theory follows Equations (7.1) and (7.2) for the DRM and HRM deployments respectively as shown in Figures 7.1 and 7.2. This can be easily extended for any combinations of ESs and bridges.

$\alpha_L$=*Listener processing time*

DECENTRALIZED RESERVATION MODEL (DRM)

$$(2n+2)d_t + 2n(\alpha_{br} + d_q) + \alpha_L \tag{7.1}$$

$\alpha_{br}$=*bridge processing time*

where $\alpha_{br}$ is the internal processing time of a TSN bridge which includes resolution, registration, resource computation and reservation done by SRP, $d_q$ is the queuing delay at the bridges, $d_t$ is the delay between bridges and ESs for transmission of MSRP packets[2], $n$ is the number of TSN bridges between the Talker and its Listener(s).

2 Note that $d_t$ includes queuing, forwarding and propagation components of the delay.

SDIN: HYBRID RESERVATION MODEL (HRM)

$$4d_{t(es)} + 2d_{t(nc_T)} + 2d_{t(nc_L)} + \alpha_{nc} + d_{q(nc)} + \alpha_L \tag{7.2}$$

$\alpha_{nc}$= *controller processing time*

where $\alpha_{nc}$ is the internal processing time of the controller which includes resolution, registration, resource computation and reservation, $d_{t(es)}$ is the transmission delay between an ES and edge bridge (i. e., $br \rightarrow es$), $d_{t(nc_T)}$ is the delay between edge bridge and controller for Talker messages, $d_{t(nc_L)}$ is the delay between edge bridge and the controller for Listener messages. $\alpha_{nc} = \alpha^t_{nc} + \alpha^l_{nc}$, where $\alpha^t_{nc}$ represents the controller processing time for a *TAdvert*. $\alpha^l_{nc}$ is the controller processing time for *LReady* or *LJoin* packets. $d_{q(nc)}$ is the queuing delay at the controller stations.

From Equation (7.2), it can be observed that while the response time of DRM increases in direct proportion to an increasing number of bridges, SDIN-HRM is not directly dependent on the number of TSN bridges. The dependence on the number of bridges is considered as a fraction of the processing time of the controller $\alpha_{nc}$ i. e., the time it takes the controller to centrally allocate resources on $n$ bridges. On the other hand, SDIN-HRM response time can be severely affected by the position of the controller w.r.t. the edge bridges. While this is a great concern in SDN deployments for data-centers or cloud networks where the controllers can be located geographically farther from the network itself as discussed in these articles [114–116], contrarily, in industrial networks the controller position is required to be on the same factory or shop floor. Usually a few meters from the bridge LAN. Hence, this effect though significant, is within the same order of response time measured for a hop in DRM.

It must be noted that the equations assume the delays ($d_t$) are symmetric in either direction. However, this is not exactly true since queuing component of delay ($d_q$) varies due to frequency of packet arrival (arrival rate) and how fast the device is can process them (service rate). Regardless, the assumption remains valid even for the physical system if the delay is considered w.r.t to transmission and propagation of control packets on the forwarding lines.

## 7.4 MODELING METHODOLOGY AND DRM/SDIN-HRM MODELS

This section describes the modeling methodology for the performance analysis. As described previously in section 7.1, Figures 7.1 and 7.2 show a non-formalized model of the communication present in industrial TSN LANs. While Figure 7.1 shows the currently employed decentralized approach based on SRP, Figure 7.2 depicts the novel hybrid approach utilizing a centralized SDIN controller as a CNC entity.

For performance evaluation, one can leverage field/test measurements, analytical (closed form/numerical) or a simulation approach

[117]. For the latter, modeling formalism such as Markov Chains- and PN-based solutions as well as QN are often employed. In this thesis, a combination of deterministic PNs and QNs is used for generating performance models for the different deployments.

The use of deterministic PN and QNs allows one to capitalize on the modeling power and expressiveness of both formalism. That is, it enables the integration of hardware as well as software aspect of the system's behavior in a single model which makes it possible to simulate simultaneous resource possession or contention as well as synchronous and/or asynchronous processes. It therefore fits perfectly to the deployment scenarios described in Section 7.1. Additional advantages of a combined PN-QN compared to the individual QNs or PNs formalism are discussed by Kounev et al. [118].

The following presents the PN-QN concept and models for DRM and SDIN-HRM.

### 7.4.1 *Petri-Nets analysis*

PETRI- OR PLACE/TRANSITION NETS The general idea of PNs is to describe state changes in a system with transitions, also known as place/transition or in short PT-nets. PNs are bipartite directed graphs that contain two sets of nodes:— transitions, $\mathcal{T}$, (i. e., events that may occur in a system), places, $\mathcal{P}$, (i. e., conditions for events). Relationships between nodes are formed by arcs. A PN is therefore bipartite in the sense that arcs cannot directly connect nodes of the same type;— rather, arcs connect places to transition (input arcs) and transitions to places (output arcs). PNs as per Cassandras et al. [119] are defined in the following:

**Definition 7.4.1** (Petri Net)**.** PN is a weighted bipartite graph given by a 4-tuple $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{W})$ where $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ is a finite set of $n$ places; $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ is a finite set of $m$ transitions; $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is a set of arcs; $\mathcal{W} : \mathcal{F} \to \{1, 2, 3, \cdots\}$ is a weight function on the arcs; $\mathcal{P} \cap \mathcal{T} = \emptyset$ and $\mathcal{P} \cup \mathcal{T} \neq \emptyset$.

In PNs, a place may contain an arbitrary number of tokens. A state, $s$, defines a possible assignment of tokens to all places in the PN such that $s \in \mathcal{S} \subset \mathbb{N}^{|p|}$, $p \in \mathcal{P}$, describes the number of tokens in each place.

A transition with an input arc is enabled if its input place contains one or more tokens. Transitions without input arcs are always enabled. An enabled transition fires by removing a token from its input place to its output place via its output arc. When a transition fires, it transforms a PN from one state to another. A state in which no transition is enabled is termed an absorbing state.

Given an initial state, $s$, a "reachability" set $\mathcal{R}_s$ defines the set of states that can be reached via any number of firing sequence. There

can be infinite state generation in PNs depending on the number of places and firings. Especially, when multiple transitions are enabled simultaneously, transitions can fire in any order hence making them nondeterministic. However, with the use of execution policies, firings can be made deterministic. This expressiveness makes PNs more suited for modeling stochastic and deterministic systems. Therefore, this thesis leverages this expressiveness of deterministic PN to model the process of stream reservation in TSN.

### 7.4.2 *Queuing network analysis*

QUEUING SYSTEMS A Queuing System (QS) is a node that consists of queue(s) and server(s). When a packet arrives at a QS, it is queued. Eventually, it is processed and thus, incurs a certain amount of delay. This delay constitutes the time the packet had to wait to be serviced and the time spent to process the packet. The resulting service time often follows a probabilistic distribution influenced by a set of conditions such as the frequency of the arrivals and rules governing how services are delivered, and how fast the service can be delivered.

**Definition 7.4.2** (Queuing System)**.** QS is defined by Kendall's notation as $\mathcal{A}|\mathcal{S}|m|\mathcal{B}|\mathcal{P}|\mathcal{Q}$, where $A$ is the probability distribution of inter-arrival times, $\mathcal{S}$ is the probability distribution of service times, $m$ is the number of servers, $\mathcal{B}$ is the buffer size, $\mathcal{P}$ is the population size, and $\mathcal{Q}$ is the queuing discipline.

Multiple QSs form a Queuing Network (QN), see Definition 7.4.3. After a packet is processed in one QS, the packet is sent to the next connected QS. If multiple connections exist, a probability is assigned to each link to determine the likelihood of packets being sent through the link. Generically, in QNs, packets are sent from one QS to another until they reach the end of the network [120].

**Definition 7.4.3** (Queuing Network)**.** A QN is a directed weighted graph $QN = (\mathcal{V}, \mathcal{E}, w)$ with $\mathcal{V} = \{\mathcal{Q}\}$, $\mathcal{E} = \{\mathcal{R}\}$, and $w(\mathcal{E}) \rightarrow \mathbb{R}$, where $\mathcal{Q}$ is a set of queuing systems and $\mathcal{R}$ is a set of directed edges with an associated weight function $w(\mathcal{E})$ representing either routing probabilities or arrival rates.

QNs are classified in two distinguishable forms

- **Closed QNs** have a fixed amount of packets sent through the network. Whenever a packet reaches the "end" of the network, it is returned to the start. Thus, Closed QNs have a fixed number of packets within the system. Although it is possible to analyze response time, throughput and utilization at specific QSs, they are not useful for modeling industrial communication Ethernet networks such as those considered in this thesis. This is because an arbitrary amount of packets pass through such networks.

- **Open QNs** are better suited, since they have a source that creates packets and sends them into the network at predefined rates (= arrival rate $\lambda$). Similar to the service time $\mu$, the arrival rate can be deterministic or randomly distributed and therefore, follows a probabilistic distribution.
  Packets exit the network at the drain of the system. The time spent to traverse the QN before the packet exiting the QN, is known as the response time.
  In the case of the presented DRM model the process is initiated with the Talker that generates the request, the absorbing state (drain) is the point in time when the fulfillment of the reservation process has occurred for one or more Listeners. Thus, this is the point in time when the Talker receives an *LReady* packet from the first Listener upon successful acknowledgment of the stream reservation.
  In the case of SDIN-HRM, however, the source of the request can also be the Listener sending a *Join* request, the drain being the time when the SDIN-CNC fulfills the request and acknowledges the Listener.

Open QNs are generally unstable if packets arrive faster than they are processed. This causes packet queuing at specific stations in the system. Short bursts of packets are allowed but the station must be able to process all packets eventually. Otherwise, the queue of the station grows indefinitely and may not be completed.

To achieve accurate or precise results, it is necessary to plot the exact service distributions of the system components in the QSs. However, a much practical approach is to fit the distribution of the system under study to well known distributions which are usually integrated in open source performance tools. The detailed workflow for this approach is discussed next.

## 7.5 PERFORMANCE WORKFLOW AND ANALYSIS

Prior to the generation of PN-QN models, the processing time of the SDIN-CNC stations for SDIN-HRM and the SRP-based bridges for DRM are empirically measured, and used to define the service parameterization of the various QSs in the PN-QN model.

The measured data of the test systems (SDIN-CNC and TSN bridge) are depicted in histograms in Figures 7.3 and 7.4. We use the goodness-fit test to find standard distributions that best fit the measured data.

In a statistical hypothesis testing, $H_{0|1}$;— the goodness-fit test has a null hypothesis, $H_0$, and an alternative hypothesis, $H_1$. That is, either;

- $H_0$: The measured data follows the hypothesized distribution;
- $H_1$: The measured data does not follow the hypothesized distribution.

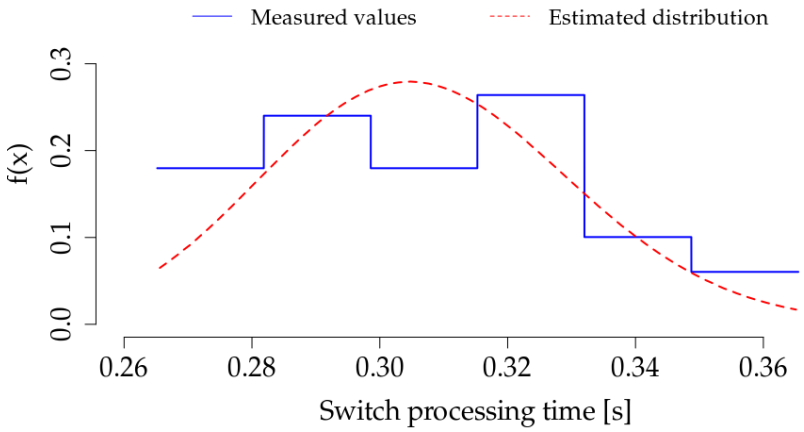



Figure 7.3: Gamma distribution estimating the switch processing time

Using this approach, Anderson-Darling [121, 122], Kolmogorov-Smirnov [123, 124] and Chi-Square [125] goodness-fit tests are used to identify fitting distributions for the processing times. The goodness-fit distribution of the TSN bridges, Listener, and controller processing times are visualized in Figures 7.3 and 7.4, as well as Table 7.1, respectively.

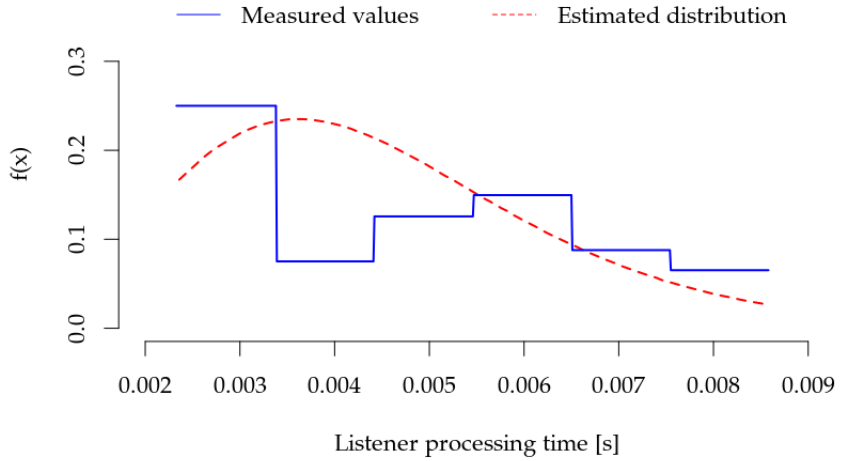



Figure 7.4: Gamma distribution estimating the Listener processing time

**Switch processing time** follows a gamma distribution (Equation (7.3)) with parameters $\alpha = 163.68$, $\theta = 0.00187$, and data mean $(x) = 298\ ms$.

$$f(x) = \frac{x^{\alpha-1}}{\Gamma(\alpha) \times \theta^{\alpha}} e^{-\frac{x}{\theta}} \tag{7.3}$$

**Listener processing time** represents the time duration between the reception of a *TAdvert* and generation of *LReady* message. It follows a gamma distribution with parameters $\alpha$ = 36.348, $\theta$ = 0.0018, and data mean $(x)$ = 6.54 $ms$. Listener processing time defines the time required by the Listener to generate *LReady* message after receiving a *TAdvert* message as shown in Equations (7.1) and (7.2).

**Controller processing time** distribution for *TAdvert* and *LReady* or *LJoin* follows a uniform distribution (Equation (7.4)) and gamma distribution (Equation (7.3)) with parameters as shown in the Table 7.1.

$$f(x) = \frac{1}{\max - \min} \tag{7.4}$$

Table 7.1: Processing time distribution for *TAdvert* and *LReady/LJoin* messages

| **Message/controller** | **Distribution** | **Parameter** | **Values** |
|---|---|---|---|
| *TAdvert* ($nc_t$) | Uniform | min $[ms]$ | 2.3 |
| | | max $[ms]$ | 7.2 |
| | | mean $[ms]$ | 4.80 |
| *LReady/LJoin* ($nc_l$) | Gamma | $\theta$ | 0.0132 |
| | | $\alpha$ | 2.127 |
| | | $x$ $[ms]$ | 28.0 |

### 7.5.1 *Design of PN-QN performance models*

In the modeling process, it is essential that the generated PN-QN models retain the most relevant attributes of the system in relation to the performance study. Therefore, only relevant aspects in each deployment should be considered. This can significantly reduce the simulation time from weeks to days if not hours. These considerations are discussed under the respective deployment models

### 7.5.2 *Modeling DRM*

The DRM model follows an exact description as shown in Figure 7.1 where three bridges, a Talker and three Listeners are considered. Figure 7.5 shows a PN-QN model developed for the DRM.

Once a token arrives at $p_1$ (ingress point of the network by Talker stream), it goes through the timed transitions $t_1$ to $br_1^T$. The timed transitions represent the $d_t$ as discussed in Equation (7.1) and (7.2). Here,

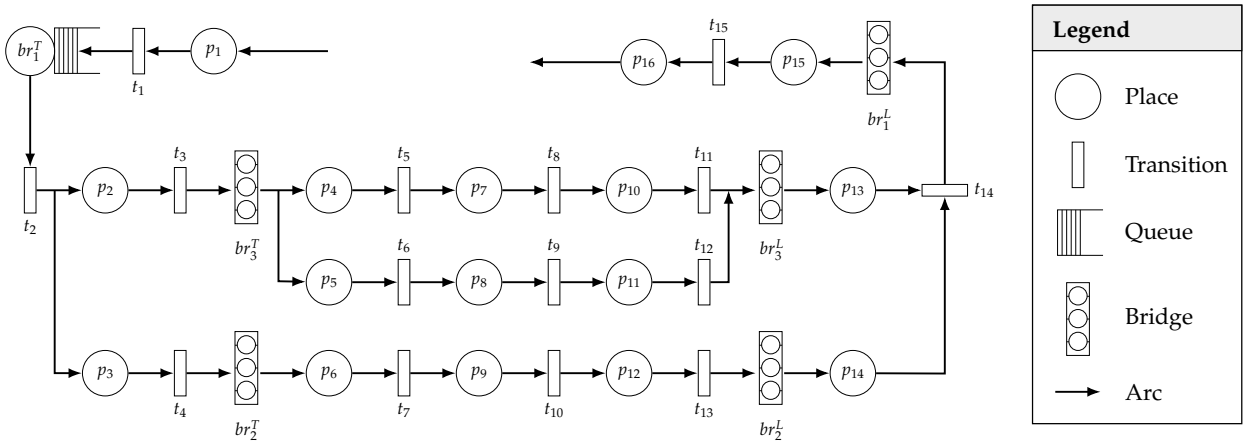



Figure 7.5: PN-QN model illustrating the DRM deployment for communication between Talker and three Listeners over TSN LAN as shown in Figure 7.1. Circles and rectangles represent places and transitions respectively.

$d_t$ is the ratio of MSRP packets (*TAdvert*=84 bytes, *LReady*=78 bytes) and the Line speed (100 Mbps). At $br_1^T$ the token (superscript $T \implies$ *TAdvert* packets are processed at this station) is duplicated and a copy of each is transmitted to $br_2^T$ and $br_3^T$ respectively for all Listener(s). This behavior of the model represents the operation of MSRP/MRP packet propagation (for multicast or broadcast) of *TAdvert* packets. A similar operation is performed at $br_3^T$ to increment the amount of tokens corresponding to the number of *TAdvert* packets received by Listener 2 & Listener 3.

Places $p_7, p_8, p_9$ represent the start point for the generation of Listener messages (*LReady/LJoin*). As explained in section 7.1 in the DRM deployment, only *LReady* messages are issued by the Listeners. Once a token of type ($T$) enters either of these places ($p_7, p_8, p_9$), it enables the firing of $L$ tokens belonging the respective Listeners ($l_1, l_2, l_3$). These tokens then transit through the network as shown in Figure 7.5 till each reaches $p_{16}$; where the process terminates (absorbing state). The superscript $T$ and $L$ at the bridges (*br*) show the type of tokens passing through each node in the system, which represent *TAdvert* and *LReady* packets respectively (customer classes).

In practice, each TSN bridge (*br*) has limited queue length, however, to study the stability of the systems in performance analysis such as these, one can theoretically assume an infinite queue size in order to estimate the system response time during the processing of large volumes of jobs within the system. Also, it is only relevant to concentrate on the queue of the bridge connected directly to the Talker ES ($br_1^T$). This is because all *TAdvert* requests are received as tokens by the network at $br_1^T$. A token dropped at $br_1^T$ does not propagate through the network hence does not influence other bridges in the model. So far as the rest of the bridges in the network have queues greater than

or equal to the edge bridge, packet drops can only happen on at the Talker edge bridge. This allows for simulating the effect of finite and infinite queuing scenarios on the system response time and system drop rate.

The model defines one customer class $T$ of type *TAdvert* even though practically as illustrated in Figure 7.1, there can be 4 different customer classes representing the Talker and Listeners (i. e., $TAdvert \rightarrow T$, $LReady \rightarrow L_{1,2,3}$ of respective Listeners). This simplification significantly reduces the model size as state-space generations are reduced using a single customer class instead of multiple classes. This subsequently reduces the simulation time because response time is computed for each customer class separately and would have to be summed up eventually to get the system response time. However, by varying the handling of tokens from a single customer class at different stations enables the simulation of the same effect as if additional three customer classes are defined for each Listener.

It must be emphasized that this simplification does not affect the result in any way. In fact, it only reduces the model size and overall simulation time because it reduces only the state-space generation.

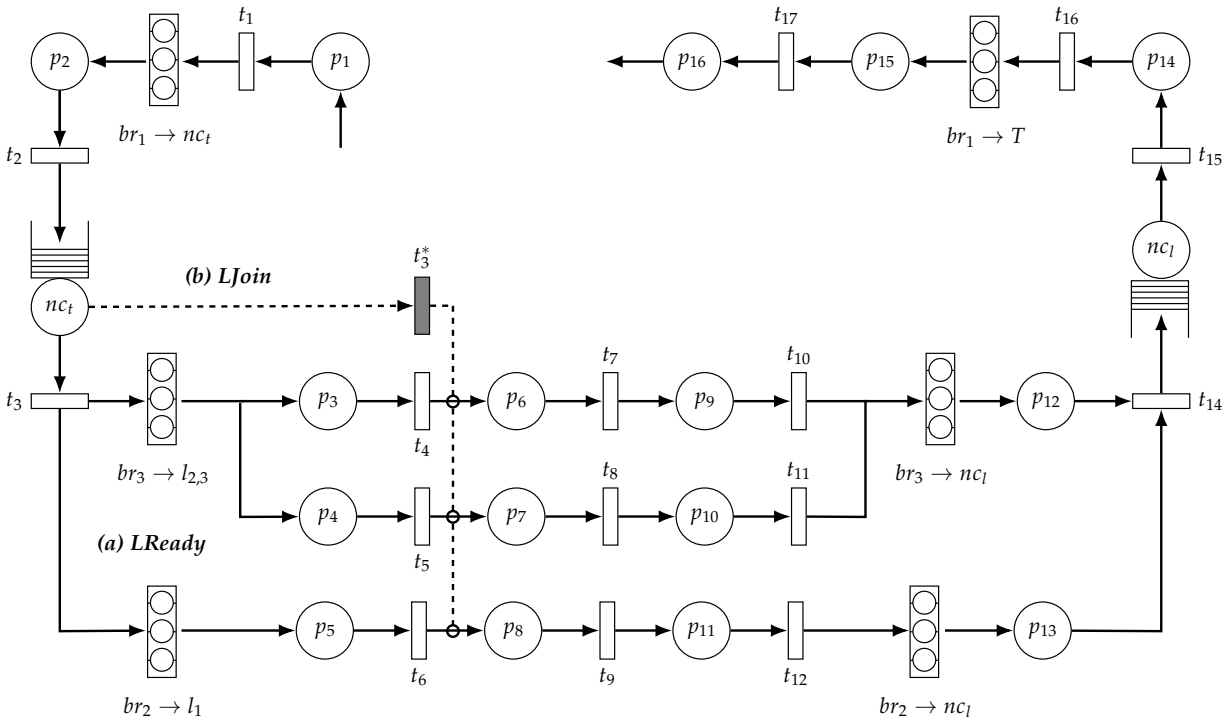



Figure 7.6: PN-QN model showing the two variants of the SDIN-HRM deployment model: ***(a) LReady***: the controller ($nc_t$) informs the Listener(s) about advertised Talker streams after resolving and registering *TAdvert* notifications, ***(b) LJoin***: Listener(s) actively send *Join* notifications to the controller ($nc_l$) as request to subscribe to Talker streams if advertised.

### 7.5.3 *Modeling SDIN-HRM*

Figure 7.6 shows the PN-QN models for the two variants of the HRM. The first, denoted as *(a) LReady*, describes a scenario where the controller ($nc_t$) informs Listeners about an advertised stream after resolving and registering a *TAdvert* packet from the Talker via $br_1$. This prompts the Listener(s) to send *LReady* notification if they desire to subscribe to the stream.

The second, denoted as *(b) LJoin*, describes a scenario where Listeners actively inform the controller ($nc_L$) of their wish to subscribe to a Talker stream if advertised.

Due to the complexity and similarities of the models at certain parts, both models are shown in Figure 7.6. The difference in the two models is highlighted at the regions where they occur (i. e. at $t_3 \rightarrow t_{6,7,8}$) in the case of ***(a)*** and $t_3^* \rightarrow t_{6,7,8}$ for ***(b)***.

In scenario ***(b) LJoin***, instead of the use of timed transition at $t_3$, an immediate transition from $t_3^*$ to the Listener start places ($p_6, p_7, p_8$) is used. This difference represents a scenario where Listener(s) are assumed to send *LJoin* requests just about the same time the SDIN controller has just finished registering an advertised Talker stream. Hence, there is no need to inform Listeners anyway. Due to difference in mechanisms (i. e., ***(a)*** and ***(b)***), it is expected that the response times for the two cases differ. This is evident from careful examination of Equation (7.2).

In the case of ***(b)***, the internal processing time at a Listener's ES is omitted, i. e., neglecting $\alpha_L$ by replacing all parts of the model from $t_3$ to $t_4, t_5, t_6$ (***(a) LReady***) with $t_3^*$ (***(b) LJoin***). This is because the Listeners are already trying to subscribe to a stream by sending *LJoin* packets whether the streams they want to subscribe to are advertised or not and thus, the processing time for generating *LReady* packets at the Listener ESs in (***(a) LReady***), is no longer necessary in (***(b) LJoin***). Therefore, although the model for both cases are shown in a single figure, the results of the performance analyses are presented separately to emphasize the effect of these subtle differences.

The two options are possible enhancements that add to the flexibility and usability of the SDIN-HRM as a deployment alternative. It also provides a similar reservation process for dynamic/plug-and-play stream reservation as provided by DRM. For timed transitions between SDIN-CNC and bridges, due to the huge influx of notifications to the controller, it has to be ensured that notifications for new *TAdvert* are not overly delayed at the bridges due to other notifications bound for the controller. Therefore, delay due to link congestion to SDIN controller must be minimized by using a high capacity link (e. g., 1Gbps or 10Gbps) to reduce congestion at the edge bridges to the controller.

## 7.6 EVALUATION OF DRM/SDIN-HRM PN-QN MODELS

In the following sections discuss the evaluation of the performance models.

### 7.6.1 *Evaluation Metrics*

A performance metric is a measurable quantity that captures precisely a value of interest. In this evaluation, the interesting thing is the time it takes each of the deployment models to complete the stream reservation process. In this regard, the system response time and the drop rate are essential metrics to consider. The description of the metrics are as follows:

- **System Response Time** ($R$) is defined as the time interval between the instant of submission of a user's request to a system and the instant the corresponding reply arrives at the user. Although, response time can be generically assessed for almost all types of systems, in computer networks, it is calculated as:

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

  where $d_s$ is the service delay of the request, $d_q$ is the queuing delay of the user's request, $d_p$ 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. It can be seen that the generic definition of response time follows a similar analogy as Equations.( 7.1) and (7.2) on close examination.

- **Drop Rate** ($D$) describes the rate at which packets arrive at a QS (see Definition 7.4.2) with a finite queue of size $z$ which is already full. The following equation calculates the probability of such events:

  $$D = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^z,$$

  where $\lambda$ is the arrival rate at the queuing system, $\mu$ its service rate. $\mu$ in this case is equal to $\alpha$ in Equations (7.1) and (7.2).

Mainly, the focus of the performance analysis is to determine the response time for job completion of the two plug-and-play deployment models (i. e., DRM and SDIN-HRM). However, it is also interesting to analyze aspects that influence the response time of the system such as:

- the effect of parallel processing of jobs, i. e., the number of parallel servers (processors) under various arrival rates ($\lambda$), measured in jobs/s.

- the drop rate and waiting time of jobs at stations of interest in the models (i. e. $nc_T$, $nc_L$, and $br_1^T$) considering finite and infinite queue size. Under finite queue size, it is interesting to see effects such as dropping jobs when they exceed the queue size compared to having job wait till they are serviced.

As explained in Section 7.4, it is interesting to know exactly the amount of different Talker streams that can be served by a TSN-based LAN and how these deployments influence the process.

### 7.6.2 *Evaluation Environment*

The test setup for evaluation of the deployment models is described. The evaluation is performed on a system with the following 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

Software-wise, the open source suite, Java Modelling Tools (JMT)[3], which consists of simulation and analytical tools for performance evaluation and modeling of computer and communication systems is used. The suite implements several state-of-the-art algorithms for Exact, Approximate, Asymptotic and Simulation algorithms for PN-QNs. The tool *JSIMgraph* from JMT is used since it is less restrictive with regard to the behavior of systems.

JMT is a discrete-event simulator which enables the analysis of models that combine the strength of PNs together with QNs. In addition, it allows for the deterministic routing of packets as well as the fine-granular mechanics of state-space methods. This makes it well suited for the performance analysis of this kind. Its simulation approach allows for tunable quality of results due to the repetition of execution of the models considering variable parameters and the derivation of relevant output measures [108].

Furthermore, the tool supports numerous distributions (e. g., Deterministic, Erlang, Exponential, Gamma, Normal, Pareto, Uniform, etc.) for characterizing service times and arrival rate ($\lambda$) as well as state-dependent and state-independent routing strategies.

3 Java Modeling Tools – Introduction, http://jmt.sourceforge.net/, last accessed: 05/03/2019

## 7.7 RESULTS ANALYSIS AND VALIDATION

The simulations are performed by varying parameters such as the queue size at the bridges and SDIN-CNC controller under different service conditions such as *Drop* or *Waiting-Queue;—Drop* refers to a service rule that allows the QS to drop packets once the queue size is exceeded,— *Waiting-Queue* refers to service rule where packets are never dropped regardless of whether the queue size is exceeded. This is equivalent to using an infinite queue and allows the measurement of the waiting time at a specified QS. A maximum queue size of 10 packets per TSN bridge is considered. This is because the physical TSN bridges used for the demonstrator test measurements has a maximum buffer size of 10.

In order to compare and validate DRM and SDIN-enabled HRM on the same set of conditions, both systems must perform a similar task such as stream reservation or time-schedule reservation. Since time-schedule reservation is not present in DRM but HRM, they are compared based on stream reservation. Also, it is important that the queue size of the SDIN controller in the HRM models is also set to 10 just as in the bridges. However, the *Waiting-Queue* service rule if enabled, allows the simulation of the system as if the QSs have infinite queue size. This also enables the observation of the system behavior under heavy traffic.

Also, given that Talkers and Listeners can request for reservation at any time, the arrival process of packet are random and independent of each participant and therefore, follows a Poisson arrival process where the inter-arrival times between packets are exponentially distributed. Therefore, an exponential distribution is used to characterize the inter-arrival time (with mean $= \frac{1}{\lambda}$), for stream advertisement from the Talker class. This is because the rate of growth of the queues in the system is proportional to number of stream requests that emanates from the Talker class.

The deployment models are evaluated for their response time under an exponential arrival of stream request considering a specified number of requests generated during the physical test measurements. The models are then further explored to examine the system behavior under varied traffic and server conditions in a what-if analysis which allows the system to analyzed by varying more than one parameter.

Figure 7.7 shows the cumulative distribution of response time for the respective models considering exponential arrival ($\lambda = 10$) under server size of 3 using a *Drop* service rule. The result shows that the SDIN-HRMs outperforms the DRM by about a factor of two, with response time of completed jobs distributed between 0.2 to 0.8 seconds whilst that of the DRM is between 0.8 to 1.3 seconds.

Figure 7.8a, 7.8b and 7.8c show the system-response time for the DRM and the two variants of the SDIN-HRM models ( *LReady* and

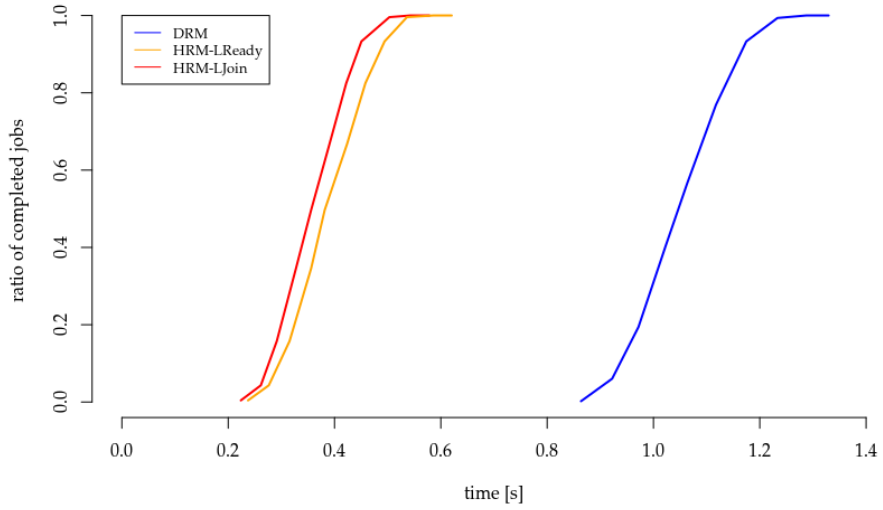


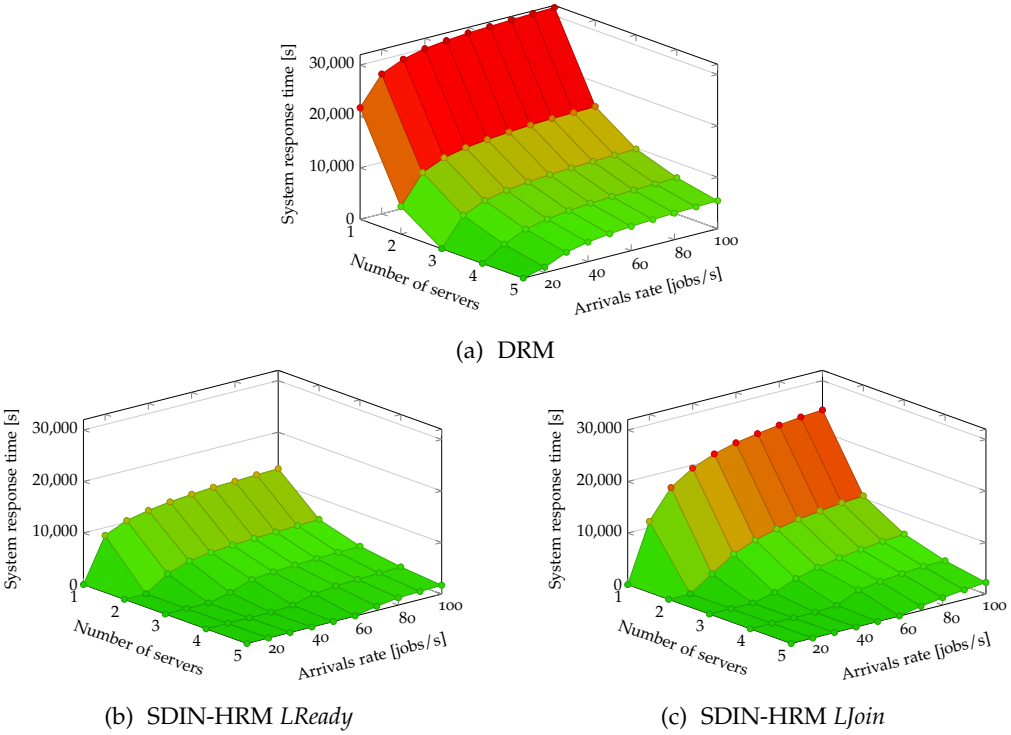

Figure 7.7: Cumulative distribution of system response, exp($\lambda$ =10), queue-size=10 and *Drop-Queue* service rule using 3 servers at 95% confidence.

*LJoin*) respectively under the *Waiting-Queue* service rule in a what-if-analysis. This result is also similar to Table 7.2, 7.4 and 7.3 using infinite queue size. This shows the consistency of the models and accuracy of the results given that Waiting-Queue service rule behaves the same way as using the infinite queues. It can be observed that at an arrival rate of 10 jobs per second, the system response time of the DRM lies between 3.13s and 1.01s (see Table 7.2) when the system uses 3 to 5 servers respectively. This result is validated when it is compared to the actual system measurement with TSN bridges with 3 processors.

Also, the results show a general trend of increased response time of the system with increasing arrivals and a decrease in response time with increasing bridge processors (servers).

In general, what the results show is that the response time of the distributed deployment model takes a very long time under higher traffic conditions if the number of employed servers are low. A similar trend is also observed with the SDIN-HRM deployment models, however, the response times for SDIN-HRM models are much lower than for DRM when the arrival process is between 10 to 30 jobs per second using 1 to 3 servers but achieves almost the same response time as DRM during higher arrival rates.

Comparing the two SDIN-HRM models, a slight increase is seen in the response-time of the *LReady* model than the *LJoin* model, yet, the difference is not significantly large as compared to the DRM. Figures 7.9a, 7.9b, and 7.9c show the response time under the *Queue-drop* service rule, where *TAdvert* packets are dropped if the queue sizes of the bridges and/or controller are exceeded.




(a) DRM

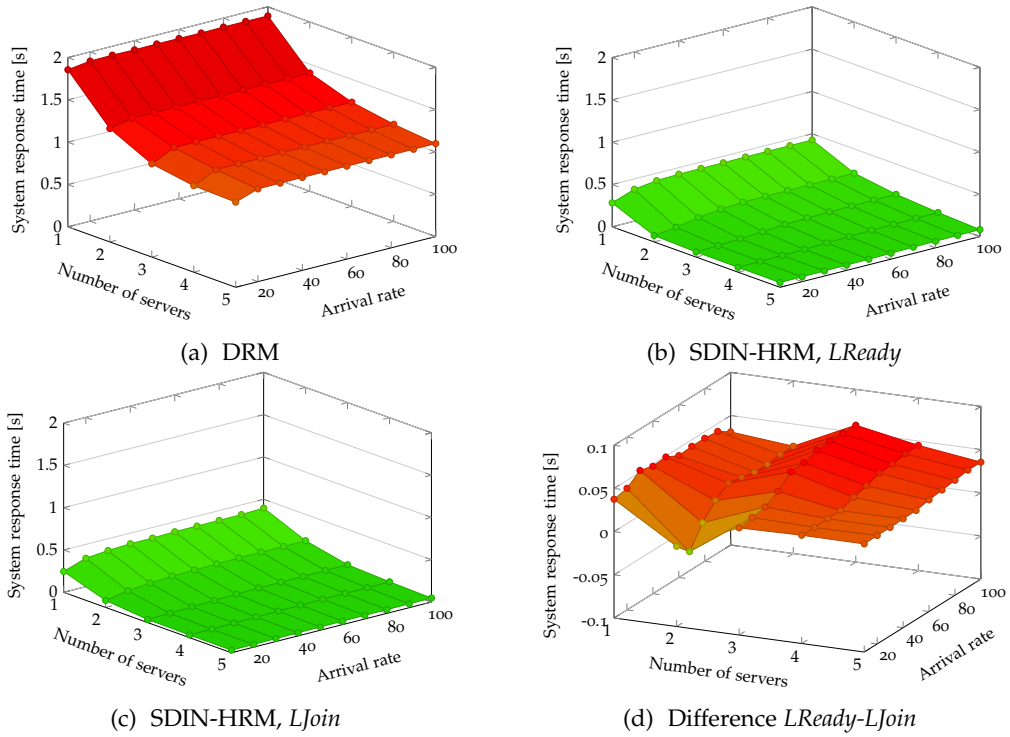
(b) SDIN-HRM *LReady*

(c) SDIN-HRM *LJoin*

Figure 7.8: System response time under increasing traffic arrival for DRM and SDIN-enabled HRM, exp($\lambda$=100), queue=10 and *waiting-Queue* service rule. Result computed at 95% confidence.

Considering Figure 7.9a, a general decrease in response time as the number of processors (servers) used within the bridges increases from 1 to 5. However, under each respective processor conditions, a relatively constant response time is observed. The response times at 1 to 3 servers are consistent with the measured values obtained from the test measurements from the demonstrator setup.

A similar trend persists in the case of the SDIN-HRM models, however, they are lower than in DRM. The difference between the two models is depicted in Figure 7.9d, which shows the difference in the response-time for *LReady* and *LJoin* models. It can be observed that the *LReady* model performs slightly better than the *LJoin*, with a lower response time under one to two servers. However, *LJoin* performs better when the number of servers increase from 2 to 5. This is expected as the internal response time of the Listener ESs are not considered anyway. In addition, the arrival of packets from all listeners at $nc_l$ can in fact overwhelm the QS, thereby, leading a high waiting time at the controller *LJoin* QS, hence requiring more servers to process them faster. In the case of *LReady*, packets from the listeners may arrive at different times due to listeners not responding to all talker advertisement. In general, a low response time of SDIN-HRM models is observed in comparison to DRM. This observation is further explained by examining the system drop rate.




(a) DRM

(b) SDIN-HRM, *LReady*

(c) SDIN-HRM, *LJoin*

(d) Difference *LReady-LJoin*

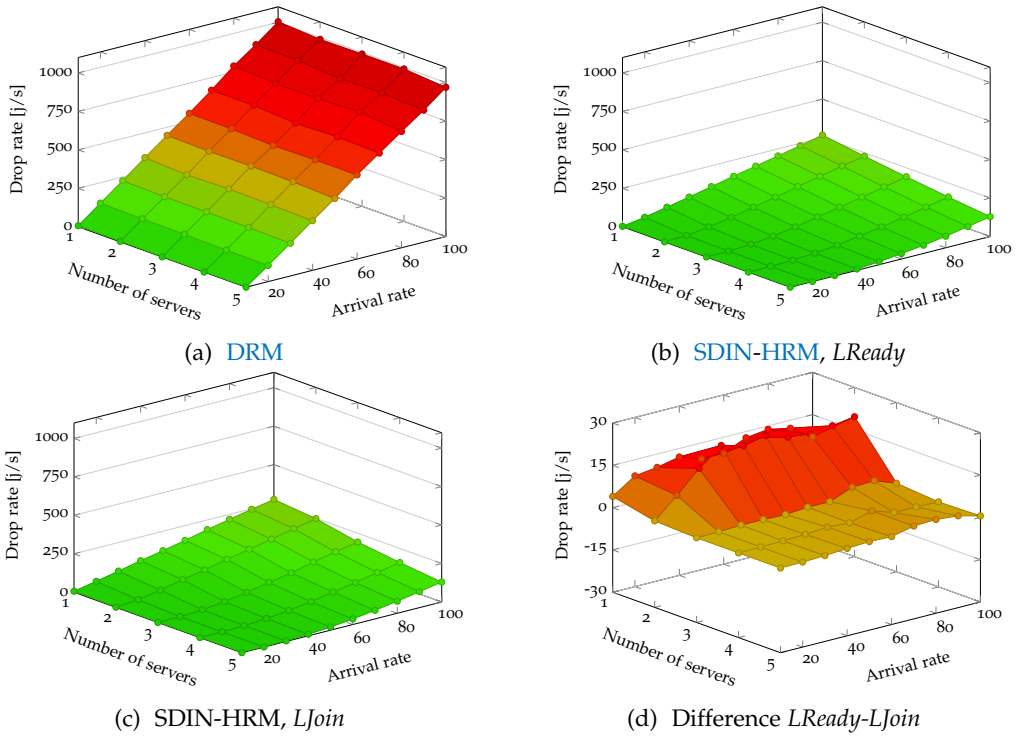
Figure 7.9: Response time analysis of DRM and SDIN-enabled HRM, exp($\lambda$=100), queue=10 and *Queue drop* service rule. Result computed at 95% confidence.

In the case of DRM, the drop test is done at $br_1^T$ because it is the ingress point to the system, therefore, once the queue size is exceeded and a packet is dropped, it does not affect the processes of the other bridges. In case of SDIN-HRM, the drop test is done at $nc_t$ since it also serves the same function as $br_1^T$ in the DRM.

The results displayed in Figure 7.10a show that the drop rate increases steeply with rising number of jobs per second. The same trend is observed with increasing number of processors with a slight decrease in the steepness. The possible explanation for this result is the general processing time of the DRM. Compared to the SDIN-HRM models shown in Figures 7.10b and 7.10c, the steepness reduces significantly with increasing number of processors. This is to be expected since the $nc_t$ processing time distribution is for a simple task of *streamID* lookup and registration, which completes much faster than in the case of DRM.

The difference between the two SDIN-HRM models is shown in Figure 7.10d where a general rise in drop rate is visible for the *LReady* model with 1 to 2 server(s). This further explains the response time differences shown in Figure 7.9d, because *LJoin* assumes that the stream is already registered; which means, it completes much faster than *LReady* with 2 servers.

Table 7.2, 7.3 and 7.4 show the response times of the different




(a) DRM (b) SDIN-HRM, *LReady*

(c) SDIN-HRM, *LJoin* (d) Difference *LReady-LJoin*

Figure 7.10: Drop rate under increasing traffic arrival, DRM and SDIN-enabled HRM, exp($\lambda$=100), queue=10 and *Queue drop* service rule. Result computed at 95% confidence.

Table 7.2: Response time of DRM stream reservation, infinite queue

| Arrival rate ($\lambda$) | System response time (s) | | | | |
|---|---|---|---|---|---|
| | Number of servers | | | | |
| | 1 | 2 | 3 | 4 | 5 |
| 10 | $2.17E^4$ | $5.36E^3$ | 1.38 | 1.03 | 1.01 |
| 20 | $2.72E^4$ | $1.09E^4$ | $5.41E^3$ | $2.69E^3$ | $1.05E^3$ |
| 30 | $2.90E^4$ | $1.27E^4$ | $7.25E^3$ | $4.51E^3$ | $2.88E^3$ |
| 40 | $2.99E^4$ | $1.36E^4$ | $8.15E^3$ | $5.98E^3$ | $4.34E^3$ |
| 50 | $3.05E^4$ | $1.41E^4$ | $8.70E^3$ | $5.98E^3$ | $4.34E^3$ |
| 60 | $3.09E^4$ | $1.45E^4$ | $9.07E^3$ | $6.34E^3$ | $4.71E^3$ |
| 70 | $3.11E^4$ | $1.48E^4$ | $9.33E^3$ | $6.61E^3$ | $4.97E^3$ |
| 80 | $3.13E^4$ | $1.50E^4$ | $9.52E^3$ | $6.80E^3$ | $5.17E^3$ |
| 90 | $3.15E^4$ | $1.51E^4$ | $9.68E^3$ | $6.95E^3$ | $5.32E^3$ |
| 100 | $3.16E^4$ | $1.53E^4$ | $9.80E^3$ | $7.08E^3$ | $5.44E^3$ |

models under infinite queue size. Under infinite queue size, there is no need for drop rate analysis as all request must be processed by the systems eventually. Therefore, the *Drop* service rule is not further investigated in this case.

Table 7.3: Response time for SDIN-HRM *LReady* stream reservation, infinite queue

| Arrival rate ($\lambda$) | System response time (s) | | | | |
|---|---|---|---|---|---|
| | Number of servers | | | | |
| | 1 | 2 | 3 | 4 | 5 |
| 10 | $3.32E^{0}$ | $7.83E^{-1}$ | $6.19E^{-1}$ | $5.75E^{-2}$ | $5.61^{-2}$ |
| 20 | $8.57E^{3}$ | $1.86E^{-1}$ | $9.34E^{-1}$ | $6.25E^{-2}$ | $5.80E^{-2}$ |
| 30 | $1.04E^{4}$ | $2.88E^{3}$ | $3.34E^{0}$ | $7.31E^{-2}$ | $6.25E^{-2}$ |
| 40 | $1.13E^{4}$ | $4.26E^{3}$ | $9.85E^{2}$ | $1.14E^{-1}$ | $7.15E^{-2}$ |
| 50 | $1.18E^{4}$ | $4.80E^{3}$ | $2.48E^{3}$ | $3.31E^{2}$ | $1.01E^{-1}$ |
| 60 | $1.22E^{4}$ | $5.17E^{3}$ | $2.83E^{3}$ | $1.45E^{3}$ | $3.40E^{1}$ |
| 70 | $1.24E^{4}$ | $5.45E^{3}$ | $3.10E^{3}$ | $1.94E^{3}$ | $8.43E^{2}$ |
| 80 | $1.26E^{4}$ | $5.62E^{3}$ | $3.29E^{3}$ | $2.14E^{3}$ | $1.40E^{3}$ |
| 90 | $1.28E^{4}$ | $5.79E^{4}$ | $3.45E^{3}$ | $2.27E^{3}$ | $1.58E^{3}$ |
| 100 | $1.29E^{4}$ | $5.90E^{4}$ | $3.57E^{3}$ | $2.40E^{3}$ | $1.71E^{3}$ |

Table 7.4: Response time for SDIN-HRM *LJoin* stream reservation, infinite queue

| Arrival rate ($\lambda$) | System response time (s) | | | | |
|---|---|---|---|---|---|
| | Number of servers | | | | |
| | 1 | 2 | 3 | 4 | 5 |
| 10 | $3.11E^{0}$ | $7.29E^{-1}$ | $5.61E^{-1}$ | $3.25E^{-2}$ | $3.0E^{-2}$ |
| 20 | $1.12E^{4}$ | $1.61E^{0}$ | $8.14E^{-1}$ | $3.73E^{-2}$ | $3.29E^{-2}$ |
| 30 | $1.67E^{4}$ | $2.89E^{3}$ | $2.32E^{0}$ | $4.85E^{-2}$ | $3.73E^{-2}$ |
| 40 | $1.95E^{4}$ | $5.58E^{3}$ | $9.83E^{2}$ | $9.09E^{-2}$ | $4.69E^{-2}$ |
| 50 | $2.11E^{4}$ | $7.25E^{3}$ | $2.64E^{3}$ | $3.49E^{2}$ | $7.76E^{-2}$ |
| 60 | $2.22E^{4}$ | $8.35E^{3}$ | $3.73E^{3}$ | $1.45E^{3}$ | $2.53E^{1}$ |
| 70 | $2.30E^{4}$ | $9.13E^{3}$ | $4.53E^{3}$ | $2.19E^{3}$ | $8.27E^{2}$ |
| 80 | $2.35E^{4}$ | $9.71E^{3}$ | $5.13E^{3}$ | $2.79E^{3}$ | $1.42E^{3}$ |
| 90 | $2.40E^{4}$ | $1.02E^{4}$ | $5.57E^{3}$ | $3.25E^{3}$ | $1.87E^{3}$ |
| 100 | $2.44E^{4}$ | $1.05E^{4}$ | $5.92E^{3}$ | $3.62E^{3}$ | $2.24E^{3}$ |

Also, it can be observed from Table 7.2 that the response time for stream setup grows very long under $1 \rightarrow 2$ servers even for jobs arrivals under 20 jobs/s, which is also the case considering the SDIN-HRM-*LReady* model, however, the *LReady* still remains lower than the DRM when Table 7.3 is examined. Generally, both models show a much reduced response time with increasing number of servers. Finally, Table 7.4 shows the response time for SDIN-HRM-*LJoin* model under infinite queue size. It can be seen that, it outperforms the DRM and SDIN-HRM-*LReady* models.

### 7.7.1 *Result validation*

For the validation process, measurements taken from the prototypical demonstrator setup are compared to result from the PN-QN performance results. To do so, the Mean Squared Error (MSE) metric is used. This is important because service distribution used in the models were derived from goodness-fit of the processing time of the prototype demonstrators. Therefore, it is important to assert the degree to which the approximations influence the models' results by examining the deviation from the measured data. An MSE gives a good representation of the variation. The MSE values are shown in Table 7.5.

Overall, the MSEs show that the prototype system and model match very well, having an MSE of $< 0.015$.

In particular, it can be observed that with the SDIN-HRM deployment model, the *LReady* model shows a higher MSE than the *LJoin*. This is expected because in the actual deployment Listeners may or may not immediately respond to *TAdvert* messages received from the SDIN-CNC. In the model, it is assumed that a Listener immediately replies to any received *TAdvert* message. This can lead to slight variations in the response time compared to the actual setup.

Table 7.5: Mean Squared Error of prototype system and performance model results for HRMs and DRM deployment

| Test # | HRM-*LJoin* | HRM-*LReady* | DRM |
|---|---|---|---|
| 1 | 0.0007 | 0.0004 | 0.0055 |
| 2 | 0.0149 | 0.0210 | 0.0006 |
| 3 | 0.0067 | 0.0027 | 0.0023 |
| 4 | 0.0014 | 0.0182 | 0.0122 |
| 5 | 0.0125 | 0.0007 | 0.0069 |
| 6 | 0.0016 | 0.0009 | 0.0009 |
| 7 | 0.0166 | 0.0044 | 0.0138 |
| 8 | 0.0015 | 0.0004 | 0.0019 |
| 9 | 0.0133 | 0.0079 | 0.0000 |
| 10 | 0.0091 | 0.0916 | 0.0079 |
| **MSE** | **0.0078** | **0.0148** | **0.0052** |

## 7.8 SUMMARY OF CHAPTER AND BIBLIOGRAPHIC COMMENTS

### 7.8.1 *Bibliographic Comments*

The mechanism for resource reservation varies widely mostly due to the application of advanced queuing and transmission techniques applied on the forwarding plane [104]. These techniques significantly impact the choice of control plane mechanisms and for that matter TSN infrastructural deployment in industrial environments, e. g., process or factory automation.

In this regard, Ehrlich et al. [111] and Herlich et al. [126] have examined and analyzed the architectural similarities of TSN deployment user models and proposed the integration of SDN concepts in TSN to enhance the evolvability of industrial networks. Their work suggest the combination of CNC and CUC as SDN controllers [111] to enable integration with other network systems. However, their work does not describe nor provide any recommendation or implementation as examined in this thesis.

Chen [104] and Nasrallah [107] et al. highlight the disadvantage w.r.t the reliability and availability of centralized controllers when needed, as well as capital and operation overheads w.r.t CRM or generic HRM. This is because their architectural analysis identifies the centralized deployment of controllers (CNC) as an Single Point of Failure (SPoF), a huge attack surface [112, 127], as well as it being relatively capital intensive (CAPEX/OPEX). Their opinion expresses how centralized control may be superfluous to or add unnecessary complexity to a small-scale network [104, 107], hence, increases CAPEX/OPEX.

Also, Nayak et al. [99] proposed an enhancement of TSN with SDN capabilities to enable an incremental flow scheduling and routing in time-sensitive SDNs. Though their work does not use the TSN compliant forwarding mechanisms (i. e., the TAS), it sets the tone for examining online-scheduling for scheduled TT using TAS.

In [85], our work on time-scheduling for TAS considers a similar approach and propose a scheduling computation algorithm highlighting its sufficiency for all deployment models (i. e., for offline and on-line TT schedule computation). This was further examined in Chapter 5 and 6 of this thesis.

To address consistency and SPoF challenges of centralized network control, two approaches are pursued in literature;— controller placement [114–116],— distributed controller clusters. However, in the industrial environment, challenges resulting from the distance of network controller(s) from the network are not of significant concern. This is because in process or factory automation networks, the network controllers are expected to be within the same shop-floor.

For the distributed controller clusters, the general idea behind scalability and reliability of SDN control plane is enabled by decentralized

replication of controller logic or functions to different physical or virtualized systems. In this context, Sakic et al. [108–110] provides an analysis of SDN-based control plane fault tolerance and consensus algorithms for distributed control plane.

In [110], the authors investigate scalability issues regarding the interplay between progressing state of distributed controllers and the consistency of the configuration of external forwarding device in the control plane under Byzantine faults. In [109], an empirical study of the impact of adaptive consistency of distributed SDN control plane is examined with the aid of an experiment consisting of five extended OpenDaylight controller instances and two network topologies within data-centers.

Similarly, Muqaddas et al. [128] investigated the load overhead of intra-cluster communication using multiple controllers in an Open Network Operating System (ONOS) cluster. These analyses show that SDN-enabled centralized control can be very scalable and reliable albeit concerns raised in [104, 107, 111].

In the context of SDN, notable work regarding performance analysis is done in [108], where the authors evaluated the performance of SDN cluster organization w.r.t the response time and availability of consensus algorithms using the Mobius framework, which is based on Stochastic Activity Nets (SANs) [129–131]. The authors analyze the response time of controller clusters in the face of adversities. SANs are not used in this thesis because the system under study requires deterministic dispatch of packets and therefore, deterministic PNs are better suited.

Furthermore, model-checking, classical system theory and discrete-time state analysis [132], and formalism such as PN and QN [133–135] have been used to evaluate the performance of network protocols and networks as a whole. A similar approach is followed by Sood et al. [136], who introduce an analytical model to study the performance of SDN switches based on an M/Geo/1 queuing model. The analytical model is validated using extensive simulations, where the influence on packet arrival rates, number of flow-table entries, and position of the target rule by corresponding packets are evaluated.

Gelberger et al. [137] also analyze and compare the performance of two SDN protocol architectures, namely OpenFlow and ProGFE, as a function of their complexity, flexibility, and potential capabilities. The authors conclude that SDN flexibility comes at the price of raw performance with an additional overhead for complex functionalities.

Araniti et al. [138] investigate the performance of OpenFlow over wireless networks and the enhancements introduced by the usage of SDN concepts. Using OMNeT++ [139] simulation, the obtained results show that SDN architecture and OpenFlow yield benefits regarding e2e delay, throughput, and jitter. However, analysis in this thesis has shown that e2e latency guarantees are dependent on the DP and the

efficient computational models derived from SDN-enablers.

Contrastingly, this thesis combines two methodologies to achieve validatable results;— first, a simulation approach based on PNs and QN models is used to gain quantitative performance results for TSN deployment scenarios;— second, measurements from actual DRM and SDIN-HRM installations were used to validate the results.

### 7.8.2 *Summary of chapter*

In this chapter, an SDN-enabled Hybrid Reservation Model (HRM) for control and management of TSN infrastructures is proposed as a complementary and/or alternative deployment solution that allows industrial network operators to harness the full potential of TSN.

The novel Software-Defined Industrial Networking (SDIN)-HRM leverages existing Multiple Stream Registration/Reservation Protocol (MSRP) mechanisms such as Multiple Stream Registration/Reservation Protocol (MSRP), Multiple MAC Registration Protocol (MMRP) and Multiple VLAN Registration Protocol (MVRP), and SDN principles to provide dynamic resource reservation coordinated by a centralized SDIN controller.

A set of requirements are formulated which specify how existing MSRP packet combined with User Network Interface (UNI) data structure can be extended to enable a flexible exchange of "scheduled-traffic" information between End-Station (ES) and SDIN controller and thus, allowing an online centralized computation of Time-Triggered (TT) schedules which could otherwise not be achieved in the Decentralized Reservation Model (DRM).

Aside from the illustrated qualitative benefits and flexibility of SDIN for TSN deployment, the chapter provides quantitative performance analysis of the proposed system compared with existing deployment models. Leveraging the expressiveness of deterministic Petri Net (PN) and Queuing Network (QN), performance models for the respective DRM and SDIN-HRM are developed to characterize the intricate operations of each deployment. The PN-QN models are analyzed via simulations, comparing deployment models based on their responsiveness under varied traffic conditions.

The results of the performance analysis indicate that in addition to the flexibility of SDIN-HRM to integrate new requirements, the obtained response times for plug-and-play stream reservation are lower than those of the decentralized model.

# 8 CONCLUSIONS

Network programmability is revolutionizing how networks are constructed and operated. Leading concepts such as Software-Defined Networking (SDN) have already been adopted for systems in telecommunication and enterprise networks. However, there is a significant hesitation for its adoption in industrial networks due to misconceptions and lack of clarity of the specifications that cover the needs of industrial operators.

This thesis shows novel ways to achieve practical deployments by leveraging Software-Defined Networking (SDN) for the control and management of next-generation industrial networks based on Time-Sensitive Networking (TSN).

## 8.1 SUMMARY OF THESIS

To better understand the building principles of SDN in the industrial context, this thesis starts by examining SDN with other programmable network concepts such as Forwarding and Control Element Separation (ForCES) from the perspective of industrial stakeholders. Notably, concerns such as industrial operators' ability to deploy solutions that can manage applications with contrasting requirements on a common network infrastructure, as well as provide the requisite Quality of Service (QoS) whilst ensuring continuity by accommodating new services, are among several challenges identified and examined in Chapter 2. With the analysis of SDN concepts, this thesis identifies and examines relevant principles enabling the concept of Software-Defined Industrial Networking (SDIN) to allow industrial operators address their concerns and resulting challenges.

With a better understanding of SDN concepts and what they can provide to aid in addressing the challenges of industrial networks, Chapter 3 identifies and examines virtualization technologies and concepts that can be leveraged for the realization of an SDN-enabled industrial infrastructure. Virtualization concepts such as network slicing and how it can be leveraged to accommodate characteristically different QoS requirements on the same network is analyzed and discussed. Paramount in the analysis is how Network Slice (NS) can be realized in production-grade SDIN controllers by employing solutions to the Virtual Network Embedding (VNE) problem.

In this regard, the underlying principles of the VNE problem were examined to reveal relevant aspects of solutions to the problem that can be optimized for flexible resource allocation and QoS provision-

ing within SDN. This includes the role of VNE algorithms in flexible service automation as well as considerations for the VNE problem formulation, and design of VNE algorithms in the context of SDN.

Based on the insights gained from VNE analysis in relation to three principles of SDN namely— Service creation, Abstraction and virtualization, and the Concept of shared resource; two distinct objectives are proposed for the VNE formulation within SDN;—Objective 1 discusses how the underlying Substrate Network (SN) and the characteristics of applications can be virtualized to achieve QoS-aware resource allocation,— Objective 2 examined how VNE algorithms must be designed for efficient resource orchestration within an SDN controller.

Based on objectives defined in Chapter 3, Chapter 4 applies a Trajectory Approach (TA) to derive computational models for delay constraint verification on next-generation TSN-enabled SN, which provides varied traffic shaping features for QoS guarantees. The traffic shaping features are analyzed and delay computation functions derived for the respective shapers or schedulers based on a novel Queue-Scheduler-Link (QSL) model considering the effect of individual operations as well as the tandem composition within a TSN-enabled Substrate Network (SN).

Subsequently, Chapter 5 proposes and develops novel online VNE algorithms to address Objective 2 which covers efficient resource orchestration within SDN. The novel Demand Based virtual Link Embedding Algorithm (DBvLEA) which integrate several features such as QoS-aware mapping, topology-aware and schedule-aware mapping are designed and developed considering proposed guidelines discussed in Chapter 3.

In Chapter 6, the proposed algorithms are evaluated to assert the underlying hypothesis of the design. This includes comparison to state-of-the-art VLiM algorithms.

Finally, to showcase the flexibility and benefits of an SDN-enabled industrial network, an SDIN-enabled Hybrid Reservation Model (HRM) is proposed in Chapter 7 as a deployment solution that allows industrial operators to utilize the full potential of TSN. Subsequently, a qualitative and quantitative performance analysis showcasing the merits of the proposed deployments are discussed.

## 8.2 LESSONS LEARNED AND IMPACT

This thesis started by examining the concerns and requirements of industrial stakeholders and their hesitations towards adopting programmable networking concepts such as SDN. In this regard, the thesis set forth to establish clarity on the subject by examining the underlying principles of programmable networks, then proposing and developing solutions that can be leveraged to address their concerns.

The following are the lessons learned and the impact of the findings of this thesis.

### 8.2.1 *Analysis of programmable networks*

The analysis of network programmability based on the principle of control and forwarding functions separation presented in Chapter 2 revealed two exciting facts;— the concept of SDN adapts application requirements (environment) to a network (organism), which is contrary to evolution from a biological point of view. This implies SDN abstracts the requirements of applications and a network to assert compatibility rather than changing the capability of the network to suit requirements of its environment. However, the ability to fulfill the requirements of applications is shown in Chapters 3 and 4 to depend solely on the features of the underlying network.

Secondly, the use of Abstraction and virtualization concepts (e. g., Concept of shared resource) and the possibility to augment network functions within the centralized controller allow for the realization of application specific requirements on a network through the use of computational and communication models. This enables on-demand service creation via automated processes within the SDIN controller. These results therefore provide a foundation and focus-points upon which solutions can be curated to address specific requirements of applications.

Finally, the knowledge of network evolvability obtained from the analysis of SDN principles gives researchers and engineers a clear understanding of how to map requirements of legacy applications on SDN-enabled infrastructures and thus, allows for control and management of heterogeneous industrial systems. This, in effect, makes SDN-enabled infrastructures backward compatible and future-proof.

### 8.2.2 *QoS provisioning and the VNE problem*

Following from the knowledge gained on SDN concept of programmability in Chapter 2, the examination of the VNE problem in the context of SDN provides considerable insights into how VNE should be applied in production-grade SDN-enabled networks.

The result of the VNE analysis in Chapter 3 can be used as a guideline for both researchers and engineers to develop algorithms and computational models that can be applied in any network where precise QoS and efficient resource usage are required.

Furthermore, the separation of the VNE problem into two distinct focus areas provides essential insights into the relevant aspects of the VNE problem in terms of what to consider and what goes into making VNE algorithms QoS-ware. Based on these guidelines, the proposed delay models on TSN-enabled infrastructures show how

existing methodologies for exact worst-case analysis of switched Ethernet networks can be used in VLiM algorithms for delay guarantees within SDN.

Also, the novel resource allocation algorithms proposed and evaluated in Chapters 5 and 6 respectively, provide resource allocation functions that enable application-specific service customization. This includes the use of a novel approach to distributed online time-schedule computation algorithm coordinated by a centralized controller.

To the best of our knowledge, the proposed algorithm is the first to examine online time-aware scheduling on a TSN-enabled infrastructure. The impact of these results implies that industrial operators no longer have to choose one solution or the other but can utilize all queuing and shaping mechanisms specified in the TSN standard on a common infrastructure controlled and managed by SDIN controllers. Lastly, demand-based resource orchestration shows a significant advantage over the greedy-based shortest path first approach for a multi-constrained resource orchestration in SDIN. It also contributes to a lightweight function which enables SDIN controllers to be deployed on low capacity compute systems.

### 8.2.3 *Flexible deployment of TSN infrastructure*

After examining the role of virtualization and algorithmic functions within SDN, Chapter 7 demonstrated the flexibility of SDIN-enabled TSN. The demonstration illustrated among other things, how communication models such as packet headers can interact with computational functions within an SDIN controller to enable dynamic service provisioning on demand.

Moreover, results of the qualitative performance analysis attest to the flexibility of an SDN-enabled TSN. This is further backed by quantitative performance analysis which shows the proposed SDIN-enabled HRM can perform just as good as DRM if not better.

The impact of the results illustrated in Chapter 7 shows the practical application of novel algorithms and concepts proposed in this thesis.

Furthermore, the open service interfaces and resources exposed by the SDIN controller enable integration with brown- and green-field technologies such as Field-bus and 5G systems as well as inter-domain communications across multiple TSN domains [11].

## 8.3 OUTLOOK

The approach and concepts proposed in this thesis are not exhaustive solutions to challenges within industrial networks, there is still the need for further research as some concepts proposed in this thesis rely on assumptions such as perfect synchronization of TSN bridges and ESs as well as an always available SDIN controller.

Contrary to real-world deployments, systems fail, and whenever such failures occur, they can lead to financial losses or injuries. This therefore requires a critical examination of SDIN controller deployments in production networks in case of failures.

Furthermore, efficiently controlled distributed processes are those coordinated by a centralized entity [140]. Though this assertion is true from classical distributed systems theory in terms of flexibility and performance, there are merits to the Single Point of Failure (SPoF) problem attributed to centralized control of distributed systems. For example, a compromised SDIN controller gives an attacker unlimited control of communication within production systems. This, therefore, implies the security in SDIN deployment is as important as the performance aspect of the system and must be further investigated.

In this regard, the prevention of man-in-the-middle attacks targeted at obtaining login credentials of operators in order to gain unauthorized access to the SDIN controller must be investigated. Finally, strategies to guide against Distributed Denial of Service (DDoS) attacks aimed at making the SDIN controllers unavailable to in order stall production must also be investigated.

# APPENDIX

## A.1 FORCES REQUIREMENTS

The following provide a summary of requirements stipulated by the ForCES architectural framework for design of protocols between the CE and FE termed Fp, as well as models of the FE and CE [12, 141].

### A.1.1 *Requirement of Fp*

- Must enable CE to discover the capabilities of FE
- Must support secure communication
- Must be capable of supporting 10s of 1000s FEs and ports
- Must support reliable payload transport
- Must support packet redirect and mirroring
- Must support FE topology discovery
- Must support dynamic association of CE and FEs
- Must be able to group an ordered set of commands to FEs
- Must support asynchronous notification of FE events to CE
- Must enable CE to perform statistical queries on FE
- Must support CE redundancy or CE fail-over
- Must support multi-hop communication between CE and FE if they are on the different layers of the OSI model using RFC2914 compliant L4 protocols e.g. Transport Connection Protocol (TCP), Secure Communication/Transport Connection Protocol (SCTCP)

### A.1.2 *Requirement of FE and FE models*

- FE models must express the kind of logical functions that can be applied to packets as they pass through an FE.
- FE models must support variations in the way logical functions are implemented on an FE.
- FE models must be capable of describing the order in which these logical functions are applied in an FE.

- FE must support a flexible infrastructure in which new logical functions such as classification, actions, and parameterization data can be easily added.
- FE model must be capable of describing the types of statistics gathered by each logical function.
- FE must support minimal set of functions however, the number of function is unrestricted.
- FE must be capable of expressing the number of ports on the device, the static attributes of each port such as speed, address, and state.
- FE models must be capable of expressing the data that can be used by the forwarding function to make a forwarding decision.
- FE must be capable of expressing its QoS capabilities in terms of, e.g., metering, policing, shaping/scheduling, and queuing functions and how they can be used to provide IntServ and DiffServ
- FE models should be extensible to adapt to new and currently unknown functionality.

## A.2 TSN STANDARDS AND KEY FEATURES

See [142] for further descriptions.

| IEEE TSN standards | Key features |
|---|---|
| 802.1Qbv | Enhancement for scheduled traffic (TAS) |
| 802.1Qcc | Enhancement for stream reservation |
| 802.1Qca | Path control and reservation |
| 802.1Qci | Ingress filtering and policing |
| 802.1Qch | Cyclic queuing and forwarding |
| 802.1Qcr | Asynchronous traffic shaping |
| 802.1Qbu | Frame preemption |
| 802.1CB | Stream identification and seamless redundancy |
| 802.1AS | Clock/time synchronization |
| 802.Qcp, CBcv, cw | YANG models for bridges, CB, and Qbv, Qbu, Qci |

## A.3 SUBSTRATE NETWORK AND SLICE REQUEST MODELS

Listing A.1: SN Model With Integrated QSL

```
{
"network": {
        "resources": {
       "nodes": [{
           "id": " ",
           "attributes": {"relaibility": 0.99,
           "port": [{
           "id": "ce:","rate": 100000000, "tx": "on/off"
              ,"rx": "on/off"}
                  ]}
                }],
       "links": [{
           "id": "src/dst",
           "attributes": {
           "propagationDelay": "0.00000010",
           "schedulers": [{
               "name": "TAS",
                "queues": [{
                    "id": "0", "maxBuffer": 100000,
                    "reservedBuffer": 0,"delay": 0,
                    "occupants": []}]},
                    {"name": "CBS",
                     "queues": [{"id": "1",
                     "maximumBufferSize": 100000,
                     "usedBuffer": 0, "delay": 0,
                     "occupants": []}]}],
             "bandwidth": {"max": "port.rate","available
                ": "max-used","used": 0}},
              "src": {"srcNodeId": "port.tx","srcPort":
                 "port.tx"},
              "dst": {"dstNodeId": "port.rx", "dstPort":
                  "port.rx"}}]}}}
```

Listing A.2: NS Request Model

```
{
"NSrequest": {
     "sliceId": "rtslice",
      "realtimeCapable": true,
      "cyclic": "true",
      "baseCycleTime": "125",
      "delayDemand":"10000",
```

```
8       "sliceEndPoints": [ "mac/ip:1", "mac/ip:2","mac/ip
           :3"],
9       "connectivityMatrix": [{
10           "src-node-id": "mac/ip:1",
11               "dst-node-id": [{
12                   "value": "mac/ip:2", "maxBurst": "10"
                        },
13                   { "value": "mac/ip:3","maxBurst": "10
                        "}]},
14                    {
15              "src-node-id": "mac/ip:2","dst-node-id": [{"
                   value": "mac/ip:1",
16                   "maxBurst": "10"}]}]}}
```

## A.4 MSRP AND SDIN-HRM DATA MODELS

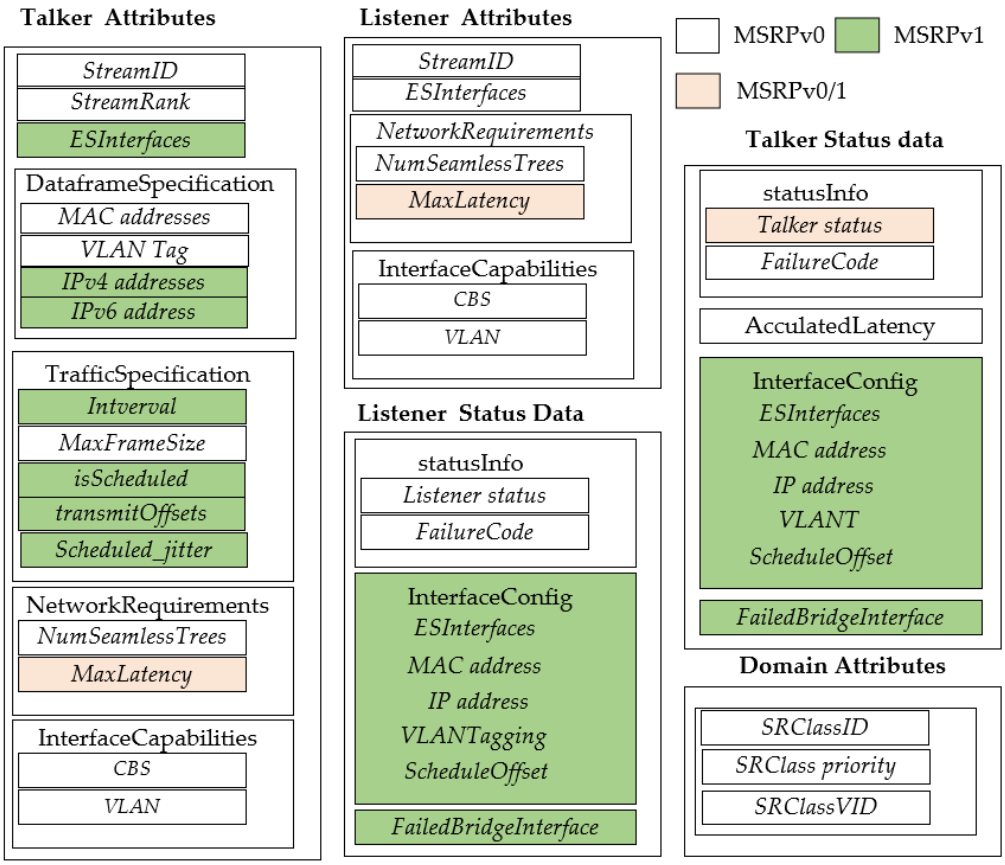



Figure A.1: MSRP attributes

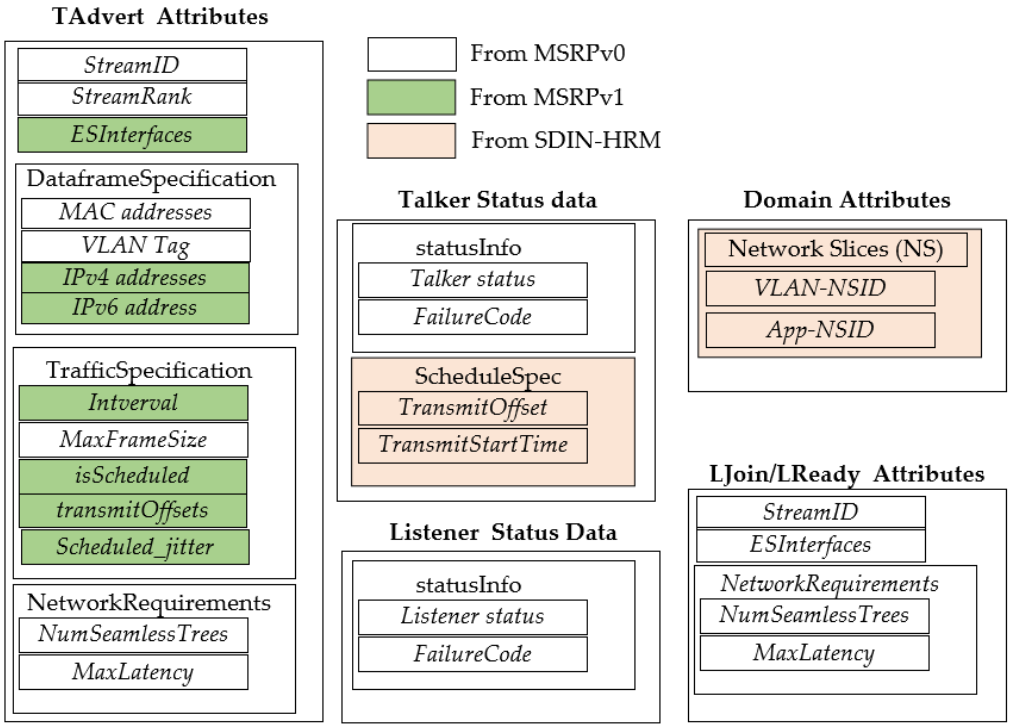



Figure A.2: Proposed attributes for SDIN-enabled HRM

# BIBLIOGRAPHY

## PUBLICATIONS BY THE AUTHOR


[11] Frimpong Ansah et al. "Controller of Controllers Architecture for Management of Heterogeneous Industrial Networks." In: *16th IEEE International Conference on Factory Communication Systems* 14.12 (2020,) (cit. on pp. 16, 28, 148).

[34] Riccardo Guerzoni, Ishan Vaishnavi, Frimpong Ansah, and Riccardo Trivisonno. "Virtual Link Mapping for delay critical services in SDN-enabled 5G networks." In: *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. 2015, pp. 1–9. DOI: 10.1109/NETSOFT.2015.7116163 (cit. on pp. 37, 40, 41, 44, 70–73, 96, 98–100, 102, 103).

[39] Riccardo Trivisonno, Riccardo Guerzoni, Ishan Vaishnavi, and Frimpong Ansah. "Network Resource Management and QoS in SDN-Enabled 5G Systems." In: *2015 IEEE Global Communications Conference (GLOBECOM)* (2015), pp. 1–7. DOI: 10.1109/GLOCOM.2015.7417376 (cit. on pp. 40, 41, 44, 49, 70–73, 93, 96, 98, 103).

[42] Frimpong Ansah, Juergen Rottmeier, Zirkler Andreas, and Hermann de Meer. "Worst-case delay slicing for Time-sensitive Applications in Softwarized Industrial Networks." In: *2020 25th IEEE International Conference on Emerging Technologies and Factory (ETFA)* 1.12 (2020), pp. 1652–1659. DOI: 10.1109/ETFA46521.2020.9211965 (cit. on pp. 41, 44).

[48] Frimpong Ansah and Hermann de Meer. "DBvLEA: A Demand-Based Approach to Virtual Link Mapping for Multi-Service Industrial Applications." In: *2019 15th International Conference on Network and Service Management (CNSM)*. 2019, pp. 1–8 (cit. on pp. 44, 69).

[64] Frimpong Ansah, Mainak Majumder, Hermann de Meer, and Jasperneite Juergen. "Network Slicing : An Industry Perspective." In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019, pp. 1367–1370. DOI: 10.1109/ETFA.2019.8869073 (cit. on p. 48).

[85] Frimpong Ansah, Mohamed Amine Abid, and Hermann de Meer. "Schedulability Analysis and GCL Computation for Time-Sensitive Networks." In: *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. 2019 (cit. on pp. 69, 77, 93, 141).

[91] Frimpong Ansah, Santiago Soler Perez Olaya, Hermann de Meer, and Martin Wollschlaeger. "Application Topology-Aware Virtual Network Mapping and Service Provisioning in Programmable Networks." In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019, pp. 1313–1316. DOI: 10.1109/ETFA.2019.8869434 (cit. on p. 76).

[105] Frimpong Ansah, Michael Niedermeier, and Hermann de Meer. "Performance Analysis of Software-defined TSN Control Plane." In: *IEEE Transactions on Network and Service Management* 14.12 (2019,(peer-review)) (cit. on p. 113).

[111] Marco Ehrlich, Dennis Krummacker, Christoph Fischer, Rene Guillaume, Santiago Soler Perez Olaya, Frimpong Ansah, Hermann de Meer, Martin Wollschlaeger, Hans D. Schotten, and Juergen Jasperneite. "Software-Defined Networking As an Enabler for Future Industrial Network Management." In: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2018. DOI: 10.1109/ETFA.2018.8502561 (cit. on pp. 122, 141, 142).

## REFFERENCES

[1] Brendan Galloway and Gerhard P. Hancke. "Introduction to Industrial Control Networks." In: *IEEE Communications Surveys & Tutorials* 15 (2013), pp. 860–880 (cit. on pp. 1, 12, 13).

[2] Thilo Sauter, Stefan Soucek, Wolfgang Kastner, and Dietmar Dietrich. "The Evolution of Factory and Building Automation." In: *IEEE Industrial Electronics Magazine* 5.3 (2011), pp. 35–48. ISSN: 1941-0115. DOI: 10.1109/MIE.2011.942175 (cit. on pp. 2, 75).

[3] IEEE 802.1Qcc/D2.1 Standard. *Bridges and Bridged Networks - Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*. 2018 (cit. on p. 2).

[4] Timothy et al. editors Gower. *The Princeton Companion to Mathematics*. Princeton University Press, 2008. ISBN: 9780691118802. URL: http://www.jstor.org/stable/j.ctt7sd01 (cit. on p. 3).

[5] Andrew Campbell, Hermann de Meer, Michael Kounavis, Kazuho Miki, John Vicente, and Daniel Villela. "A Survey of Programmable Networks." In: *Computer Communication Review* 29 (Apr. 1999), pp. 7–23. DOI: 10.1145/505733.505735 (cit. on pp. 4, 31).

[6] Alex Galis, Spyros Denazis, Celestin Brou, and Cornel Klein. *Programmable Networks for IP Service Deployment*. Norwood, MA, USA: Artech House, Inc., 2004. ISBN: 1580537456 (cit. on pp. 4, 17–19, 28, 31, 32).

[7] Samson AG. *Communication Networks (Technical information)*. last accessed January 2019. URL: https://www.samson.de/document/l155en.pdf (cit. on pp. 9–12).

[8] Eckehard Steinbach, Sandra Hirche, Marc Ernst, Fernanda Brandi, Rahul Chaudhari, Julius Kammerl, and Iason Vittorias. "Haptic Communications." In: *Proceedings of the IEEE* 100.4 (2012), pp. 937–956. ISSN: 1558-2256. DOI: 10.1109/JPROC.2011.2182100 (cit. on p. 10).

[9] Jonas Moll and Eva-Lotta Sallnäs. "Communicative Functions of Haptic Feedback." In: *Haptic and Audio Interaction Design*. Ed. by M. Ercan Altinsoy, Ute Jekosch, and Stephen Brewster. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–10. ISBN: 978-3-642-04076-4 (cit. on p. 10).

[10] C. Dovrolis and J. Todd Streelman. "Evolvable Network Architectures: What Can We Learn from Biology?" In: *SIGCOMM Comput. Commun. Rev.* 40.2 (Apr. 2010), pp. 72–77. ISSN: 0146-4833. DOI: 10.1145/1764873.1764886. URL: http://doi.acm.org/10.1145/1764873.1764886 (cit. on pp. 14, 24, 32).

[12] L. Yang, R. Dantu, T. Anderson, and R. Gopal. *Forwarding and Control Element Separation Framework*. (ForCES) Framework. RFC 3746. April 2004 (cit. on pp. 17, 19, 151).

[13] H. Khosravi and T. Anderson. *Requirement for Separation of IP Control and Forwarding*. (ForCES) Framework. RFC 3654. November 2003 (cit. on pp. 19, 20).

[14] Open Networking Foundation (ONF). "Software-Defined Networking (SDN) Architecture." In: ONF TR-521, Issue 1.1, 2016 (cit. on pp. 21, 22).

[15] Ghaffarkhah Alireza, Gopalpur Devjit, O'Connor Brian, Chau Uyen, and Wanderer Jim. "Stratum: Enabling the next generation of SDN." 2019. URL: https://www.netsia.com/brochures/5-Jim-Wanderer-Stratum_%20Enabling-the-Next-Generation-of-SDN.pdf (cit. on p. 29).

[16] J. Biswas, A. A. Lazar, J. Huard, L. Koonseng, S. Mahjoub, L. Pau, M. Suzuki, S. Torstensson, W. Weiguo, and S. Weinstein. "The IEEE P1520 standards initiative for programmable network interfaces." In: *IEEE Communications Magazine* 36.10 (1998), pp. 64–70. ISSN: 1558-1896. DOI: 10.1109/35.722138 (cit. on p. 31).

[17] Samphel Norden and Kenneth F. Wong. "ANMAC: An Architectural Framework for Network Management and Control Using Active Networks." In: *Active Networks*. Ed. by Stefan Covaci. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 212–219. ISBN: 978-3-540-48507-0 (cit. on p. 31).

[18] Ken L. Calvert. *Architectural framework for active networks*. 1999. URL: https://www.cc.gatech.edu/projects/canes/papers/arch-1-0.pdf (cit. on p. 31).

[19] Open Networking Foundation. *OF-Config; OpenFlow Management and Configuration Protocol* (cit. on p. 32).

[20] Open Networking Foundation. *OpenFlow Switch Specification* (cit. on p. 32).

[21] Alexandru Stancu, Alexandru Avram, Martin Skorupski, Alexandru Vulpe, and Simona Halunga. "Enabling SDN application development using a NETCONF mediator layer simulator." In: *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. 2017, pp. 658–663. DOI: 10.1109/ICUFN.2017.7993873 (cit. on p. 32).

[22] Thomas Kunz and Karpakamurthy Muthukumar. "Comparing OpenFlow and NETCONF when interconnecting data centers." In: *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. 2017, pp. 1–6. DOI: 10.1109/ICNP.2017.8117598 (cit. on p. 32).

[23] M. Dallaglio, N. Sambo, F. Cugini, and P. Castoldi. "Control and management of transponders with NETCONF and YANG." In: *IEEE/OSA Journal of Optical Communications and Networking* 9.3 (2017), B43–B52. DOI: 10.1364/JOCN.9.000B43 (cit. on p. 32).

[24] Stefan Wallin and Claes Wikström. "Automating Network and Service Configuration Using NETCONF and YANG." In: *Proceedings of the 25th International Conference on Large Installation System Administration*. Boston, 2011 (cit. on p. 32).

[25] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. "OpenDaylight: Towards a Model-Driven SDN Controller architecture." In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. 2014, pp. 1–6. DOI: 10.1109/WoWMoM.2014.6918985 (cit. on p. 32).

[26] Pankaj Berde et al. "ONOS: Towards an Open, Distributed SDN OS." In: *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. HotSDN '14. Chicago, Illinois, USA: ACM, 2014, pp. 1–6. ISBN: 978-1-4503-2989-7. DOI: 10.1145/2620728.2620744. URL: http://doi.acm.org/10.1145/2620728.2620744 (cit. on p. 32).

[27] Andreas Fischer, Hermann de Meer, and Wolfgang. Kellerer. *An Evaluation Methodology for Virtual Network Embedding*. Universität Passau, 2016. URL: https://books.google.de/books?id=hepIswEACAAJ (cit. on pp. 34, 35, 39, 42, 43, 49, 95, 96, 103).

[28] Andreas Berl, Andreas Fischer, and Hermann de Meer. "Virtualisierung im Future Internet." In: *Informatik-Spektrum* 33.2 (2010), pp. 186–194. ISSN: 1432-122X. DOI: 10.1007/s00287-010-0420-z. URL: https://doi.org/10.1007/s00287-010-0420-z (cit. on p. 34).

[29] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. "Network virtualization: state of the art and research challenges." In: *IEEE Communications Magazine* 47.7 (2009), pp. 20–26. ISSN: 1558-1896. DOI: 10.1109/MCOM.2009.5183468 (cit. on pp. 34, 41).

[30] N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. "A Survey of Network Virtualization." In: *Comput. Netw.* 54.5 (Apr. 2010), pp. 862–876. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2009.10.017. URL: http://dx.doi.org/10.1016/j.comnet.2009.10.017 (cit. on pp. 34, 41).

[31] A. T. Campbell, M. E. Kounavis, D. A. Villela, J. B. Vicente, H. G. De Meer, K. Miki, and K. S. Kalaichelvan. "Spawning networks." In: *IEEE Network* 13.4 (1999), pp. 16–29. ISSN: 1558-156X. DOI: 10.1109/65.777438 (cit. on p. 34).

[32] Zoran Despotovic, Artur Hecker, Ahsan Naveed Malik, Riccardo Guerzoni, Ishan Vaishnavi, Riccardo Trivisonno, and Sergio A. Beker. "VNetMapper: A fast and scalable approach to virtual networks embedding." In: *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. 2014, pp. 1–6. DOI: 10.1109/ICCCN.2014.6911859 (cit. on pp. 37, 41, 43, 70, 96, 98, 99).

[33] Riccardo Trivisonno, Ishan Vaishnavi, Riccardo Guerzoni, Zoran Despotovic, Artur Hecker, Sergio A. Beker, and David Soldani. "Virtual Links Mapping in Future SDN-Enabled Networks." In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. 2013, pp. 1–5. DOI: 10.1109/SDN4FNS.2013.6702562 (cit. on pp. 37, 43, 49, 70, 73, 93, 96, 98–100).

[35] Ashraf A. Shahin. "Virtual Network Embedding Algorithms Based on Best-Fit Subgraph Detection." In: *Computer and Information Science* 8 (2015), pp. 62–73 (cit. on p. 37).

[36] Jens Lischka and Holger Karl. "A Virtual Network Mapping Algorithm Based on Subgraph Isomorphism Detection." In: *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*. VISA '09. Barcelona, Spain: ACM, 2009, pp. 81–88. ISBN: 978-1-60558-595-6. DOI: 10.1145/1592648.

1592662. URL: http://doi.acm.org/10.1145/1592648.1592662 (cit. on p. 37).

[37] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach. "Virtual Network Embedding: A Survey." In: *IEEE Communications Surveys Tutorials* 15.4 (2013), pp. 1888–1906. DOI: 10.1109/SURV.2013.013013.00155 (cit. on pp. 37, 69).

[38] Cunqian Yu, Weigang Hou, and Lei Guo. "A Survey on Virtual Network Embedding in Optical Cloud Data Center Networks." In: *2016 International Conference on Software Networking (ICSN)*. 2016, pp. 1–5. DOI: 10.1109/ICSN.2016.7501922 (cit. on pp. 37, 69).

[40] M. Li, C. Chen, C. Hua, and X. Guan. "Intelligent Latency-Aware Virtual Network Embedding for Industrial Wireless Networks." In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 7484–7496. DOI: 10.1109/JIOT.2019.2900855 (cit. on p. 40).

[41] Andreas Berl, Roman Weidlich, Michael Schrank, Helmut Hlavacs, and Hermann de Meer. "Network Virtualization in Future Home Environments." In: *Integrated Management of Systems, Services, Processes and People in IT*. Ed. by Claudio Bartolini and Luciano Paschoal Gaspary. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 177–190. ISBN: 978-3-642-04989-7 (cit. on p. 41).

[43] N. M. Mosharaf Kabir Chowdhury, M. R. Rahman, and R. Boutaba. "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping." In: *IEEE/ACM Transactions on Networking* 20.1 (2012), pp. 206–219. ISSN: 1063-6692. DOI: 10.1109/TNET.2011.2159308 (cit. on p. 42).

[44] N. M. Mosharaf Kabir Chowdhury, M. R. Rahman, and R. Boutaba. "Virtual Network Embedding with Coordinated Node and Link Mapping." In: *IEEE INFOCOM 2009*. 2009, pp. 783–791. DOI: 10.1109/INFCOM.2009.5061987 (cit. on p. 42).

[45] Edoardo Amaldi, Stefano Coniglio, Arie M. C. A. Koster, and Martin Tieves. *On the computational complexity of the virtual network embedding problem*. International Network Optimization Conference (INOC). 2015 (cit. on p. 43).

[46] David G. Andersen. *Theoretical Approaches To Node Assignment*. 2002. DOI: https://doi.org/10.1184/R1/6610829.v1 (cit. on p. 43).

[47] Andreas Fischer, Juan F. Botero, Michael Duelli, Daniel Schlosser, Xavier Hesselbach, and Hermann de Meer. "ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms." In: *Electronic Communications*

*of the EASST* 37 (2011). Ed. by Tiziana Margaria, Julia Padberg, Gabriele Taentzer, Horst Hellbrueck, Norbert Luttenberger, and Volker Turau, pp. 1–12. ISSN: 1863-2122. URL: http://www.net.fim.uni-passau.de/pdf/Fischer2011a.pdf (cit. on pp. 44, 49, 95, 96, 103).

[49] Michael Till Beck and Claudia Linnhoff-Popien. "On delay-aware embedding of virtual networks." In: *The sixth international conference on advances in future internet, AFIN*. Citeseer. 2014 (cit. on pp. 44, 49, 66).

[50] Giorgos Chochlidakis and Vasilis Friderikos. "Low latency virtual network embedding for mobile networks." In: *2016 IEEE International Conference on Communications (ICC)*. 2016, pp. 1–6. DOI: 10.1109/ICC.2016.7510997 (cit. on pp. 44, 49).

[51] Xiaoting Li. "Worst-case delay analysis of real-time switched Ethernet networks with flow local synchronization." 2013. URL: http://oatao.univ-toulouse.fr/10305/ (cit. on pp. 45, 66).

[52] Jean-Luc. Scharbarg and Christian Fraboul. "Simulation for end-to-end delays distribution on a switched Ethernet." In: *2007 IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007)*. 2007, pp. 1092–1099. DOI: 10.1109/EFTA.2007.4416904 (cit. on pp. 45, 66).

[53] O. Dolejs and Z. Hanzalek. "Simulation of Ethernet for real-time applications." In: *IEEE International Conference on Industrial Technology, 2003*. Vol. 2. 2003, 1018–1021 Vol.2. DOI: 10.1109/ICIT.2003.1290801 (cit. on pp. 45, 66).

[54] Jean-Luc Scharbarg, Frédéric Ridouard, and Christian Fraboul. "A Probabilistic Analysis of End-To-End Delays on an AFDX Avionic Network." In: *IEEE Transactions on Industrial Informatics* 5 (2009) (cit. on pp. 45, 66).

[55] Orna Grumberg and David E. Long. "Model Checking and Modular Verification." In: *ACM Trans. Program. Lang. Syst.* 16 (1994), pp. 843–871 (cit. on pp. 45, 66).

[56] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008. ISBN: 026202649X (cit. on pp. 45, 66).

[57] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification: Model-Checking Techniques and Tools*. 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 3642074782, 9783642074783 (cit. on pp. 45, 66).

[58] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. "Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network." In: *2009 IEEE Conference on Emerging Technologies Factory Automation*. 2009, pp. 1–8. DOI: 10.1109/ETFA.2009.5347083 (cit. on pp. 45, 47, 54, 66).

[59] S. Martin and P. Minet. "Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class." In: *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*. 2006, 8 pp.–. DOI: 10.1109/IPDPS.2006.1639424 (cit. on pp. 45, 47, 54, 66).

[60] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. "Improving the Worst-Case Delay Analysis of an AFDX Network Using an Optimized Trajectory Approach." In: *IEEE Transactions on Industrial Informatics* 6.4 (2010), pp. 521–533. ISSN: 1551-3203. DOI: 10.1109/TII.2010.2055877 (cit. on pp. 45, 47, 54, 66).

[61] Jean-Yves Le Boudec. "Application of network calculus to guaranteed service networks." In: *IEEE Transactions on Information Theory* 44.3 (1998), pp. 1087–1096. ISSN: 0018-9448. DOI: 10.1109/18.669170 (cit. on pp. 45, 66).

[62] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Heidelberg: Springer-Verlag, 2001. ISBN: 3-540-42184-X (cit. on pp. 45, 66).

[63] J. Georges, E. Rondeau, and T. Divoux. "Evaluation of switched Ethernet in an industrial context by using the Network Calculus." In: *4th IEEE International Workshop on Factory Communication Systems*. 2002, pp. 19–26. DOI: 10.1109/WFCS.2002.1159696 (cit. on pp. 45, 66).

[65] Xin Li, Mohammed Samaka, H Anthony Chan, Deval Bhamare, Lav Gupta, Chengcheng Guo, and Raj Jain. "Network slicing for 5G: Challenges and opportunities." In: *IEEE Internet Computing* 21.5 (2017), pp. 20–27 (cit. on p. 48).

[66] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K. Marina. "Network slicing in 5G: Survey and challenges." In: *IEEE Communications Magazine* 55.5 (2017), pp. 94–100 (cit. on p. 48).

[67] Jose Ordonez-Lucena, Pablo Ameigeiras, Diego Lopez, Juan J. Ramos-Munoz, Javier Lorca, and Jesus Folgueira. "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges." In: *IEEE Communications Magazine* 55.5 (2017), pp. 80–87 (cit. on p. 48).

[68] Alexandros Kaloxylos. "A survey and an analysis of network slicing in 5G networks." In: *IEEE Communications Standards Magazine* 2.1 (2018), pp. 60–65 (cit. on p. 48).

[69] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions." In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2429–2453 (cit. on p. 48).

[70] Xuan Zhou, Rongpeng Li, Tao Chen, and Honggang Zhang. "Network slicing as a service: enabling enterprises' own software-defined cellular networks." In: *IEEE Communications Magazine* 54.7 (2016), pp. 146–153. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7509393 (cit. on p. 48).

[71] Riccardo Trivisonno, Massimo Condoluci, Xueli An, and Toktam Mahmoodi. "mIoT Slice for 5G Systems: Design and Performance Evaluation." In: *Sensors* 18.2 (2018) (cit. on p. 49).

[72] Anders Ellersgaard Kalør, René Guillaume, Jimmy Jessen Nielsen, Andreas Mueller, and Petar Popovski. "Network slicing in industry 4.0 applications: Abstraction methods and end-to-end analysis." In: *IEEE Transactions on Industrial Informatics* 14.12 (2018), pp. 5419–5427 (cit. on p. 49).

[73] Amal S. Alzahrani and Ashraf A. Shahin. "Energy-Aware Virtual Network Embedding Approach for Distributed Cloud." In: *CoRR* abs/1710.11590 (2017). arXiv: 1710.11590. URL: http://arxiv.org/abs/1710.11590 (cit. on p. 49).

[74] Rongping Lin, Shan Luo, Haoran Wang, and Sheng Wang. "Energy-aware virtual network embedding in flexi-grid networks." In: *Opt. Express* 25.24 (2017), pp. 29699–29713. DOI: 10.1364/OE.25.029699. URL: http://www.opticsexpress.org/abstract.cfm?URI=oe-25-24-29699 (cit. on p. 49).

[75] IEEE *802.1Qbv Task Group-Enhancements for Scheduled Traffic*. URL: http://www.ieee802.org/1/pages/802.1bv.html (cit. on pp. 51, 80, 91).

[76] Jochen W. Guck and Wolfgang Kellerer. "Achieving end-to-end real-time Quality of Service with Software Defined Networking." In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. 2014, pp. 70–76. DOI: 10.1109/CloudNet.2014.6968971 (cit. on p. 53).

[77] Jane W. S. Liu. *Real-Time Systems*. Prentice Hall, 2000. ISBN: 9780130996510. URL: https://books.google.de/books?id=855QAAAAMAAJ (cit. on pp. 57, 80, 83, 110).

[78] Francesco Bianchi and Francesco Lo Presti. "A Latency-Aware Reward Model Based Greedy Heuristic for the Virtual Network Embedding Problem." In: *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS'16. Taormina, Italy: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2017, pp. 271–278. ISBN: 978-1-63190-141-6. DOI: 10.4108/eai.25-10-2016.2266742. URL: https://doi.org/10.4108/eai.25-10-2016.2266742 (cit. on p. 66).

[79] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Ittai Abraham, Mahesh Balakrishnan, Archit Gupta, and Aditya Akella. "A Unified Network Coordinate System for Bandwidth and Latency." In: (Apr. 2011) (cit. on p. 66).

[80] Liao Shengquan, Wu Chunming, Zhang Min, and Jiang Ming. "An efficient virtual network embedding algorithm with delay constraints." In: *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*. 2013, pp. 1–6 (cit. on p. 66).

[81] Dahlia Malkhi, Ittai Abraham, Mahesh Balakrishnan, and Rama Ramasubramanian. *A Unified Network Coordinate System for Bandwidth and Latency*. Tech. rep. MSR-TR-2008-124. 2008, p. 15. URL: https://www.microsoft.com/en-us/research/publication/a-unified-network-coordinate-system-for-bandwidth-and-latency/ (cit. on p. 66).

[82] Barbara Martini, Federica Paganelli, Paola Cappanera, Stefano Turchi, and Piero Castoldi. "Latency-aware composition of Virtual Functions in 5G." In: *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. 2015, pp. 1–6. DOI: 10.1109/NETSOFT.2015.7116188 (cit. on pp. 66, 93).

[83] Zihui Yan, Ning Wei, Qizhen Jin, and Xiaobo Zhou. "Latency-Aware Resource-Efficient Virtual Network Embedding in Software Defined Networking." In: *2019 28th Wireless and Optical Communications Conference (WOCC)*. 2019, pp. 1–5. DOI: 10.1109/WOCC.2019.8770635 (cit. on p. 66).

[84] Felipe Rodrigo De Souza, Charles Christian Miers, Adriano Fiorese, and Guilherme Piegas Koslovski. "QoS-Aware Virtual Infrastructures Allocation on SDN-Based Clouds." In: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 2017, pp. 120–129. DOI: 10.1109/CCGRID.2017.57 (cit. on pp. 66, 93).

[86] Mikael Capelle, Slim Abdellatif, Marie-José Huguet, and Pascal Berthou. "Online virtual links resource allocation in Software-Defined Networks." In: *2015 IFIP Networking Conference (IFIP Networking)*. 2015, pp. 1–9. DOI: 10.1109/IFIPNetworking.2015.7145320 (cit. on pp. 70, 72, 73).

[87] Long Gong, Yonggang Wen, Zuqing Zhu, and Tony Lee. "Toward profit-seeking virtual network embedding algorithm via global resource capacity." In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2014, pp. 1–9. DOI: 10.1109/INFOCOM.2014.6847918 (cit. on p. 70).

[88] Martin Wollschlaeger, Thilo Sauter, and Juergen Jasperneite. *The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things*. IEEE Industrial Electronics Magazine, vol. 11, no. 1. 2017 (cit. on p. 75).

[89] Santiago Soler Perez Olaya, Martin Wollschlaeger, Dennis Krummacker, Christoph Fischer, Hans D. Schotten, René Guillaume, Joachim W. Walewski, and Norman Franchi. "Communication Abstraction Supports Network Resource Virtualisation in Automation." In: *2018 IEEE 27th International Symposium on Industrial Electronics (ISIE)*. 2018, pp. 697–702. DOI: 10.1109/ISIE.2018.8433801 (cit. on p. 75).

[90] *Digital data communication for measurement and control – Fieldbus for use in industrial control systems – Part 500: Application Layer service definition (IEC 65C/435/CDV:2006)*. DIN EN 61158-500:2006. 2006 (cit. on p. 76).

[92] Silviu S. Craciunas and Ramon Serna Oliver. "Combined Task- and Network-level Scheduling for Distributed Time-triggered Systems." In: *Real-Time Syst.* 52.2 (Mar. 2016), pp. 161–200. ISSN: 0922-6443. DOI: 10.1007/s11241-015-9244-x. URL: http://dx.doi.org/10.1007/s11241-015-9244-x (cit. on pp. 80, 93).

[93] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann de Meer, and Xavier Hesselbach. "Virtual Network Embedding: A Survey." In: *IEEE Communications Surveys Tutorials* 15.4 (2013), pp. 1888–1906. ISSN: 2373-745X. DOI: 10.1109/SURV.2013.013013.00155 (cit. on p. 93).

[94] Abdeltouab Belbekkouche, Mahmud Md. Hasan, and Ahmed Karmouch. "Resource Discovery and Allocation in Network Virtualization." In: *IEEE Communications Surveys Tutorials* 14.4 (2012), pp. 1114–1128. ISSN: 2373-745X. DOI: 10.1109/SURV.2011.122811.00060 (cit. on p. 93).

[95] Francisco Pozo. "Synthesis of Extremely Large Time-Triggered Network Schedules." In: *Proceedings* 1.3 (2017). ISSN: 2504-3900. DOI: 10.3390/IS4SI-2017-04018. URL: http://www.mdpi.com/2504-3900/1/3/171 (cit. on p. 93).

[96] Silviu S. Craciunas and Ramon Serna Oliver. "SMT-based Task- and Network-level Static Schedule Generation for Time-Triggered Networked Systems." In: *Proceedings of the 22Nd International Conference on Real-Time Networks and Systems*. RTNS '14. Versaille, France: ACM, 2014, 45:45–45:54. ISBN: 978-1-4503-2727-5. DOI: 10.1145/2659787.2659812. URL: http://doi.acm.org/10.1145/2659787.2659812 (cit. on p. 93).

[97] Silviu S. Craciunas, Ramon Serna Oliver, and Wilfried Steiner. "Formal Scheduling Constraints for Time-Sensitive Networks." In: *CoRR* abs/1712.02246 (2017). arXiv: 1712.02246. URL: http://arxiv.org/abs/1712.02246 (cit. on p. 93).

[98] Guy Avni, Shibashis Guha, and Guillermo Rodriguez-Navas. "Synthesizing time-triggered schedules for switched networks with faulty links." In: *2016 International Conference on Embedded Software (EMSOFT)*. 2016, pp. 1–10. DOI: 10.1145/2968478.2968499 (cit. on p. 93).

[99] Naresh Ganesh Nayak, Frank Dürr, and Kurt Rothermel. "Incremental Flow Scheduling Routing in Time-sensitive Software-defined Networks." In: *IEEE Transactions on Industrial Informatics* (2017) (cit. on pp. 93, 141).

[100] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. "Satisfiability modulo theories." English (US). In: *Handbook of Satisfiability*. 1st ed. Vol. 185. Frontiers in Artificial Intelligence and Applications 1. 2009, pp. 825–885. ISBN: 9781586039295. DOI: 10.3233/978-1-58603-929-5-825 (cit. on p. 93).

[101] Tri Trinh, Hiroshi Esaki, and Chaodit Aswakul. "Quality of service using careful overbooking for optimal virtual network resource allocation." In: *The 8th Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association of Thailand - Conference 2011*. 2011, pp. 296–299. DOI: 10.1109/ECTICON.2011.5947831 (cit. on pp. 96, 98).

[102] Ermin Sakic, Vivek Kulkarni, Vasileios Theodorou, Anton Matsiuk, Simon Kuenzer, Nikolaos E Petroulakis, and Konstantinos Fysarakis. "VirtuWind - An SDN- and NFV-based Architecture for Softwarized Industrial Networks." en. In: *GI/ITG Measurement, Modelling and Evaluation of Computing Systems (MMB) 2018*. Erlangen, Germany, 2018. DOI: 10.1007/978-3-319-74947-1_17 (cit. on p. 96).

[103] 5G-ACIA. *5G for Connected Industries and Automation*. ZVEI - German Electrical and Electronic Manufacturers' Association. 2018 (cit. on p. 96).

[104] Feng Chen. *Resource Allocation Protocol (RAP) based on LRP for Distributed Configuration of Time-Sensitive Streams*. 2017. URL: http://ieee802.org/1/files/public/docs2017/tsn-chen-RAP-whitepaper-1117-v02.pdf (cit. on pp. 113, 116, 121, 141, 142).

[106] Levi Pearson. *Stream Reservation Protocol AVnu Best Practices*. 2014. URL: https://avnu.org/wp-content/uploads/2014/05/AVnu_Stream-Reservation-Protocol-v1.pdf (cit. on p. 114).

[107] Ahmed Nasrallah, Venkatraman Balasubramanian, Akhilesh Thyagaturu, Martin Reisslein, and Hesham ElBakoury. "Reconfiguration Algorithms for High Precision Communications in Time Sensitive Networks: Time-Aware Shaper Configuration with IEEE 802.1Qcc (Extended Version)." In: *CoRR* abs/1902.02537 (2019). arXiv: 1906.11596. URL: http://arxiv.org/abs/1906.11596 (cit. on pp. 121, 141, 142).

[108] Ermin Sakic and Wolfgang Kellerer. "Response Time and Availability Study of RAFT Consensus in Distributed SDN Control Plane." In: *CoRR* abs/1902.02537 (2019). arXiv: 1902.02537. URL: http://arxiv.org/abs/1902.02537 (cit. on pp. 122, 133, 142).

[109] Ermin Sakic and Wofgang Kellerer. "Impact of Adaptive Consistency on Distributed SDN Applications: An Empirical Study." In: *IEEE Journal on Selected Areas in Communications* 36.12 (2018), pp. 2702–2715. ISSN: 0733-8716. DOI: 10.1109/JSAC.2018.2871309 (cit. on pp. 122, 142).

[110] Ermin Sakic and Wolfgang Kellerer. "BFT Protocols for Heterogeneous Resource Allocations in Distributed SDN Control Plane." In: *CoRR* abs/1902.02519 (2019). arXiv: 1902.02519. URL: http://arxiv.org/abs/1902.02519 (cit. on pp. 122, 142).

[112] Levent Ertaul and Krishnakumar. Venkatachalam. "Security of Software Defined Networks (SDN)." In: *International Conference on wireless Networks*. July 17-20, 2017. DOI: inLasVegas,Nevada,USA. (cit. on pp. 122, 141).

[113] Ijaz Ahmad, Suneth Namal, Mika Ylianttila, and Andrei Gurtov. "Security in Software Defined Networks: A Survey." In: *IEEE Communications Surveys Tutorials* 17.4 (2015), pp. 2317–2346. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2474118 (cit. on p. 122).

[114] Abha Kumari and Ashok Singh Sairam. "A Survey of Controller Placement Problem in Software Defined Networks." In: *CoRR* abs/1905.04649 (2019). arXiv: 1905.04649. URL: http://arxiv.org/abs/1905.04649 (cit. on pp. 123, 141).

[115] Guodong Wang, Yanxiao Zhao, Jun Huang, and Yulei Wu. "An Effective Approach to Controller Placement in Software Defined Wide Area Networks." In: *IEEE Transactions on Network and Service Management* 15.1 (2018), pp. 344–355. ISSN: 1932-4537. DOI: 10.1109/TNSM.2017.2785660 (cit. on pp. 123, 141).

[116] Ahmad Jalili, Manijeh Keshtgari, Reza Akbari, and Reza Javidan. "Multi Criteria Analysis of Controller Placement Problem in Software Defined Networks." In: *Computer Communications* 133 (2019), pp. 115 –128. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2018.08.003. URL: http://www.sciencedirect.com/science/article/pii/S0140366418300045 (cit. on pp. 123, 141).

[117] Boudewiji R. Haverkort, Raymond Marie, Gerardo Rubino, and Kishor S. Trivedi. *Performability Modelling Techniques and Tools*. John Wiley & Sons Academic Press Professional, Inc., 2001. 338 pp. ISBN: 978-0-471-49195-8 (cit. on p. 124).

[118] Samuel Kounev, Simon Spinner, and Philipp Meier. "Introduction to Queueing Petri Nets: Modeling Formalism, Tool Support and Case Studies." In: *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*. ICPE '12. Boston, Massachusetts, USA: ACM, 2012, pp. 9–18. ISBN: 978-1-4503-1202-8. DOI: 10.1145/2188286.2188290. URL: http://doi.acm.org/10.1145/2188286.2188290 (cit. on p. 124).

[119] Christos G. Cassandras and Stephane 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 (cit. on p. 124).

[120] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor 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 (cit. on p. 125).

[121] F. W. Scholz and M. A. Stephens. "K-Sample Anderson–Darling Tests." In: *Journal of the American Statistical Association* 82.399 (1987), pp. 918–924. DOI: 10.1080/01621459.1987.10478517. eprint: https://doi.org/10.1080/01621459.1987.10478517. URL: https://doi.org/10.1080/01621459.1987.10478517 (cit. on p. 127).

[122] T. W. Anderson and D. A. Darling. "A Test of Goodness of Fit." In: *Journal of the American Statistical Association* 49.268 (1954), pp. 765–769. DOI: 10.1080/01621459.1954.10501232. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1954.10501232. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1954.10501232 (cit. on p. 127).

[123] Walter Böhm and Kurt Hornik. "A Kolmogorov-Smirnov Test for R Samples." In: *Fundam. Inf.* 117.1-4 (Jan. 2012), pp. 103–125. ISSN: 0169-2968. URL: http://dl.acm.org/citation.cfm?id=2385103.2385110 (cit. on p. 127).

[124] Taylor B. Arnold and John W. Emerson. "Nonparametric Goodness-of-Fit Tests for Discrete Null Distributions." In: *The R Journal* 3.2 (2011), pp. 34–39. DOI: 10.32614/RJ-2011-016. URL: https://doi.org/10.32614/RJ-2011-016 (cit. on p. 127).

[125] Yadolah Dodge. "Chi-square Goodness of Fit Test." In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 72–76. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_55. URL: https://doi.org/10.1007/978-0-387-32833-1_55 (cit. on p. 127).

[126] Matthias Herlich, Jia Lei Du, Fabian Schörghofer, and Peter Dorfinger. "Proof-of-concept for a Software-defined Real-time Ethernet." In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016, pp. 1–4. DOI: 10.1109/ETFA.2016.7733605 (cit. on p. 141).

[127] Michael Brooks and Baijian Yang. "A Man-in-the-Middle Attack Against OpenDayLight SDN Controller." In: *Proceedings of the 4th Annual ACM Conference on Research in Information Technology*. RIIT '15. Chicago, Illinois, USA: ACM, 2015, pp. 45–49. ISBN: 978-1-4503-3836-3. DOI: 10.1145/2808062.2808073. URL: http://doi.acm.org/10.1145/2808062.2808073 (cit. on p. 141).

[128] Abubakar Siddique Muqaddas, Andrea Bianco, Paolo Giaccone, and Guido Maier. "Inter-Controller Traffic to Support Consistency in ONOS Clusters." In: *IEEE Transactions on Network and Service Management* 14.4 (2017), pp. 1018–1031. ISSN: 1932-4537. DOI: 10.1109/TNSM.2017.2723477 (cit. on p. 142).

[129] William H. Sanders and John F. Meyer. "Stochastic Activity Networks: Formal Definitions and Concepts." In: *Lectures on Formal Methods and PerformanceAnalysis: First EEF/Euro Summer School on Trends in Computer Science Bergen Dal, The Netherlands, July 3–7, 2000 Revised Lectures*. Ed. by Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 315–343. ISBN: 978-3-540-44667-5.

DOI: 10.1007/3-540-44667-2_9. URL: https://doi.org/10.1007/3-540-44667-2_9 (cit. on p. 142).

[130] Willaim H. Sanders and John F. Meyer. "Lectures on Formal Methods and Performance Analysis." In: ed. by Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen. New York, NY, USA: Springer-Verlag New York, Inc., 2002. Chap. Stochastic Activity Networks: Formal Definitions and Concepts, pp. 315–343. ISBN: 3-540-42479-2. URL: http://dl.acm.org/citation.cfm?id=567305.567314 (cit. on p. 142).

[131] David Daly, Daniel D. Deavours, Jay M. Doyle, Patrick G. Webster, and William H. Sanders. "Möbius: An Extensible Tool for Performance and Dependability Modeling." In: *Computer Performance Evaluation.Modelling Techniques and Tools*. Ed. by Boudewijn R. Haverkort, Henrik C. Bohnenkamp, and Connie U. Smith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 332–336. ISBN: 978-3-540-46429-7 (cit. on p. 142).

[132] Paul J. Kuehn, Sebastian Scholz, Siyuan Cao, and Fei Li. "Performance modeling of networked control systems." In: *2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. 2017, pp. 32–39. DOI: 10.1109/ICUMT.2017.8255161 (cit. on p. 142).

[133] Marco Bertoli, Giuliano Casale, and Guiseppe Serazzi. "Java Modelling Tools: An Open Source Suite for Queueing Network Modelling and Workload Analysis." In: *Proceedings of QEST 2006 Conference*. Riverside, US: IEEE Press, 2006, pp. 119–120 (cit. on p. 142).

[134] Marco Bertoli, Giuliano Casale, and Guiseppe Serazzi. *An Overview of the JMT Queueing Network Simulator*. Tech. rep. TR 2007.2. Politecnico di Milano - DEI, 2007 (cit. on p. 142).

[135] Marco Bertoli, Giuliano Casale, and Guiseppe Serazzi. "The JMT Simulator for Performance Evaluation of Non-Product-Form Queueing Networks." In: *Annual Simulation Symposium*. Norfolk,VA, US: IEEE Computer Society, 2007, pp. 3–10. ISBN: 978-0-7695-2814-4 (cit. on p. 142).

[136] Keshav Sood, Shui Yu, and Yong Xiang. "Performance Analysis of Software-Defined Network Switch Using $M/Geo/1$ Model." In: *IEEE Communications Letters* 20.12 (2016), pp. 2522–2525. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2016.2608894 (cit. on p. 142).

[137] Alexander Gelberger, Niv Yemini, and Ran Giladi. "Performance Analysis of Software-Defined Networking (SDN)." In: *IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*. 2013, pp. 389–393. DOI: 10.1109/MASCOTS.2013.58 (cit. on p. 142).

[138] G. Araniti, J. Cosmas, A. Iera, A. Molinaro, R. Morabito, and A. Orsino. "Openflow Over Wireless Networks: Performance Analysis." In: *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. 2014, pp. 1–5. DOI: 10.1109/BMSB.2014.6873559 (cit. on p. 142).

[139] András Varga and Rudolf Hornig. "An Overview of the OMNeT++ Simulation Environment." In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 60:1–60:10. ISBN: 978-963-9799-20-2. URL: http://dl.acm.org/citation.cfm?id=1416222.1416290 (cit. on p. 142).

[140] Maarten van Steen and Andrew S. Tanenbaum. "A brief introduction to distributed systems." English. In: *Computing* 98.10 (2016), pp. 967–1009. ISSN: 0010-485X. DOI: 10.1007/s00607-016-0508-7 (cit. on p. 149).

[141] L. Yang et al. "ForCES Forwarding Element Functional Model." In: *RFC 5812* (March 2003). URL: http://www.rfc-editor.org/info/rfc5812. (cit. on p. 151).

[142] Simon Brooks and Ecehan Uludag. *Time-Sensitive Networking: From Theory to Implementation in Industrial Automation*. 2019. URL: https://www.tttech.com/wp-content/uploads/TSN-in-industrial-automation.pdf (cit. on p. 152).

# DECLARATION

I hereby declare that this thesis titled "Performance and Optimization Technologies for Software Defined Industrial Networks" is my own work and has not been submitted as an excise nor examination for a degree in this or any other university.

The thesis has not been published in full. For the published parts of this thesis, I have duly cited and remarked where similarly they have been used.

I agree to deposit copies of this thesis in the university's library.

*Munich, Germany, May 2021*

Frimpong Ansah

## COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/