Redundancy in Communication Networks for Smart Grids

THÈSE N^{O} 6973 (2016)

PRÉSENTÉE LE 28 AVRIL 2016

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 2
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Miroslav POPOVIC

acceptée sur proposition du jury:

Prof. P. Frossard, président du jury
Prof. J.-Y. Le Boudec, directeur de thèse
Prof. D. Hutchison, rapporteur
Prof. H. Kirrmann, rapporteur
Prof. M. Paolone, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2016

To my parents…

Acknowledgements

First and foremost, I would like to thank my thesis director, Professor Jean-Yves Le Boudec, for his constant guidance and motivation, both on a scientific and personal level. His dedication to his students goes far beyond professional level and is something exceptional nowadays. The influence he had on me is invaluable and will follow me through all my life. I will always be proud of being his student.

The perfect atmosphere in LCA2 is yet another accomplishment of his. I had the privilege to be part of a team composed of extraordinary people. Irina was my first office mate, I experienced my first professional successes with Ramin and Nicolas. Dan was someone you could really count on when you need it. With Tech and Nadia, we went together through all the ups and downs of the PhD life, and Maaz showed up toward the end of my thesis work as a perfect complement. I hope we will all stay in touch and that we will continue our excursions to Athens, Hyderabad, and, who knows, perhaps even to Addis Ababa, if Tech does not forget something again.

Patricia, Holly (correcting-my-articles-and-adding-hyphens lady), Angela, Danielle, Marc-André and Yves, you were simply perfect support. They make the life of PhD students much smoother, and I have great appreciation for all the effort you invest.

At this point, I also have to mention the DESL crew that is led by my unofficial co-advisor, Professor Mario Paolone. His vision and energy were an important driving force throughout my thesis work and it was a real pleasure, not only to work with Paolo, Lorenzo and Marco, but also to play football and basketball with them.

I would also like to thank Professors David Hutchinson, Hubert Kirrmann and Pascal Frossard for being part of my PhD jury and for their valuable comments.

On a non-professional side, Nikola, Cojba, Vladan, Velja and Kopta were amazing company for lunch at EPFL (before they all finished ahead of me and left EPFL). Nothing would be the same without Turbo Galaxy and Black Sheep, two multiple-championship-winning teams composed of great people. I thank Milos, Moki, Jeki, Miha, Luka, Filip, David, Yota, for being such great friends all of these years.

Last but not least, I am grateful to my whole family. My parents laid the foundation for everything in my life with their unconditional and incomparable love, support and belief in me. Jelena was, is, and will always be the best sister in the world. By far! This thesis is Teodora's as much as mine, as only she knows how she survived all my quirks while still showing endless love and patience for me. Nikola and Marko made my life toward the end of my thesis work a little bit more difficult, but infinitely more beautiful.

Abstract

Traditional electric power grids are currently undergoing fundamental changes: Representative examples are the increase in the penetration of volatile and decentralized renewable-energy sources and the emerging distributed energy-storage systems. These changes are not viable without the introduction of automation in grid monitoring and control, which implies the application of information and communication technologies (ICT) in power systems. Consequently, there is a transition toward smart grids. IEEE defines smart grid as follows [1]: "The integration of power, communications, and information technologies for an improved electric power infrastructure serving loads while providing for an ongoing evolution of end-use applications" .

The indispensable components of the future smart grids are the communication networks. Many well-established techniques and best practices, applied in other domains, are revisited and applied in new ways. Nevertheless, some gaps still need to be bridged due to the specific requirements of the smart-grid communication networks. Concretely, a challenging objective is to fulfill reliability and low-delay requirements over the wide-area networks, commonly used in smart grids.

The main “playground" for the work presented in this thesis is the smart-grid pilot of the EPFL campus. It is deployed on the operational $20kV$ medium-voltage distribution network of the campus. At the time of the writing of this thesis, the real-time monitoring of this active distribution network has been already put in place, as the first step toward the introduction of control and protection. The monitoring infrastructure relies on a communication network that is a representative example of the smart-grid communication networks.

Keeping all this in mind, in this thesis, the main topic that we focus on, is the assurance of data communication over redundant network-infrastructure in industrial environments.

This thesis consists of two parts that correspond to the two aspects of the topic that we address. In the first part of the thesis, we evaluate existing, well-established, technologies and solutions in the context of the EPFL smart-grid pilot. We report on the architecture of the communication network that we built on our campus. In addition, we go into more detail by reporting on some of the characteristics of the devices used in the network. We also discuss security aspects of the MPLS Transport Profile (MPLS-TP) which is one of the proposed technologies in the context of smart grids.

Abstract

In the second part of this thesis, we propose new solutions. While designing our campus smart-grid network, we analyzed the imposed requirements and recognized the need for a solution for reliable packet delivery within stringent delay constraints over a redundant network-infrastructure. The existing solutions for exploiting network redundancy, such as the parallel redundancy protocol (PRP), are not viable for IP-layer wide-area networks, a key element of emerging smart grids. Other solutions (MPLS-TP for example) do not meet the stringent delay requirement. To address this issue, we present a transport-layer solution: the IP-layer parallel redundancy protocol (iPRP).

In the rest of the thesis, we analyze the methods for implementing fail-independent paths that are fundamental for the optimal operation of iPRP, in SDN-based networks. We also evaluate the benefits of iPRP in wireless environments. We show that, with a help of iPRP, the performance of the communication based on the Wi-Fi technology can be significantly improved.

Key words: communication networks, smart grid, redundancy, reliability, availability, mission-critical applications, industrial communications, low-latency, parallel redundancy protocol, packet replication, real-time communication, multicast, EPFL smart-grid testbed, software-defined networking, disjoint paths, disjoint trees, fail-independent paths, fail-independent trees, MPLS-TP, smart-grid security, security vulnerabilities, authentication, SHDLS, traffic engineering, traffic shaping, packet losses, wireless communications, Wi-Fi, measurements, performance evaluation.

Résumé

Les réseaux électriques traditionnels subissent actuellement des changements fondamentaux : des exemples représentatifs sont l'augmentation de la pénétration des sources d'énergies renouvelables (décentralisées et volatiles) et les systèmes distribués de stockage d'énergie. Ces changements ne sont pas viables sans l'introduction de l'automatisation dans la surveillance et le contrôle du réseau, ce qui implique l'application des technologies de l'information et de la communication (ICT) dans les systèmes électriques. Par conséquent, une transition vers les "smart grids" (réseau de distribution d'électricité "intelligent") est présente.

Les réseaux de communication sont des composants indispensables des futurs smart grids. La plupart des techniques qui sont déjà connues dans d'autres domaines s'appliquent ici, parfois de manière innovante. Néanmoins, certaines lacunes apparaissent en raison des exigences spécifiques des réseaux de communication des smart grids. Concrètement, un objectif ambitieux est de satisfaire aux exigences de fiabilité et de délais de transmission très courts sur les réseaux étendus (wide-area networks - WANs), couramment utilisés dans les smart grids.

Le principal banc de test pour le travail présenté dans cette thèse est le test expérimental d'un smart grid sur le campus de l'EPFL. Il est déployé sur le réseau électrique opérationnel du campus. Au moment de l'écriture de cette thèse, le suivi en temps réel de ce réseau de distribution actif a été déjà mis en place. C'est la première étape vers l'introduction du contrôle et de la protection. L'infrastructure de surveillance est basée sur un réseau de communication qui est un exemple représentatif des réseaux de communication de smart grids. En gardant tout cela à l'esprit, dans cette thèse, notre thème principal est le support de transmission de données sur une infrastructure de réseau redondant dans des environnements industriels.

Dans la première partie de la thèse, nous évaluons, les technologies et les solutions existantes dans le contexte du smart grid de l'EPFL. Nous décrivons l'architecture du réseau de communication que nous avons construit sur notre campus. En outre, nous montrons certaines des caractéristiques des dispositifs utilisés dans le réseau. Nous discutons également des aspects de sécurité de MPLS-TP (MPLS Transport Profile), qui est une des technologies proposées dans le cadre de smart grids.

Dans la deuxième partie de cette thèse, nous proposons de nouvelles solutions. Pendant la conception de notre réseau de communication, nous avons analysé les exigences qui sont imposées et nous avons reconnu la nécessité d'une solution pour

la transmission de paquets fiable et avec des contraintes strictes de délai sur une infrastructure réseau redondante. Les solutions existantes pour exploiter la redondance du réseau, tels que le protocole de redondance parallèle (parallel redundancy protocol - PRP), ne sont pas viables pour des réseaux étendus basés sur IP, un élément clé de l'émergence des réseaux intelligents. Les autres solutions (MPLS-TP, par exemple) ne répondent pas à l'exigence de délai strict. Pour résoudre ce problème, nous présentons une solution dans la couche transport : IP-layer parallel redundancy protocol (iPRP).

Dans le reste de la thèse, nous analysons les méthodes pour fournir des chemins de communication redondants qui sont fondamentales pour le fonctionnement optimal d'iPRP, dans les réseaux basés sur SDN. Nous évaluons également les avantages d'iPRP pour la communication sans fil. Nous montrons qu'iPRP améliore considérablement la performance de Wi-Fi.

Mots clefs : réseaux de communication, smart grid, redondance, fiabilité, disponibilité, communication industrielle, faible délai, parallel redundancy protocol, réplication de paquets, communication en temps réel, multicast, EPFL smart grid, Software Defined Networking, chemins de communication redondants, arbres indépendants, MPLS-TP, sécurité de smart grid, failles de sécurité, authentification, SHDSL, mise en forme du trafic, pertes de paquets, communications sans fil, Wi-Fi, mesures, évaluation de performance.

List of Abbreviations

ADN	Active distribution network
BFD	Bidirectional forwarding detection
CDF	Cumulative distribution function
CPU	Central processing unit
DSLAM	Digital subscriber line access multiplexer
DTLS	Datagram transport layer security
ECMP	Equal-cost multipath routing
FIFO	First in, first out
GPS	Global positioning system
ICT	Information and communication technologies
IND	iPRP network subcloud discriminator
IP	Internet protocol
iPRP	IP-layer parallel redundancy protocol
LACP	Link aggregation control protocol
LAN	Local-area networks
LER	Label edge router
LSP	Label-switched path
MAC	Media access control
MPLS	Multiprotocol label switching
MPLS-TP	Multiprotocol label switching - transport profile
MPTCP	Multipath transmission control protocol
MTU	Maximum transmission unit
OS	Operating system
OSPF	Open shortest path first
PDC	Phasor-data concentrator
PMU	Phasor-measurement unit
PSC	Protection state coordination
PRP	parallel redundancy protocol
RFC	Request for comments
RIP	Routing information protocol
RSTP	Rapid spanning tree protocol
RSVP	Resource reservation protocol

RTT	Round-trip time
SDH	Synchronous digital hierarchy
SDN	Software-defined networking
SE	State estimator
SHDSL	Symmetrical high-speed digital subscriber line
SONET	Synchronous optical networking
SSM	Source-specific multicast
STP	Spanning tree protocol
TCP	Transmission control protocol
TLS	Transport layer security
UDP	User datagram protocol
VPN	Virtual private network
WAN	Wide-area network

Contents

1 Introduction

1.1 Motivation

Lately, smart grid has become a buzzword that is heard in many different contexts. Perhaps the definition that is the simplest, and at the same time, sufficiently broad, is that smart grid is the convergence of traditional power grids, and information and communications technologies. Thus, an indispensable part of smart grids are communication networks.

The primary objective of smart-grid communication networks is to support traffic for all applications; this includes real-time traffic (e.g., for grid monitoring and control). Packet losses or excessive packet delays due to communication-network failures can result in economic losses or, even worse, human lives can be endangered in cases when these failures affect protection mechanisms. Therefore, high communication-network availability is an imperative (according to [2], even 99.999%, which translates to five minutes of downtime per year, might not be enough).

A typical technique for increasing communication-network availability is to build a redundant network-infrastructure (cloned networks, rings, rings of rings, etc.). However, the existence of redundant network-infrastructure alone is not enough, as it has to be accompanied by proper mechanisms for exploiting the existing redundancy in case of a failure and because of many other challenges. We can find many techniques in the literature that are successfully applied to other domains. But, due to specific timing and reliability requirements imposed on smart-grid communication networks, these techniques are not always suitable for achieving the desired levels of reliability. In this thesis, we address this problem and we offer solutions for effective exploitation of redundant network-infrastructure in the context of smart grids.

For a more general outlook, we discuss the global trends that led to the emergence of smart grids in Section 1.1.1. In addition, the development of the smart-grid pilot at

EPFL is described in Section 1.1.2. It served as a realistic base for the research that is presented in this thesis.

1.1.1 Global Trends

For many years, the predominant sources of electric power were large power plants. The produced power was transferred to consumers through a system of transmission and distribution substations where the voltage levels were progressively reduced, as it approaches the customers' premises. The whole process was characterized by one-way flows of electricity - from the large producers toward the customers. As such, the system was rather predictable, which is why the simple monitoring and control systems were acceptable. Operators were able to steer the system successfully in most cases, even if the time-granularity of the system-state updates was in the order of 15 minutes.

This well-established landscape started to change toward the end of the last century. Driven by the negative effects of the climate change, the introduction of renewable energy sources was inevitable. They are usually found in the form of distributed energy resources, which led to two-way power flows. In addition, the renewables can be highly intermittent (e.g., the sudden appearance of a cloud can dramatically change the power production of photo-voltaic panels). Hence, it is very difficult to predict power production. It became much more challenging to maintain the system stability and to guarantee service quality (e.g., voltage levels and frequency need to be in a certain range). Consequently, a much more sophisticated control and monitoring system became essential, which led to the introduction of information and communication technologies, i.e. smart grids. Furthermore, this transition has opened the doors for demand-response mechanisms and grid modernization in general.

Needless to say, smart-grid functionalities rely upon the reliable and timely exchange of information between intelligent electronic devices deployed in the field (controllers, actuators...) Thus, a reliable communication network represents the heart of the smart grid.

1.1.2 EPFL-Campus Smart-Grid Pilot

In order to understand the context in which we operate, we briefly describe the real-time monitoring infrastructure of the smart-grid pilot on the EPFL campus. The smart-grid pilot is deployed on the medium-voltage electrical grid of the EPFL campus. It is considered to be a challenging active distribution network (ADN) due to its characteristics, such as the presence of distributed power-generation (photo-voltaic systems and fuel cells), the presence of energy storage, short lines and a largely variable load-demand.

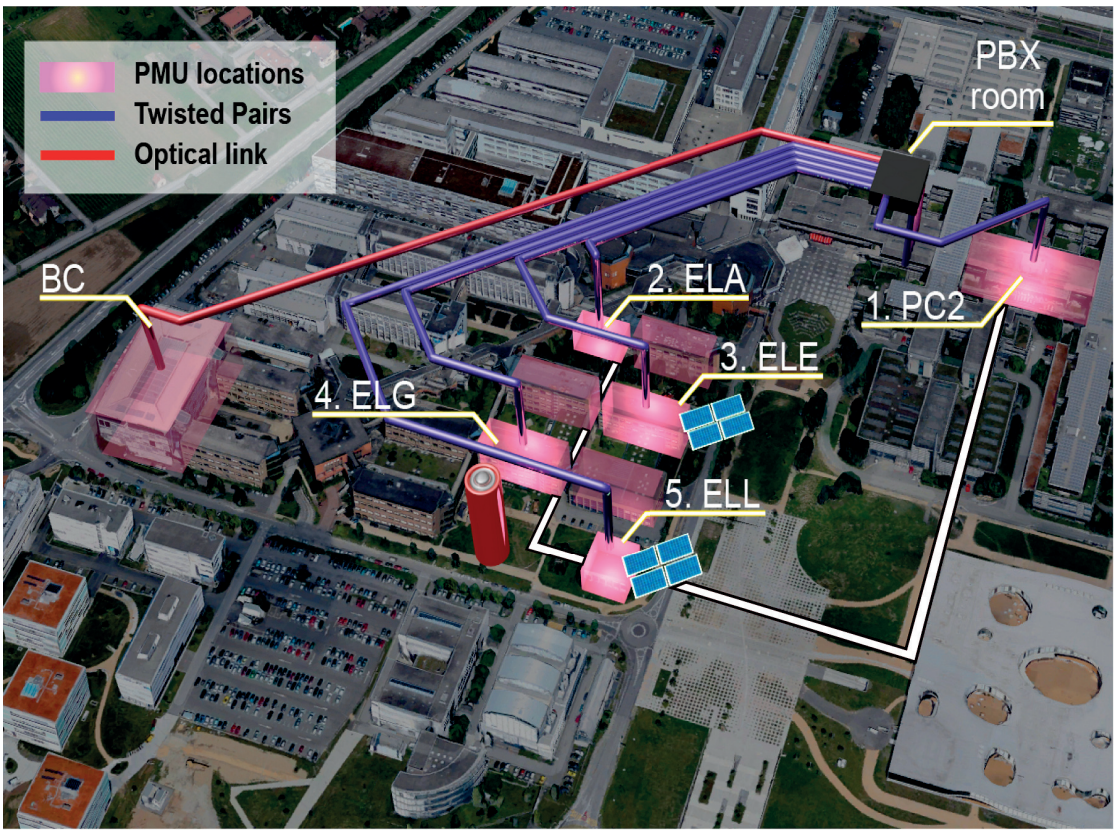


Figure 1.1: EPFL campus with locations of medium-voltage transformers that are monitored. The white line represents the electrical feeder for which the state estimation is performed. Acronyms in the Figure are the names of different buildings of the EPFL campus.

The real-time operation of an ADN uses its knowledge of the global state of the electrical network, which requires performing quasi-simultaneous, accurate, high-frequency measurements of physical quantities (such as voltage and phase) in buses of the electrical network. They are obtained by phasor measurement units (PMUs) equipped with synchronized clocks (e.g., via GPS) that are interfaced to medium-voltage transformers through specialized sensors. The data is then conveyed via a communication network to a phasor-data concentrator (PDC). The PDC concentrates the data and prepares it for processing by an electrical network state estimator (SE). Concretely, at EPFL, one feeder is monitored; this feeder spans five sites, each of them equipped with a PMU [3]. Figure 1.1 depicts locations of medium-voltage transformers that are monitored. And, in Figure 1.2, we show the block diagram that describes how the system elements are combined together.

For the moment, the output of the EPFL SE is used only for electrical-network monitoring. Nevertheless, this information produced by the SE can be used by a network controller for management purposes, which is expected to be implemented in

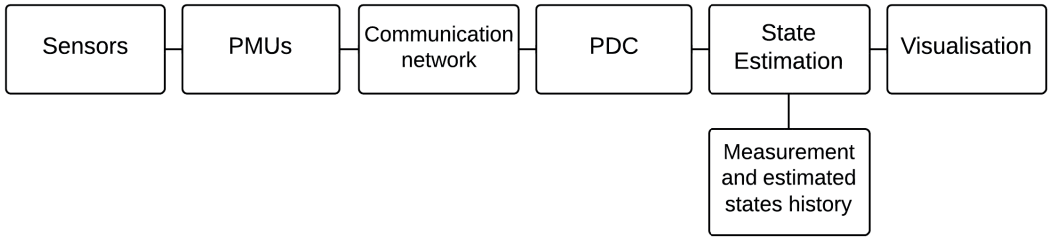


Figure 1.2: System-architecture block diagram

the future. Using the estimated state of the electrical network, the network controller should take actions that control the electrical network within a predefined (safe) range of operating set-points. Furthermore, the same information can be also used by other smart-grid applications such as fault location in power networks or active load-management using model predictive control strategy.

The EPFL-campus smart-grid pilot is a representative example of an active distribution network. As such, it is a perfect “playground" for the work presented in this thesis. In addition, it gave us a unique opportunity to experience the real-life challenges of building an ADN communication network. These challenges were invaluable in discovering the problems that need to be solved, which helped us to define our research directions.

1.2 Dissertation Outline

In the first part of the thesis, we evaluate existing, well-established, technologies and solutions in the context of the EPFL smart-grid pilot. Concretely, in Chapter 3, we report on the architecture of the communication network that we built on our campus. In addition, we go into more detail by reporting on some of the characteristics of the devices used in the network and the effect of these characteristics on the network operation. This work led us to conclusions about the importance of traffic engineering for the operation of critical network infrastructure.

MPLS Transport Profile (MPLS-TP) is one of the proposed technologies for smart-grid WAN connectivity. Therefore, it was a natural starting point for studying the approaches to building communication networks in smart grids. In particular, we were interested in the protection-switching mechanism of MPLS-TP, so we built a testbed to investigate its performance. While studying the protocol, we discovered some security vulnerabilities. We report on them in Chapter 4, where we show that, when it comes to the security aspects of the standard, there is a discrepancy between RFCs and the Cisco implementation for MPLS-TP that we evaluated, which is not surprising given

the complexity of RFCs. The revealed inconsistencies also show the importance of having a comprehensive approach to building systems, which includes security from the very beginning.

During the design of our campus smart-grid network, we analyzed the imposed requirements and recognized the need for a solution for reliable packet delivery within stringent delay-constraints over a redundant network-infrastructure. One of the conclusions of the investigation of the protection-switching mechanism of MPLS-TP, which is supposed to take advantage of the network redundancy, was that it does not satisfy the stringent time requirements imposed by mission-critical applications. The parallel-redundancy protocol (PRP) is another solution that satisfies the delay requirements. PRP works best in local area networks, e.g., sub-station networks, by replicating all packets at the MAC layer over parallel paths. It is not, however, viable for IP-layer wide-area networks, a key element of emerging smart grids. These findings initiated further research, and in Chapter 5 we present our solution to the problem of ensuring effective exploitation of redundant network-infrastructure: the IP-layer parallel redundancy protocol (iPRP).

The existence of fail-independent paths is fundamental for the optimal operation of iPRP. It is not difficult to enforce such paths in a dedicated and fully controlled network infrastructure. But, when a *shared* network infrastructure is used, this task becomes very challenging. We address this problem in Chapter 6, where we evaluate different algorithms (proposed in the literature) for providing node-redundant multicast-distribution trees. We study specifically how to adapt these algorithms in an SDN network and compare them based on the parameters that are relevant in an industrial environment.

Finally, in Chapter 7 we analyze the benefits of iPRP in wireless environments. Typically, wireless communication is not seen as a good candidate for low-delay communication with high-reliability requirements. We show that, with a help of iPRP, the performance of the communication based on the Wi-Fi technology can be significantly improved.

1.3 Contributions

The following is the list of the main contributions of this thesis.

• We built a dedicated communication network to support the needs of the EPFL-campus smart-grid pilot. It enables researchers from the collaborating Distributed Electrical Systems Laboratory (DESL) lab to put into practice theoretical concepts from the field of power systems. It enables us to gain first-hand experience of challenges in the processes of design and exploitation of smart-grid communication networks.

• Through the experiments in the network that we built, we defined its operating region, which was an important step toward successful network-operation. We also demonstrated the importance of traffic engineering.

• We built a testbed to evaluate the security of MPLS-TP. This enabled us to reveal security vulnerabilities of the MPLS-TP implementation of a major network manufacturer. We exploited those vulnerabilities by designing and putting into place number of attacks that result in partial or, in some attacks, the complete network being compromised.

• We designed and implemented the IP-layer parallel redundancy protocol (iPRP): a solution to the problem of effective utilization of redundant communication-networks. The design of iPRP poses non-trivial challenges in the form of selective packet-replication, soft-state and multicast support, which we overcame successfully. iPRP provides support for multicast, widely used in smart-grid networks. It has a characteristic of selectively replicating only the traffic desired by a user (e.g., time-critical UDP traffic). iPRP only requires a simple software installation on the end-devices. There are no modifications needed for the existing monitoring application, for the end-device operating system or for the intermediate network devices.

• We successfully deployed iPRP on the EPFL-campus smart-grid pilot thus demonstrating that iPRP contributes to the reliability of the message exchange between the hosts that are part of the EPFL-campus smart-grid pilot.

• We studied how to construct node-redundant multicast trees that could be used in parallel over a shared network infrastructure in the context of smart grids, and that are necessary for the optimal operation of iPRP. We evaluated different algorithms proposed in the literature for providing such trees. We studied specifically how to adapt these algorithms in an SDN network and compare them based on (1) the number of forwarding rules that need to be installed on SDN switches, (2) the number of hops between source-destination pairs given the installed forwarding rules, and (3) the number of sources that can be placed in the network given the capacity constraints. In addition, we discussed the effects of topology changes (node failures, new source arrival and new destination arrival) on the activity of the SDN control plane.

• Finally, we demonstrated the benefits of iPRP in a wireless environment. We built a roof-top testbed and conduct experiments where we show that iPRP reduces the number of packet losses perceived by an application that runs on top of iPRP. Hence, iPRP can contribute in building cost-effective communication networks based on wireless technologies.

2 State of the Art

We begin this chapter with an overview of the mechanisms for exploiting communication-network redundancy (Section 2.1). Later on, we focus on the smart-grid case where link or device failures are among the possible reasons for packet-delivery failures. These failures can be overcome by building redundant network-infrastructure. Therefore, we give particular attention to the mechanisms for exploiting network redundancy that are proposed in the context of smart grids (Section 2.2). Security goes hand-in-hand with the reliability of smart-grid communication networks, so we conclude the chapter by giving some details about the best practices in securing such networks from cyber-attacks (Section 2.3).

2.1 Overview of Mechanisms for Exploiting Network Redundancy

Typical examples of techniques for exploiting network redundancy are summarized in Table 2.1, based on [2].

Protocol	Recovery Time	Applicable to IP networks?
Routing protocols (e.g. OSPF)	tens of seconds	YES
Ethernet link aggregation (e.g. LACP)	few seconds	YES
Rapid spanning tree protocol	few seconds	YES
SONET/SDH rings	$50ms$	YES
MPLS-TP	$50ms$	YES
SDN-based solutions	tens of milliseconds	YES
Parallel redundancy protocol (PRP)	$0ms$	NO

Table 2.1: Techniques for exploiting network redundancy.

We proceed by giving main characteristics of these techniques.

Routing protocols. Based on the information exchanged and processed by the routers, they define the paths taken by packets in a communication network. We distinguish *distance-vector* routing protocols and *link-state* routing protocols. Distance-vector routing protocols can be based on the Bellman-Ford algorithm [4], and notable examples are RIP [5], RIPv2 [6] and IGMP [7]. Link-state routing protocols are based on creating a connectivity map of the entire network and notable examples are OSPF [8] and IS-IS [9].

Ethernet link aggregation. The main purposes of this technique are to increase throughput and to provide redundancy. It achieves this goals by grouping physical ports of a bridge which are then treated together. This way, failure of a single link from a group, once detected, does not affect connectivity. An example of such protocol is the link aggregation control protocol (LACP) [10].

Rapid spanning tree protocol (RSTP). The main purpose of RSTP [11] is to prevent loops in bridged networks. It achieves this by computing a tree that covers all the bridges in the network and that disables unused links. When necessary, RSTP also handles link or device failures by performing tree-reconfiguration. It represents an evolution of the older spanning tree protocol (STP) [12] and it is characterized by a faster convergence time upon topology changes, compared to STP.

SONET/SDH rings. The synchronous optical networking (SONET) [13] and the synchronous digital hierarchy (SDH) [14] are two similar technologies that are based on the fiber infrastructure, in the form of a ring, that interconnects network nodes and associated end-hosts. Communication is possible in both directions of the ring. Thus, any single link failure (e.g., fiber cut), once detected, does not disconnect the network nodes and the associated end-hosts.

MPLS-TP. The multiprotocol label switching (MPLS) [15] is a mechanism for packet forwarding that is based on path labels instead of IP addresses that are used by routing protocols. Each label identifies one label-switched path (LSP) that is established between MPLS routers. MPLS can coexist with non-MPLS networks (e.g., IP networks) through the concept of a label edge router (LER) that encapsulates or decapsulates IP packets on the edges of an MPLS network. An important functionality of MPLS is the support for traffic engineering. MPLS-TP (multiprotocol label switching - transport profile) [16] represents the evolution of MPLS, with improved features concerning

failure circumvention [17], which are important for mission-critical network infrastructure.

Software-defined networking (SDN). It represents an emerging approach to network control and management [18] and has the potential advantage of lower cost and higher modularity and flexibility, compared to traditional approaches. The main characteristics of SDN are the control and data planes that are decoupled. The centralized controller gathers all the information about the network and communicates the resulting forwarding rules on the SDN-switches. This concept is contrary to the one applied in the case of routing protocols where the decisions are made in a decentralized manner. OpenFlow [19] is the main representative of SDN-based solutions for network administration.

Parallel redundancy protocol (PRP). The parallel redundancy protocol (PRP) [20,21] is an IEC standard used to increase reliability. PRP can be used by devices with double network interfaces. It takes advantage of specific network topologies, where each device is connected through its network interfaces to fail-independent communication networks. An essential requirement of PRP is that the two networks need to be identical copies of each other. Hence, the two network interfaces of a same device (each of them connected to a different network) need to have *the same* MAC address. PRP works as follows: When a MAC frame is transmitted by a host, it is replicated on its two network interfaces, after being tagged with a redundancy control trailer. The destination MAC address is the same on both networks, as explained. The destination host receives the frames on its two interfaces. The added trailer contains a frame sequence number that, along with the source MAC address, enables the receiver to detect and eliminate redundant frames. Thus, a packet loss on one of the paths is repaired in zero time.

We are mainly interested in the recovery time that is needed for a failure repair, provided by the aforementioned techniques. This time depends on a failure-detection mechanism which is fundamentally the same for all the protocols — except for the parallel redundancy protocol (PRP) — and is based on some form of keep-alive signaling. The absence of several (usually three) consecutive keep-alive messages is recognized as a failure, which triggers the process of activating alternative paths. The shorter the keep-alive period is, the sooner the failure will be detected. In the same time, a shorter keep-alive period means more traffic overhead.

As mentioned before, the parallel redundancy protocol (PRP) [20] has a different approach. Packets are sent in parallel over cloned networks, and only the one that arrives first at the destination is accepted, the other one is silently discarded. This way, even if one of the cloned networks is not operational, packets are delivered over the healthy network to the applications that expect them.

Application	Delay allowance [ms]	Relative priority 0 (max)-100 (min)	Application type
Delay $\leq$ 10ms			
High-speed protection	8	2	Teleprotection
Load shedding	10	20	SCADA
10ms $<$ Delay $\leq$ 20ms			
Breaker reclosers	16	15	Teleprotection
PMU data (for protection)	20	12	Synchrophasors
20ms $<$ Delay $\leq$ 100ms			
PMU data	60	10	Synchrophasors
Time synchronization	100	20	Synchrophasors
100ms $<$ Delay $\leq$ 250ms			
VoIP bearer	175	50	Business voice
Critical business data	250	70	Business data
250ms $<$ Delay $\leq$ 1s			
Noncritical operations data	500	80	SCADA
Noncritical business data	500	80	Business data
1s $<$ Delay			
Image files	1000	90	SCADA
Best effort	2000	100	Many

Table 2.2: Sample of applications with their delay requirements and relative priorities for smart-grid use, based on [2].

In practice, often a combination of the techniques from Table 2.1 can be used. For example, a variant of the spanning-tree protocol and a routing protocol often go together, even in relatively simple networks and networks without high-availability requirements. Naturally, there is a question about whether these techniques are readily applicable in smart grids. It is not straightforward to answer this question because the term smart grid has different meanings to different people. For some, it is about collecting the non-critical information from smart meters [22]. At the other extreme, it is about using ICT infrastructure for protection in modern power grids [23]. Consequently, the delay allowances for the messages of interest vary in a similar way. Table 2.2, based on [2], represents a sample of different applications and the corresponding delay allowances and (relative) priorities. The delay allowances can be in the order of several milliseconds up to several seconds. The approaches in coping with the link or device failures depend on how stringent these timing requirements are. In addition, there is a question about the size of the communication network, which translates to whether the network can be local or not. In the next section, we discuss the applicability of these techniques in smart grids with more protocol details where needed.

2.2 Dealing with Component Failures in Smart-Grid Communication Networks

Table 2.2 gives an overview of applications found in smart grids and for which communication needs to be ensured. For the non-critical traffic (e.g., regular software update), a possible approach is to use TCP protocol at the transport layer, in combination with some mechanisms for connectivity reestablishment (e.g., RSTP and OSPF). This process can be lengthy, as a routing-protocol convergence-time can be in the order of minutes. But the real challenge is meeting the timing requirements for the applications that can be found in smart grids with very hard delay-constraints (e.g., protection). This is why the Multiprotocol Label Switching - Transport Profile (MPLS-TP) [17, 24–27] is often referenced as one of the proposed technologies for smart-grid IP networks [2, 28, 29]. In particular, its protection-switching mechanism, with 50-ms reaction-time guarantee, is considered as the state-of-the-art approach in reaching the goal of highly available smart-grid communication networks. This mechanism pre-allocates alternative paths that are activated within 50-ms following a link or device failure, thus re-establishing end-to-end connectivity.

In addition, MPLS-TP is compatible with Integrated services (IntServ) architecture [30] that provides quality-of-service guarantees and is implemented through the resource reservation protocol (RSVP) [31]. Hence, traffic-priority policies, such as the one from Table 2.2, can be enforced with a goal of timely delivery of critical messages.

Software-defined networking (SDN) [18] is another proposed technology in the context of smart grids [32–37] for meeting both delay and quality-of-service requirements. It represents an alternative to traditional approaches to network control and management, such as the aforementioned MPLS-TP. Still, MPLS-TP (or some of its features) can be a basis for an SDN solution, as there are two approaches as to how SDN can be applied in smart-grid communication networks. In the first one, an SDN controller implements the full MPLS-TP specifications. To this end, there is a trend of adding more and more MPLS functionalities to the SDN architecture. This effort is evident from the evolution of the OpenFlow specifications [38] (OpenFlow protocol is an enabler of SDN).

The second approach to using SDN technology is to perform a higher level of customization and to selectively apply only the desired functionalities that may or may not be inspired by the MPLS-TP specifications. For example, authors in [37] demonstrate an addition of the network-protection support based on the bidirectional forwarding detection (BFD) protocol. This protocol is also used in MPLS-TP for link-failure detection.

PRP represents the state-of-the-art technology for exploiting network redundancy in substation local area networks (LANs) where operators have full control over the

design process. Therefore, dedicated and cloned networks can be built and, with the help of PRP, $0ms$-packet-repair time can be reached.

2.2.1 Inconsistencies Between Best Practices and Requirements

Reaching the goal of a reliable and secure smart-grid communication network is far from being straightforward, despite all the above-mentioned recommendations and our awareness of the requirements that need to be fulfilled when building such a network.

Perhaps the most obvious example, of where there is a gap between the requirements and the proposed solutions, is about meeting the delay requirements for time-critical smart-grid applications. Concretely, in Table 2.2, we can see that there are quite a few applications that have a delay allowance below $50ms$. Nevertheless, as mentioned before, many industry-oriented sources refer to MPLS-TP as the technology on which smart-grid communication networks should be based despite the (higher) $50ms$-switchover time, guaranteed by the protection-switching feature of MPLS-TP. A possible explanation for this discrepancy is in the fact that it is very tempting (and cost-effective) for vendors and equipment manufacturers to offer technologies and products that are conceived for applications in other fields with different requirements. Simply putting a smart-grid label on the existing products, however, is often not enough.

We find similar inconsistencies also in more academic-oriented sources. Link-failure or node-failure detection-mechanisms, proposed in the context of SDN-based communication networks, require time that ranges from several seconds [34, 35] to several tens of milliseconds at best [36, 37], thus also failing to meet stringent delay-requirements for time-critical smart-grip applications. Interestingly enough, when it comes to the failure detection in the SDN-based comunication networks, the best results are achieved when applying the bidirectional forwarding detection (BFD) mechanism used in MPLS-TP [37].

The only technology that guarantees $0ms$-packet-repair time is PRP. Nevertheless, PRP is conceived to support substation automation where LANs are dominant and, as such, is not applicable for IP networks that are a key element for smart grids. Still, PRP serves as an inspiration for solutions presented in this thesis (see Chapter 5).

The main focus of this thesis is to bridge the identified gaps with the objective of enabling the redundant communication-networks to meet the delay requirements for time-critical smart-grid applications.

2.3 Cyber-Security of Smart-Grid Communication Networks

When it comes to security, many problems that are present in other fields are appearing in smart grids as well. Hence, ideally, best-practice techniques, which are used to secure the ICT infrastructure in other fields, should be applied. In addition, there are references that focus on smart-grid security; probably the most extensive reference is the report of the US national institute of standard and technology (NIST) [39].

Nevertheless, due to some of the peculiarities of specific smart-grid systems or applications, it is not always possible to apply directly the recommended techniques. For example, due to stringent time requirements imposed on critical-messages delivery, providing authentication and confidentiality for such messages with the commonly-used transport layer security (TLS) protocol can be infeasible [40]. Another potential source of difficulties comes from the fact that in power systems it is not unusual to find installed in the field devices with very long expected lifetimes (sometimes even tens of years). Applying state-of-the-art security techniques to such legacy equipment can be simply impossible due to device limitations.

This being said, before opting for certain security solution all the specifics of the smart grid of interest need to be taken into account. Then, based on potential threats, suitable security measures should be determined. For example, an analysis of the potential threats to active distribution network is given in [41], where authors list possible attacks. This list includes unauthorized access to smart-grid devices, man-in-the-middle attacks following the intrusion in the communication channel, rogue-device installation, denial-of-service attacks and malicious software-patching. The authors also suggest a set of applicable security solutions and best practices.

3 EPFL-Campus Smart-Grid Communication Network

3.1 Introduction

In this chapter, we describe the communication network built as part of the EPFL-campus smart-grid pilot. The purpose of the pilot is the experimental validation of the concept of real-time state estimation for an active distribution network. The main infrastructure components are (1) phasor mesurement units (PMUs) that output physical quantities (such as voltage and phase) every $20ms$, (2) a dedicated communication network, and (3) electrical-network state estimation (SE) process for real-time monitoring, which also comprises a phasor-data concentration (PDC) process. In the following, we focus on the communication network that has been built. We give details about the network design (Section 3.2) that resulted from the system requirements.

During the equipment installation phase, and later on, during the exploitation of our network, we performed some experiments to define the operating region of the infrastructure that is based on the single-pair high-speed digital subscriber line (SHDSL) technology. We present the results of these experiments. They concern mainly the SHDSL infrastructure. In Section 3.3 we define the operating region of the SHDSL communication channel, and in Section 3.4 we detail experiments that reveal some limitations of the used DSLAMs, devices that concentrate SHDSL channels. The lessons learned from these experiments are important for the successful exploitation of the network, as concluded in Section 3.5.

3.2 Communication Network Architecture

As the controller relies on up-to-date state estimates, the communication network plays a crucial role in the operation of an active distribution network. It needs to ensure the delivery of PMU data within stringent time delays and with minimal loss to the central control point. Additionally, it needs to be immune to power cuts because, at

such critical moments, its availability is essential. For these reasons alone, a dedicated communication infrastructure is required (as typical communication networks are inoperative during blackouts). A dedicated communication network is preferred; also for security reasons.

As we built our dedicated network from scratch, we decided to use IPv6 rather than IPv4, in order to prevent future transition issues. We avoided expensive cabling deployments by re-using existing twisted pair cables, originally installed for telephony. These cables are passive and are star-wired from a central point, the "PBX room" in Figure 1.1, where the backup power is available. Communication over twisted pair cables uses the single-pair high-speed digital subscriber line (SHDSL) technology, as the cables that are in place are too long for Ethernet. Traffic from all PMUs is concentrated at the SHDSL concentrators (called DSLAMs) located in the PBX room. This would also be the natural place to locate the phasor-data concentrator (PDC) and state estimator (SE) machines; however, we had only very restricted access to it. Hence, we had to place the PDC and SE machines in a more convenient location; for communications from the PBX room to PDC, we had to use more expensive optical fibers at 100 Mb/s, as the bitrate of SHDSL (2 Mb/s) would not have been sufficient here. The whole network is resilient to up to 8 hours of power outage; it is traffic-engineered to ensure enough capacity for the generated traffic.

The entire communication network is in fact duplicated, as seen in Figure 3.1. We developed an IP version of the parallel redundancy protocol, called iPRP (presented in Chapter 5); it takes care of duplicating UDP packets (at PMUs) and removing duplicates (at the PDC). This provides 0-ms repair of packet losses. We implemented iPRP as a transport-layer solution in the Linux operating systems of the PMUs and PDC; this has the benefit of not requiring any changes to any PMU/PDC applications or to any network devices. Raw measurements and state estimator outputs are made public through a web interface[1]. Every stored file contains one hour of data; it is a self-explanatory text file that is digitally signed.

[1] http://smartgrid.epfl.ch

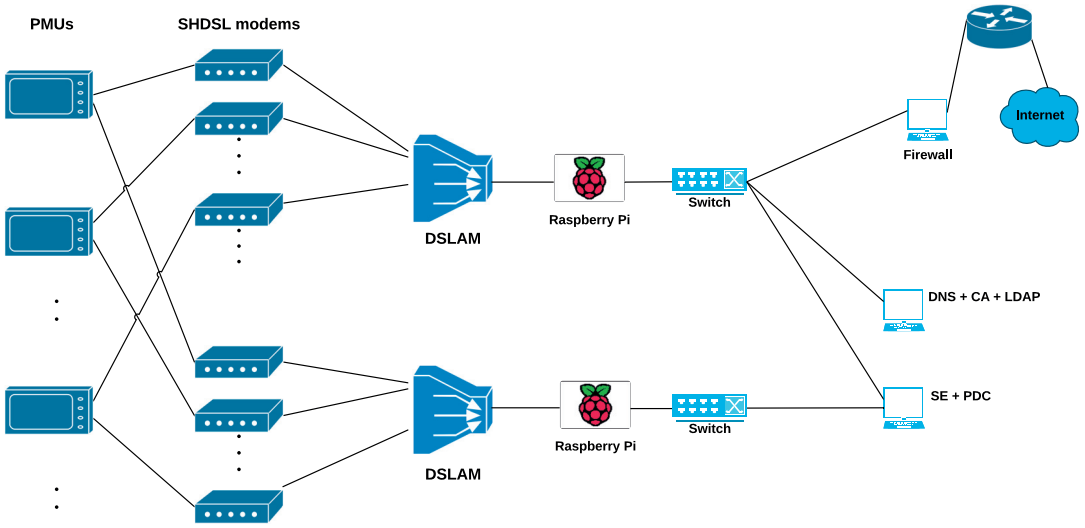


Figure 3.1: EPFL-campus smart-grid communication network.

3.3 Defining the Operating Region of the SHDSL Communication Channel

To understand the physical-layer limits of the SHDSL communication channels that are to be deployed, we investigate the behavior of the network in situations of congestion when the load is slightly above the capacity limit for short periods of time, because non-determinism that characterizes software used in network components can cause unexpected behavior. For example, packets containing measurement results could be backlogged due to a software update or a scheduled security session key replacement. As a result, there occurs a burst of packets that, for a short period of time, requires a larger capacity than available (which implicitly results in packet losses). Nevertheless, even when packet losses are inevitable, we need to ensure that they are not excessive. In this section we show that with off-the-shelf equipment excessive packet losses are possible in some practical scenarios. Hence, as part of the design phase we need to specify under which conditions this can happen, i.e., we need to define the operating region of the whole system.

Our approach was to begin by measuring the maximum throughput that can be achieved on an SHDSL communication channel. When we conducted further tests

they led to surprising results. We observe that when the PDC data transmission rate slightly exceeds the capacity of the SHDSL link, even for a short period of time, goodput drops catastrophically. Specifically, we measure a maximum achievable goodput of 1.98Mbps at the destination (the capacity of the SHDSL link). However, when the PDC generates data at 2.2Mbps, we obtain drastically different goodput at the destination, depending on the frame size: For a frame size of 2178, the goodput is 1.25Mbps, i.e., 37% loss, whereas for a frame size of 1815B, the goodput is 850kbps, i.e., 57% loss.

We explain this phenomenon by the combination of two separate factors: IP fragmentation that splits each frame in two IP packets with very different sizes, and FIFO/tail-drop queuing discipline within line terminal devices at the source end. We conclude that following the guidelines from C37.118.2-2011 standard [42] is not sufficient for designing a PMU data transfer layer. This comes from the fact that the standard defines messaging including types, use, contents and data-formats, whereas traffic management issues are out of its scope. At the end of this chapter we give some guidelines on how the problem can be mitigated by implementing traffic shaping within PMUs.

3.3.1 Experimental Setup

The experimental setup is shown in Figure 3.2.

We use ZyXEL SHDSL line terminals [43] that implement G.SHDSL.bis technology [44]. They are connected via a twisted-pair loopback circuit. The length of the twisted-pair corresponds to the typical distance between two end-points on campus. The two PCs run Ubuntu Linux. They are connected to SHDSL line terminals and are used for several purposes: to emulate PDCs/PMUs, to run tools like `iperf` and `ping`, or to run other custom applications written in C++ designed for a specific experiment. The connection between each PC and its line terminal is a 100Mbps Ethernet link. The line terminal forwards the data on the twisted-pair.

The very first experiments evaluate maximum available throughput that can be achieved, as well as the round-trip time (RTT). To this end we used the standard `iperf` and `ping` tools, respectively. On average we found a maximum available throughput of 1.98 Mbps and a RTT of 3.4 ms.

3.3.2 Discovering Line-Terminal Queue Size and Queuing Discipline

Due to the lack of documentation from the manufacturer of SHDSL line terminals, we performed experiments to discover the characteristics of the queues that are implemented within a device. This was necessary in order to understand the packet-loss patterns presented in the next subsection. The input interface capacity is roughly 50

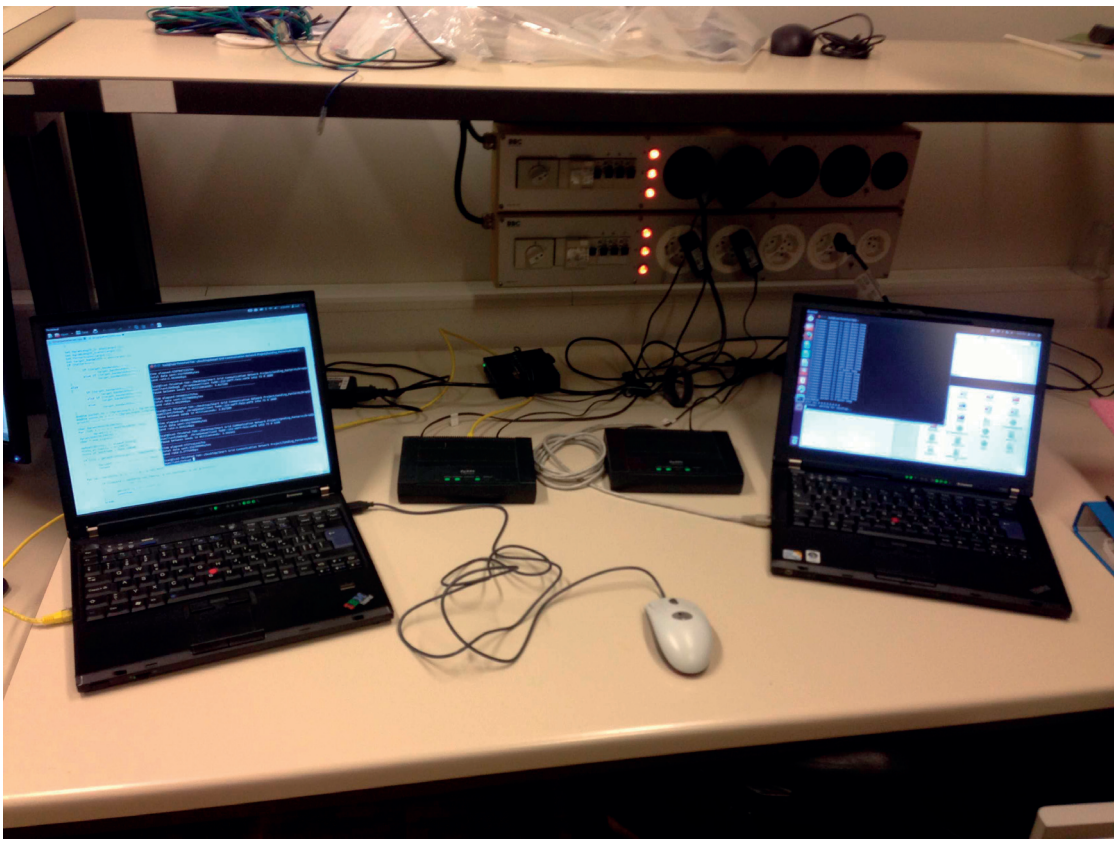


Figure 3.2: Experimental setup. SHDSL line terminals (black boxes in the middle) are connected by a twisted-pair loopback circuit.

times higher than the output interface capacity. The bottleneck where packets are dropped is a queue that corresponds to the outgoing interface of the line terminal (see Figure 3.3).

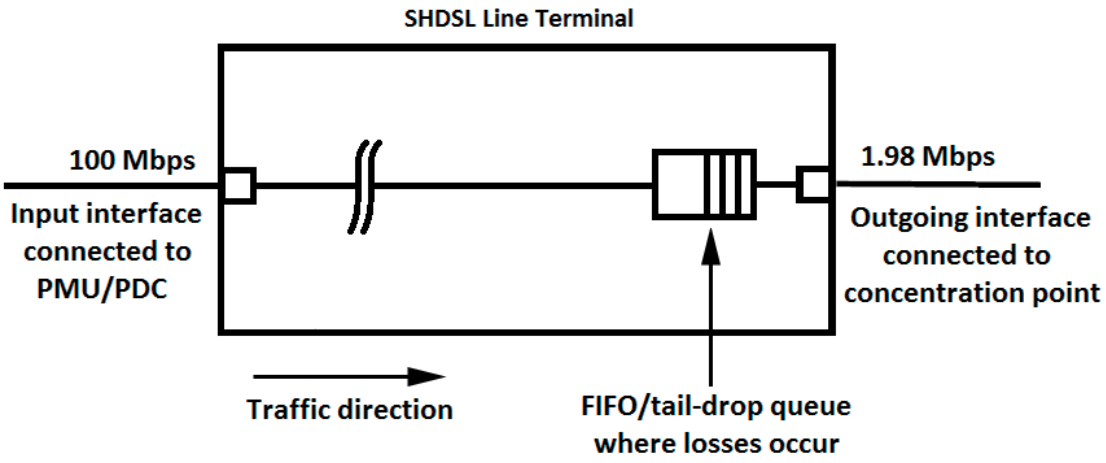


Figure 3.3: FIFO/tail-drop queue within SHDSL line terminal where losses occur.

We run a sequence of experiments to determine the queue size and the queuing discipline. On the sender side, we send bursts of packets, and, on the receiver side, we examine the received packets. We vary the burst size between 1 and 100 packets. We also vary the size of the payload (50 B, 500 B, 1000 B and 1452 B). Each packet sent contains a sequence number that is inspected at the receiving end. The results of these experiments are depicted in Figure 3.4.

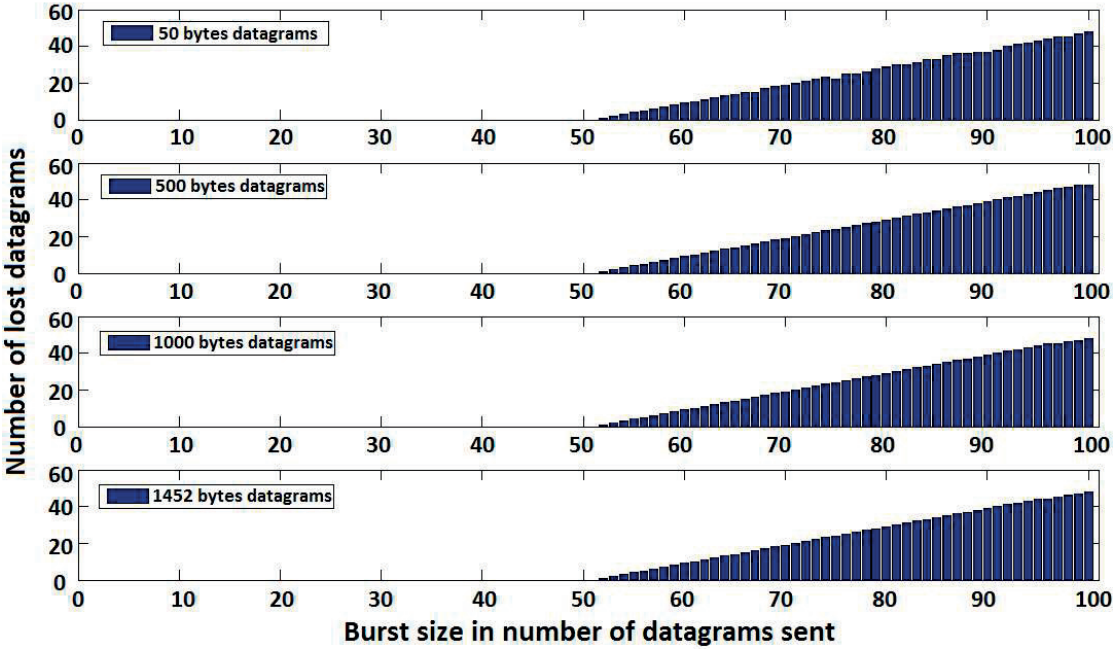


Figure 3.4: Number of lost packets for different datagram and burst sizes.

We observe that regardless of the packet size, the 52nd packet is always the first one to be lost. Furthermore, by analyzing the sequence numbers of the received packets, we observe that all the following packets are also lost. We conclude that the outgoing buffers associated with SHDSL line-terminal interfaces are implemented as FIFO queues with tail-drop queue management algorithm, with a queue size of 50 packets.

3.3.3 When Fragmentation Occurs, Information Loss Can Be Significantly above Expected Rate.

As mentioned in the introduction, it is important to investigate the behavior of the network in situations when the load, for short periods of time, is slightly above the link capacity limit. Although in this case packet losses cannot be avoided, we want to quantify how many packets are discarded.

The role of a PDC is to aggregate measurements - received from a number of

PMUs or other PDCs - to a single stream that is then forwarded to the control point. Measurements are correlated by time-tags. Depending on the number of streams that are aggregated, the resulting UDP datagrams could exceed the threshold size beyond which IP fragmentation occurs.

This size is dictated by the maximum transmission unit (MTU) of the underlying protocol. In our case, we use Ethernet that has a MTU of 1500 B. If the total size of the UDP payload, the UDP header (8 B) and the IPv6 header (40 B) is above 1500 B, IP fragmentation will take place, and one UDP datagram will be encapsulated in two (or more, if necessary) IP packets. At the receiver's side a UDP datagram can be reassembled only if all IP packets that carry its fragments are correctly received.

If any fragment is lost, the whole UDP datagram is considered to be lost as the transport layer is not able to reassemble the datagram. In a toy example in Figure 3.5, we compare the consequences of two different loss patterns that are due to two different scheduling strategies in the bottleneck queues. These are the best-case and the worst-case loss patterns. In both cases the same fraction of bits is discarded. In the best case the overall information loss equals the fraction of lost bits, whereas in the worst case all the information is lost.

This suggests that when UDP datagrams are fragmented into two IP packets, an optimal scheduler should not discard only one of the two corresponding IP packets. The transmission of the other would use network resources without any benefit, as it would be discarded anyway at the receiver side. Hence, if one of the UDP datagram fragments is discarded, the optimal scheduler would also discard all other IP packets that carry fragments of the same UDP datagram. In the case when the offered traffic is above line capacity, a lower bound on the resulting loss probability, is

$$\text{loss probability} \geq \frac{\text{offered traffic} - \text{line capacity}}{\text{offered traffic}} \tag{3.1}$$

In other words, if the offered traffic is only slightly above the line capacity, ideally only a small fraction of information should be lost.

We perform experiments to measure how close the system is to this optimal loss probability, given the equipment and technology at our disposal. The IEEE Standard C37.118.2-2011 states that a variable number of PMU measurements can be included within a single frame. We develop a PDC traffic emulator that enables us to vary the size of the packets on the sender side. We examine two sending patterns. *Sending Pattern I* (Figure 3.6(a)) mimics the scenario where no IP fragmentation occurs. One UDP datagram results in one IP packet, and there is always time spacing between two consecutive packets. Beginning with the target throughput, we calculate this time-

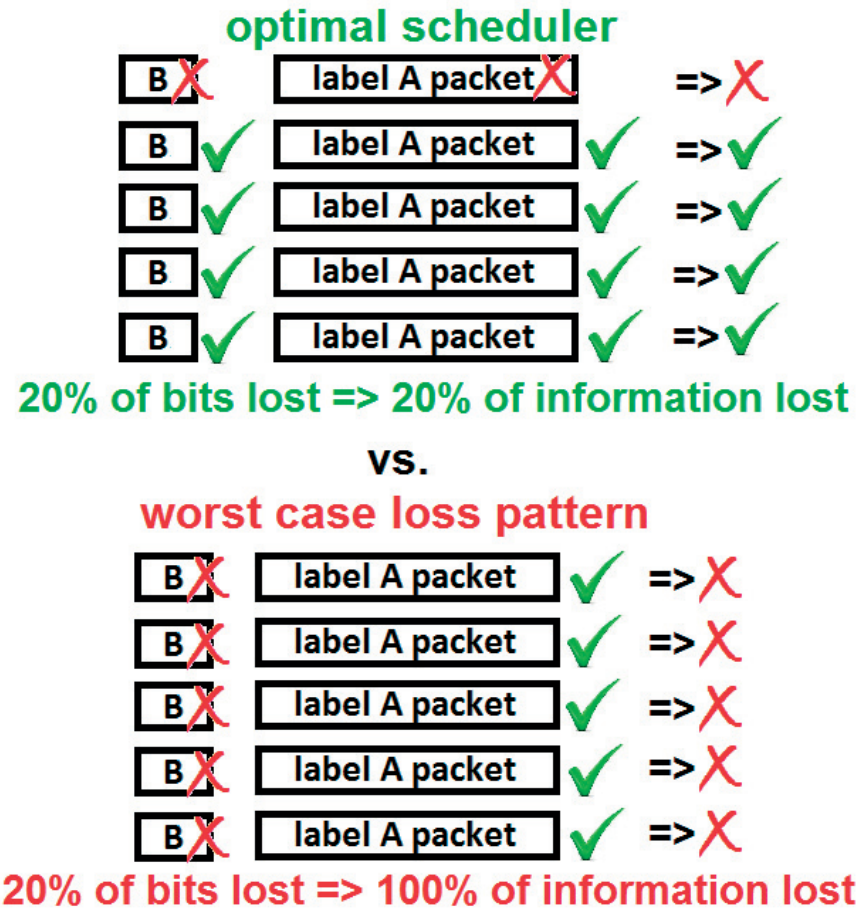
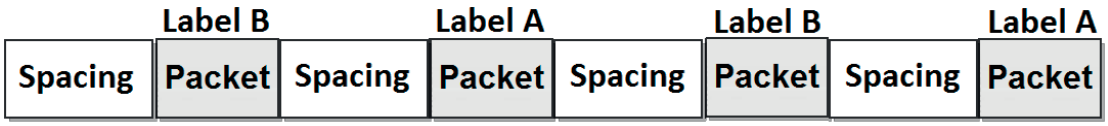


Figure 3.5: Toy example. Label A and Label B packets represent two fragments of the the same UDP datagram. Depending on the loss pattern the effect at the receiver might be dramatically different.

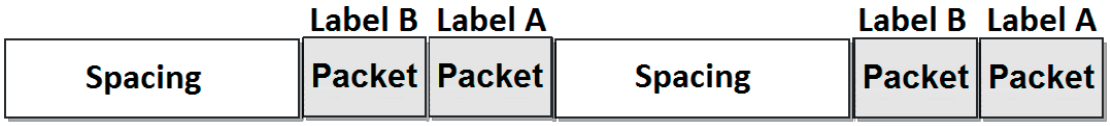
spacing duration between two consecutive packets. *Sending Pattern II* (Figure 3.6(b)) emulates the situation when IP fragmentation occurs. In this case, one UDP datagram results in two IP packets that are sent back-to-back. As in the first case, waiting times (now between groups of two IP packets) are calculated to meet the target throughput. In both cases, alternate packets are labeled with A or B; Label A packets are sent first, Label B packets follow (see Figure 3.6).

We are interested in the number of successfully received bytes at the destination end, as seen by the transport layer for various target throughputs and packet sizes. We keep track of the number and labels of lost packets.

We use the UDP transport-layer protocol because TCP retransmits lost packets, which is in our case superfluous as fresh measurements are generated at a high frequency. For every experiment, we send 10000 IP packets labeled with A and 10000 packets labeled with B and we specify sending pattern (I or II), size of packets labeled



(a) Pattern I (with spacing), Label A packet first then Label B packet.



(b) Pattern II (back to back), Label A packet first then Label B packet.

Figure 3.6: Sending patterns.

with A/B, and the target throughput.

We consider three datagram sizes (and thus implicitly three different label B packet sizes). Measurement results are depicted in Figures 3.7, 3.8, and 3.9. Each point is obtained via a single experiment. For each considered datagram size, on the left panel we plot the number of successfully transmitted Label A and B packets; and on the right panel, the number of successfully decoded bytes at the transport layer (UDP datagram payload). We also plot the curve that corresponds to the theoretical minimal-loss probability (labeled "optimal") in terms of number of bytes, as expressed in Equation 3.1. In the case of Sending Pattern I, measurement results for UDP datagram payload follow very well the optimal curve. However, for Sending Pattern II, we observe that the UDP datagram payload drops significantly. We notice that the number of Label B packets lost is also dramatically higher than for Pattern I.

3.3.4 Smaller Label B Packets Are More Likely to Be Discarded

Another observation we can make from Figures 3.7, 3.8, and 3.9 is that, when Sending Pattern II occurs, the smaller Label B packets are, the more of them we lose. For example, when the target throughput is 2.4 Mbps ($\sim 20\%$ over the SHDSL link capacity), and we set Label B packet size to 1452B, 726B, and 363B, we lose 37%, 56%, and 93% of Label B packets, respectively. This is contrary to what might be expected, as smaller packets require less time to be processed.

3.3.5 High Label-B Loss-Probability Leads to High Loss of Information

Keeping in mind the discussion about IP fragmentation and by examining the results presented in Figures 3.7, 3.8, and 3.9 when Sending Pattern II occurs, we remark the correlation between Label B packet loss and the significant drop in goodput. As a result, the number of UDP datagrams that can be used by the receiver drops dramatically.

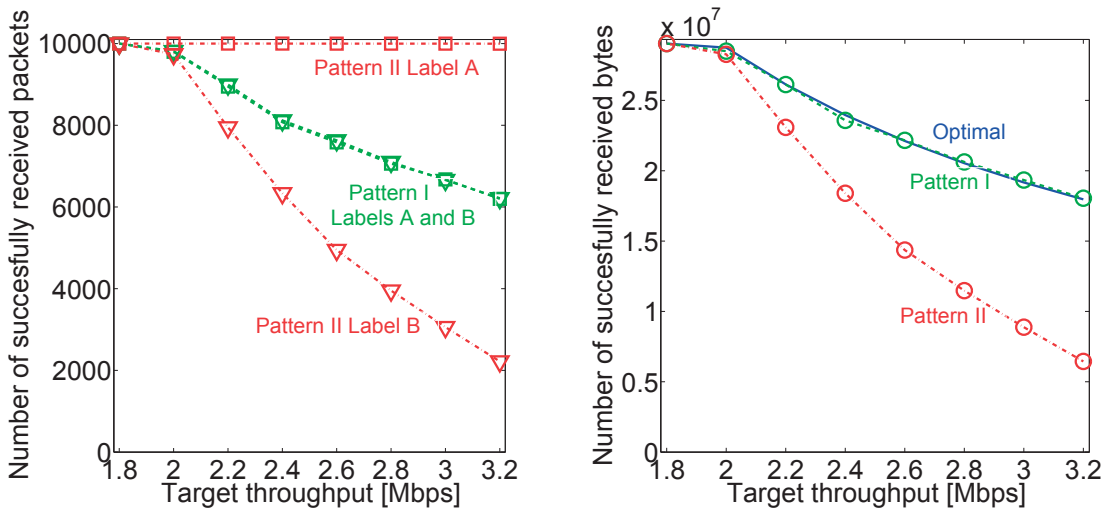


Figure 3.7: Experimental results. Label A packet size: 1452 B, Label B packet size: 1452 B. 10000 packets of each kind are sent.On the left: number of successfully received packets. On the right: number of successfully received bytes as seen by transport layer.

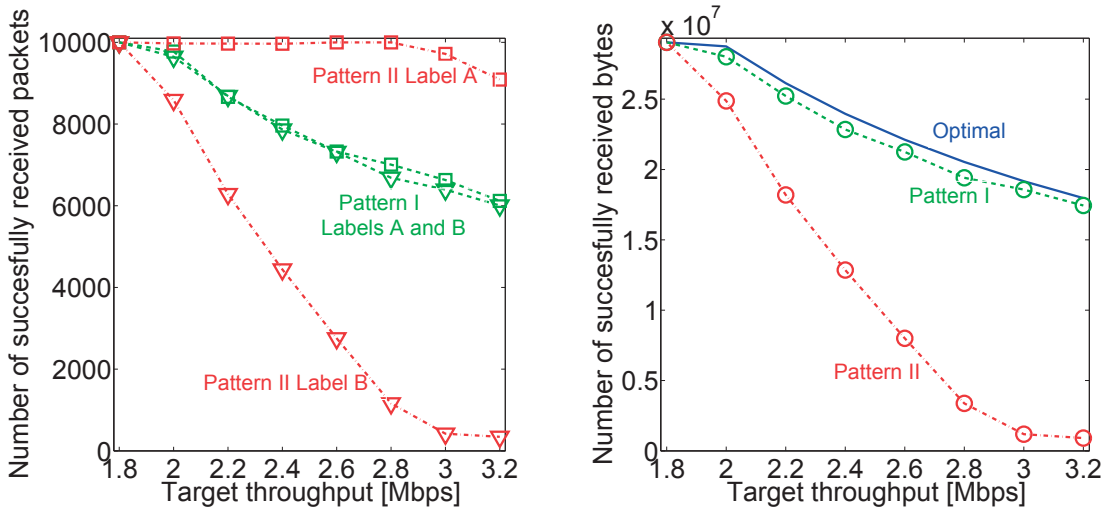


Figure 3.8: Same as Figure 3.7, except for Label B packet size: 726 B.

For example, when the target throughput is 2.4 Mbps ($\sim$ 20% over the SHDSL link capacity) and when Label B packet size is 363B, we can only retrieve up to 7% of the measurement results that are sent.

Next, for Sending Pattern II, we compare experimental results with simulation results, and we explain the phenomenon.

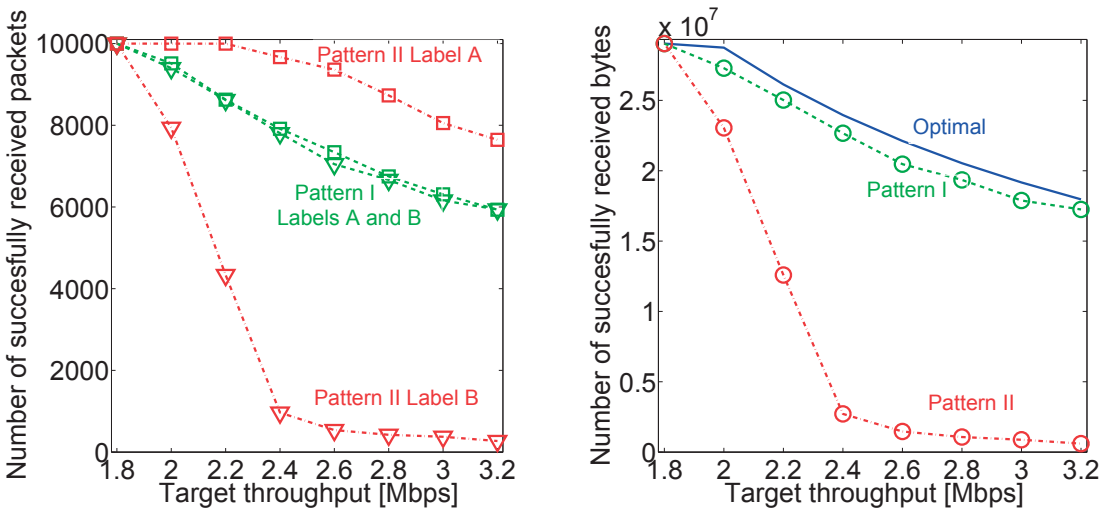


Figure 3.9: Same as Figures 3.7 and 3.8, except for Label B packet size: 363 B.

3.3.6 SHDSL Modem Simulations

In order to verify and explain the results detailed above, we design a simulation program. We use a discrete-event simulation to describe arrivals and departures from a queue that corresponds to the outgoing interface of the line terminal. Based on the experimental results from Section 3.3.2 we set the queue size to 50 and the queuing discipline to FIFO/tail-drop.

Simulation results match experimental results

Throughout this section, we analyze the results for Sending Pattern II because we identified previously that this sending pattern is the critical one.

As mentioned before, the purpose of having a simulator is to verify what we observed in our experiments. To this end we show in Figures 3.10, 3.11 and 3.12, a comparison of the results already shown in Figures 3.7, 3.8 and 3.9 with the results we obtain from our simulator for Sending Pattern II (back to back). We observe that, in all scenarios, measurement and simulation results are in accordance with each other.

Small buffers with tail-drop policies cause undesired loss patterns

We use our discrete-time FIFO/tail-drop simulator to better understand the nature of the phenomenon. We show in Figure 3.13(a), for a specific scenario (details in the figure caption), the evolution over time of the cumulative number of packets of each type that are successfully received when transmission follows Pattern II. The plotted

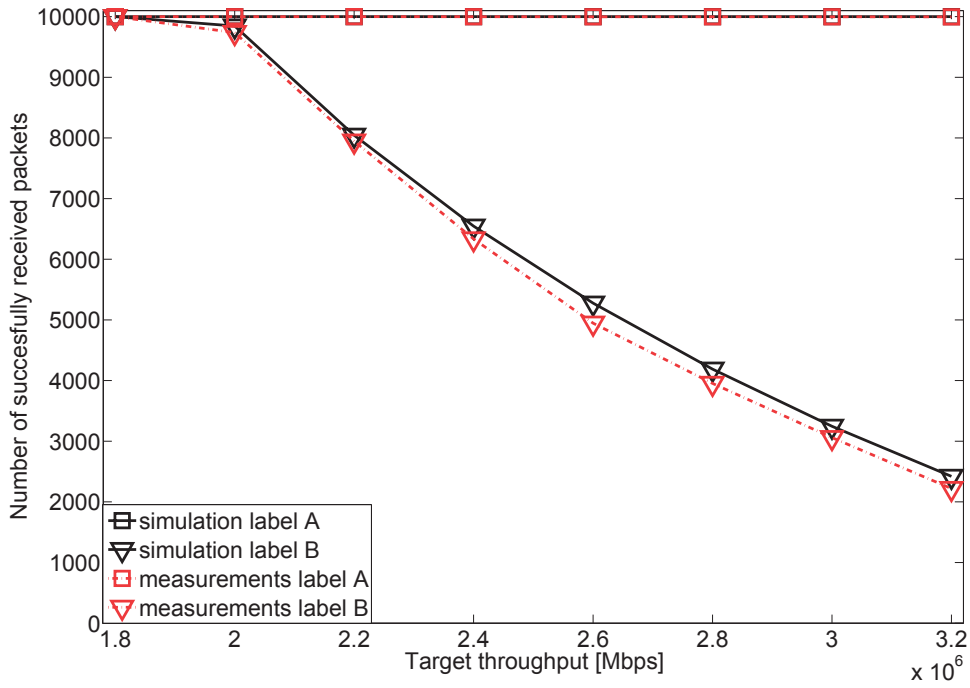


Figure 3.10: Simulation vs measurement results for Sending Pattern II (back to back). Label A packet size: 1452 B, Label B packet size: 1452 B. 10000 packets of each kind are sent.

trend is similar in all scenarios with sending Pattern II. We observe that, after an initial transient period, almost no Label B packets are successfully transmitted. Figure 3.13(a) also shows that, even when starting with an empty queue, it takes less then one second for Label B packets to start being excessively dropped. This shows that Label B packets are systematically discriminated against and that, after a short transient period, very few go through; whereas, Label A packets receive much better treatment.

In Figure 3.14 we show a possible occupancy of the queue at the arrival moment of two back-to-back packets. There is room for only one packet. The tail-drop queuing policy accepts the first packet and forces the second to be dropped. We conjecture that this is a typical situation (i.e., one free slot in the queue) and it results in Label B packets always being dropped as they follow Label A packets.

We use the same parameters to simulate a system, with a difference that we replace the tail-drop queuing policy with the datagram-aware scheduler: If a Label B packet cannot be accepted, the scheduler also discards the corresponding Label A packet, because they both carry fragments of the same UDP datagram. The expected effect is similar to the expected behaviour of an optimal scheduler that discards all the fragments of the same IP packet. We observe that the resulting effect corresponds to

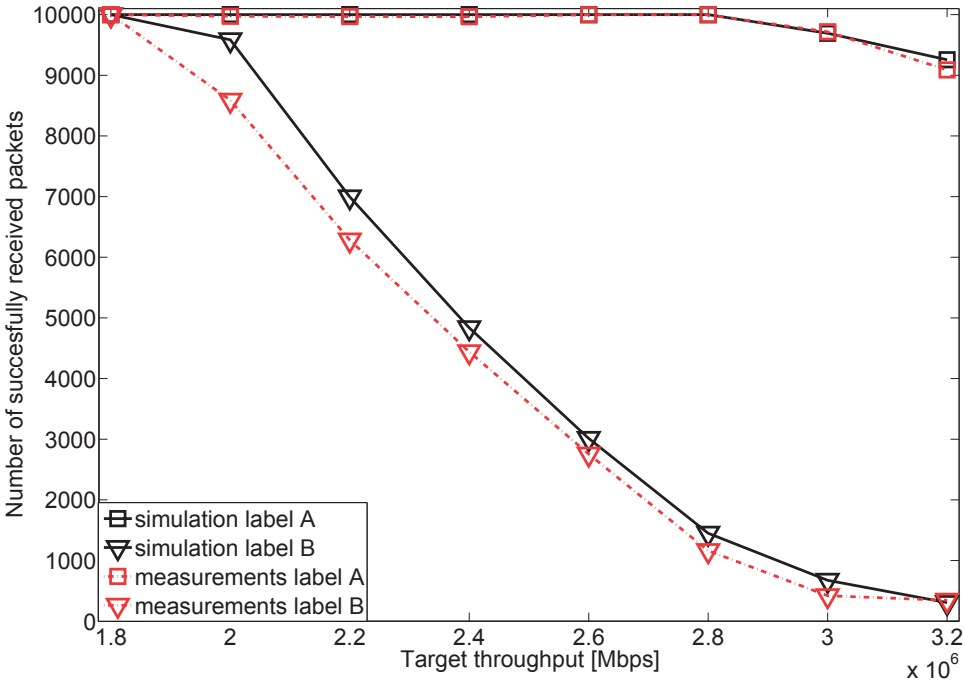


Figure 3.11: Same as Figure 3.10, except for Label B packet size: 726 B.

the theoretical optimum as expressed in Equation 3.1.

In Figure 3.13(b) we see the number of successfully received bytes as seen by transport layer. If we compare two scheduling policies, we observe that the undesired symptom disappears when we replace the tail-drop queuing policy with our datagram-aware scheduler. In the case of the tail-drop queuing policy, the receiver would be able to retrieve around 3% of the datagrams, whereas the datagram-aware scheduler ensures retrieval of around 75% of datagrams. We conclude that the tail-drop queuing policy (combined with a small buffer size) is indeed the cause of the undesired effect we observe.

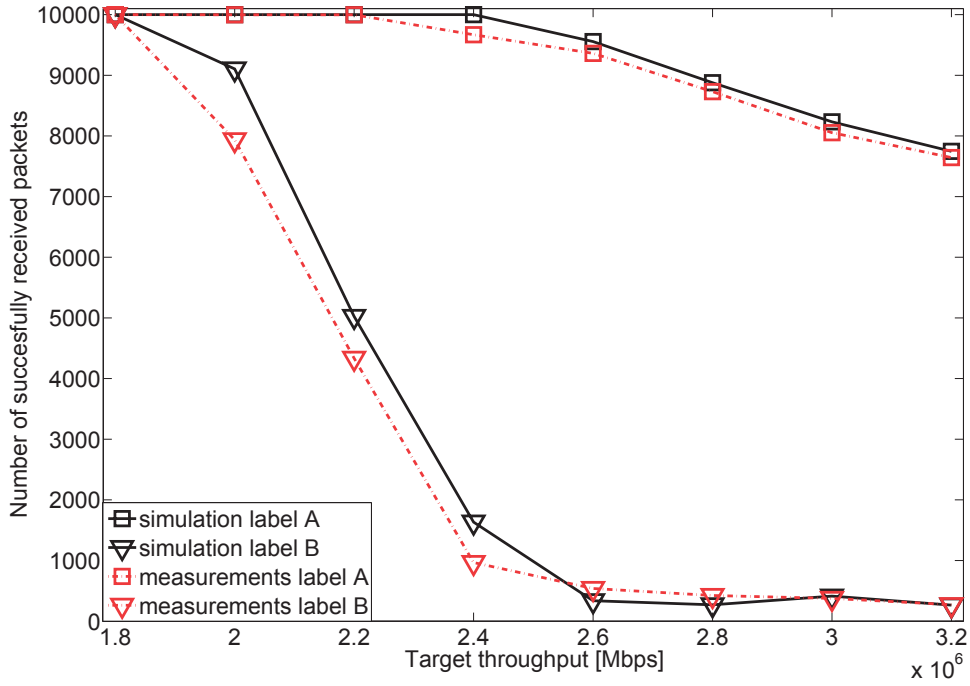
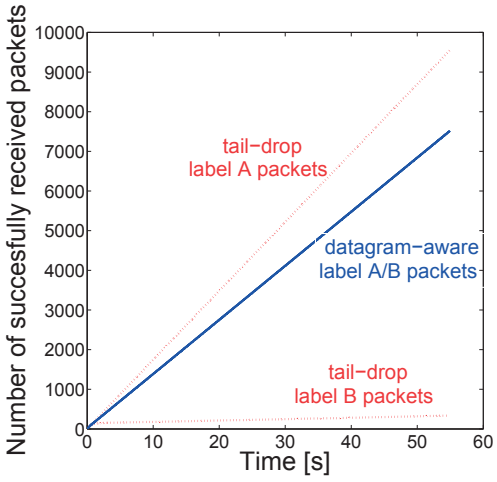
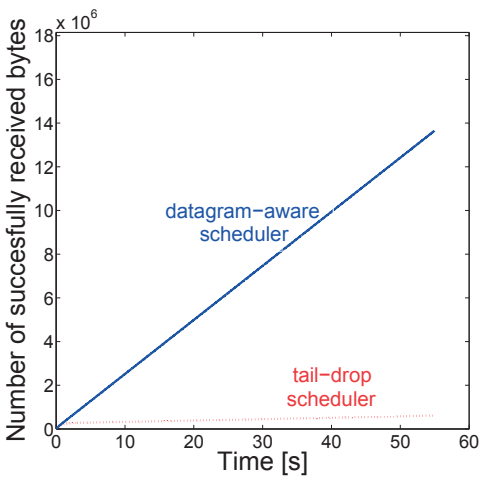


Figure 3.12: Same as Figures 3.10 and 3.11, except for Label B packet size: 363 B.



(a) Evolution of number of packets of each type that are successfully transmitted.



(b) Evolution of number of successfully received bytes as seen by transport layer.

Figure 3.13: Simulation results. Comparison of two scheduling policies. Datagram A size is 1452B, Datagram B size is 363, Target throughput is 2.6 Mbps, sending pattern is II.

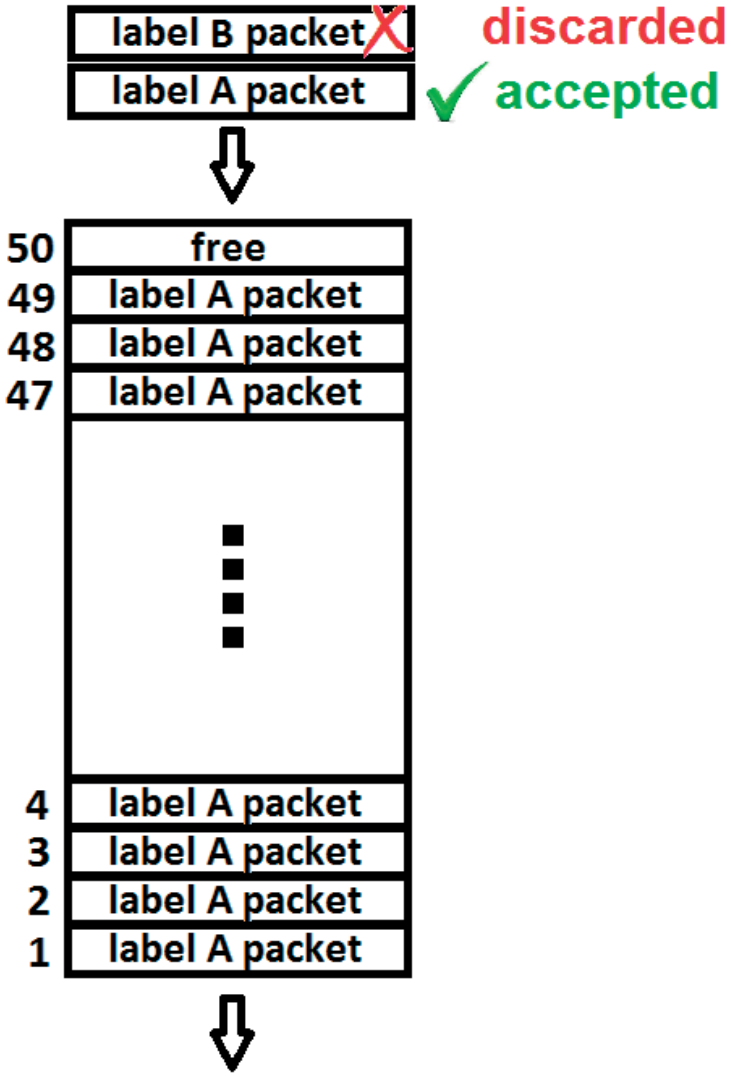


Figure 3.14: Conjectured typical situation at the FIFO/tail-drop queue that corresponds to the outgoing interface of the line terminal when the load is slightly above the available throughput and once the transient period is finished. Label A and Label B packets that arrive consecutively see a queue that can accommodate only the first one (Label A). This will result in almost no Label B packets in the queue, i.e., almost no Label B packets successfully transmitted.

3.4 Discovering Performance Limitations of DSLAMs

During the exploitation of our network, unexpected delay patterns were detected. In the following, we describe the observed problem.

PMUs are GPS-synchronized and they output measurement packets in the same moment. This is done periodically, every $20ms$, and it takes around $50ms$ from the

moment the measurements are taken until the moment the PDC processes them. Given that all the paths between PMUs and PDC are symmetrical, we would expect that the packets are delivered nearly at the same time. But, this is not what is observed at the PDC. Packets with the same time-stamp arrive with time-differences in the order of several milliseconds. Given that this jitter represents around 10% of the overall computation time, it is highly undesirable and, as it is unexpected, further investigation is required. For this analysis, we show a simplified version of the communication-network infrastructure in Figure 3.15.

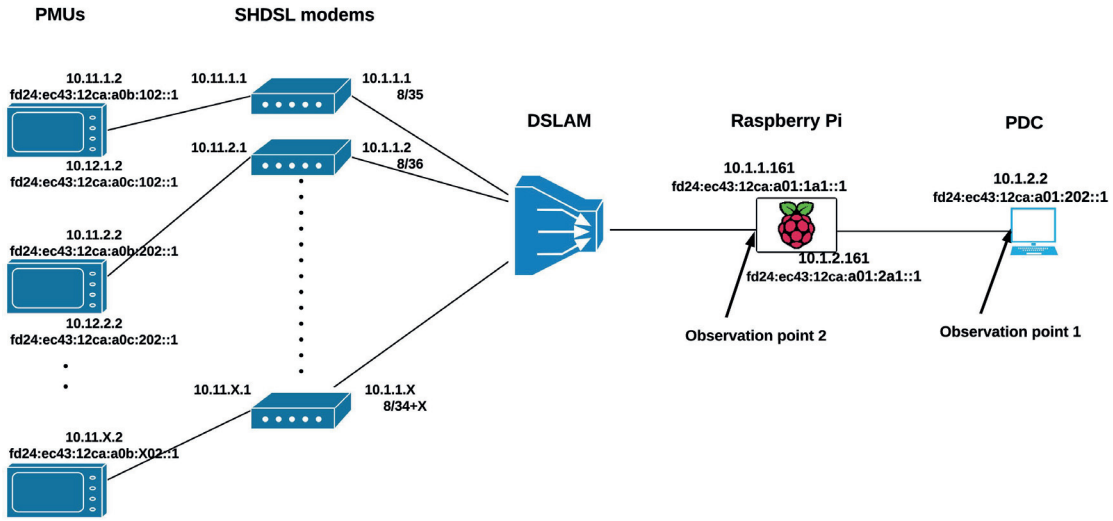


Figure 3.15: EPFL-campus smart-grid communication network - simplified representation.

3.4.1 Problem Quantification and Analysis

The very first step was to quantify what was observed. To that end we ran the `tcpdump` tool to capture traffic. We did it at the Obervation point 1 from Figure 3.15. This is the incoming interface of PDC. We see the (simplified) results below in Figure 3.16

There are several observations to make:

- There are 6 PMUs (2-7) that send traffic periodically. We distinguish them based on the source port $(43002 - 43007)$.
- All the packets in blue belong to the same measurement generation, i.e., they have the same time-stamp. This pattern repeats every $20ms$.
- Among the packets that belong to the same generation, we always observe packet from PMU2 as the first one, and packet from PMU7 as the last one. Packets from PMUs 3-6 are always in between packets from PMU2 and PMU7, with the order that varies.

```
17:30:23.720 IP6 fd24:ec43:12ca:a0b:202::1.43002 > fffe::2:  UDP, 74B
17:30:23.721 IP6 fd24:ec43:12ca:a0b:402::1.43004 > fffe::2:  UDP, 74B
17:30:23.722 IP6 fd24:ec43:12ca:a0b:602::1.43006 > fffe::2:  UDP, 74B
17:30:23.722 IP6 fd24:ec43:12ca:a0b:302::1.43003 > fffe::2:  UDP, 74B
17:30:23.725 IP6 fd24:ec43:12ca:a0b:502::1.43005 > fffe::2:  UDP, 74B
17:30:23.729 IP6 fd24:ec43:12ca:a0b:702::1.43007 > fffe::2:  UDP, 146B
17:30:23.740 IP6 fd24:ec43:12ca:a0b:202::1.43002 > fffe::2:  UDP, 74B
17:30:23.743 IP6 fd24:ec43:12ca:a0b:402::1.43004 > fffe::2:  UDP, 74B
17:30:23.743 IP6 fd24:ec43:12ca:a0b:602::1.43006 > fffe::2:  UDP, 74B
17:30:23.743 IP6 fd24:ec43:12ca:a0b:302::1.43003 > fffe::2:  UDP, 74B
17:30:23.744 IP6 fd24:ec43:12ca:a0b:502::1.43005 > fffe::2:  UDP, 74B
17:30:23.748 IP6 fd24:ec43:12ca:a0b:702::1.43007 > fffe::2:  UDP, 146B
17:30:23.759 IP6 fd24:ec43:12ca:a0b:202::1.43002 > fffe::2:  UDP, 74B
17:30:23.763 IP6 fd24:ec43:12ca:a0b:302::1.43003 > fffe::2:  UDP, 74B
17:30:23.763 IP6 fd24:ec43:12ca:a0b:602::1.43006 > fffe::2:  UDP, 74B
17:30:23.763 IP6 fd24:ec43:12ca:a0b:402::1.43004 > fffe::2:  UDP, 74B
17:30:23.765 IP6 fd24:ec43:12ca:a0b:502::1.43005 > fffe::2:  UDP, 74B
17:30:23.768 IP6 fd24:ec43:12ca:a0b:702::1.43007 > fffe::2:  UDP, 146B
17:30:23.780 IP6 fd24:ec43:12ca:a0b:202::1.43002 > fffe::2:  UDP, 74B
17:30:23.782 IP6 fd24:ec43:12ca:a0b:602::1.43006 > fffe::2:  UDP, 74B
17:30:23.782 IP6 fd24:ec43:12ca:a0b:402::1.43004 > fffe::2:  UDP, 74B
17:30:23.782 IP6 fd24:ec43:12ca:a0b:302::1.43003 > fffe::2:  UDP, 74B
17:30:23.785 IP6 fd24:ec43:12ca:a0b:502::1.43005 > fffe::2:  UDP, 74B
17:30:23.788 IP6 fd24:ec43:12ca:a0b:702::1.43007 > fffe::2:  UDP, 146B
```

Figure 3.16: Captured traffic at Observation point 1.

- Packets from the same generation are arriving with a time-window of around $8ms$.

We calculate the observed throughput as follows. After adding $62B$ of headers to $74B$ of payload for PMUs 2-6 and $146B$ of payload for PMU7, we have $888B$ in total - processed within $8ms$. The observed throughput is then approximately $1Mbps$. Given that all the interconnecting links after DSLAM, which is the concentration point, have capacities far above what is observed, we suspect that there might be a bottleneck in one of the two devices that are shared by packets from all PMUs: DSLAM and Raspbery Pi. Hence, we proceed by analyzing the throughputs supported by them.

Analysis of Raspberry Pi: Raspberry Pi is not intended to be used for high-performance routing. In addition, in our setting, it is configured as an end-point of multiple IPv6-in-IPv4 tunnels, so we suspect it can be the bottleneck. We conduct two simple experiments. In the first one, we analyze the throughput of Raspberry Pi when routing only IPv4 packets. The measured throughput was around $50Mbps$. Then we put additional load by sending IPv6 packets. This way, we forced Raspberry Pi to perform additional

processing due to IPv6-in-IPv4 tunneling, which is the usual workload in our network. The measured throughput was around $30Mbps$. In both cases, we used `iperf` as a tool for packet generation and bandwidth measurement.

Hence, the conclusion is that Raspberry Pi is not the bottleneck, as the measured throughput is well above $1Mbps$, which is the observed bottleneck throughput.

Analysis of DSLAM: Unlike with Raspberry Pi, which is an open Debian-based system, we have less control over DSLAM and its internals. Hence, we need to analyze this device as a black-box. To this end, we run `tcpdump` at the Obervation point 2 from Figure 3.15. This is the incoming interface of Raspberry Pi. We see the (simplified) results below in Figure 3.17.

```
16:18:46.219 IP6 fd24:ec43:12ca:a0b:202::1.43002 > fffe::2:  UDP, 74B
16:18:46.222 IP6 fd24:ec43:12ca:a0b:602::1.43006 > fffe::2:  UDP, 74B
16:18:46.222 IP6 fd24:ec43:12ca:a0b:402::1.43004 > fffe::2:  UDP, 74B
16:18:46.222 IP6 fd24:ec43:12ca:a0b:302::1.43003 > fffe::2:  UDP, 74B
16:18:46.223 IP6 fd24:ec43:12ca:a0b:502::1.43005 > fffe::2:  UDP, 74B
16:18:46.227 IP6 fd24:ec43:12ca:a0b:702::1.43007 > fffe::2:  UDP, 146B
16:18:46.238 IP6 fd24:ec43:12ca:a0b:202::1.43002 > fffe::2:  UDP, 74B
16:18:46.241 IP6 fd24:ec43:12ca:a0b:602::1.43006 > fffe::2:  UDP, 74B
16:18:46.241 IP6 fd24:ec43:12ca:a0b:402::1.43004 > fffe::2:  UDP, 74B
16:18:46.242 IP6 fd24:ec43:12ca:a0b:302::1.43003 > fffe::2:  UDP, 74B
16:18:46.244 IP6 fd24:ec43:12ca:a0b:502::1.43005 > fffe::2:  UDP, 74B
16:18:46.247 IP6 fd24:ec43:12ca:a0b:702::1.43007 > fffe::2:  UDP, 146B
```

Figure 3.17: Captured traffic at Observation point 2.

All the conclusions from the analysis above from `tcpdump` output at Observation point 1 still hold. As the DSLAM is the first device common to all the paths between PMUs and PDC, it represents the bottleneck.

3.5 Conclusion

In this chapter, we have presented the design of the EPFL smart-grid communication network and the results of the experiments performed on its infrastructure. When designing the first set of experiments, to test the operating region of the SHDSL communication channel, we followed data transfer requirements described in the IEEE C37.118.2-2011 standard. However, we were able to identify critical scenarios where following these guidelines is not sufficient.

We used off-the-shelf SHDSL line terminals. When IP fragmentation occurs, two back-to-back packets arrive at the line terminal. When the reception rate exceeds the

transmission rate, the packet queue is often full, and one or both incoming packets are dropped, thus rendering the reassembly impossible. In some of the considered scenarios, we find that packets arriving first are almost never lost, whereas a high loss-rate of the packets arriving second dictates the drop in goodput. We attribute this asymmetric behavior to the tail-drop queuing policy and small size of the queues implemented inside line terminals.

The resulting effect is that, even if the load is just slightly above the capacity of the SHDSL link, the goodput drops catastrophically. For example, when the offered load is 2.2Mbps, depending on the size of the packets, there are between 20% and 57% of lost Label B packets, although the capacity is exceeded by only 10%. When the capacity is exceeded by 20%, there are between 37% and 90% Label B packets that are lost. Thus, if measurement results are sent in fragmented packets, there is a risk that there is at least the same fraction of lost measurement results as the fraction of Label B packets that are lost. Hence, the effect on our monitoring system would be severe.

There are two different approaches for mitigating the aforementioned issues. First, a quick fix is to implement datagram-aware schedulers for the bottleneck queues (we have shown the benefits of this solution in Figure 3.13).

In the second approach, if we want to avoid losses altogether, PMUs/PDCs should implement traffic shaping on machines that run real-time operating systems (Real-Time Linux [45] for example). This would guarantee enough resources for processing and sending/forwarding packets within the desired boundaries, and packets would never be backlogged.

In the second set of experiments we conclude that DSLAM (ZyXEL SAM1316-22) is the bottleneck in the system. It causes that the packets with the same time-stamps arrive at the destination with time differences larger than expected. It is specified in the device documentation that the outgoing interface is compliant with 10/100Base-TX standards; hence, it is able to output packets at 10/100Mbps. Nevertheless, switching-fabric throughput is not specified in the document and we suspect that this represents the bottleneck in our system. Therefore, it needs to be ensured that the traffic offered by PMUs does not exceed 1Mbps, which is the throughput that can be supported by the DSLAMs. Such a result is surprisingly bad and, in retrospect, this device probably should have been avoided.

In the analysis of the identified problem, we use an assumption that the bottleneck is located after (or within) the concentration point. Theoretically, it could be that some PMUs or SHDSL modems need more time for processing and, as a consequence, packets from those PMUs arrive later at DSLAM. We consider this option to be not very likely (but we still keep it in mind), because the packet that is always observed first (PMU2) comes from the measurement point PC-2 that is the closest to the CM building

where DSLAM is installed. This means that the connecting cable is the shortest one for this measurement site, which translates to the shortest propagation time; this is the explanation for why packets from PMU2 are always observed first.

For the final recommendation after describing both sets of experiments, we conclude that a traffic management mechanism (currently out of scope of IEEE C37.118.2-2011 standard) is necessary if we want to ensure that networking resources are sufficient. A possible solution is the integrated-services architecture (IntServ) [30] that would guarantee Quality of Service. A sending device should go through a setup phase for resource reservation. During this phase all intermediate networking devices accept or reject the reservation depending on available resources and the traffic specifications advertised by the sender, namely maximum packet size, peak rate, burst tolerance, and sustainable rate.

If we implement traffic shaping mechanism on devices with real-time operating systems with the integrated services network architecture, the system becomes much more deterministic and packet losses are avoided.

4 Security Vulnerabilities of the Cisco IOS Implementation of the MPLS Transport Profile

4.1 Introduction

When studying technologies for smart-grid communication networks, it immediately becomes evident that the MPLS Transport Profile (MPLS-TP) is one of the key technologies [28]. Hence, when we began working on the topic, the natural first step was to evaluate MPLS-TP in detail. Therefore, we conducted a protocol analysis in our custom-made testbed, during which some security vulnerabilities emerged. In this chapter, we report on our findings about the attacks that are possible in MPLS-TP-based communication networks.

In smart grids, MPLS-TP is mainly used for inter-control center communication of measurement data (e.g, synchrophasor data from Phasor Measurement Units), and control and protection commands over WANs. Given the critical nature of these type of communications for the reliable operation of a smart grid, the communication infrastructure is required to satisfy high availability with bounded delay. MPLS-TP satisfies these requirements in that it supports traffic engineering to guarantee deterministic delay for high priority traffic, and it provides end-to-end protection - ensuring network reliability and high availability. End-to-end protection is achieved by the MPLS-TP OAM (Operations, Administration and Maintenance) framework that provides protection switching feature, controlled by bidirectional forwarding detection (BFD) and protection state coordination (PSC) protocols. BFD detects failures in label-switched paths (LSP), and PSC coordinates the protection switching.

The fact that an MPLS-TP network extends over a (usually unprotected) wide area renders the communication network vulnerable to cyber intrusions by an attacker with a malicious intention of compromising the smart grid's operations. Hence, one of the major challenges for a smart grid utility is to implement proper cyber security

protection methods for its MPLS-TP based WAN.

There is a whole family of MPLS-TP-related RFCs [17, 24–27] and several among them are security-related. However, we find that the RFC-based security analysis of MPLS-TP is complex due to fragmentation of pieces of information needed to understand the big picture of which the pieces are spread among multiple RFCs. To some extent, security-related problems are treated as “hot potatoes"; the responsibility of securing different aspects of MPLS-TP in different RFCs is sometimes outsourced to another RFC some of which are only informational and majority of which is without straightforward guidelines. Consequently, the lack of holistic approach in securing the MPLS-TP network makes the whole process difficult to follow. For example, RFC 5085 [27] relies on IPsec to provide the security of MPLS-TP, but this is not always applicable as there are some non IP flows in some smart grid and other contexts that use MPLS-TP. This led us to pose as a hypothesis that vendor solutions might not correctly implement all of the required security. To test this hypothesis, we started an analysis of several MPLS-TP implementations. In this chapter, we report on our findings with the Cisco IOS implementation, Cisco being one of the leading manufacturers of network equipment and with a large investment in smart grids.

In Section 4.2, we give an overview of the MPLS-TP features that are necessary to understand the experiments we conduct. In Section 4.3 we describe the testbed, that we build in the lab environment: it consists of eight virtual routers each running a Cisco IOS image that supports MPLS-TP.

In Section 4.4, we describe how we were able to conduct the attacks. This was possible in spite of us implementing the security guidelines that are available from Cisco for IOS and MPLS-TP (which include IPSec and authentication of the control plane). In some scenarios (not reported here) the available security measures were sufficient. However, we identified other scenarios, described below, where these security measures were not sufficient. In these latter scenarios, the attacker harms the network at several points (e.g., disabling both working and protection LSPs) by using the access to only one or two interfaces of a switch (e.g., on the protection LSP). We achieved this by inserting forged BFD or PSC messages into the network, which induces the label edge router (LER) into believing false information about the LSP status. In two attacks, the LER disables the operational LSP. In another attack, the LER continues to believe that a physically destroyed LSP is up and running. These spoofing attacks rely on the assumption that an attacker gains physical access to cables in the network. This assumption is consistent with reality because it is common to find several unmanned facilities, such as remote substations, in smart grid networks. Such substations are where fiber-to-copper converters or optical terminators of an MPLS-TP network likely reside. Moreover, some smart grid utilities also deploy pole-mounted optical repeaters at every few kilometres along their electric transmission lines. Gaining physical access to such physically exposed locations is usually achieved by breaking a window in a

substation or climbing up a pole.

In addition to the ease of physical access to the attack locations, the equipment required to mount the spoofing attacks is rather affordable. In MPLS-TP networks where only optical links are used (no copper) an attacker can afford a NetFPGA-10G card [46] with SFP+ modules in order to gain access to the network. Besides, it is not uncommon to find copper cables between optical termination nodes (fiber-to-copper converters) and MPLS-TP routers (see Figure 4.4). The cost of equipment required to launch a spoofing attack in such networks is almost negligible. An attacker can use an off-the-shelf switch to connect his laptop computer to the network at the copper cable segments. The attacker only injects very modest amounts of bogus traffic from his laptop computer and does not need to intercept legitimate traffic, which remains untouched. As such, our described attacks require only low-end, cheap equipment that is readily available in all consumer electronics shops. In our experiments, we used the second method to access the network in order to demonstrate the spoofing attacks discussed in this chapter. Note that the nature of attacks remains the same no matter which method of gaining access to the network is used.

In two of our attacks, the attacker needs to access only one cable; in another one, he needs access to two cables. In all cases, an attacker gains more power to damage the physical electrical infrastructure as a result of a more intelligent communication network, i.e., an attacker can now bring down a physical target in the power grid from a remote location by selectively manipulating traffic at only one, or in some cases two, locations in the communication infrastructure. In conventional power grid networks, an attacker would have required physical access to sabotage the target. If we want to compare the attacks described here to the one that involves simple wire cutting, we can see that cutting wires harms only LSPs directly affected by the cut whereas we show that our attacks are more powerful (LSPs in the other parts of the network are also affected). In some sense, this is in contradiction with the expected benefits of smart grids: they should not make the electrical systems weaker than they are today. In Section 4.5 we discuss countermeasures that are likely to thwart the described attacks.

4.2 MPLS-TP Protocol Overview

A label-switched path (LSP) is defined as a one-way path that data follows from one particular node (a router able to do label-switching) to a different node, where intermediate nodes can be traversed. The two nodes where data enters and leaves the LSP are called label edge routers (LER), and nodes traversed within the LSP are called label-switching routers (LSR). MPLS-TP mandates that all LSPs go by pairs traversing the same nodes and links on each direction, and that they be signaled as a single entity (one single LSP identifier is used); this characteristic is called co-routed bidirectional.

In MPLS-TP, the control plane for signalling and recovering LSPs can be static or dynamically configured. In smart grids, a static configuration is commonly used by network operators for security reasons, to avoid interacting with dynamic control protocols (i.e., RSVP-TE, T-LDP) from other service providers; as we are in the context of smart-grid communication networks, we use static configuration of MPLS-TP LSPs for our analysis.

Recovery is the ability of the network to become operational following the failure or degradation of traffic delivery caused by a network fault or a denial-of-service attack on the network [25]. There are several types of recovery methods in MPLS-TP, and in our testbed we analyze protection switching. Protection switching is a well-suited method that provides fast repair and exists side-by-side with the static configuration of the control plane commonly used in smart grids. This method pre-allocates an alternative bidirectional LSP to divert traffic during a fault condition, uses BFD for link failure detection with an almost immediate response time (i.e., less than 50ms) and has a mechanism for coordinating the state of the protection provided to a working LSP between both LERs. Within the protection switching scope, a working LSP is defined as the LSP where data runs under normal operating conditions; furthermore, a protection LSP is the pre-allocated LSP where data is diverted from the working LSP in case of network failure or degradation. The service delivered by both working and protection LSPs is called an “MPLS-TP tunnel".

There are two different schemes for providing protection switching to an LSP. The first scheme is called “1+1", where the label edge router at the ingress of the MPLS-TP tunnel transmits simultaneously the data on both working and protection LSPs, and the label edge router at the egress of the MPLS-TP tunnel selects between working or protection LSPs based on some predetermined criteria. The second scheme is called “1:n" where the working LSP handles data under normal operating conditions; and only if there is a defect, failure, degradation or request from network operator, the traffic is switched to protection LSP. For simplicity, in this chapter we discuss a particular case of the second scheme called “1:1" where one pre-allocated protection LSP serves one particular working LSP. The nature of the attacks we conduct is such that the security weaknesses revealed in our testbed can be exploited in schemes “1+1" and “1:n" as well, with exactly the same results.

Ver	Diag	Sta|P|F|C|A|D|M	Detect Mult	Length
My Discriminator				
Your Discriminator				
Desired Min TX Interval				
Required Min RX Interval				
Required Min Echo RX Interval				
Auth Type	Auth Length	Authentication Data		

Figure 4.1: Format of a BFD Control Packet.

4.2.1 Bidirectional Forwarding Detection (BFD)

BFD in MPLS-TP is a protocol that detects link failures within the MPLS-TP tunnel. From the three types of messages described in [47], our focus is on the BFD control packet that we are spoofing in our attack. The BFD control packet verifies the continuity of an LSP. A BFD session is established between label edge routers for each LSP within the MPLS-TP tunnel. A LER sends a BFD control packet every 3.3 ms (this is the interval recommended by IETF [25]); the BFD control packet is sent in-band in the LSP (i.e. is switched at intermediate routers exactly like data packets in the LSP) and is intercepted by the LER that terminates the LSP. When an LER observes that 3 consecutive BFD control packets are not received, it declares the incoming LSP to be broken; as the two directions of co-routed bidirectional LSPs can fail independently, the receiving LER sends a BFD “Session Down" message in the reverse direction of the LSP to inform the LER at the other end of the failure; then protection switching is triggered. In Cisco’s implementation BFD control packets are sent every 4 ms instead of 3.3 ms.

In order to understand the attacks described in section 4.4, we include the format of a BFD control packet in Figure 4.1 and provide relevant information for the fields concerning our attacks. The first of two fields for our testbed is the diagnostics (*Diag*) field, which is five bits long and reports a fault or defect condition between label edge routers; from the 32 possible codes we are interested in two of them, a 0 means there is no fault or defect condition to report and a code 1 stands for “Control Detection Timer Expired", which in normal operating conditions is set when we miss three consecutive BFD control packets. The second field is the state (*Sta*) field, which is two bits long and refers to the state of the BFD session between label edge routers; here a value of zero means “Administrative Down" (given by network operator), one stands for “Down", two for “Init" (used during BFD session setup) and three for “Up".

Ver	Request	Prot. Type	Revertive	Reserved1	FPath	Path
TLV Length			Reserved2			
Optional TLVs						

Figure 4.2: Format of a PSC Control Packet.

4.2.2 Protection State Coordination (PSC)

The PSC protocol is used to ensure that — whenever protection switching is triggered in one of the unidirectional LSPs of an MPLS-TP tunnel — the protection switching is also triggered for the remaining unidirectional LSP. This is in line with the co-routed bidirectional feature of MPLS-TP LSPs described in Section 4.2.1. PSC protocol also tells the LER whether the protection LSP is available, and if there is any inconsistency in protection switching configuration (timers, revertive functionality, etc.) between LERs.

There are six different PSC protocol states, among which the normal state is the default state when protection switching is enabled; and the unavailable state, which is used when protection switching is disabled by network operator or unavailable due to a failure on the protection LSP. A label edge router calculates the next PSC protocol state, based on the priorities of the requests issued by three sources: local requests (which can come from the network operator, control plane, management plane or specific timers), the PSC message received from the peer label edge router, and the current PSC protocol state. According to [17], the highest priority of the requests issued by any of the sources described above corresponds to network-operator commands (“Clear", “Lockout of Protection" and “Forced Switch"). Network-operator commands are used by the attackers to launch the attacks on the PSC protocol described in Section 4.4.2. For these attacks, we discuss two fields of the PSC control packet format (Figure 4.2). First we have the 4-bit request (*Request*) field that represents the PSC protocol state of the local label edge router that is sent to the peer label edge router to be considered on its next state computation. In this field code 14 stands for “Lockout of Protection" and indicates that protection switching is down as a result of a network operator command. The second field is the 8-bit fault path (*FPath*) field that indicates which LSP (working or protection) shall be affected by the *Request* field.

4.3 Testbed Description

In this section, we describe the setting we used to evaluate MPLS-TP security. The network topology that we used is shown in Figure 4.3 [48]. Depicted routers run Cisco IOS that supports MPLS-TP; namely, we use Cisco Cloud Service router 1000V (CSR1000V) IOS. We mount eight VMWare virtual machines (VM) configured with

CSR 1000V images on two physical machines (four virtual routers per each of the two physical machines). Each physical machine has an eight-core processor and 16GB of RAM. We assign one processor core and 3GB of RAM for each of the four virtual routers within one physical machine. The CSR 1000V 60-day evaluation license that we had at our disposal gave us full access to all the CSR 1000V features at a throughput of 50 Mbps. To the best of our knowledge, Cisco's other commercially available routers that support MPLS-TP have no additional security-related features.

We configure our MPLS-TP network to follow a one-to-one (1:1) protection mechanism by configuring a working LSP and a protection LSP between R2 and R7. A working LSP follows the path R2-R1-R3-R5-R7, whereas a protection LSP follows the path R2-R4-R6-R8-R7. We configure LSPs statically and use RSVP to reserve network resources. In order to connect each virtual router, as well as two physical machines, we use 1 Gbps links with full duplex configuration. As MPLS TP recommends co-routed bidirectional LSPs as described in Section 4.2, both directions of each link are assigned 25Mbps of bandwidth. Inside both the working and protection LSPs, we configure a BFD session, described in Section 4.2.1, with a 4ms message interval and a 12ms detection interval [49].

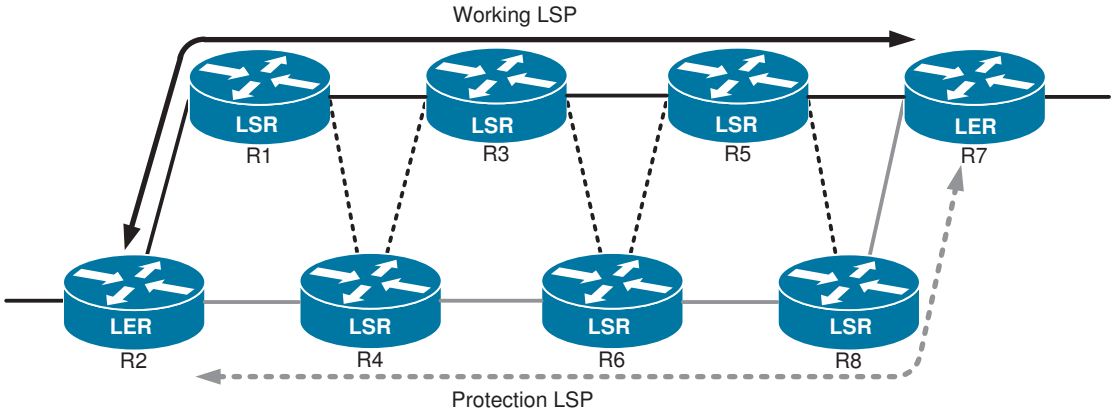


Figure 4.3: The Network topology with 1:1 Protection used in our MPLS-TP testbed.

4.4 Vulnerabilities in MPLS-TP Protocol

In this section, we describe three attacks. In Section 4.4.1 we describe two BFD spoofing attacks: In the first one, we remove protection from a target LSP; and in the second one, we disable fault detection in an LSP. In Section 4.4.2 we describe a PSC spoofing attack in which we bring down an operational MPLS-TP tunnel. As discussed in Section 4.1, we assume that the attacker can access the cables at one point (for the first and third attacks) or two points (for the second attack).

4.4.1 BFD Spoofing Attacks

As described in Section 4.2.1, BFD control messages are used to proactively monitor the continuity of an LSP. In this section, we describe spoofing attacks associated with BFD messages that enable an attacker to launch targeted attacks on a specific LSP.

Scenario I - Removing Protection from a Target LSP

In this attack, the attacker's goal is to falsely inform a label edge router that a working LSP is broken, even though there is no actual failure. This attack forces label edge routers of an LSP to unnecessarily switch from a working LSP to a protection LSP.

In our experiment, first we connected our switch to the cable of the working LSP at point *a* in Figure 4.4. Then we connected our laptop *b* to the switch and sniffed for BFD packets in order to gather information such as MAC addresses of LSRs $R3$ and $R5$, MPLS label, the TTL value, and the BFD session number of the target LSP between LERs $R2$ and $R7$. Finally, using the sniffed information, we created the forged packets by using the Scapy [50] tool and sent them to $R7$ through the switch by using a simple python code. The packets were manipulated such that we modified the *Diagnostic (Diag)* field and the *State (Sta)* field of a BFD control packet shown in Figure 4.1; the *Diag* field was set to "*Control Detection Time Expired*" and the *Sta* to "*Down*" (Figure 4.5). Note that other *Diag* field codes could also be used with similar results.

Upon reception of *BFD Down* packets, $R7$ is deceived into believing that the forward working LSP from $R2$ to $R7$ is down. In order to observe the result of this attack, we sniffed at points *c* and *d* of Figure 4.4[1]; we observed that $R7$ starts the protection switching by sending to $R2$ three PSC packets via the protection LSP with a "*Signal Fail (SF)*" message, and by sending BFD control packets via the working LSP with the *Sta* field set to "Down". $R7$ does this despite the fact that it also receives legitimate BFD packets from $R2$ with *Sta* field set to "Up". When $R2$ receives the BFD and PSC messages from $R7$ through the working LSP and protection LSP (respectively), it

[1] Note that packet sniffing at points c and d is not part of the attack, we used it only to establish that the attack works.

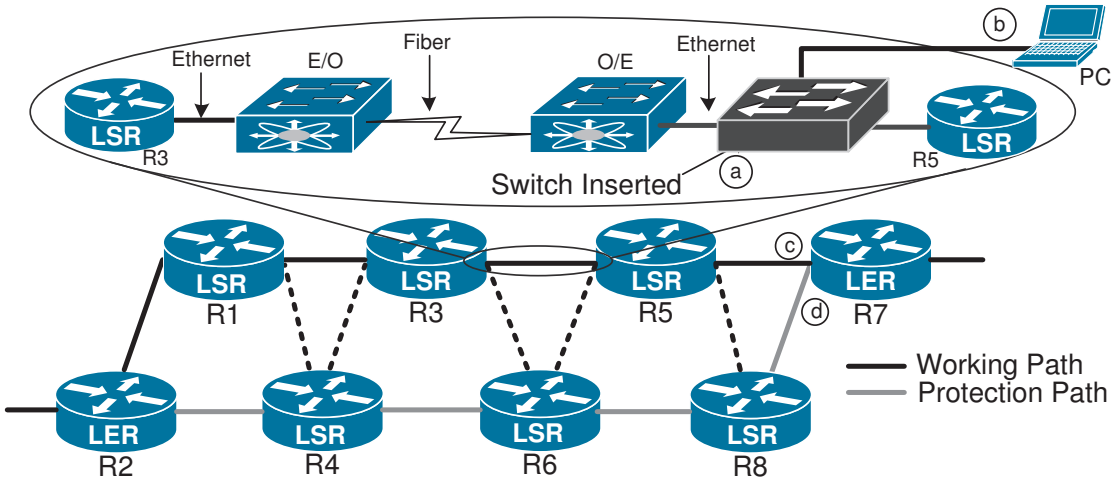


Figure 4.4: BFD Spoofing Attack : Removing protection from a target LSP.

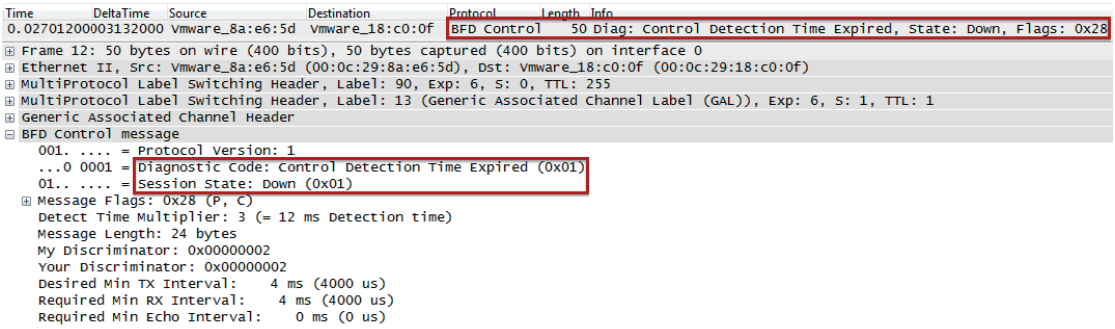


Figure 4.5: A wireshark capture of a spoofed BFD packet to remove protection from a target LSP.

triggers the protection switching and sends to $R7$ three PSC control packets with a "*Signal Fail (SF)*" message.

We observed that, by continuously injecting the forged BFD Down packets destined to $R7$, we impede the working LSP from getting back on its feet - effectively removing protection from the target LSP. This is possible because there is no mechanism for detecting that the frequency with which the messages are coming is different from what is expected. In other words, no matter how many forged messages we insert within the expected period of $4ms$, it does not raise any suspicion and those messages are accepted as legitimate. Similarly, no suspicion is raised when a node receives a steady mixture of "Up" and "Down" messages. As a final outcome of the attack, in the case of a fault in the protection LSP, the LERs would not have any alternative LSP to switch to.

Scenario II - Disabling LSP Fault Detection

The attacker's goal in this attack is the inverse of the attack introduced above, i.e., he aims to falsely inform label edge routers that a working LSP is up and running, while in reality it is down due to a link failure somewhere along the path. The link failure can be due to deliberate sabotage or a result of a random failure. In our experiment, we used the set up in Figure 4.6 to demonstrate this attack. On the working LSP, we connected two malicious switches *b* and *d* to the cables between $R1$ and $R2$ and between $R5$ and $R7$. Then we connected two laptops *a* and *c* to the MPLS-TP network through these two switches.

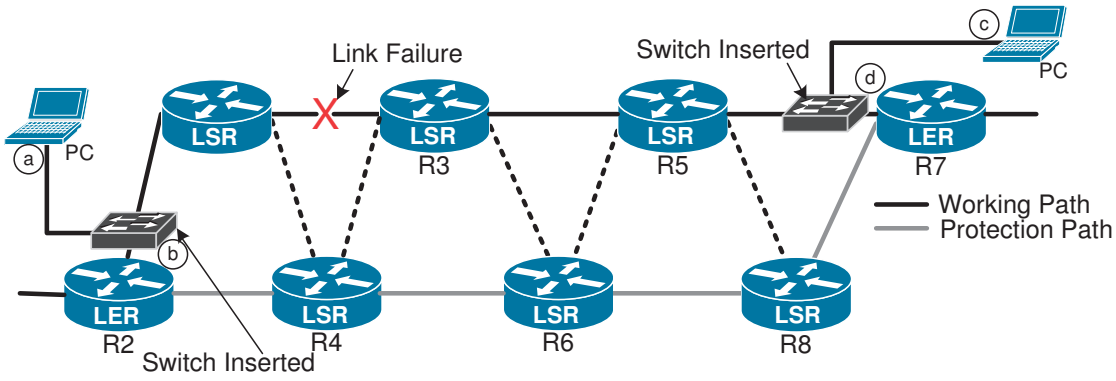


Figure 4.6: BFD Spoofing Attack: Disabling fault detection in an LSP.

From laptop *a*, we injected forged BFD *"Session Up"* packets (the *Sta* field set to "*Up*") destined to LER $R2$ as if they were sent from LER $R7$. Likewise, we injected forged *"Session Up"* packets from laptop *c* destined to LER $R7$ as if they were sent from LER $R2$. The two LERs processed these forged packets without raising any alarms in spite of receiving more BFD *"Session Up"* packets than expected, as a result of the packets injected from our laptops.

We then broke the link between $R1$ and $R3$, thus emulating a random link failure condition, while we continued sending the forged *"Session Up"* BFD packets to both label edge routers $R2$ and $R7$ from our two laptops. Again, we observed that the two label edge routers $R2$ and $R7$ did not notice the change in the rate of received BFD *"Session Up"* messages as a result of the missing authentic BFD messages from each other. Hence, the forged BFD messages from our laptops tricked both label edge routers into believing that the working LSP was still up and running, while in reality the link between $R1$ and $R3$ was down. To test our hypotheses, we sent *ping* commands from $R2$ to $R7$ and sniffed for traffic in the working LSP at laptop *a*. At this location, we observed ping requests sent from $R2$ to $R7$. We also observed that $R2$ did not receive any ping replies on either of its interfaces. Hence, we conclude that $R2$ actually sent the ping requests through the working LSP as if everything was fine on this LSP.

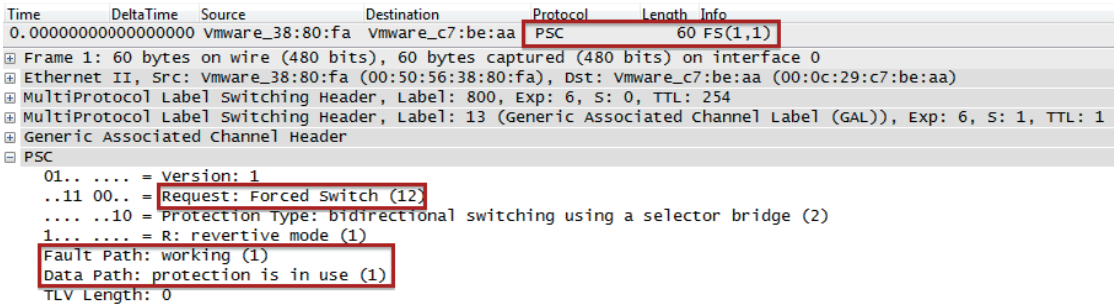


Figure 4.7: A wireshark capture of a spoofed PSC message to shutdown a working LSP

The general conclusion we can make from this experiment is that sending forged BFD *"Session Up"* packets to both LERs of an LSP disables protection switching in the presence of a link failure. Therefore, data sent through the working LSP from either end of the LSP is silently dropped at the broken link. Such loss of data can have undesirable consequences especially to real-time systems such as smart-grid applications.

4.4.2 PSC Spoofing Attack

In this attack, the goal is to instruct a label edge router to completely shutdown a target MPLS-TP tunnel for particular working and protection LSPs. Based on our testbed-experiment results, carrying out PSC spoofing attack is simpler compared to a BFD spoofing attack due to the priority hierarchy of the protocol and to the plainness of the requests for the change of the PSC protocol state, as described in section 4.2.2; nevertheless, the results of the PSC spoofing attack can be devastating compared to a BFD spoofing attack because the PSC message can completely stop the operation of a target MPLS-TP tunnel.

For the attack carried out in the testbed, we inserted a switch between routers $R4$ and $R6$, and we plugged a laptop to the switch, as shown at points *a* and *b* in Figure 4.9. The only attributes we needed to gather were the MAC address of the target label-switching router ($R6$) and the MPLS label, as opposed to several attributes needed for BFD spoofing attacks.

We created two different types of PSC packets; one targeting the working LSP and another one targeting the protection LSP. Both messages emulate an operator's "shutdown" commands executed at R2 for both the working and protection LSPs. We generate the spoofed packets by manipulating the *Request* and the *Fault Path* fields. For the PSC packet targeting the working LSP, we set the *Request* field to "Forced switch (12)" and the *Fault Path* field to "Working (1)" (Figure 4.7) and for the one targeting the protection LSP the *Request* field is set to "'Lockout of protection (14)" and the *Fault Path* field to "Protection (0)" (Figure 4.8) . Finally, we sent both packets to LER $R7$ via $R6$.

```
Time          DeltaTime  Source              Destination         Protocol  Length Info
0.000000000000000000     Vmware_38:80:fa     Vmware_c7:be:aa     PSC              60 LO(0,0)
⊞ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: Vmware_38:80:fa (00:50:56:38:80:fa), Dst: Vmware_c7:be:aa (00:0c:29:c7:be:aa)
⊞ MultiProtocol Label Switching Header, Label: 800, Exp: 6, S: 0, TTL: 254
⊞ MultiProtocol Label Switching Header, Label: 13 (Generic Associated Channel Label (GAL)), Exp: 6, S: 1, TTL: 1
⊟ Generic Associated Channel Header
    .... 0000 = Channel Version: 0
    Channel Type: Protection State Coordination Protocol (PSC) (0x0024)
⊟ PSC
    01.. .... = Version: 1
    ..11 10.. = Request: Lockout of protection (14)
    .... ..10 = Protection Type: bidirectional switching using a selector bridge (2)
    1... .... = R: revertive mode (1)
    Fault Path: protection (0)
    Data Path: protection is not in use (0)
    TLV Length: 0
```

Figure 4.8: A wireshark capture of a spoofed PSC message to shutdown a protection LSP

In order to observe the effectiveness of this attack, we setup a sniffing session on both interfaces of router $R7$ (point *c* of Figure 4.9) (as before, such a sniffing session is *not* part of the attack, only part of our observation). We observed that when $R7$ received the forged PSC packet targeting the working LSP, it assumed LER ($R2$) was instructed by a network operator to switchover to protection LSP. As a result $R7$ also locked out the working LSP and sent three PSC control packets back to $R2$ with the *Request* field set to "Forced switch", *Fault Path* set to “Working". When $R7$ received the second forged PSC packet that targeted the protection LSP, again $R7$ assumed that $R2$ was instructed by the network operator to lockout the protection LSP. Thus it also locked out the protection LSP and replied back to $R2$ with three PSC packet with *Request* field set to "Lockout of protection (14)" and *Fault Path* to “Protection".

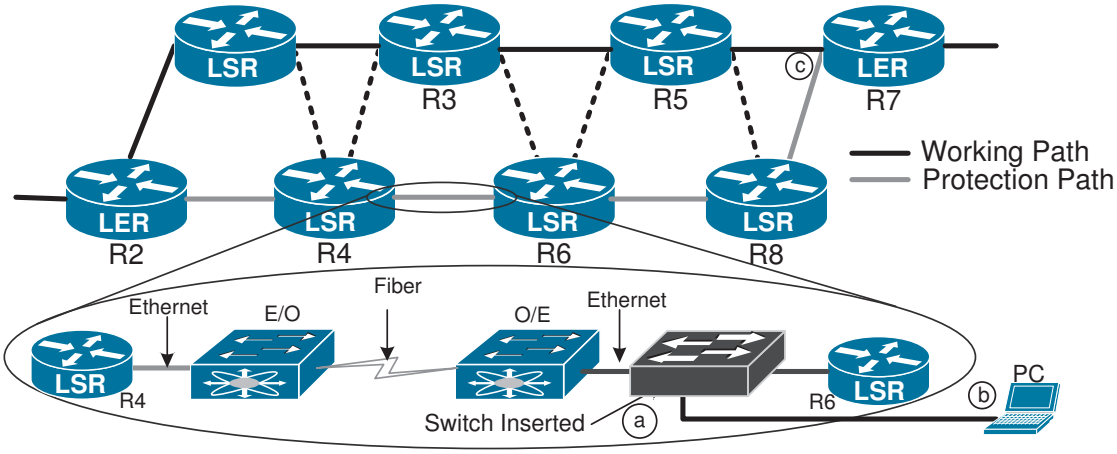


Figure 4.9: Network setup for PSC Spoofing attack.

The consequence of the attack is a complete shutdown of the MPLS-TP tunnel for the transmission of data. Since the forged packets inserted during the attack have the structure of a network operator’s command, the LERs cannot bring the MPLS-TP tunnel up by themselves. Thus the network operator is required to explicitly issue a

command to bring back the MPLS-TP tunnel to normal function.

4.5 Discussion and Countermeasures

In the previous section, we discussed spoofing attacks on BFD and PSC messages. These attacks disrupt the proper operation of an MPLS-TP network. Such attacks are possible because a label edge router does not have a means to verify whether received BFD and PSC messages truly originate from the label edge router on the other end of an LSP or whether they were forged messages. Therefore, an obvious solution to these attacks is to implement message-origin authentication mechanisms.

RFC5880 [51] proposes an optional authentication scheme for protecting BFD messages from spoofing attacks. Such a solution, if implemented with a proper key management scheme, could prevent the BFD spoofing attacks similar to those introduced in Section 4.4.1. However, the Cisco IOS MPLS-TP implementation, which we used for our experiments, does not implement this authentication option.

Unlike BFD, the PSC protocol does not have any built-in security to protect it from spoofing attacks. One solution for protection against PSC messages spoofing attacks is to craft a built-in authentication mechanism, similar to the optional BFD authentication (RFC5880), by using one of the optional TLV fields in a PSC packet.

If an MPLS-TP core network supports IP services, OAM messages such as BFD and PSC messages can be tunnelled on top of an IP tunnel. In such cases, standard IP security solutions such as IPsec or (D)TLS between label edge routers can be used to provide end-to-end security, thereby preventing spoofing attacks. RFC5085 [27] proposes IPsec as a solution to protect OAM protocols of MPLS/GMPLS networks if the core network supports IP, VPN, or transport services. However, not all core networks are required to support IP. For example, a smart grid MPLS-TP network that transports IEC 61850 based Multicast Sampled Value (MSV) and Generic Object Oriented Substation Event (GOOSE) messages between substations or between a substation and a controller is often implemented without IP [52].

An alternative solution for preventing spoofing attacks on BFD and PSC messages in non-IP MPLS core networks is to use hop-by-hop security (MACsec). However, Cisco does not support MACsec in its routers (it does support it in switches). One minor drawback of using MACsec is that if any of the network devices in an LSP are compromised, MACsec fails to achieve its purpose, i.e., forged BFD and/or PSC packets injected at the compromised device will be processed as valid packets by a receiving label edge router. Nonetheless, such attacks can be prevented by incorporating tamper resistant security solutions such as Trusted Platform Module (TPM) to protect sensitive data and by enforcing proper access control mechanisms to deny unauthorised access

to the network devices. Note that implementing an ACL alone would not solve the problem as the attack is conducted with spoofed packets. Since our findings show that lack of MAC layer security exposes smart grid networks to various cyber-attacks, we recommend that utilities implement MACsec or a variant of it in their MPLS-TP networks.

4.6 Conclusion

MPLS-TP is one of the proposed technologies for WAN connectivity in the context of smart grid. In this chapter, we have shown that, when it comes to the security aspects of the standard, there is a discrepancy between RFCs and the Cisco IOS implementation for MPLS-TP we evaluated, which is not surprising given the complexity of RFCs. More specifically, we have observed that the Cisco IOS does not implement security recommendations for OAM protocols, such as BFD and PSC, thus exposing them to different spoofing attacks. In our testbed, to launch spoofing attacks on these two OAM protocols we exploited the identified security vulnerabilities. By launching spoofing attacks, we have shown that we could degrade the performance of an MPLS-TP network by removing a protection LSP of an MPLS-TP tunnel. We have also demonstrated that we can disable detection of a link failure in LSP by tricking label edge routers into believing a failed link is still up and running. Finally we have shown that we can bring the whole MPLS-TP tunnel down by sending forged operator PSC commands.

Our experiments with the Cisco IOS show that no protection against spoofing attacks is provided for non-IP MPLS-TP OAM messages. Therefore, we recommend that RFCs be more directive in proposing built-in security for OAM protocols. They should mandate source authentication mechanisms for both BFD and PSC messages or mandate MACsec or a variant of it as an alternative authentication solution when built-in security is absent.

5 iPRP: Parallel Redundancy Protocol for IP Networks

5.1 Introduction

Specific time-critical applications (found for example in electrical networks, industrial processes, high-frequency trading, online gaming, etc.) have such strict communication-delay constraints that retransmissions following packet loss can be both detrimental and superfluous. In smart grids, critical control applications require reliable information about the network state in quasi-real time, within hard delay-constraints of the order of approximately 10 ms. Measurements are streamed periodically (every 20 ms for 50 Hz systems) by phasor measurement units (PMUs) to phasor data concentrators (PDCs). In such settings, retransmissions can introduce delays for successive, more recent data that in any case supersede older ones. Moreover, IP multicast is typically used for delivering the measurements to several PDCs. Hence, UDP is preferred over TCP, despite its best-effort delivery approach. Increasing the reliability of such unidirectional (multicast) UDP flows is a major challenge.

5.1.1 Problems with MAC-Layer Parallel Redundancy Protocol

The parallel redundancy protocol (PRP) IEC standard [20] was proposed as a solution for deployments inside a local area network (LAN) where there are no routers. Communicating devices need to be connected to two cloned (disjoint) bridged networks. The sender tags MAC frames with a sequence number and replicates it over its two interfaces. The receiver discards redundant frames based on sequence numbers.

PRP works well in controlled environments, such as a substation LAN, where network setup is entirely up to the substation operator, who ensures that the requirements of PRP are met (e.g., all network devices are duplicated). At a larger scale (for example, a typical smart grid communication network that spans an entire distribution network) routers are needed and PRP can no longer be used. Thus, a new solution is needed for IP wide area networks (WANs).

In addition to extending PRP functionality to WANs, the new design should also avoid the drawbacks of PRP. The most limiting feature of PRP is that the two cloned networks need to be composed of devices with *identical* MAC addresses. This contributes to making network management difficult. Furthermore, PRP duplicates all the traffic unselectively, which is acceptable for use in a LAN, but which cannot be done in a WAN, because links can be expensive and unnecessary traffic should be avoided. Moreover, PRP has no security mechanisms.

Note that a parallel redundancy protocol for IP WANs needs to support IP multicast, as this is used by modern smart grid applications.

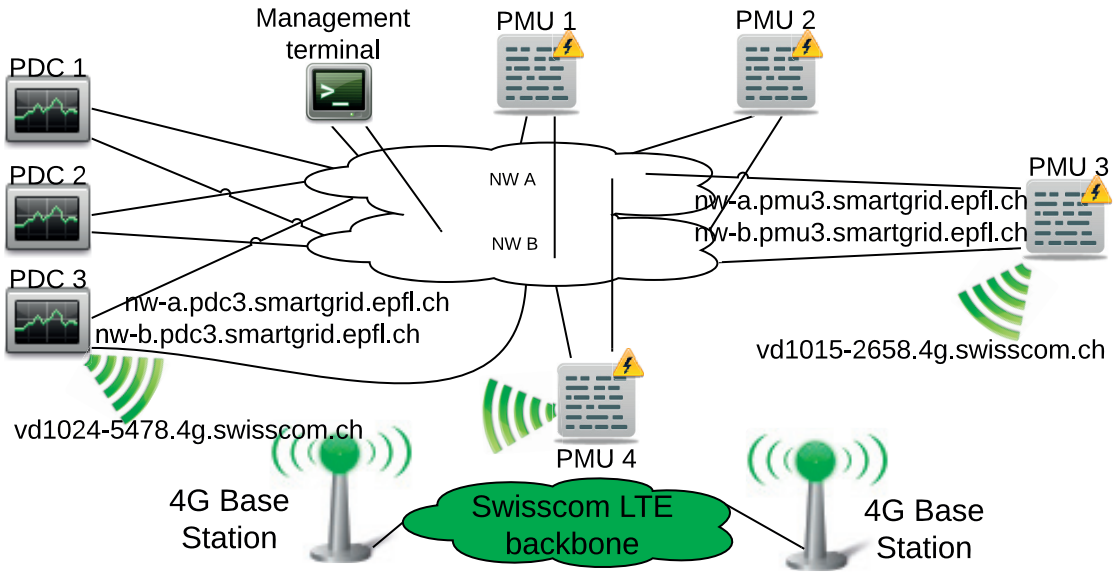


Figure 5.1: A typical iPRP use-case in the context of smart grids. Devices (Phasor Data Concentrators (PDC), Phasor Measurement Units (PMU)) are connected to two overlapping network subclouds (labeled A and B). Some devices use an additional LTE connection providing a low latency cellular service [53]. Every PMU streams data to all PDCs, using UDP and IP multicast.

Concretely, Fig. 5.1 depicts a smart grid WAN where PRP cannot be directly deployed: devices are multi-homed and each interface is assigned a different IP address. Most devices have two interfaces connected to a main network cloud made of two fail-independent network subclouds labeled "A" and "B", while some have a third interface connected to a 4G cellular wireless service (labeled "Swisscom LTE backbone" in the figure). It is assumed that paths between interfaces connected to the "A" network subcloud stay within it (and similarly with "B"). The "A" and "B" network subclouds could be physically separated, however in practice they are most likely interconnected for network management reasons.

A simple way to achieve the arrangement described before is to divide the network into two logical subclouds, *A* and *B*. Then, by adjusting the routing weights of the links

interconnecting the *A* and *B* subclouds, we can ensure that $A \to A$ and $B \to B$ traffic stays within *A* and *B* subclouds, respectively, thereby giving rise to fail-independent paths. In such a setting, the interconnections will be used only for $A \leftrightarrow B$ traffic.

We need a solution that, similarly to PRP, takes advantage of network redundancy for increasing the reliability of UDP flows, and that works in scenarios such as the one in Fig. 5.1.

The existence of fail-independent paths is fundamental for the optimal operation of such a solution. However, in the event of a network-component failure, the paths can partially overlap. Then, the solution should reap the maximum possible benefits by operating in a degraded-redundancy mode. In other words, if complete end-to-end redundancy is no longer possible, the solution should continue to work.

In order for our solution to be easily deployed, we also require it to be transparent to *both* the application and network layers: it should only require installation at end-devices and no modifications to running application software or to intermediary network devices (routers or bridges).

In this chapter we present the design and implementation of iPRP (the IP parallel redundancy protocol), a transport layer solution for transparent replication of unidirectional unicast or multicast UDP flows on multihomed devices.

5.1.2 iPRP

An iPRP host has to send different copies of the same packet over different paths. With the current technology, a device cannot control the path taken by an IP packet, beyond the choice of a destination address, exit interface and a type-of-service value. Other fields, such as the IPv6 flow label or source routing header extensions, are either ignored or rejected by routers. Also, the type-of-service field is used by applications and should not be tampered with by iPRP. Hence, we assume that a choice of the path is done at the sources by choosing communication interface and the destination address. The job of iPRP is then to transparently replicate packets over the different interfaces for the UDP flows that need it, match corresponding interfaces, remove duplicates at the receiver, and do this in a way that is resilient to crashes (see Section 5.5.7).

Not all traffic requires replication, only certain devices and certain UDP flows do (time-critical data). Hence, replication needs to be selective: a failure-proof mechanism, transparent to applications, is required for detecting and managing packet replication. It needs to correctly match the interfaces, so that independent paths are used whenever they exist.

The iPRP protocol design is such that it does not interfere with the existing security

mechanisms and does not introduce any new security weaknesses (see Section 5.6).

iPRP assumes that the network is traffic-engineered; the critical UDP data streams receive enough resources and are not subject to congestion. iPRP instantly repairs packet losses due to failures or transient problems such as transmission losses. It *does not* solve congestion problems due to under-dimensioned network links. TCP flows are not affected.

Our iPRP implementation is for IPv6, as it is being installed in our smart-grid communication network (smartgrid.epfl.ch), that uses IPv6 (following the argument that new network environments should avoid future transition problems and embrace IPv6 from the start). Our implementation is available at http://goo.gl/N5wFNt. Adaptation to IPv4 is straightforward.

5.2 Related Work

As mentioned in Section 5.1, iPRP overcomes the limitations of PRP [20]. The authors of [54] are aware of the fact that PRP is limited to LANs and suggest a direction for developing PRP in an IP environment. Their suggestion is neither fully designed nor implemented. Also, it requires that the intermediate routers preserve the PRP trailers at the MAC layer, which in turn requires changes in all of the routers in the networks. It does not address all the shortcomings of PRP (diagnostic tools, lack of IP multicast support, need of special hardware). In contrast, our transport layer approach does not have these drawbacks.

Multipath TCP (MPTCP) [55] is used in multi-homed hosts. It allows TCP flows to exploit the host's multiple interfaces, thus increasing the available bandwidth for the application. Like MPTCP, iPRP is a transport layer solution and is transparent to network and application. Unlike MPTCP, iPRP replicates the UDP packets on the parallel paths, while MPTCP sends one TCP segment on only one of them. In a case of loss, MPTCP resends the segment on the same path until enough evidence is gathered that this path is broken. So, a lost packet is repaired after several RTTs (not good for time-critical flows).

Similarly, link aggregation control protocol (LACP) [56] and equal-cost multi-path routing (ECMP) [57] require seconds for failover. LACP enables the bundling of several physical links together to form a single logical channel. The failure of a link is discovered through the absence of keep-alive messages that are sent every $1-30$ s. ECMP can be used together with most routing protocols in order to balance traffic over multiple best paths when there is a tie. In a case of failure, it relies on the reconfiguration of the underlying routing protocol, that is commonly detected by the absence of keep-alive messages.

Network coding exploits network redundancy for increasing throughput [58], and requires intermediary nodes to recode packets (specialized network equipment needed). Also, it is not suitable for time-critical applications as typically packets are coded across "generations" which introduces decoding delays. Source coding (e.g. Fountain codes [59]) can be useful for the bursty transmissions of several packets. However, it adds delay, as encoding and decoding are performed across several packets (not suitable for UDP flows with hard-delay constraints).

Multiprotocol-label-switching transport-profile (MPLS-TP) $1+1$ protection feature [25] performs packet duplication and feeds identical copies of the packets in working and protection path. On the receiver side, there exists a selector between the two; it performs a switchover based on some predetermined criteria. However, some time is needed for fault detection and signaling to take place, after which the switchover occurs. Hence, a 0-ms repair cannot be achieved.

Multi-topology routing extends existing routing protocols (e.g. [60]) and can be used to create disjoint paths in a single network. It does not solve the problem of transparent packet replication, but can serve as a complement to iPRP in the following way. On top of the underlying network (base topology) additional class-specific topologies can be created as a subset of base topology. We can use this feature to define fail-independent A and B subclouds in order to ensure fail-independent paths between sources and destinations.

Another method to ensure the discovery of fail-independent paths is software-defined networking (SDN) [19]. The centralized controller is aware of the overall network topology and can impose routing rules in a way that guarantees independent paths/trees between all the hosts.

5.3 A Top-Down View of iPRP

In this section we first go over high-level design decisions we had to make during the development of the iPRP protocol and then succinctly describe the resulting design. iPRP aims to provide reliable end-to-end communication between multi-homed hosts. The very first question that emerges is the choice of a TCP/IP layer where iPRP should be placed. Among others, iPRP needs to support scenarios where the traffic is carried over a shared network infrastructure e.g., a telecom network operator provides connectivity for smart-grid services [61]. In this case, end-users do not control intermediate routers. Hence, the routers should not be aware of the existence of iPRP. Consequently, we put network transparency as a requirement which leads us to a solution that places iPRP *above* the network layer. Taking this into account a possible solution can be to place iPRP at the application layer. This would imply that all legacy applications that are traditionally used would need to undergo changes in order to be compatible with

the iPRP protocol. Again, we opt for an application-transparent solution which leads us to a choice of a transport-layer solution.

The next choice to make was whether all the traffic originated at a sender should be replicated. Having in mind that bandwidth over WAN links can be of limited capacity we opt here for a solution that replicates traffic selectively. The control of this feature is left to users through a simple configuration of the UDP port numbers that correspond to services that demand high reliability of packet delivery.

The next choice was that of a mechanism to inform a sender about the alternate IP addresses (unicast or multicast) of the receivers, so as to establish redundant paths. Classic solutions like PRP use cloned address networks. Cloned IP addresses are not an option of iPRP as users do not necessarily control and manage all the interconnecting networks.We solve this problem by designing a lightweight, secure and crash-resilient signaling protocol. It also takes into account specificities of the multicast communication e.g., it avoids flooding of senders with signaling messages from large groups of multicast receivers. It is a plug and play protocol initiated whenever a new sender emerges and completely transparent to the application layer.

Furthermore, we needed a mechanism for discard of duplicates, which can cope with packet reordering due to the network, and crash failures of the hosts. Existing duplicate-discard mechanisms such as the one used by PRP do not perform well under such packet-reordering and host failures. So, we also put in place a stateless protocol for redundant-packets removal.

The resulting design of iPRP is as follows. iPRP senders and receivers are expected to have multiple network interfaces. Applications are identified by the UDP port used to receive data. To enable iPRP for a certain application at the receiver, the port on which the application is listening needs to be added to the list of iPRP monitored ports (see Section 5.4.3). Receiving data on an iPRP monitored port automatically triggers an iPRP session between the sender and the receiver (see Section 5.5). Within this session, the iPRP software running on the source host learns the receiver's network interfaces from the iPRP software running on the receiver. It uses the configured rules to match local interfaces to the receiver's remote ones. It then proceeds to capture the application's outgoing packets, encapsulates them in iPRP data messages addressed to the receiver's remote interfaces (according to the determined matching), and replicates them over the local interfaces. At the receiver, the iPRP software decapsulates the original packets, discards duplicates (Section 5.5.5), and delivers them to the receiver application.

Hosts having multiple interfaces is not a strict requirement, but is desirable. In cases where the sender or the receiver have a single interface, iPRP still works, but the paths taken by the replicated packets join at a certain point, which becomes a single point of failure.

5.4 Operation of iPRP

5.4.1 How to Use iPRP

iPRP is installed on end-devices with multiple interfaces: on streaming devices (the ones that generate UDP flows with hard delay constraints) and on receiving devices (the destinations for such flows).

Streaming devices (such as PMUs) do not require special configuration. Streaming applications running on such devices benefit from the increased reliability of iPRP without being aware of its existence. iPRP operates as a modification to the UDP layer.

On receiving devices the only thing that needs to be configured is the set of UDP ports on which replication is required. For example, say that an application running on a PDC is listening on some UDP port for measurement data coming from PMUs. After iPRP is installed, this port needs to be added to the list of iPRP monitored ports in order to inform iPRP that any incoming flows targeting this port require replication. The application does not need to be stopped and is not aware of iPRP.

Nothing else needs to be done for iPRP to work. In particular, no special configuration is required for intermediary network equipment (routers, bridges).

5.4.2 General Operation: Requirements for Devices and Network

iPRP provides $1 + n$ redundancy. It increases, by packet replication, the reliability of UDP flows. It does not impact TCP flows.

iPRP-enabled receiving devices configure a set of UDP ports as *monitored*. When a UDP packet is received on any of the monitored ports, a one-way *soft-state iPRP session* is triggered between the sender and the receiver (or group of receivers, if multicast is used). *Soft-state* means that: (i) the state of the communication participants is refreshed periodically, (ii) the entire iPRP design is such that a state-refresh message received after a cold-start is sufficient to ensure proper operation. Consequently, the state is automatically restored after a crash, and devices can join or leave an iPRP session without impacting the other participants.

Within an iPRP session, each replicated packet is tagged with an iPRP header (Section 5.5.4). It contains the same *sequence number* in all the copies of the same original packet. At the receiver, duplicate packets with the same sequence number are discarded (Section 5.5.5). The original packet is reconstructed from the first received copy and forwarded to the application.

In multicast, all devices in the group of receivers need to run iPRP. If only some of

the receivers support iPRP, these trigger the start of an iPRP session with the sender and benefit from iPRP; however, the others stop receiving data correctly. The use of source-specific multicast (SSM) is recommended (see [62]).

All iPRP-related information is encrypted and authenticated. Existing mechanisms for cryptographic key exchange are applied (security considerations in Section 5.6) .

5.4.3 UDP Ports Affected by iPRP

iPRP requires two system UDP ports (transport layer) for its use: the *iPRP control port* and the *iPRP data port* (in our implementation 1000 and 1001, respectively). The iPRP control port is used for exchanging messages that are part of the soft-state maintenance. The iPRP data port receives data messages of the established iPRP sessions. iPRP-capable devices always listen for iPRP control and data messages.

The set of monitored UDP ports, over which iPRP replication is desired, are *not* reserved by iPRP and can be any UDP ports. UDP ports can be added to/removed at any time from this set during the iPRP operation. Reception of a UDP packet on a monitored port triggers the receiver to initiate an iPRP session. If the sender is iPRP-capable, an iPRP session is started (replicated packets are sent to the iPRP Data Port), else regular communication continues.

5.4.4 Matching the Interconnected Interfaces of Different Devices

One of the design challenges of iPRP is determining an appropriate match between the interfaces of senders and receivers, so that replication can occur over fail independent paths. To understand the problem, consider Figure 5.1 where the PMUs and PDCs have at least two interfaces. The A and B network subclouds are interconnected. However, the routing is designed such that a flow originating at an interface connected to subcloud A with a destination in A will stay in subcloud A. A potential problem can arise if a sender's interface, say SA, intended to be connected to the A subcloud, is mistakenly connected to the B subcloud, and vice-versa. Then one path from source to destination will go from SA (on subcloud B) to the destination interface DB (on subcloud B), and conversely on the other path. Following the routing rules, these flows will use interconnecting links between A and B subclouds. This is not desirable as these links can be of insufficient capacity because they are not intended to carry such traffic. Furthermore, it is no longer guaranteed that such paths are disjoint. PRP avoids this problem by requiring two physically separated and cloned networks. iPRP does not impose these restrictions. Hence, iPRP needs a mechanism to match interfaces connected to the same network subcloud.

To facilitate appropriate matching, each interface is associated with a 4-bit identi-

fier called *iPRP Network subcloud Discriminator (IND)*, which qualifies the network subcloud it is connected to. The iPRP software in end-devices learns each of the interfaces' INDs automatically via simple preconfigured rules. Network routers have no notion of IND. A rule can use the interface's IP address or its DNS name. In our implementation, we compute each interface IND based on its fully qualified domain name. In Figure 5.1, the rule in the iPRP configuration maps the regular expression `nw-a*` to the IND value `0xa`, `nw-b*` to IND `0xb`, and `*swisscom.ch` to IND `0xf`, respectively.

The receiver periodically advertises the IP addresses of its interfaces, along with their INDs to the sender (via `iPRP_CAP` messages). The sender compares the received INDs with its own interface INDs. Only those interfaces with matching INDs are allowed to communicate in iPRP mode. In our example, IND matching prevents iPRP to send data from a PMU A interface to a PDC B interface. Moreover, each iPRP data packet contains the IND of the network subcloud where the packet is supposed to transit (see Section 5.5.4). This eases the monitoring and debugging of the whole network. It allows us to detect misconfiguration errors that cause a packet expected on an A interface to arrive on a B interface.

5.5 Protocol Description

The iPRP message exchange is divided into two planes: control plane and data plane. The control plane is responsible for exchange of messages required to establish and maintain an iPRP session. The data plane is responsible for replication and deduplication of time-critical UDP flows. Note that control plane messaging is non-time critical and far less frequent than data plane (data plane $\sim$ ms, control plane $\sim$ s).

The data plane operation is divided into two phases: replication phase and duplicate discard phase. Next, we discuss the operation of each plane and the description of key elements of the iPRP protocol in detail.

5.5.1 Control Plane

The control plane is used for exchange of messages to establish and maintain an iPRP session. The iPRP session establishment is triggered when a UDP packet is received at some monitored UDP port p. In Fig. 5.2, UDP port p is made monitored at t_1 at the receiver, by adding it to the list of monitored ports. This triggers the establishment of an iPRP session, i.e., the receiver's *soft-state-maintenance* functional block (Fig. 5.3) adds the sender to the list of active senders (Alg. 1).

The *iPRP-capability-advertisement* functional block (Fig. 5.3) at the receiver, sends `iPRP_CAP` to the control port of the sender every T_{CAP} seconds (t_2 in Fig. 5.2, Alg. 2). This message informs the sender that the receiver is iPRP enabled and provides

Algorithm 1: (At the receiver) Soft-state maintenance (keeps the list of active senders up-to-date)

```
1  while true do
2      remove inactive hosts from the list of active senders (last-seen timer expired);
3      for every packet received on one of the monitored ports or on iPRP Data Port
       do
4          if the source is in the list of the active senders then
5              update associated last-seen timer;
6          else
7              put sender in the list of active senders;
8          end
9      end
10 end
```

Algorithm 2: (At the receiver) iPRP capability advertisement

```
1  while true do
2      compute T_backoff (Section 5.5.6);
3      listen for iPRP_ACKs until T_backoff expires;
4      send iPRP_CAP messages to all hosts in the list of active senders from which
       no iPRP_ACKs are received;
5      sleep T_CAP − T_backoff;
6  end
```

information required for selective replication over alternative paths. It contains: (1) the iPRP version; (2) INDs of the network subclouds to which the receiver is connected, to facilitate IND matching (see Section 5.4.4); (3) the source and destination UDP port numbers of the packet that triggered the establishment of the iPRP session; (4) in multicast, the multicast IP address of the group; (5) in unicast, IP addresses of all receiver interfaces; (6) a symmetric, short-lived cryptographic key for authentication and encryption of the iPRP header (Section 5.6)

On receiving the `iPRP_CAP`, the *iPRP-session-maintenance* functional block (Fig. 5.3) at the sender acknowledges it with an `iPRP_ACK`. The `iPRP_ACK` contains the list of sender IP addresses which are used by the receiver to subscribe to alternate network subclouds to receive data through SSM. In multicast, the receivers send `iPRP_CAP` after a back-off period (Section 5.5.6) to avoid flooding. The `iPRP_ACK` message also serves as a terminating message for impending `iPRP_CAP`s, thereby preventing a flood (Alg. 2).

To complete the iPRP session establishment, the iPRP-session-maintenance functional block performs IND matching (section 5.4.4) and creates a *peer-base* entry (t_3 in Fig. 5.2, Alg. 3). The *peer-base* contains all information needed by the sender for replication of data packets.

Algorithm 3: (At the sender) iPRP session maintenance

```
1  while true do
2      remove aged entries from the peer-base;
3      for every received iPRP_CAP message do
4          if there is no iPRP session established with the destination then
5              if IND matching is successful then
6                  establish iPRP session by creating new entry in the peer-base;
7                  send iPRP_ACK message;
8              end
9          else
10             update the keep-alive timer;
11         end
12     end
13 end
```

The second goal of the control plane it to maintain an iPRP session. To this end, the `iPRP_CAP` messages are used as keep-alive messages (Alg. 3). The iPRP session is terminated if no `iPRP_CAP` message is received for a period of $3T_{CAP}$. These messages are sent to a sender as long as it is present in the list of active senders. The list of active senders is maintained by the soft-state-maintenance functional block by updating the *last-seen timer* (Alg. 1) when a new data packet is received. Sessions that are inactive for more than $T_{inactivity}$ are terminated.

For each new iPRP session, a corresponding iPRP session establishment is triggered. If any of the required steps could not be completed due to message loss or to an iPRP incapability, an iPRP session is not established and packets are not replicated.

Addition or removal of new interfaces at the sender or receiver is communicated by the `iPRP_CAP` messages and the *peer-base* is updated accordingly. Specifically, when an `iPRP_CAP` is received for already established iPRP sessions, the peer-base is updated in the following ways. Newly received INDs, which are successfully matched are added to the peer-base. On the contrary, INDs in the peer-base that cannot be matched with any of the received INDs, are removed from the peer-base after confirmation from multiple consecutive `iPRP_CAP`s (to handle the effect of the backoff algorithm in Section 5.5.6).

5.5.2 Data Plane: Replication Phase

The replication phase occurs at the sender to send out data plane messages once the iPRP session is established. The *replication* functional block (Fig. 5.3) on the sender intercepts all outgoing packets destined to UDP port p of the receiver. These packets are subsequently replicated and iPRP headers (section 5.5.4) are prepended to each copy of the payload. iPRP headers are populated with the iPRP version, a sequence-

number-space ID (SNSID - unique identifier of an iPRP session), a sequence number, an original UDP destination port, and IND. The 32-bit sequence number is the same for all the copies of the same packet. The destination port number is set to the iPRP data port for all the copies. An authentication hash is appended and the whole block is encrypted. Finally, the copies are transmitted as iPRP data messages over the different matched interfaces (see Alg. 4, t_4 in Fig. 5.2).

Algorithm 4: (At the sender) Packet replication

```
1  for every outgoing packet do
2      check the peer-base;
3      if there exists an iPRP session that corresponds to the destination socket then
4          replicate the payload;
5          append iPRP headers incl. seq. number;
6          send packet copies;
7      else
8          forward the packet unchanged;
9      end
10 end
```

5.5.3 Data Plane: Duplicate Discard Phase

The duplicate discard phase occurs at the receiver once an iPRP session is established to ensure that only one copy of replicated packets is forwarded to the application. Upon reception of packets on the iPRP data port, the associated last-seen timer is updated (see Alg. 1) and the packets are forwarded to the *duplicate-discard* functional block (Alg. 5). It decrypts the iPRP header at the beginning of the payload using the symmetric key used in `iPRP_CAP` message. Then, function `isFreshPacket` (Section 5.5.5 - Alg. 6) is called. Based on the sequence-number-space ID (SNSID - unique identifier of an iPRP session) and the sequence number, the packet is either forwarded to the application or discarded. The first received copy should reach the application, subsequent copies are discarded. The replication is thus rendered transparent to the sender and receiver applications. In Fig. 5.2 we show two scenarios after the time t_4; in one case both copies are delivered, in the other, one packet is lost.

5.5.4 The iPRP Header

Fig. 5.4 shows the position and the fields of the iPRP header used in data packets. The SNSID is used to identify an iPRP session. This identifier is unique across all iPRP sessions terminating at the same receiver, thereby allowing multiple iPRP sessions on the same machine. In our implementation, it is chosen as a concatenation of the source IPv6 address, the source UDP port number of the socket to which the application writes the packet and a 16-bit reboot counter.

Algorithm 5: (At the receiver) Duplicate discard

```
1  for every packet received on iPRP data port do
2      get sequence number space ID (SNSID);
3      get sequence number (SN);
4      if it is the first packet from this SNSID then
5          SNSID.HighSN← SN;                                  // Bootstrap
6          remove iPRP header;
7          reconstruct original packet;
8          forward to application;
9      else
10         if isFreshPacket(SN, SNSID) then
11             remove iPRP header;
12             reconstruct original packet;
13             forward to application;
14         else
15             discard the packet;
16         end
17     end
18 end
```

The SNSID is used by a receiver to tie the packets with different source IP addresses that belong to the same iPRP session. When a new receiver joins a multicast group with an already established iPRP session, it uses the source IP address in the SNSID to uniquely identify the sender of the packets and the source port number in the SNSID to uniquely identify the streaming application on the sender. However, in case of a crash and reboot of the sender, the sequence number is reset. Then, a new reboot counter in the iPRP header differentiates packets belonging to the new iPRP session from those of the old iPRP session, thereby ensuring a seamless recovery at the receiver.

To maintain the format of the iPRP header for an IPv4 implementation, we suggest repeating source IPv4 address four times at the place of source IPv6 address. The original destination UDP port number is included to allow for the reconstruction of the original UDP header. The iPRP header is placed after the inner-most UDP header. So, iPRP works well, even when tunneling is used (e.g., 6to4).

Like many protocols (such as DTLS, VPN, VXLAN, 4in6, etc.), iPRP adds its own header to the packet payload. In order to avoid packet fragmentation, we adopt the same solution as any tunneling protocol: at the sender, iPRP reduces the interface MTU size to the minimum of 1280 bytes required by IPv6. In practice, typical MTU values are closer to the IPv6-recommended 1500 bytes. This leaves a margin for the inclusion of the iPRP and other tunneling protocol headers.

5.5.5 The Discard Algorithm

The redundant copies of a packet are eliminated by a discard algorithm running at the receiver. In scenarios where the packets are received out-of-order, the discard algorithm proposed for PRP [63] delivers several copies of the same packet to the application. The function `isFreshPacket` (Alg. 6) avoids this issue. It is used by Alg. 5 to decide if a packet sequence number corresponds to a fresh packet. We use 32-bit unsigned integer sequence numbers, large enough to avoid the wrap-around problem.

Algorithm 6: Function to determine whether a packet with sequence number `CurrSN` corresponds to a fresh packet in the sequence number space ID `SNSID`. The test "`x` follows `y`" is performed for 32-bit unsigned integers using subtraction without borrowing as "`(x-y)>>31==0`".

```
1  function isFreshPacket(CurrSN, SNSID)
2  if CurrSN==SNSID.HighSN then
3  |   return false ;                                   // Duplicate packet
4  else if CurrSN follows SNSID.HighSN then
5  |   put SNs [SNSID.HighSN+1, CurrSN-1] in SNSID.ListSN;
6  |   remove the smallest SNs until SNSID.ListSN has MaxLost entries;
7  |   SNSID.HighSN ← CurrSN ;                          // Fresh packet
8  |   return true;
9  else
10 |   if CurrSN is in SNSID.ListSN then
11 |   |   remove CurrSN from SNSID.ListSN;
12 |   |   return true ;                                 // Late packet
13 |   else
14 |   |   return false;                    // Already seen or very late
15 |   end
16 end
```

Alg. 6 tracks the following variables per iPRP session, identified by a sequence number space ID (`SNSID`):

- `HighSN` – highest sequence number of a packet received before the current packet,
- `ListSN` – sequence-number list of delayed packets.

`ListSN` is bounded to a maximum of `MaxLost` $< 2^{31}$ entries. `MaxLost` is the maximum sequence-number difference accepted by the application. In practice, we can take `MaxLost` $> R \times T_{late}$, where R is an upper bound on packet rate of the streaming application that corresponds to an iPRP session and T_{late} is the time after which packets are deemed out-of-date, thus irrelevant. Consequently, if a packet is received with a sequence number that precedes `HighSN` by more than `MaxLost`, it is deemed "very late" and dropped.

The value of `MaxLost` is configurable and depends on the targeted application. For example, in our smart-grid setting, there is a hard delay-constraint of 20 ms (any packet older than this can be safely discarded). To be conservative, we allow packets with the delays of up to $T_{late} = 50$ ms. We set `MaxLost` to 1024, high enough to support any realistic PMU streaming rate.

iPRP and its discard algorithm are able to recover after unexpected events (crashes and reboots). A problem can occur if, after a reboot of a sender, the same sequence numbers are reused. Then, fresh packets can be wrongly discarded as the receiver would be deceived into believing that it had already delivered such packets. This problem can be fixed by imposing handshakes between senders and receivers. However, such a solution is not appropriate if multicast is used and, furthermore, it would violate soft-state property. Our solution is to have a sender maintain a reboot counter that defines different sequence-number spaces within the same sender machine (see Section 5.5.4). Therefore, when a new reboot counter is encountered, the receiver creates a new `SNSID`, thereby resetting `HighSN`. Following a reboot of a receiver, all the receiver's counters are initialized upon the reception of the first iPRP data packet.

As mentioned earlier, the algorithm keeps track of one variable and of one list per iPRP session. The most expensive operation is searching the list (line 10). However, in practice, `ListSN` is limited to few entries. The algorithm can be further optimized for a $\mathcal{O}(1)$ time complexity by using a hash table implementation for `ListSN`. Additionally, the algorithm is designed to have a fixed memory usage: size(`ListSN`) bytes.

Before proving the correctness of the algorithm, we need to introduce some definitions. We say that a received packet is *valid* if it arrives in order or if it is out-of-order but not later than T_{late}. Formally, this means that a packet received at time t with $\texttt{SN} = \alpha$ is not valid if some packet with $\texttt{SN} = \beta > \alpha + \texttt{MaxLost}$ was received before t.

Furthermore, let Δ be an upper bound on the delay jitter across all network subclouds. Formally, for any two packets i, j sent over any two network subclouds k, l: $\Delta \geq \left(\delta_i^k - \delta_j^l\right)$, where δ denotes the one-way network latency. Also, recall that $T_{inactivity}$ is used to terminate inactive sessions (Section 5.5.1).

Theorem 1 (Correctness of the discard algorithm)**.** *If $R \times \Delta < 2^{31}$ and $R \times (T_{inactivity} + \Delta) < 2^{31}$, then Alg. 6 guarantees that: (1) no duplicates are forwarded to the application and (2) the first received valid copy of any original packet is forwarded to the application.*

The proof is lengthy and is given in the appendix of this chapter. To understand the practicality of the conditions in the theorem, note that $T_{inactivity}$ is in the order of seconds and is much larger than Δ. Therefore, the only condition to verify is $R \times (T_{inactivity} + \Delta) < 2^{31}$, which for, say $T_{inactivity} = 10s$ and $\Delta = 100ms$, requires $R < 2 \times 10^8$ packets per second – a rate much higher than ever expected.

5.5.6 The Backoff Algorithm

The soft-state in a multicast iPRP session is maintained by periodic advertisements (`iPRP_CAP`) sent to the source by each member in the multicast group of receivers. We want to prevent "message implosion" at the source for groups of receivers ranging from several hosts to millions. Failing to do so can have a similar effect as a denial-of-service attack. The source would be overwhelmed with processing `iPRP_CAPs` if all the multicast group members would send them. Nevertheless, if the source waits too long before receiving at least one `iPRP_CAP`, the start of the iPRP operation would be delayed. This is why we also require the source to receive an `iPRP_CAP` within at most $D = 10$s after the start of the loop in Alg. 2 (executed periodically every $T_{CAP} = 30$s).

A similar problem was studied in the literature on reliable multicast, where ACK implosion at the source needs to be avoided. To our knowledge, the solution that best fits our scenario was proposed by Nonnenmacher and Biersack [64]. We adopt it in our design: each receiver performs a random backoff before transmitting an `iPRP_CAP`. The source acknowledges each `iPRP_CAP` by an `iPRP_ACK`. The reception of an `iPRP_ACK` before the expiry of the backoff timer inhibits any receiver from sending its `iPRP_CAP`. The backoff timer follows a flipped truncated exponential distribution (inaptly called "exponential" in [64]), defined by a PDF on $[0, D]$ that increases toward D, $f_X(x; \lambda, D) \stackrel{\text{def.}}{=} \lambda e^{\lambda x}(e^{\lambda D} - 1)^{-1} \cdot \mathbb{1}_{\{x \in [0,D]\}}$.

Due to the back-off algorithm, a multicast iPRP sender may not receive `iPRP_CAPs` from the same receiver in two consecutive cycles. As different receivers can have interfaces with different INDs active, the consecutive `iPRP_CAPs` seen by the sender can have different INDs. In cases when an `iPRP_CAP` is received from a receiver with fewer interfaces, if the missing INDs would be immediately removed from the sender's peer-base, the replication on these network sub-clouds would be adversely affected. To mitigate this problem, removal of INDs from the peer-base is done only after confirmation from multiple consecutive `iPRP_CAPs` (see Section 5.5.1).

We implement the backoff computation of [64] by CDF inversion. A uniform random variable $U \in [0, 1]$ is obtained via a random number generator. Next, the backoff is set to $T_{\text{backoff}} = \lambda^{-1} \ln(1 + (e^{\lambda D} - 1)U)$ (Alg. 2, line 2). We pick $\lambda = 25/D$. See [62] for a further discussion.

5.5.7 Robustness and Soft-state

iPRP is a soft-state protocol that is robust against host failures and supports joining or leaving the hosts from the network at any time, independently of each other. In a multicast case, it is expected that a new iPRP-capable receiver can show up (or simply crash and reboot) after an iPRP session with other receivers was established. Then, the new receiver will immediately be able to process packets received at the iPRP data port

without the need to exchange control messages.

The iPRP control-message exchange does not rely on the availability of any particular network subcloud, making our protocol robust to network failures. Once the soft-state maintenance functional block learns about alternative network subclouds, `iPRP_CAP` messages are sent over all of them. Furthemore, the control plane communication to the reserved iPRP control port is secured (see Section 5.6). The security algorithm for iPRP header protection can be chosen as part of the configuration.

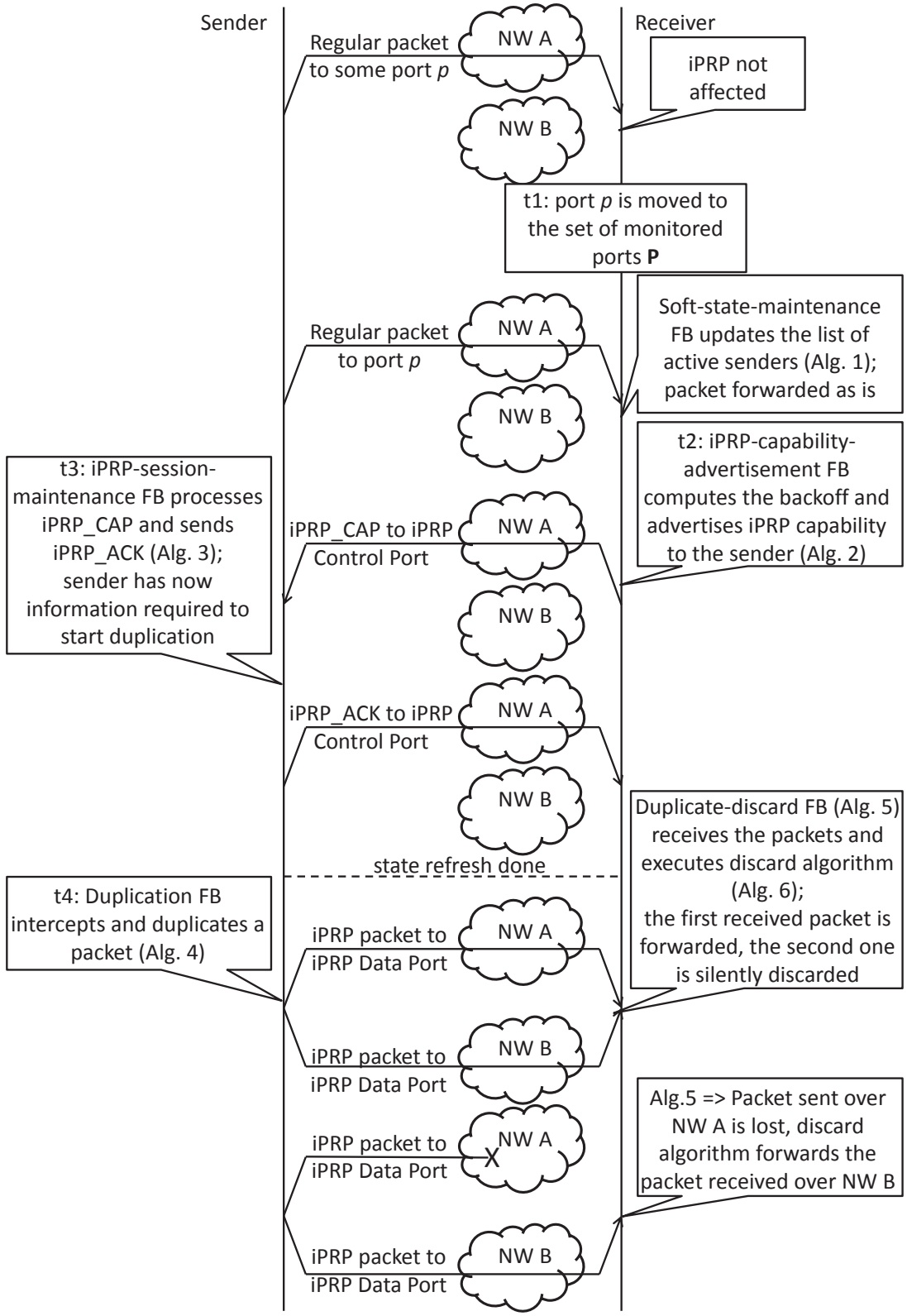


Figure 5.2: Message sequence chart for typical scenario when iPRP-capable devices are starting iPRP operation.

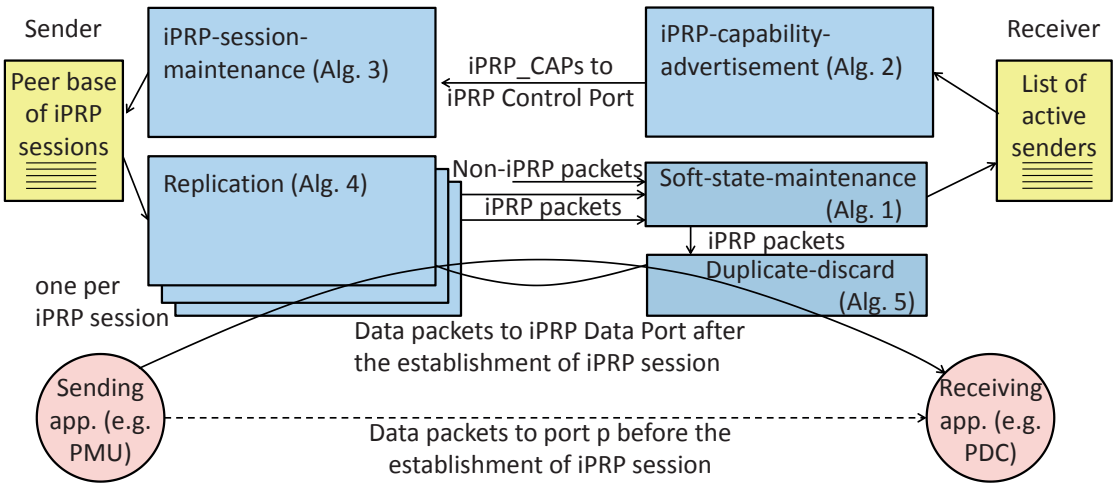


Figure 5.3: Overview of the functional blocks.

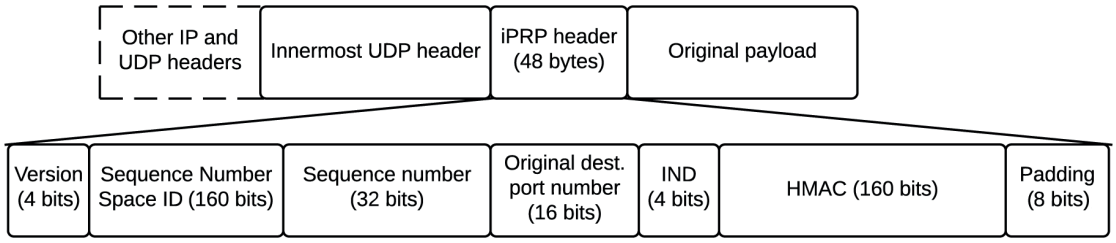


Figure 5.4: Location and fields of the iPRP header.

5.6 Security Considerations

The iPRP protocol design is such that it does not interfere with upper-layer security protocols. However, in addition, we needed to provide security for the iPRP header itself, as there are attacks that can stay undetected by upper-layer security protocols. Concretely, if an attacker manages to alter the sequence-number field of iPRP packets transmitted over one (compromised) network subcloud, the discard algorithm can be tricked in a way that the packets from both (compromised and non-compromised) network subclouds are discarded. Note that similar attacks exist for PRP, where an attacker, with access to one network, can force the discard of valid frames on another network. For example, say an attacker has access to network subcloud A. A PRP frame is represented as $A5$, where A is the network subcloud it belongs to and 5 is the sequence number. If $A5$ and $B5$ were received and the attacker retransmits the frame $A5$ by altering the sequence number as $A6$, then the actual $A6$ and $B6$ frames will both be discarded. In other words, an unsecured PRP or iPRP could weaken the network instead of making it more robust. Yet another argument for protecting the iPRP protocol is that by doing so we minimize the exposure for prospective attacks in the future.

The iPRP control messages are encrypted and authenticated. This guarantees that the security of replicated UDP flows is not compromised by iPRP and that it does not interfere with application layer encryption/authentication.

Specifically, `iPRP_CAP` messages and the corresponding `iPRP_ACK` messages are transmitted over a secure channel. The iPRP header inserted in the data packets is authenticated and encrypted with a pre-shared key. Thus, replay attacks and forged messages insertion are avoided.

We establish the secure channel for the transmission of `iPRP_CAP` messages depending on the type of communication, unicast or multicast. Details follow below.

Unicast: In unicast mode, a DTLS (datagram transport layer security) [65] session is maintained between the sender and the receiver. It is initiated by the receiver upon the arrival of the first UDP datagram from the source. `iPRP_CAP` messages are transmitted within this session. So, the iPRP capabilities of the receiver are transmitted only to an authenticated source. `iPRP_ACK`s are not required in unicast (since message implosion can occur in multicast only).

Unicast `iPRP_CAP` messages contain a symmetric key used to authenticate and encrypt the iPRP header. This key is updated periodically during a unicast iPRP session. Hosts keep a small fixed number of valid past keys to prevent losing the iPRP session because of delayed receiption of a new key. The oldest key is discarded upon reception of a new one.

Multicast: In multicast, iPRP relies on any primitive that establishes a secure channel with the multicast group. For example MSEC (multicast security) [66] can be used for group key management and for establishing a group security association.

In this setting, both `iPRP_CAP` and `iPRP_ACK` messages, as well as the iPRP headers inserted in the replicated packets, are authenticated and encrypted with the group key. Thus, there is no need to include an additional key in the `iPRP_CAP` .

5.7 iPRP Diagnostic Toolkit

As iPRP is designed to be IP friendly, it facilitates the exploitation of the diagnostic utilities associated with TCP/IP. The diagnostics include verification of connectivity between hosts and the evaluation of the corresponding RTTs (similar to `ping`), the discovery of routes to a host (similar to `traceroute`), etc. Furthermore, the toolkit also adds some more tools that are specific to iPRP and it gives iPRP a significant edge in network diagnostics and statistics collection over PRP. The toolkit comprises the following tools:

```
iPRPtest <Remote IP Address> <Port>
         <Number of packets> <Time period>
iPRPping <Remote IP Address>
iPRPtracert <Remote IP Address>
iPRPsenderStats <IP Address>
iPRPreceiverStats <IP Address>.
```

Imagine a typical scenario where an application on an iPRP-enabled host that is subscribed to a particular multicast group (G) experiences packet losses. To troubleshoot this problem, the user at the receiving host would use the `iPRPreceiverStats` tools to consult the local list of active senders, to check for the presence of an iPRP session associated with any host sending multicast data to group G. If an iPRP session exists, then the tool returns the statistics of packets received over different networks in the iPRP session. Then, to understand if the problem is caused by multicast routing or lossy links, the user moves to the sending host.

First, with `iPRPtest` and by using the remote IP address of the receiver, the user establishes a temporary, unicast iPRP session with the host. If successful, the `iPRPping` tool is used to obtain the packet loss and RTT statistics over the multiple networks. Also, the `iPRPtracert` tool is used to verify the hop-by-hop UDP data delivery over multiple networks. For any iPRP session between two hosts, the `iPRPsenderStats` is used by the sending host to query the remote host about the statistics of the packets accepted and dropped by the duplicate discard functional block on that remote host. The operation of each tool is described in detail in [62].

5.8 Implementation

We opted for a Linux-based user-space implementation that has the following properties: (1) Enable the selective filtering of IP packets so that the iPRP sequence of operation can be applied; (2) allow for packet mangling where the iPRP header can be inserted and packets can be replicated at the sender and duplicates can be discarded and original packet can be restored at the receiver; and (3) minimal CPU overhead.

To this end, we use the `libnetfilter_queue` (NF_QUEUE) framework from the Linux *iptables* project. NF_QUEUE is a userspace library that provides a handle to packets queued by the kernel packet filter. It requires the `libnfnetlink` library and a kernel that includes the `nfnetlink_queue` subsystem (kernel 2.6.14 or later). It supports all Linux kernel versions above 2.6.14. We use the the Linux kernel 3.11 with iptables-1.4.12.

The main challenge encountered in the implementation was to ensure that the delay and processing overhead was low. For this purpose, we categorized the various instructions in iPRP as time-critical or non time-critical. For instance, adding the iPRP header to a packet is time-critical, whereas updating the peer-base to enable replication on a new network is non time-critical. Then, we used batching of non time-critical instructions to reduce the total number of system-calls. In this way, we achieved a lower-overhead while maintaining the same real-time performance.

Currently, we are deploying iPRP on our EPFL smart-grid communication network (smartgrid.epfl.ch).

5.9 Performance Evaluation

In order to evaluate the performance of our implementation, we have set up a lab test-bed and do two types of assessment. The first one is to evaluate the operation of iPRP and its discard algorithm in different scenarios. The latter set of experiments is to assess the processing delay due to iPRP and the additional CPU usage used by iPRP software of our proof-of-concept implementation. Our test bed consists of two Lenovo ThinkPad T400 laptops with a 64-bit Ubuntu OS. The laptops (Table 5.1) are connected over two Ethernet based wired networks (one via USB adapter) and one ad-hoc Wi-Fi network. We label the interface eth0 as nwA, eth1 as nwB and wlan0 as nwC. To evaluate the real-time operation, we patched the Linux kernel 3.11.6 with the associated real-time Linux patch *rt29*.

Component	Version Specifications
CPU	Intel C2Duo, 2.53 Ghz
Ethernet Controller	Intel 82567LM Gigabit Card
Wireless Controller	Intel Wifi N d5300
Operating System	Ubuntu 12.04 LTS, 64-bit
Kernel	3.6.11-rt29 (RT-Linux Patch)

Table 5.1: Specifications of hosts used in the test bed

5.9.1 iPRP Behavior in the Presence of Asymmetric Delays and Packet Losses

Our goal here is to validate the design and implementation of iPRP by quantifying the packet losses and delays perceived by an application. We stress-test the discard algorithm with heavy losses and asymmetric delays and compare the performance with that in theory. The packet losses and delays are emulated using the Linux `tc-netem` [67] tool on the test bed described in Table 5.1.

In Table 5.2 we summarize settings used in different scenarios. To mimic the traffic created by PMUs, we send a 280 byte UDP datagram every $20ms$, long enough to have stationary behavior. We emulate delays that are uniformly distributed within $10ms \pm 5ms$ (small differences in network topologies and/or loads), and within $1s \pm 0.2s$ (significant differences in network topologies or serious perturbations in network functioning). We emulate both independent and bursty losses. In both cases the overall packets loss rate is 5%. To produce bursty losses with `tc-netem` we use the Gilbert-Elliot model [68] with $p = 0.01$, $r = 0.19$, $1-k = 0.01$, $1-h = 0.81$.

Scenario	`tc-netem` delay : loss nature		
	nwA	nwB	nwC
0	S:IL	S:IL	S:IL
1	Z:IL	S:IL	not used
2	Z:BL	S:BL	not used
3	Z:IL	L:IL	not used
4	Z:BL	L:BL	not used
5	S:IL	S:IL	not used

Table 5.2: Scenarios used for performance evaluation. `tc-netem` added delay : "Z" means 0 , "S" means small uniform $10ms \pm 5ms$, and "L" means large uniform $1s \pm 0.2s$. Loss nature: "IL" means 5% independent and "BL" means 5% bursty losses.

We use Scenario 0 to evaluate the operation of iPRP in the presence of more than two networks. In Scenarios 1-4, we test the discard algorithm by making asymmetric delays and losses, thus forcing it to keep track of delayed/missing packets. With Scenario 5, we test the expected iPRP side-benefit of having lower average one-way network latency, given that the iPRP duplicate-discard functional block always for-

wards the first packet delivered over any of the available networks. We measure delays and losses over individual networks and as experienced by an application located on top of iPRP.

In Table 5.3, we show the measurement results. We assume that the losses on different networks are independent. Under this assumption, the expected actual loss percentage can be approximated with the product of observed loss percentages on different networks. We compare the observed actual losses (iPRP column) with the expected actual loss percentage (theory column). A deviation would mean anomalies in the iPRP protocol and implementation. The accordance between the last two columns in Table 5.3 shows that iPRP performs as expected in significantly reducing the actual packet losses.

Scen.	nwA	nwB	nwC	iPRP	theory
0	5.061	4.913	5.1537	0.0126	0.0128
1	5.057	5.002	not used	0.253	0.254
2	5.132	5.059	not used	0.259	0.254
3	5.014	5.013	not used	0.251	0.249
4	5.022	4.981	not used	0.247	0.249
5	5.051	5.002	not used	0.251	0.253

Table 5.3: Loss percentages in various scenarios

In Fig. 5.5 we show the CDF of one-way network latency for a specific packet (d_{iPRP}) for Scenario 5. In theory, it should be $d_{iPRP} = min(d_{nwA}, d_{nwB})$, where d_{nwA}, d_{nwB} are the one-way network latencies of the same packet on network sub-clouds A and B. What we measured matches the theory very well. This is a confirmation of the anticipated side-benefit of iPRP: the delays perceived by the application are improved when iPRP is used, compared to those when only one of the individual networks is used.

CDFs are not shown for Scenarios 1-4 as, by construction, it is almost deterministic which network has the shortest latency. For example, in Scenario 1 most of the times $d_{iPRP} = min(d_{nwA}, d_{nwB}) = d_{nwA}$.

5.9.2 Processing Overhead Caused by iPRP

In this subsection, we evaluate processing delays and the additional CPU load when iPRP is used on the test bed described in Table 5.1. We conduct several runs of Scenario 1 (see Table 5.2) and use *GNU gprof* [69] to assess the average processing delay incurred by an iPRP data packet at the *iPRP sender daemon* (ISD) and the *iPRP receiver daemon* (IRD). In an ISD, a data packet encoutners only the *replicator* function which adds the iPRP header and replicates packets over multiple interfaces. This operation takes 0.8

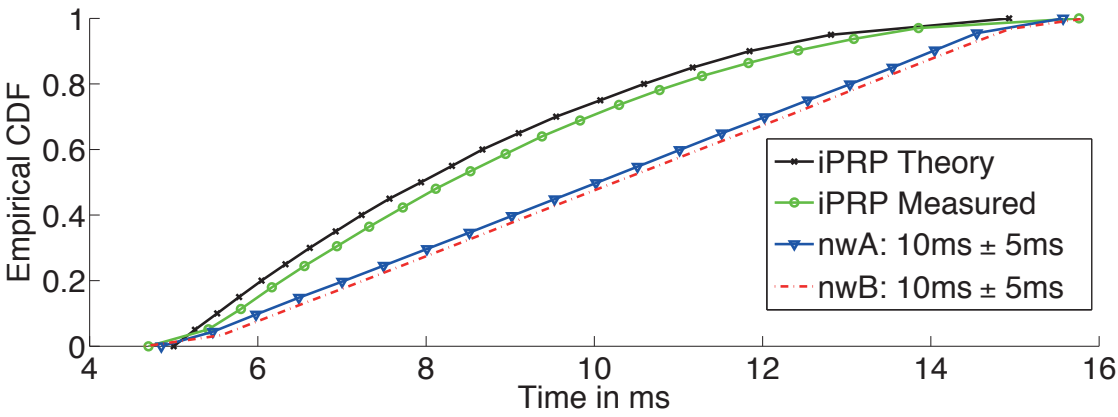


Figure 5.5: An iPRP side benefit: the delays perceived by the application are improved when iPRP is used, compared to those when only one of the individual networks is used.

μs on average. In an IRD, a data packet encounters three functions. The *packet handler* copies a packet into user-space, verifies the fields of the iPRP header and prepares a packet for the *duplicate discard* function which indicates if a packet is to be dropped or forwarded. These operations take 0.8 μs and 0.4 μs on average respectively. Lastly, if a packet is to be forwarded, the iPRP header is removed and checksum is recomputed in 2.4 μs. On average, a data packet incurs a delay overhead of 4.4 μs due to iPRP.

In order to assess the additional CPU load when iPRP is used, we perform two experiments in which we record the CPU usage by iPRP daemons on the sender and on the receiver. The results are summarized in Table 5.4. In Experiment 1 we keep constant aggregate packet rate of 1000 packets per second (pps) for all established iPRP sessions (1 iPRP session of 1000 pps, 2 iPRP sessions of 500 pps each, etc.). The CPU usage with iPRP is quasi-constant at 15 % for the sender and 12 % for the receiver. In Experiment 2 we keep a constant packet rate of 10 pps for every individual iPRP session (1 iPRP session - 10 pps in total, 2 iPRP sessions - 20 pps in total, etc.). At the sender, the CPU usage increases, at first, at a rate of 0.9 % per iPRP session and the increase rate per iPRP session decreases to 0.32 % for larger number of iPRP sessions. At the receiver, the increase rate of CPU usage per each additional iPRP session goes from 0.8 % to 0.22 %.

Following the results from Table 5.4, the additional % of CPU usage at sender (U_s) and receiver (U_r) due to iPRP can be approximated by the relations:

$$U_s = 3.7 + 0.28 \times (\text{\# of iPRP sessions}) + 0.01 \times (\text{packets/s})$$

$$U_r = 0.9 + 0.08 \times (\text{\# of iPRP sessions}) + 0.01 \times (\text{packets/s})$$

Number of sessions	Exper. 1: Aggregate of pps for all iPRP sessions kept constant to 1000		Exper. 2: pps per iPRP session kept constant to 10	
	Send. [%]	Rec. [%]	Send. [%]	Rec. [%]
0 (Idle)	3.7	0.9	3.7	0.9
1	14.5	11.8	4.5	2.2
2	14.1	11.9	5.6	2.4
4	15	11.3	5.7	2.3
10	15	12	7.3	3.2
20	15	12	10	5.2

Table 5.4: CPU usage with iPRP and varying loads

5.10 Conclusion

We have designed iPRP, a transport layer solution for improving reliability of UDP flows with hard-delay constraints, such as smart grid communication, industrial processes, high-frequency trading and online gaming. iPRP is application- and network-transparent, which makes it plug-and-play with existing applications and network infrastructure. Furthermore, our soft-state design makes it resilient to software crashes. Besides unicast, iPRP supports IP multicast, making it a suitable solution for low-latency industrial automation applications requiring reliable data delivery. We have equipped iPRP with diverse monitoring and debugging tools (iPRP diagnostic toolkit - `iPRPtest`, `iPRPping`, `iPRPtracert`, `iPRPsenderStats`, `iPRPreceiverStats`), which is quasi impossible with existing MAC layer solutions. With our implementation, we have shown that iPRP can support several sessions between hosts without any significant delay or processing overhead. To achieve a low delay and processing-overhead, we use batching of non time-critical instructions thereby reducing the total number of system calls.

Interworking with legacy systems could be handled by developing adequate proxies. Legacy hosts that cannot be upgraded with iPRP software could be placed behind an iPRP proxy that would handle all iPRP functions and could thus communicate with an iPRP-enabled host. This would add redundancy to the path between the iPRP proxy and the other end of communication. Nevertheless, the iPRP proxy would now be a single-point-of-failure. A very interesting case arises if the legacy host is equipped with PRP. In such cases, the solution with an iPRP proxy still applies but could be improved using the concept of split proxies in order to remove the single-point-of-failure. Split proxies would each be singly-attached to the PRP LAN and to the IP network. They would insert the iPRP header to each duplicate packet received from the PRP LANs. The challenge, left for future work, is to design a distributed algorithm between the split proxies in order to ensure consistency of iPRP sequence numbers.

Our implementation is publicly available and is currently being installed in our

campus smart-grid [3]. In the future, we intend to do extensive measurements on our smart-grid and study the performance of iPRP in real networks.

To further reduce the delay-overhead due to iPRP, one might think of a more efficient kernel-space implementation. However, given the low delay-overhead of our user-space implementation, it fully satisfies the needs of proof-of-concept implementation. Another possible approach is to push the implementation of iPRP functionalities to the network adapter itself, similarly to the TCP segmentation offload technique [70]. Also, given the slow adoption of IPv6, porting the implementation to IPv4 can be of interest for the future work.

Appendix

5.A Proof of Theorem 1

To prove the statement of Theorem 1, we need the following lemmas.

Lemma 1. *If* $R \times \Delta < 2^{31}$ *and* $R \times (T_{inactivity} + \Delta) < 2^{31}$*, then the wrap-around problem does not exist.*

Proof. The wrap-around problem can arise in two scenarios.

Case 1: A late packet arrives with $\texttt{CurrSN} < \texttt{HighSN} - 2^{31}$. As $R \times \Delta < 2^{31}$, the time required by the source to emit 2^{31} packets is longer than Δ. Hence, HighSN cannot precede CurrSN for more than 2^{31} and this scenario is not possible.

Case 2: A fresh packet is received with $\texttt{CurrSN} > \texttt{HighSN} + 2^{31}$. This means that from the point of view of the receiver, there were more than 2^{31} iPRP packets lost in succession. As $R \times (T_{inactivity} + \Delta) < 2^{31}$, the time for more than 2^{31} consecutive packets to be sent is greater than $(T_{inactivity} + \Delta)$. Hence, the time between reception of any two packets differing by SNs more than 2^{31} is greater than $(T_{inactivity})$. Therefore, during this time the iPRP session would be terminated and a new session will be initiated when the fresh packet is received. Hence, this scenario is also not possible. □

Therefore, in the rest of the proof, we can ignore the wrap-around problem and do as if SNs of received packets were integers of infinite precision. Also, a notation such as $\texttt{HighSN}_{t^-}$ [resp. $\texttt{HighSN}_{t^+}$] denotes the value of HighSN just before [resp. after] time t.

Lemma 2 (Monotonicity of HighSN)**.** *If at time* t*, a packet with* $\texttt{SN} = \alpha$ *is received, then* $\texttt{HighSN}_{t^+} = \max(\texttt{HighSN}_{t^-}, \alpha)$*. Therefore,* HighSN *increases monotonically with time.*

Proof. From Alg. 6, when $\alpha > \texttt{HighSN}_{t^-}$ (line 4) then the value of HighSN is changed to α (line 7). Otherwise, when $\texttt{HighSN}_{t^-} \geq \alpha$ (lines 2 and 9), HighSN is unchanged, i.e.,

$\texttt{HighSN}_{t^+} = \texttt{HighSN}_{t^-}$. The two cases combined together give $\texttt{HighSN}_{t^+} = \max(\texttt{HighSN}_{t^-}, \alpha)$. □

Lemma 3 (Fresh packet is never put in ListSN)**.** *If at time t, a packet with $\texttt{SN} = \alpha$ is forwarded to the application then $\alpha \notin \texttt{ListSN}_{t'^+} \forall\, t' \geq t$.*

Proof. Let us prove by contradiction. Assume that $\exists\, t' > t$ such that $\alpha \in \texttt{ListSN}_{t'}$. Hence, $\exists\, t_1 \in (t, t']$ when α was added to ListSN. As $t_1 > t$, from Lemma 2, we conclude that $\texttt{HighSN}_{t_1^-} \geq \texttt{HighSN}_{t^+} \geq \alpha$. Now, from Alg. 6, we know that only SNs $> \texttt{HighSN}_{t_1^-}$ can be added to ListSN. Hence, α cannot be added to ListSN at time t_1. Therefore, we have a contradiction. □

Lemma 4. *At any time t, $\texttt{HighSN}_{t^-}$ is equal to SN of a packet received at some time $t_0 < t$ or no packet has been received yet.*

Proof. HighSN is modified only at line 7, where it takes the value of the SN received. Hence, HighSN cannot have a value of a SN that has not been seen yet. □

Now, we proceed with the proof of the theorem. First, we prove statement (1). Assume we receive a duplicate packet with $\texttt{SN} = \alpha$ at time t. It means that a packet with $\texttt{SN} = \alpha$ was already seen at time $t_0 < t$. Then, from Lemma 2 it follows that $\alpha \leq \texttt{HighSN}_{t^-}$. Then, either $\alpha = \texttt{HighSN}_{t^-}$ (line 2) or $\alpha < \texttt{HighSN}_{t^-}$ (line 10).

Case 1: When $\alpha = \texttt{HighSN}_{t^-}$, the packet is discarded according to line 3.

Case 2: When $\alpha < \texttt{HighSN}_{t^-}$, line 10 is evaluated as false due to Lemma 3. Hence, the packet is discarded by line 14.

Next, we prove statement (2) by contradiction. Assume we receive a first copy of a valid packet with $\texttt{SN} = \alpha$ at time t but we do not forward it. This can happen either due to line 3 (case 1) or due to line 14 (case 2).

Case 1: Statement from line 2 was evaluated as true, which means that $\alpha = \texttt{HighSN}_{t^-}$. As $\texttt{SN} = \alpha$ is seen for the first time, Lemma 4 is contradicted. Hence, this case is not possible.

Case 2: Statement from line 10 was evaluated as false, which means that $\alpha < \texttt{HighSN}_{t^-}$ and $\alpha \notin \texttt{ListSN}_{t^-}$. We show by contradiction that this is not possible, i.e., we now assume that $\alpha < \texttt{HighSN}_{t^-}$ and $\alpha \notin \texttt{ListSN}_{t^-}$. Now, there are three cases when $\alpha \notin \texttt{ListSN}_{t^-}$ can be true:

(i) $\texttt{SN} = \alpha$ was added to and removed from ListSN before time t because it was seen (line 11) which is impossible as the packet is fresh.

(ii) $\texttt{SN} = \alpha$ was added to $\texttt{ListSN}$ and later removed at time $t_0 < t$ because the size of $\texttt{ListSN}$ is limited to $\texttt{MaxLost}$ entries (line 6). This means that at time $t_0 < t$ a packet with $\texttt{SN} = \beta$ was forwarded and $\beta - \alpha > \texttt{MaxLost}$ (line 6). However, this means that the packet with $\texttt{SN} = \alpha$ was not valid at time t_0 and therefore is also not valid at time $t > t_0$.

(iii) $\texttt{SN} = \alpha$ was never added to $\texttt{ListSN}$. Consider the set $\mathbb{T} = \{\tau \geq 0 : \texttt{HighSN}_{\tau^+} > \alpha\}$. $\mathbb{T}$ is non-empty because $t \in \mathbb{T}$, by hypothesis of our contradiction. Let $t_0 = \inf \mathbb{T}$. Then, necessarily $\texttt{HighSN}_{t_0^-} \leq \alpha < \texttt{HighSN}_{t_0^+}$ (say, $= \beta$). β is the $\texttt{SN}$ of a packet received at time t_0. Since α is valid, $\beta - \alpha < \texttt{MaxLost}$. Otherwise, α would be invalid at time t_0, therefore at time t, which is excluded. Then we have two subcases possible:

a) $\texttt{HighSN}_{t_0^-} < \alpha$. Then, by line 5, α is added to $\texttt{ListSN}$, which is a contradiction.

b) $\texttt{HighSN}_{t_0^-} = \alpha$. But, by Lemma 4 a packet with $\texttt{SN} = \alpha$ must have been received before t_0 which is a contradiction because α is a fresh packet at $t \geq t_0$.

6 Performance Comparison of Node-Redundant Multicast-Distribution Trees

6.1 Introduction

Some of the industrial processes with hard real-time constraints require the creation of redundant multicast-distribution trees to support 0-ms packet repair-time. For example, in the emerging smart-grids there are applications such as grid protection or grid control that are considered critical. We were faced with such requirements when we designed the communication network as part of the EPFL smart-grid project (smartgrid.epfl.ch).

Solutions such as PRP [20] and iPRP [71] have been deployed in these settings; they achieve 0-ms packet repair-time by duplicating packets over node-redundant multicast-distribution trees. This is commonly done in industry by having dedicated and duplicated networks. In this chapter, we study how to construct node-redundant multicast trees that can be used *in parallel* over a *shared network infrastructure*. This is of interest to telecom operators and manufacturers who want to propose solutions over their shared infrastructures [61].

Our work differs from [34] and [72], as the solutions in [34] and [72] establish one multicast tree as a backup tree that is activated only when a link or node failure is detected in the primary tree. This require extra repair-time delay as the detection of a failure is a time consuming process. Our solution however relies on the parallel use of node-redundant trees, hence it provides a 0-ms packet repair-time.

The approach that we undertook is applicable to any mechanisms for network control that have global information about the state of the communication network. The working assumption is such that there exists a centralized or distributed controller

that can impose routing rules to all the nodes that are part of the network.

In this paper, we focus our study on SDN-based networks because we believe that this is the technology that is very compatible with the assumption and that would be deployed in practice in this case by most network managers. SDN technology is the most likely candidate for the design of 5G systems [73, 74] as it provides a fully programmable operator-network interface. An SDN controller has global information of the situation in the network and can enforce appropriate forwarding rules to the SDN switches. This should therefore facilitate the implementation of the proposed algorithms. Nevertheless, there is a question of the choice of the appropriate algorithm to be implemented for the purpose of creating node-redundant multicast-distribution trees.

This problem is proven to be NP-complete [75] and approximation methods for solving it are the only possible. To that end, we evaluate the performance of three algorithms that we adapted to the smart-grid setting: (i) `ReducedCostV` [76] computes a pair of node-redundant spanning trees for a source, (ii) `MADSWIP` [77] computes a pair of maximally disjoint paths between a source and each of its destinations, and (iii) `Takahashi - Matsuyama` [75] finds a pair of minimum-cost Steiner trees between a source and all its destinations. Adaptations of the first two algorithms provide us with a pair of node-redundant multicast trees, whereas the third one constructs a pair of node-disjoint multicast trees.

We analyze the following metrics: network-utilization efficiency (number of sources placed, minimum/maximum number of hops to the destination) and the number of forwarding rules that need to be installed at SDN switches, with and without aggregation. We quantify performance of different methods based on these metrics when applied on networks with structured (operator) topology or on random networks. Our results show the following:

- In terms of the numbers of sources placed, for general networks, `ReducedCostV`[1] and `MADSWIP` perform the best as in some cases, especially for random topologies with an increasing number of destinations, `Takahashi-Matsuyama` is unable to place the second tree for any sources, hence it is not a viable solution. Nevertheless, for structured, dual-plane-like networks, `Takahashi-Matsuyama` gives results comparable to the two other algorithms.
- In terms of the number of hops to the destinations, `MADSWIP` and `Takahashi-Matsuyama` (if source placement is possible) outperform `ReducedCostV` as they construct paths that are 2 to 3 times shorter than those established by `ReducedCostV`. They can therefore provide a much better delay guarantee.

[1] We call "`ReducedCostV`" the adaptation of `ReducedCostV` and similarly with `MADSWIP` and `Takahashi-Matsuyama`.

- Using MADSWIP and Takahashi-Matsuyama (if source placement is possible), the total number of rules that need to be installed in the network is 5 to 6 times smaller than ReducedCostV when no flow-rule aggregation is applied. If applied, the number of installed rules is reduced to handful of rules for all the algorithms.
- In terms of the SDN-control-plane activity in the case of change in network topology (node failure or new source), MADSWIP and Takahashi-Matsuyama require fewer changes compared to ReducedCostV. If the change of interest is an arrival or departure of a destination within a multicast group, ReducedCostV requires no changes, hence it outperforms the two other algorithms in this case.

The rest of this chapter is structured as follows. In the next section, we cover the related work. In Section 6.3, we provide the problem formulation. In Section 6.4, we introduce three different types of algorithms that we study here, to construct node-redundant multicast trees. In Section 6.5, we analyze their performance and in Section 6.6 we discuss the effects of topology changes on the SDN control-plane activity. In Section 6.7, we provide the conclusion.

6.2 State of the Art

An extensive overview of the algorithms for network survivability was done by Kuipers in [78]; it served as an excellent reference in classifying methods for the creation of multicast-distribution trees.

We distinguish three different families of algorithms that are of our interest and we choose for this evaluation one representative from each of the families. First, there are algorithms that find *spanning trees*. In the literature, we find many solutions that are based on placing one tree and then removing links and nodes already used, before placing the second trees. Such solutions are suboptimal and can lead to the inability to place the second tree. A better solution is presented by Médard et al. in [79] and we choose its extension (algorithm ReducedCostV) by Zhang et al. [76] as the representative of the algorithms that find node-redundant spanning trees.

The second family of algorithms find *Steiner trees*. As for the spanning trees, there are many solutions based on placing one Steiner tree and then removing links and nodes already used, before placing the second one. Surprisingly, to the best of our knowledge, there are no practical algorithms for concurrent tree-construction. We leave this problem for future work and for this comparison we use the method of sequential placing of Steiner trees constructed with the algorithm from [75]. We refer to this algorithm as Takahashi - Matsuyama.

Finally, we have algorithms that find *disjoint paths*, where many algorithms are inspired by the work of Suurballe and Tarjan [80]. We single out the work of Nina

Taft-Plotkin et al. [77] (MADSWIP algorithm) and Guo et al. [81] (DIMCRA algorithm). MADSWIP computes maximum-bandwidth maximally disjoint paths and minimizes the total weight as the secondary objective, whereas DIMCRA finds two link-disjoint paths subject to multiple quality-of-service constraints. In our smart-grid setting, metrics of interest are only bandwidth and delay, hence, for the comparison, we choose MADSWIP as the representative of the algorithms that find disjoint paths.

The application of SDN technology for reliable smart-grid communication networks is considered in [34] and [72]. Neverthless, despite insisting on the need for reliable packet delivery, the described settings are not compatible with a reliability brought by iPRP-like $0ms$ repair-time. In [34], the authors describe `Pcount`, an algorithm that uses OpenFlow to detect link failures that should trigger the activation of backup multicast-distribution trees. However, the time-scale in which `Pcount` operates is in the order of seconds, which is unacceptable for critical processes in smart grids. Similarly, in [72], the approach of bypassing the link that has failed *after* the failure occurs with the precomputed backup segment is not acceptable for critical traffic.

The parallel redundancy protocol (PRP) [20] requires redundant, and, moreover, cloned networks; hence it cannot be used in a shared infrastructure. TCP-like solutions are acceptable for applications with less strict delay-requirements, as it takes several round-trip times for losses to be repaired. Network coding [82] is commonly used to improve network's throughput and reliability but can perform poorly in terms of latency. Source coding (e.g., Fountain codes [59]) is suitable in the case of bursty packet transmissions where encoding and decoding are performed across several packets. However, in smart grids, usually there is a single packet per time-slot and it should be sent as soon as it is available to avoid unnecessary delays.

6.3 Problem formulation

Let $(\mathcal{V}, \mathcal{E})$ be a *directed* graph representing our network, with the set of vertices $\mathcal{V}$ and the set of edges $\mathcal{E}$. Let $B(i,j)$ be the bandwidth assigned to $(i,j) \in \mathcal{E}$ and $C(i,j)$ be the cost of using the link. We denote by $M(s) = \{s, D(s)\}$ a multicast session with source $s \in \mathcal{V}$ and destinations $D(s) \subseteq \mathcal{V} - s$. We study algorithms to construct a pair of node-redundant multicast trees for this session such that a failure of a node (vertex) in the network leaves each destination in $D(s)$ still connected to the source by using at least one of the trees.

These node-redundant trees are not necessarily disjoint and can contain some nodes or edges in common. Hence, our study is different from those that propose the use of disjoint spanning trees; those studies are based on placing one tree and then removing links and nodes already used, before placing the second tree, because they are suboptimal and can lead to the inability to place the second tree [79]. For an

example of one such case, we refer to Figure 6.1. Node 1 is the source of packets and there are two destinations of interest: nodes 5 and 7. The blue tree follows the path 1-3-5-6-7, whereas the red tree follows the path 1-4-7-6-5. We observe that the two trees share node 6 thus making them non-disjoint. Nevertheless, both destinations can be reached upon a failure of a single node (node-redundancy property), which is the goal we want to accomplish. Note that in this example it is not even possible to construct two trees that are disjoint.

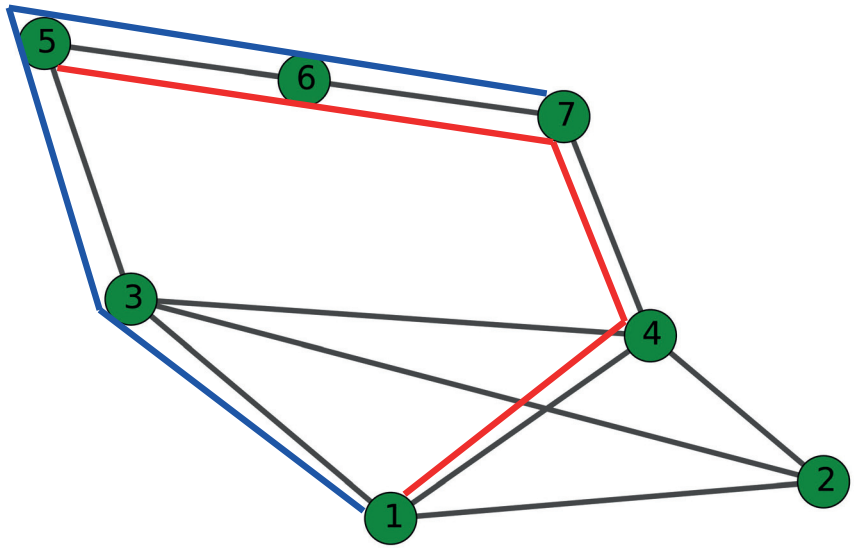


Figure 6.1: Source: Node 1. Destinations of interest: 5 and 7. Blue and red trees are node-redundant but they are not disjoint (node 6 is shared).

For comparison, we define the following metrics of interest:

- Number of sources placed. This metric illustrates how many sources can be served, depending on the approach taken. We say that a source is placed only if it is possible to establish node-redundant trees that comprise all the destinations in the group of receivers, subject to the capacity constraints.
- Minimum number of hops between each source and each destination. Given that in all cases we have two paths between each source and each destination, this metric illustrates the effective number of hops (delay) in the case when both paths are operational.
- Maximum number of hops between each source and each destination. Given that in all cases we have two paths between each source and each destination, this metric illustrates the effective number of hops (delay) in the worst case if one path becomes unavailable.
- Number of flow rules installed per source placed. This metric illustrates how big

the SDN flow tables are. It is desirable to have a smaller number of flow-table entries as the switches can be of limited resources.

- Number of flow rules installed per source placed. This metric illustrates how many rules need to be installed at the switches by the controller and how large the SDN flow tables are. The larger the number of rules installed is, the higher the volume of control traffic exchanged is between the controller and switches. This can, on one hand, overload the controller and, on the other hand, saturate the control plane. A larger number of installed rules also means larger flow tables, which can affect the performance of SDN switches with limited resources.

6.4 Methods Used in this comparison and necessary adaptations

As mentioned in Section 6.2, we choose `ReducedCostV`, `Takahashi - Matsuyama` and `MADSWIP` as the basis for the methods based on spanning trees, Steiner trees, and disjoint paths, respectively. In order to compare these approaches, it was necessary to harmonize the assumptions and to adapt the algorithms so that they all produce multicast-distribution trees. We discuss below the necessary adaptations, after describing the algorithms themselves.

6.4.1 `ReducedCostV`

This algorithm finds a pair of directed node-redundant spanning trees (referring to as T^R and T^B trees) for a given source in the network. As stated by Xue et al. [83], the construction of these trees is related to ear decomposition of the graph. Adapting this idea, `ReducedCostV` constructs T^R and T^B from a DFS (depth first search) tree $\mathcal{T}$ by applying the ear decomposition technique. This is done by adding an ear whenever a back edge from $\mathcal{T}$ to the current T^R and T^B, which span a subset of the network nodes, is encountered. The resulting T^R and T^B trees have a total cost that is minimum among all possible node-redundant spanning trees.

As mentioned in Section 6.3, we analyze directed graphs. `ReducedCostV` takes as input undirected graphs, whereas it outputs directed graphs. This means that, after placing the first source and updating the available link capacities, the resulting graph has available link capacities that are asymmetric. Hence, it cannot be treated as undirected, which is the problem for the next iteration of the simulation when the next source needs to be placed. For a solution, for every link, we keep track of the available capacities in both directions; and for the input for the algorithm, we take the minimum of the two.

Furthermore, for each source that offers traffic `ReducedCostV` computes a pair of

spanning trees that already have a property of having no single point of failure. By construction, all the destinations from a group of receivers that belong to the same multicast group will be reachable, hence no further adaptation is required. We sort the sources based on the offered traffic and we treat them in that order. Before placing a source, we remove the links that do not have enough capacity to support the offered traffic.

6.4.2 Takahashi - Matsuyama

[75] proposes a heuristic to compute minimum-cost Steiner tree for a given source node and a set of destinations in a network. This heuristic performs as follows. It first finds the closest destination to the source and constructs the path between the source and the destination. It then selects the closest non-spanned destination to this tree and updates the tree by adding the path between this destination and the tree. It repeats this process until all the destination nodes are covered. The resulting Steiner tree has the minimum total cost.

Takahashi - Matsuyama also expects the input in the form of undirected graphs, whereas it outputs directed graphs. We deal with this aspect in the same way as in the case of the method based on spanning trees (see the description of ReducedCostV approach).

An additional adaptation is needed as Takahashi - Matsuyama provides a single Steiner tree (not a pair). Hence, we use the method (e.g., described in [84]) where after placing the first Steiner tree, we remove the already used nodes and links. After this we run again the Takahashi - Matsuyama algorithm for the creation of the second Steiner tree, that is now node-redundant by construction. We sort the sources based on the offered traffic and that is the order in which we treat them. Before placing a source, we remove the links without enough capacity to support the offered traffic.

6.4.3 MADSWIP

This algorithm computes a pair of maximally disjoint paths between the source and its destinations such that either the total cost of paths is minimized or the bandwidth is maximized. It performs as follows. First, a shortest-path tree from the source to the destination is computed.The edge costs and bandwidths are updated, based on the computed shortest path, and the nodes are labeled. Using this labelling information, a pair of disjoint paths from the source to all destinations is computed.

In the case of MADSWIP, both input and output graphs are directed. Hence, in this respect, no changes are needed. Still, there is another adaptation that is required. Our goal is to find node-redundant multicast trees, whereas MADSWIP produces link-disjoint

paths. Therefore, we needed firstly to adapt the MADSWIP to produce node-disjoint paths (instead of link-disjoint) before combining them into node-redundant multicast trees. The method that we used is described in [78]; we split each node u into two nodes u_1 and u_2, with a directed link (u_1, u_2), and the incoming links of u connected to u_1 and the outgoing links of u departing from u_2.

Once we have node-disjoint paths, there is a question of how to combine them into node-redundant trees. Each source has a number of destinations that correspond to the same multicast group. Each execution of [77] produces a pair of paths originating from the source, one pair per destination. This raises the question of how to combine disjoint paths into trees: For a single path-pair, which path should join which of the two output trees? We use the following heuristic to solve this problem. Again, we sort the sources based on the offered traffic and that is the order in which we treat them. For each source, we pick randomly the first destination (from the group of multicast receivers) and we compute disjoint paths. The two solution trees (named blue and red) are initialized with these paths. After every subsequent computation of disjoint paths (for the other destinations from the same multicast group), we decide which path should be combined with which tree, based on the minimal number of *new* links that need to be added. We also remove the links without enough capacity to support the offered traffic for the next computation.

Furthermore, to have a fair comparison, we accept paths from MADSWIP only if they are *completely* disjoint (MADSWIP finds *maximally* disjoint paths). For the same reason, we implemented a version of MADSWIP that computes minimum-cost disjoint paths (instead of maximum bandwidth).

6.5 Performance Evaluation

In this section, we provide performance analysis. We first describe our simulation setting.

6.5.1 Network scenarios

Random (ad-hoc) topology: We generate a topology that resembles topologies of wireless sensor networks (see Figure 6.2). We place 100 switches in a $1000m \times 1000m$ area as follows: The positions are generated uniformly at random, with restrictions that no switches are within $75m$, and the connectivity between switches is ensured if the distance between them is below $150m$. All the links have capacity of 1.

We randomly select a subset of nodes that serve as destinations. Depending on the number of destinations, we distinguish scenarios that correspond to *sparse* multicast (3, 4, ... , 10 receivers) and *dense* multicast (15, 20, 25, ... , 40 receivers). In the sparse

multicast case, we select destinations in such a way that the distance between any pair of destinations is at least $100m$. The idea is that several receivers are required for reliability reasons and thus they should not be too close to each other.All the nodes that are not designated as receivers are the candidate sources with the traffic rate uniformly distributed in the interval $[0.025, 0.05]$. Our algorithms try to compute node-redundant trees for as many of them as possible.

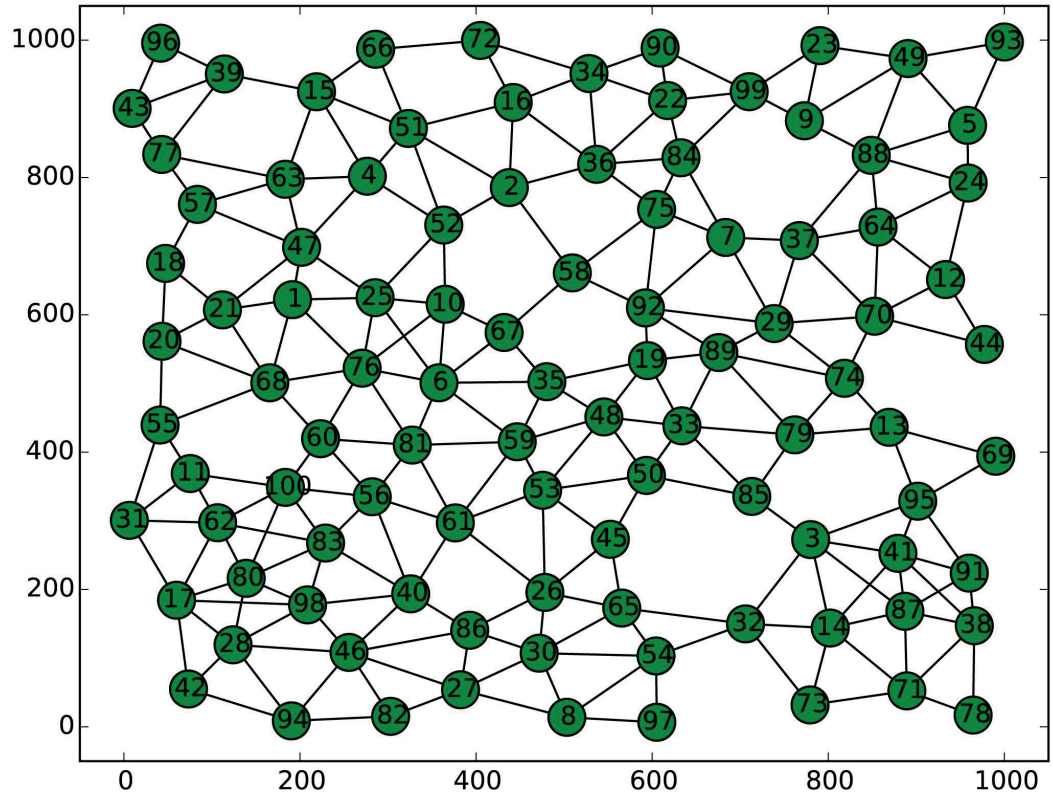


Figure 6.2: Random (ad-hoc) topology.

Structured (operator) topology: We also study the performance of mechanisms proposed in the previous section on a topology generated from [85, 86]; it represents the backhaul of carrier networks. It is hierarchical and consists of three layers: access, aggregation, and core.

The access layer consists of clusters of 20 access switches, each connected to two neighboring aggregation switches (to provide redundancy between access and aggregation layers). The aggregation layer consists of 4 *pods*, each with 4 switches connected together in a full mesh.

Two of the switches in a pod are connected to each of 20 access switches in a cluster and the remaining two switches are connected to two core switches. The core layer consists of 4 switches connected together in a full mesh. We therefore have a network of size 100 with 80 access switches, 16 aggregation switches and 4 core switches.

The links between access switches and aggregation switches have a capacity of 1. The links between aggregation switches (within a pod) have a capacity of 20. The links between aggregation switches and core switches have a capacity of 40. The resulting network is depicted in Figure 6.3.

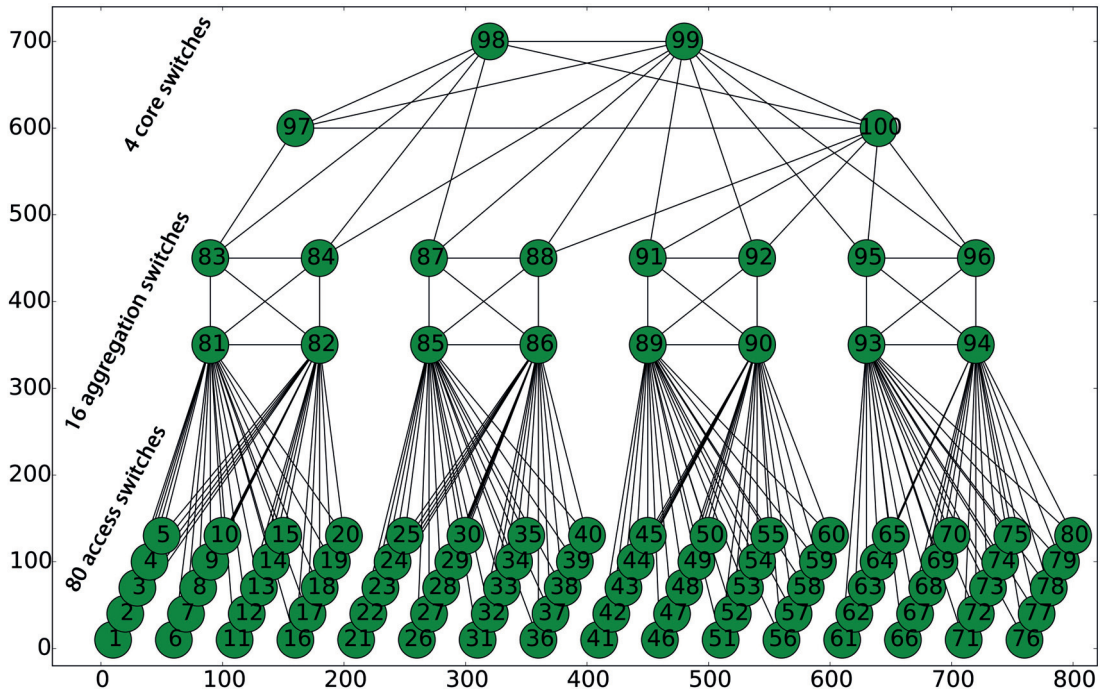


Figure 6.3: Structured (infrastructure) topology.

We randomly select subset of nodes that serve as destinations from the group of access nodes. As for the random topology case, depending on the number of destinations, we distinguish scenarios that correspond to *sparse* multicast (3, 4, ... , 10 receivers) and *dense* multicast (15, 20, 25, ... , 40 receivers). All the nodes from the group of access nodes that are not designated as receivers are the candidate sources with the traffic rate uniformly distributed in the interval $[0.025, 0.05]$. Our algorithms try to compute node-redundant trees for as many of them as possible.

6.5.2 Results

The results presented here are the output of 20 simulations with different seeds. Hence, every simulation has a different group of sources/destinations and different offered traffic for the sources. We show the average values and the confidence interval for the number of sources placed and the number of installed rules (with and without aggregation).

Random (ad-hoc) topology, sparse multicast:

As shown in Figure 6.4, there is no significant difference in the number of sources that can be placed when the number of destinations is fewer than seven. Once the number of destinations grows bigger, `Takahashi - Matsuyama` starts paying the price of the fact that it is the only algorithm that does not construct trees in parallel. Consequently, with more than seven destinations, after placing the first of the two trees, it becomes impossible to place the second tree for more than only a few sources. The main bottleneck in how many sources can be placed for the other two algorithms is the aggregated capacity of the links to the destination node with minimum degree, among all destinations. For example, node 93 in the upper-right corner (Figure 6.2) has only two edges that connect this node to the rest of the graph. Hence, the aggregated capacity that is shared among two tree branches that reach this destination is 2. Therefore, we cannot place more than $21-23$ sources, as the aggregated traffic of this many sources in decreasing order of offered traffic is close to 1. In the case of node 42 (lower-left corner) this capacity is 3. So, if the node 42 is the node with minimum number of edges in the multicast group, the limiting capacity is 50% higher than in the case of node 93, and more sources can be placed (~ 35). The more destinations we have the higher probability is that one of the nodes with two edges will be in the group of receivers; and this is why we converge to $21-23$ sources placed.

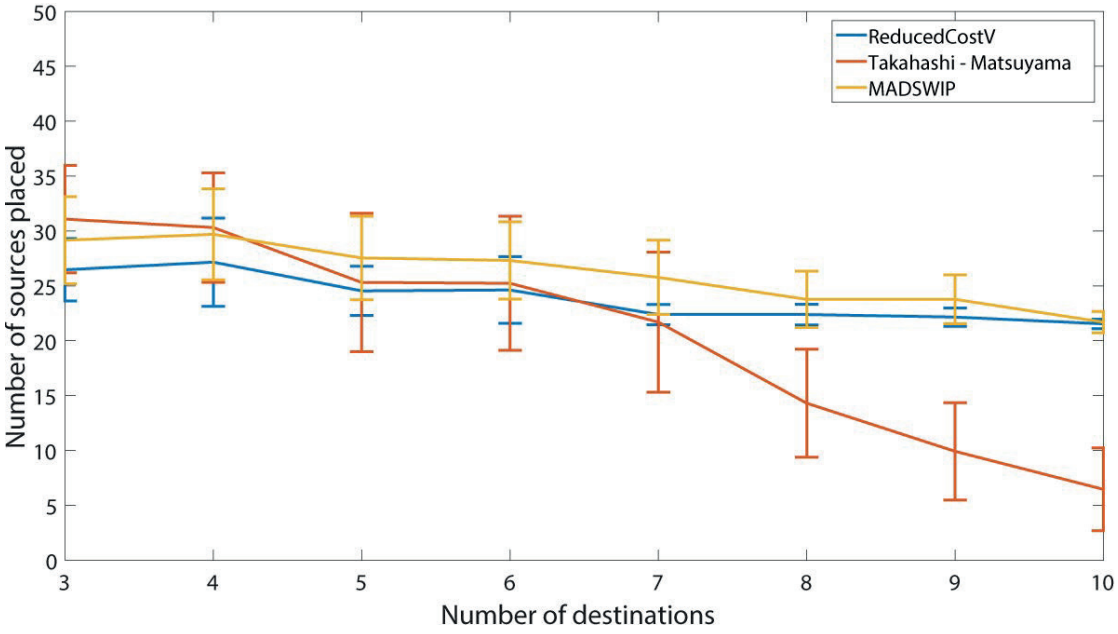


Figure 6.4: Number of sources placed for random topology, sparse multicast.

When it comes to the minimum number of hops between sources and destinations (CDFs depicted in Figures 6.5 and 6.6), `ReducedCostV` is dominated by the other two. This is expected, as `ReducedCostV` does not have as a goal creation of short paths to specific nodes; the goal is simply to cover all the nodes of a given graph.

Looking at the number of rules that need to be installed, both with and without rule

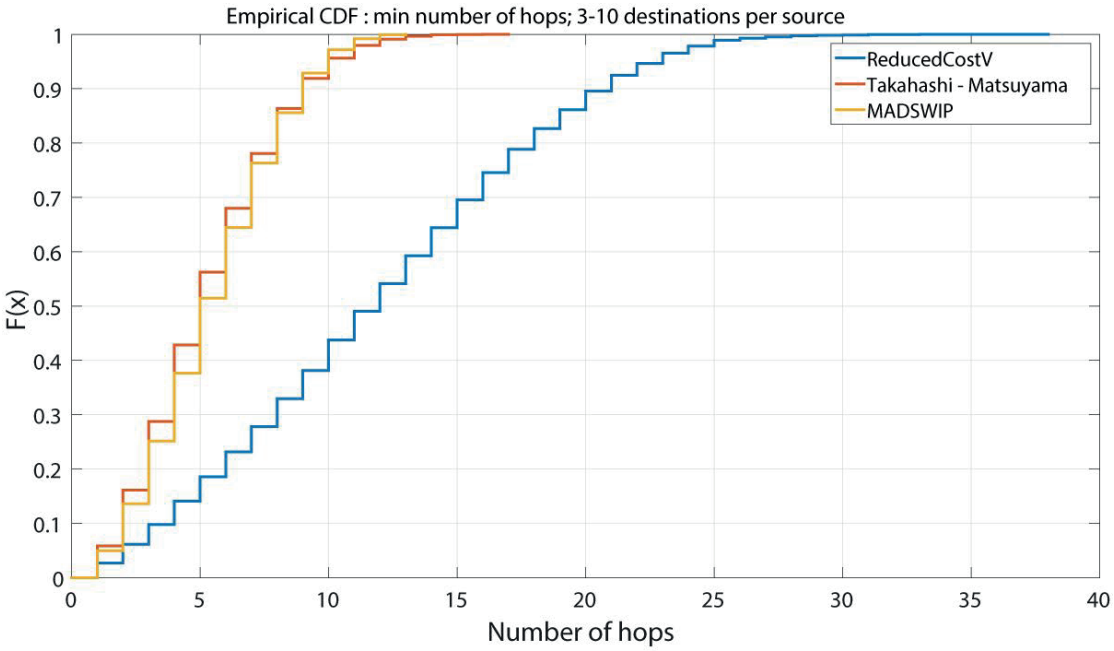


Figure 6.5: Minimum number of hops for random topology, sparse multicast.

aggregation (Figures 6.7 and 6.8), we see that `ReducedCostV` is the worst one; but the difference is not so significant if the aggregation is applied. We do not show results for `Takahashi - Matsuyama` for more than seven destinations as, afterward, the number of sources placed becomes very low. For fewer destinations, we see that `Takahashi - Matsuyama` gives slightly better results than `MADSWIP`, as every additional destination is added by minimizing the distance from the nearest node in an already established tree to other destinations within the same multicast group. Whereas, in the case of `MADSWIP`, every destination within a multicast group is treated completely separately.

Random (ad-hoc) topology, dense multicast:

The trends stay the same when we analyze dense multicast case. The number of sources that can be placed stays stable for `ReducedCostV` and `MADSWIP`, whereas the decreasing trend for `Takahashi - Matsuyama` continues, which means that no, or very few, sources can be placed (Figure 6.9).

The conclusions are unchanged when it comes to the minimum number of hops between sources and destinations (CDF shown in Figure 6.10), `MADSWIP` is better than `ReducedCostV` and `Takahashi - Matsuyama` is not shown because almost no sources were placed. The same conclusions are valid for the maximum number of hops between sources and destinations (CDF shown in Figure 6.11). However, `MADSWIP` loses its dominance when it comes to the number of rules that need to be installed both with and without rules aggregation (Figures 6.12 and 6.13). Simply, with the increasing number of destinations, the number of affected nodes grows for `MADSWIP` and, in the

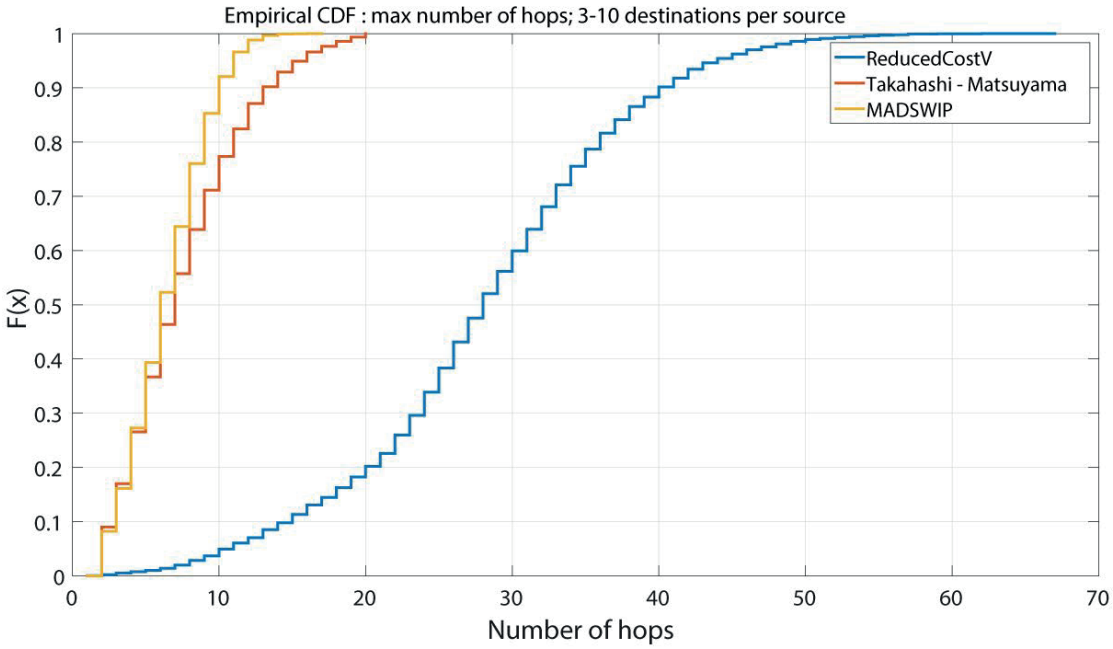


Figure 6.6: Maximum number of hops for random topology, sparse multicast.

case of `ReducedCostV`, all the nodes are affected no matter how many destinations there are.

Structured topology, sparse multicast:

Before we analyze the results for the structured topology, we should make one observation. This topology resembles dual-plane topologies and, given that the capacities at the core and aggregation levels are sufficient, the bottlenecks will simply be the links that are connected to the destinations that are part of the multicast group. All the algorithms are “smart enough" to discover two planes, even though they are not explicitly defined so the number of sources places is comparable in all cases, see Figure 6.14. Again, we converge to $21 - 23$ sources placed, as the aggregated traffic of that many sources (in decreasing order of offered traffic) is close to 1, which is the capacity of the links to the destinations mentioned above.

As for the rest of the results (Figures 6.15 - 6.18), for the number of hops, we see again that `ReducedCostV` is dominated by the two others that are comparable. The same holds for the number of SDN rules that need to be installed, and again, the difference is not so significant with route aggregation.

Structured topology, dense multicast:

In this scenario, the bottlenecks are the same as for the sparse case: the links that are connected to the destinations that are part of the multicast group. Consequently,

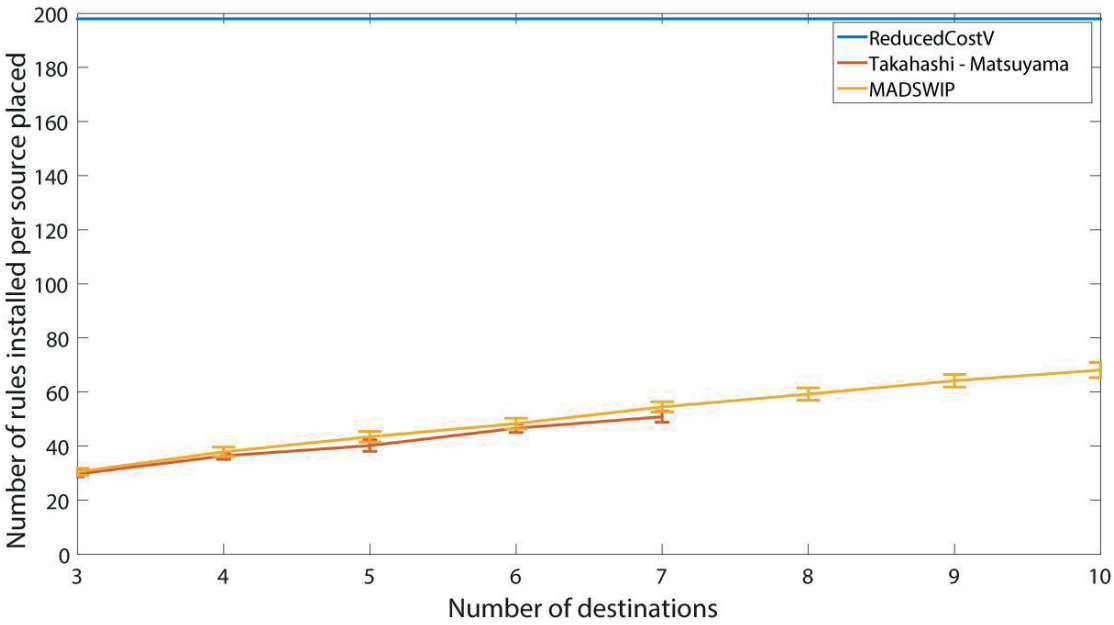


Figure 6.7: Number of rules installed without aggregation for random topology, sparse multicast.

all the conclusions are the same as for the sparse case, and the corresponding graphs are depicted in Figures 6.19 - 6.23.

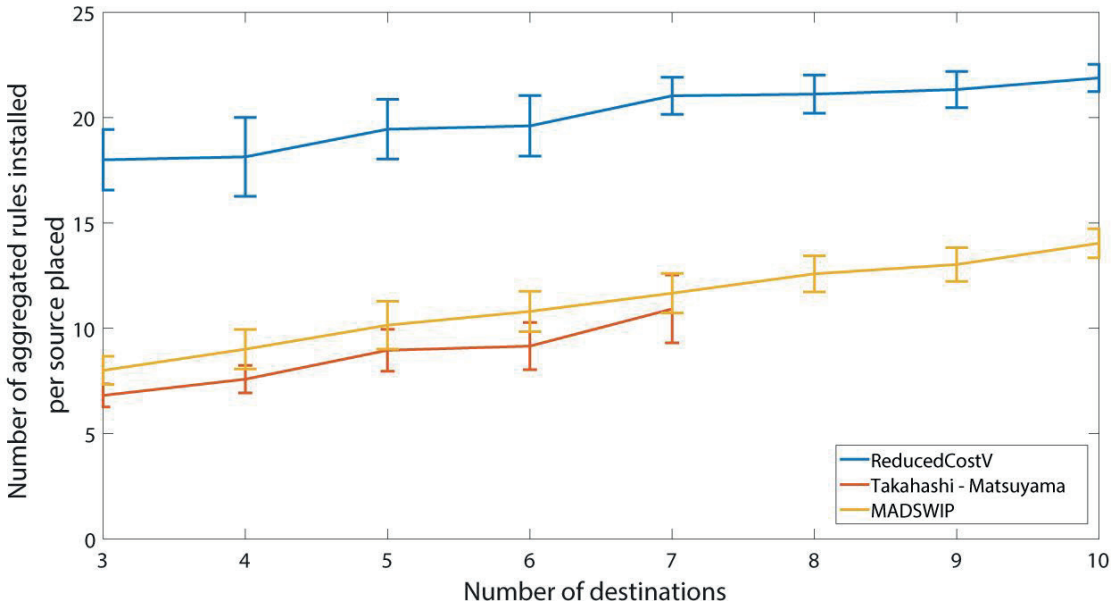


Figure 6.8: Number of rules installed with aggregation for random topology, sparse multicast.

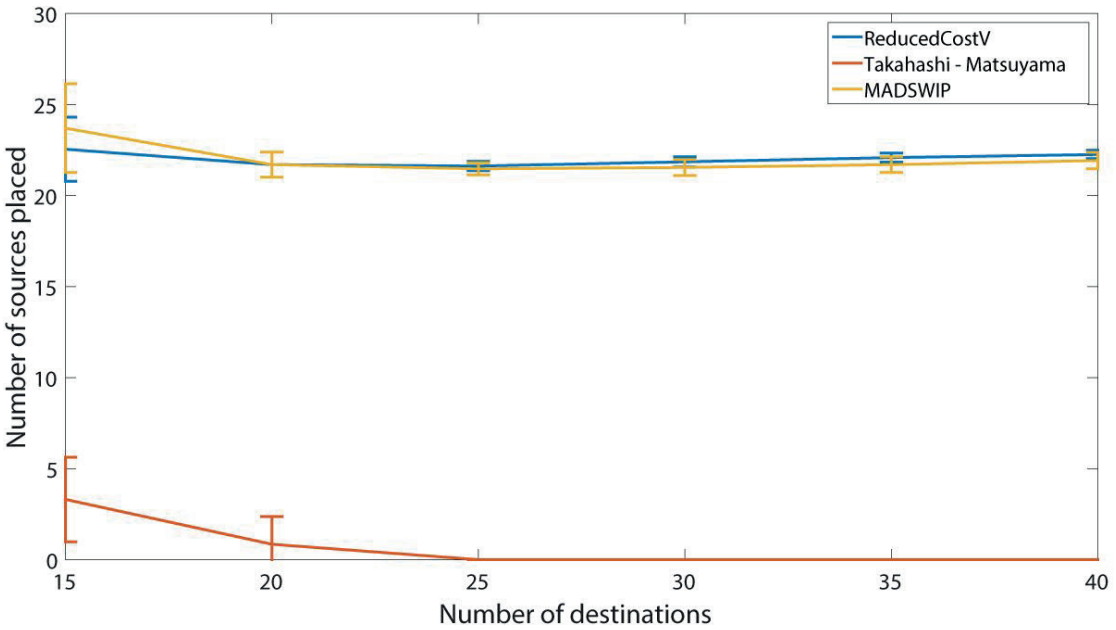


Figure 6.9: Number of sources placed for random topology, dense multicast.

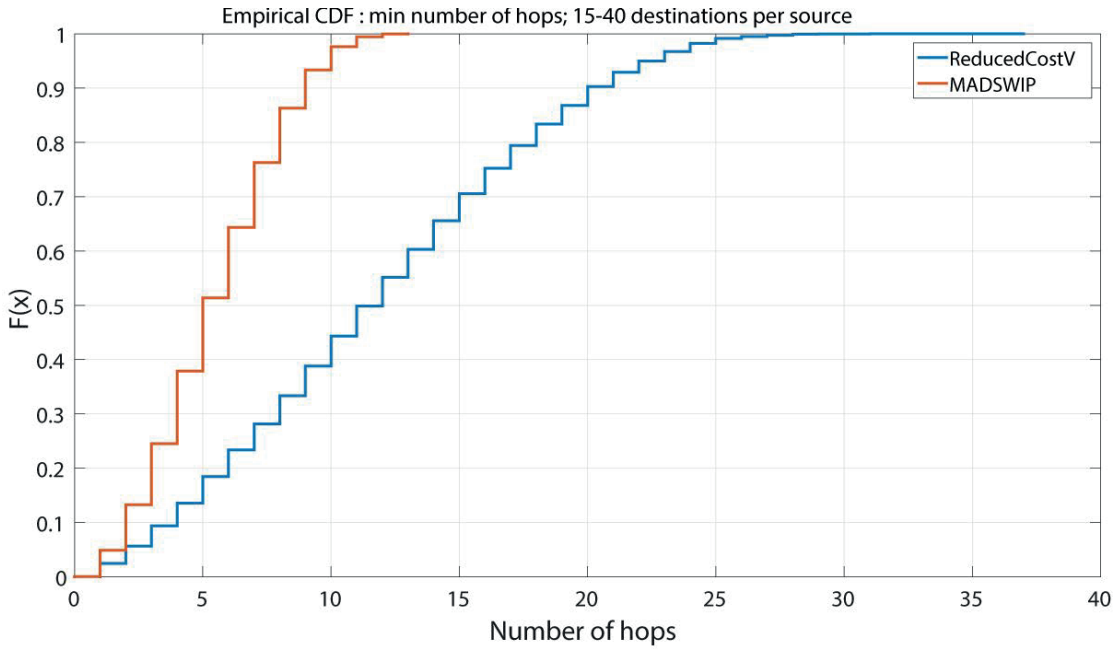


Figure 6.10: Minimum number of hops for random topology, dense multicast.

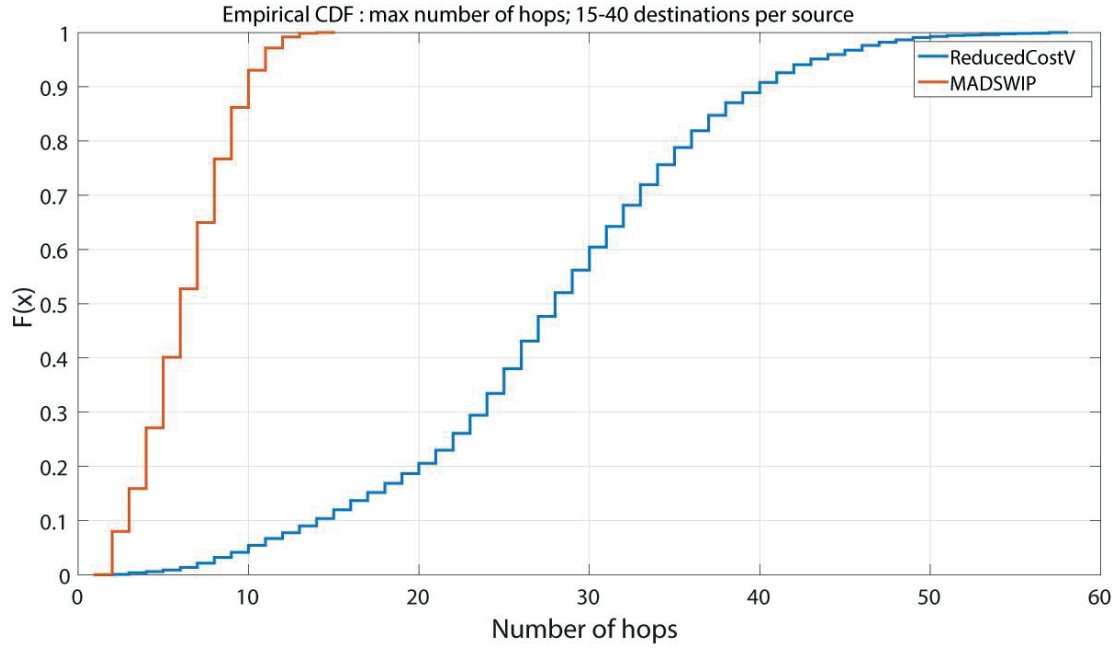


Figure 6.11: Maximum number of hops for random topology, dense multicast.

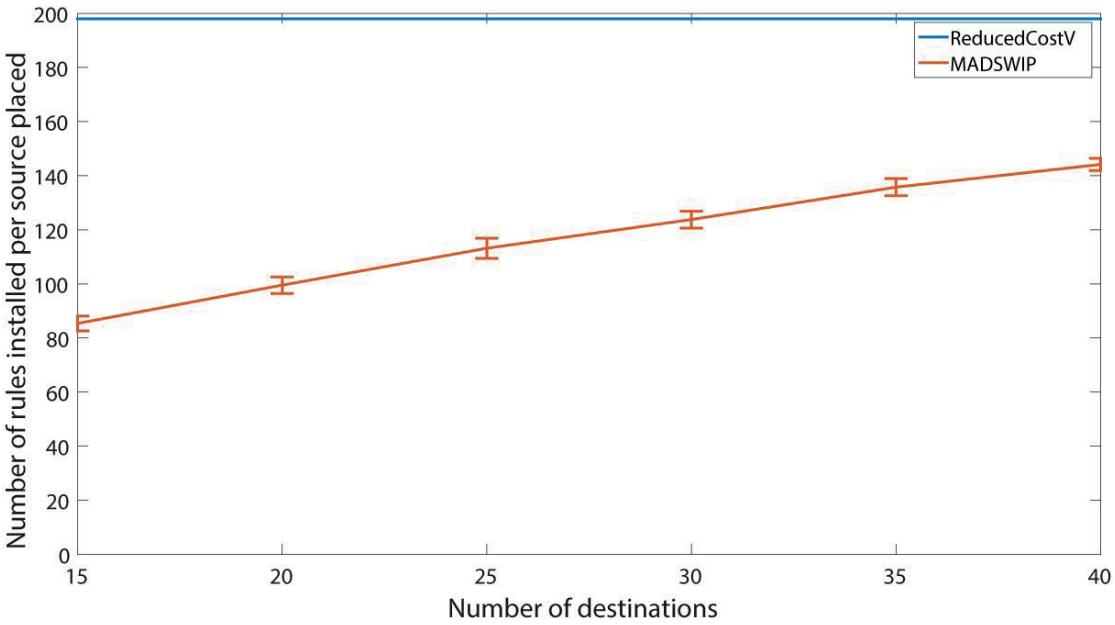


Figure 6.12: Number of rules installed without aggregation for random topology, dense multicast.

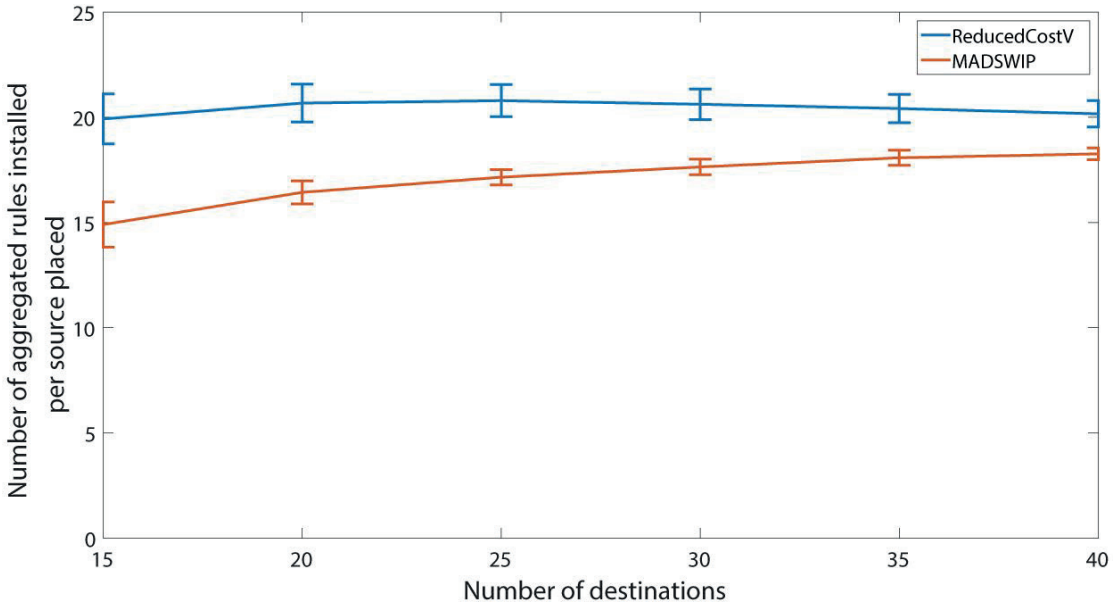


Figure 6.13: Number of rules installed with aggregation for random topology, dense multicast.

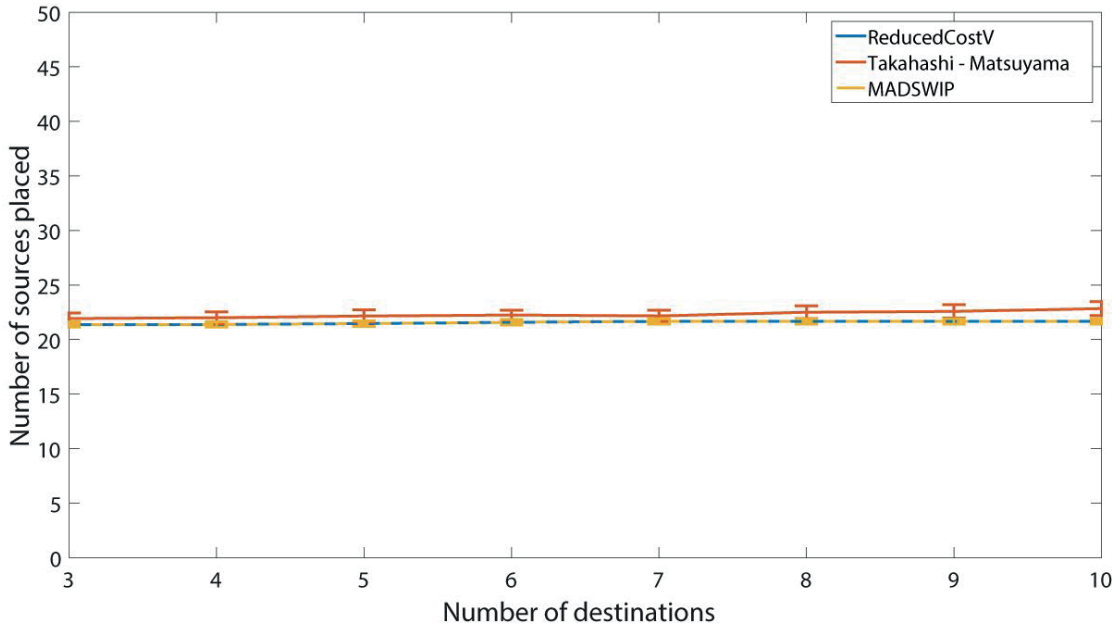


Figure 6.14: Number of sources placed for structured topology, sparse multicast.

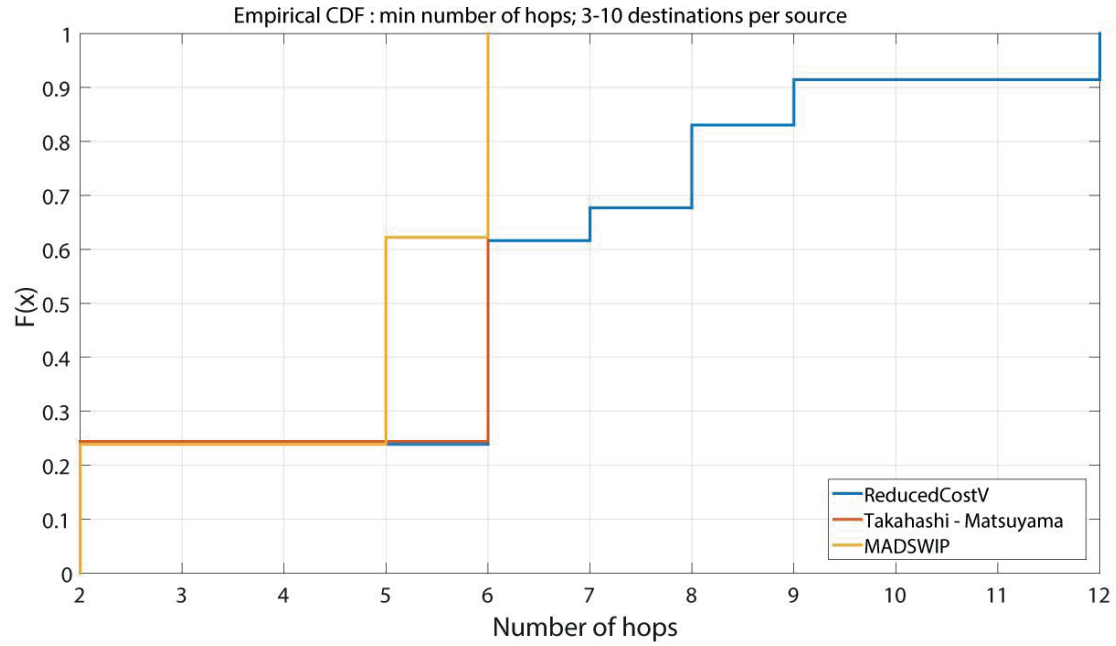


Figure 6.15: Minimum number of hops for structured topology, sparse multicast.

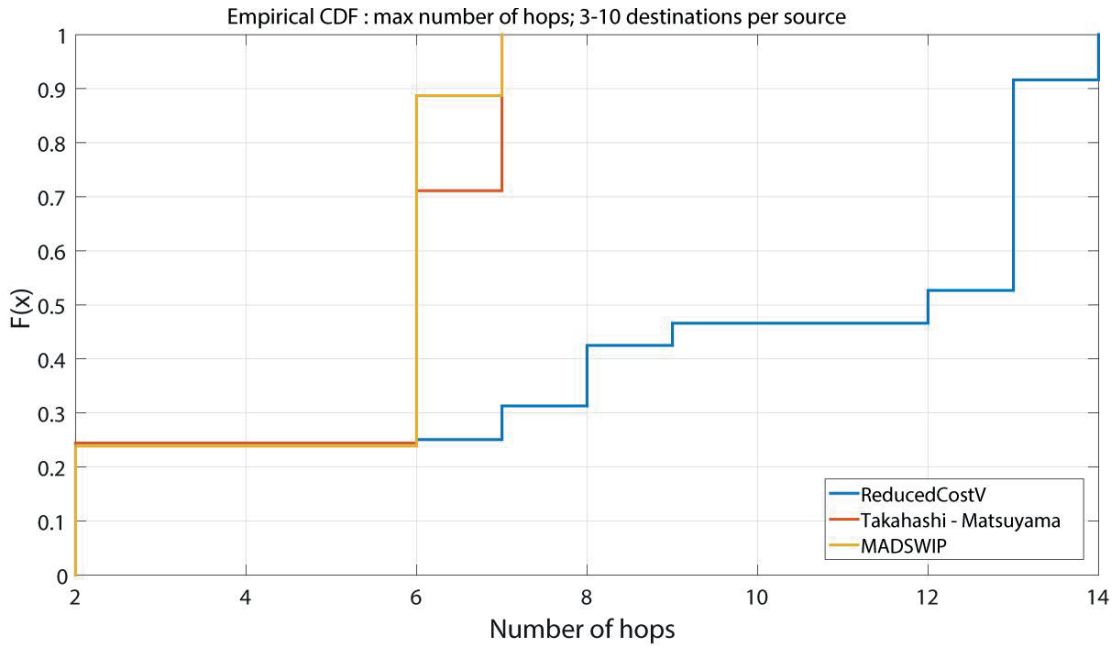


Figure 6.16: Maximum number of hops structured topology, sparse multicast.

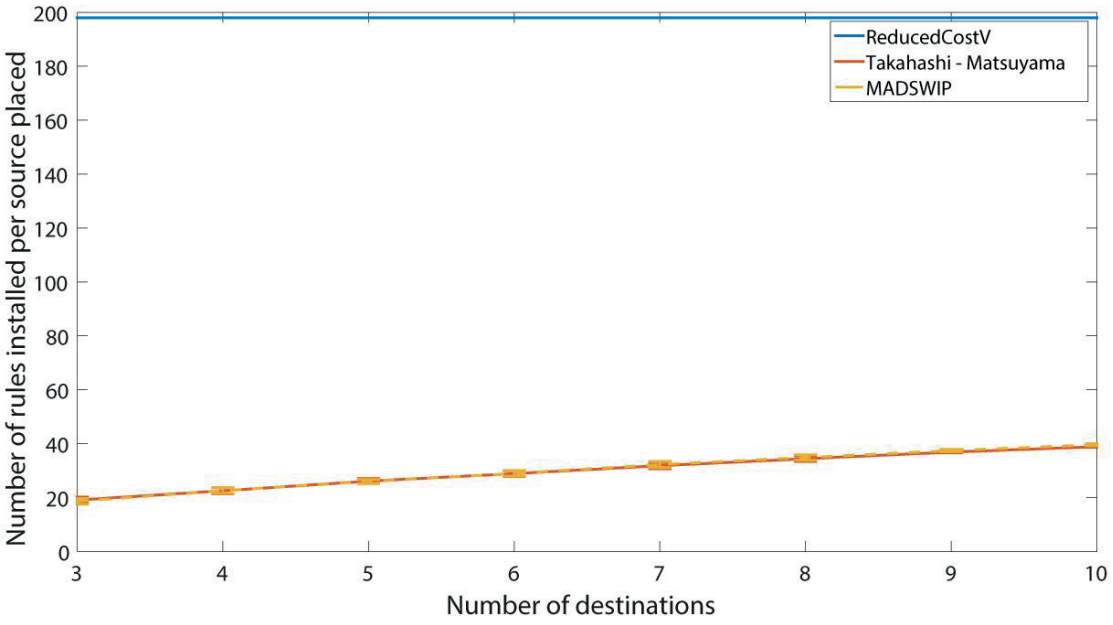


Figure 6.17: Number of rules installed without aggregation for structured topology, sparse multicast.

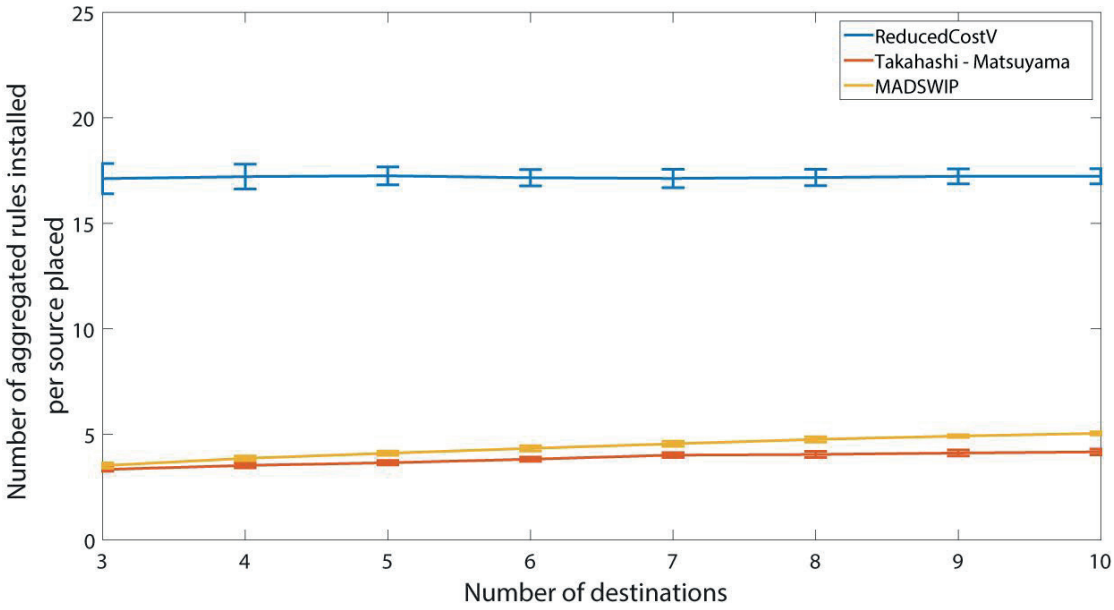


Figure 6.18: Number of rules installed with aggregation for structured topology, sparse multicast.

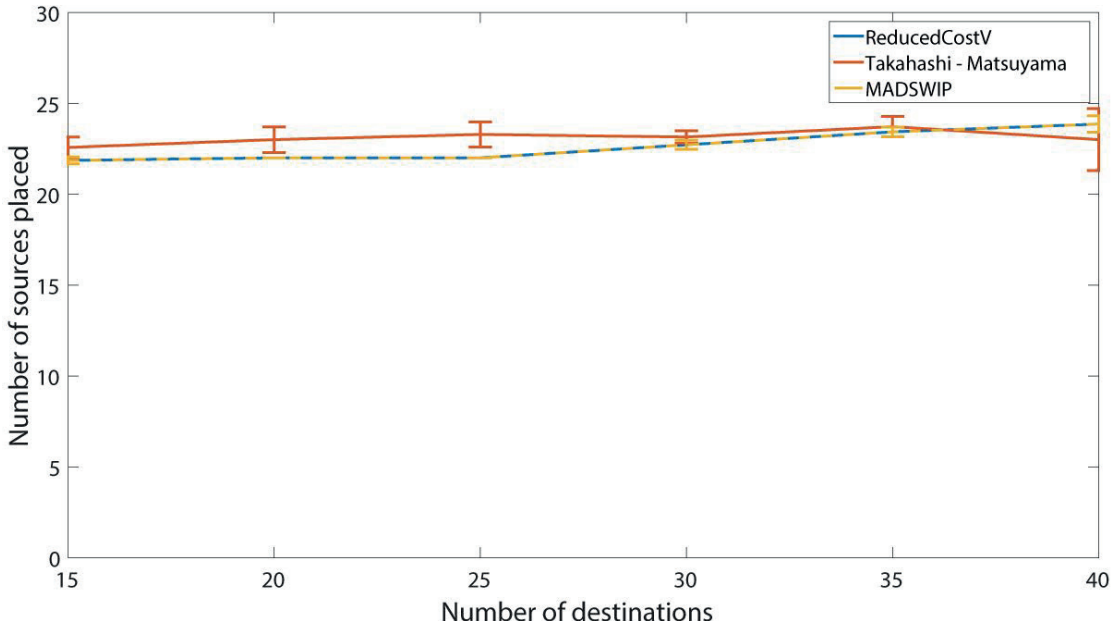


Figure 6.19: Number of sources placed for structured topology, dense multicast.

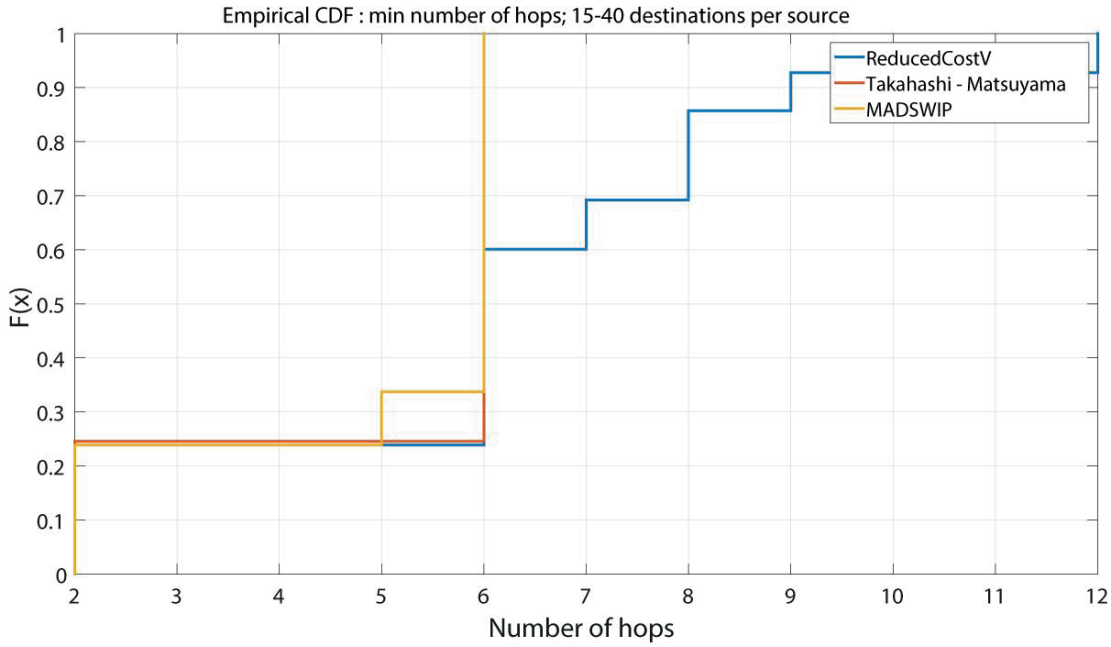


Figure 6.20: Minimum number of hops for structured topology, dense multicast.

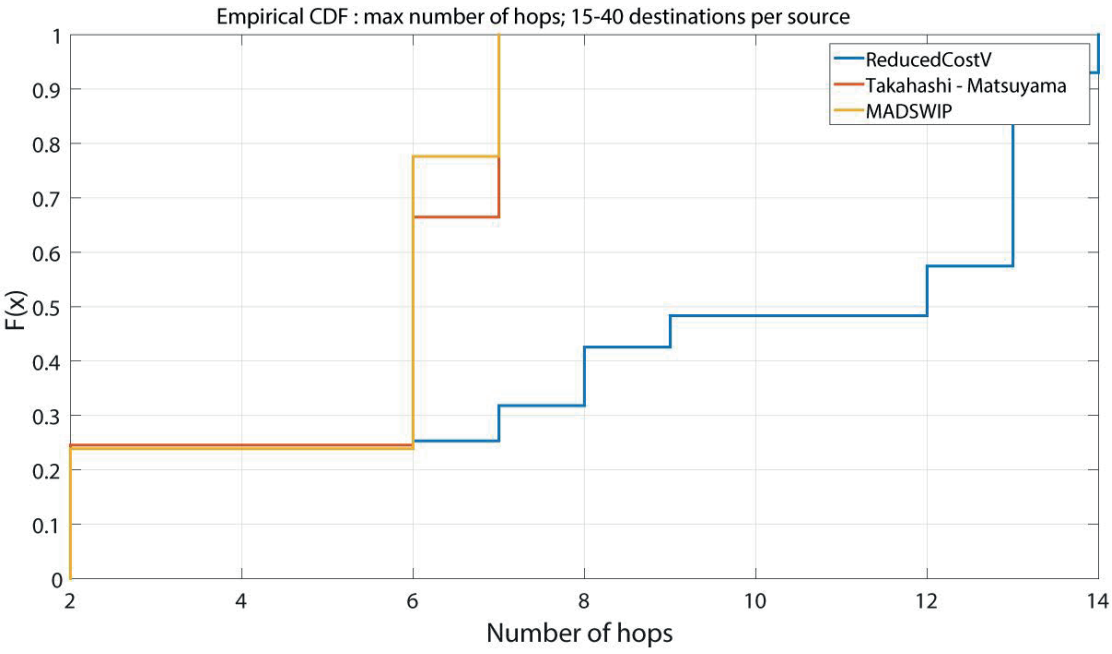


Figure 6.21: Maximum number of hops for structured topology, dense multicast.

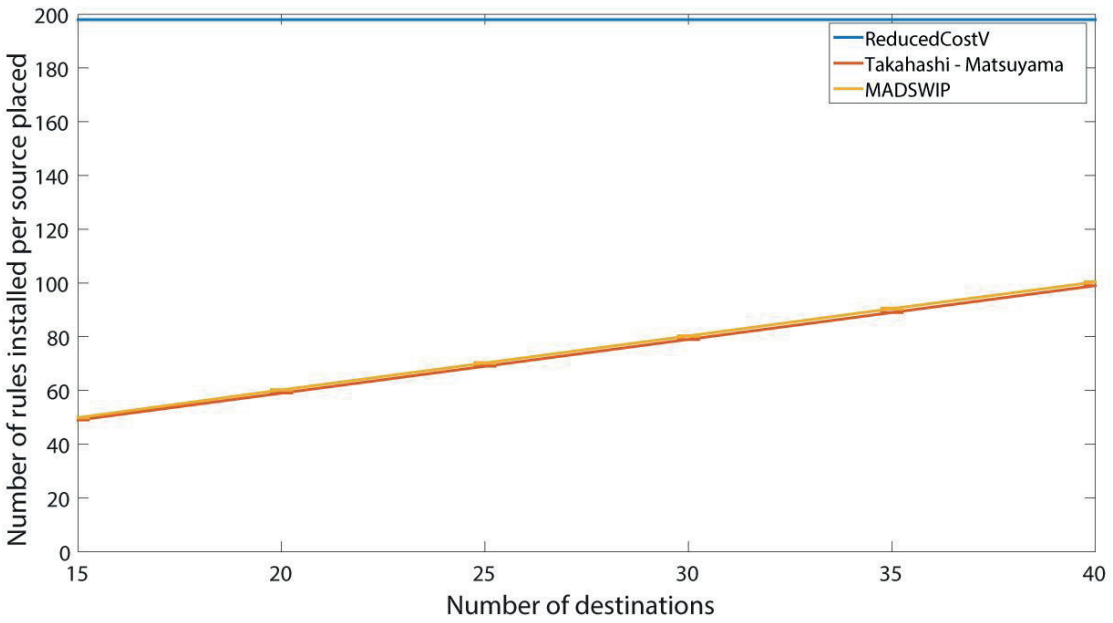


Figure 6.22: Number of rules installed without aggregation for structured topology, dense multicast.

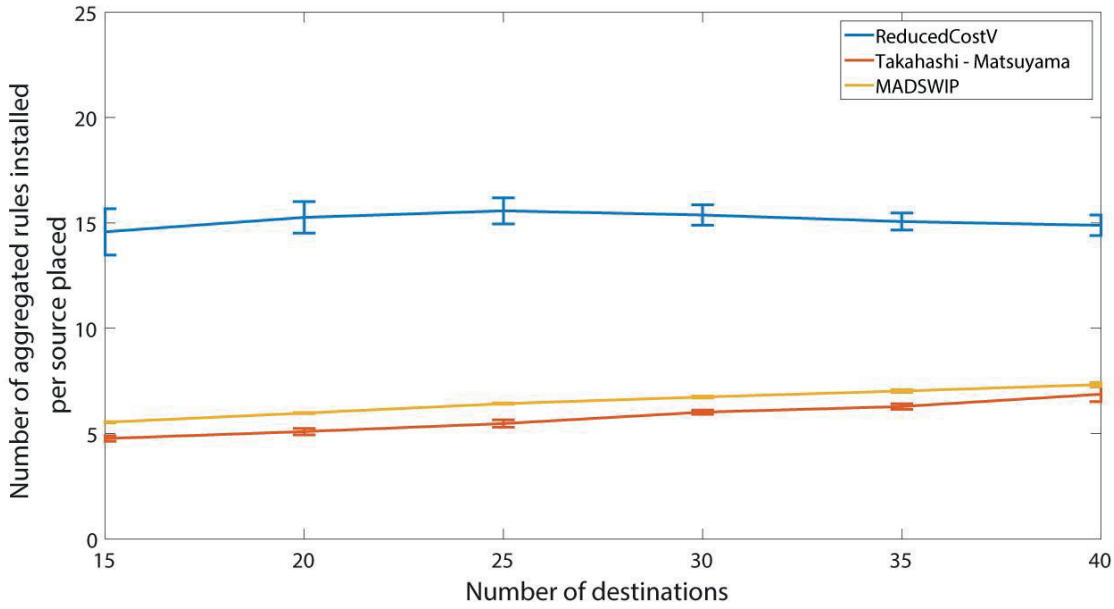


Figure 6.23: Number of rules installed with aggregation for structured topology, dense multicast.

6.6 Discussion about SDN-rules update-activity provoked by topology changes

In this section, we analyze the effect of different changes in scenarios to the activity of the SDN control plane. Concretely, we are interested in situations when there is a permanent change in the system that triggers the update of already installed forwarding rules. The goal of the reconstruction is the (re)establishment of node-redundancy property for all the sources and destinations, without disrupting services that are in-progress. Specifically, we analyze the SDN-control-plane activity in case of (i) node failure, (ii) the arrival of a new destination within a multicast group, and (iii) the arrival of a new source. This analysis is carried out under the assumption that, after a change, it is still possible to construct node-redundant multicast-distribution trees without disrupting the connections that are already put in place.

SDN-rules update-activity in the case of node failure Depending on the position of the failed node, and on the algorithm in use, the effect of such an event can be different. First, we analyze the case of random topology. For `ReducedCostV`, a node failure certainly breaks both spanning trees that are constructed. As a consequence, a new pair of spanning trees has to be computed. For `Takahashi - Matsuyama` and `MADSWIP` the answer is less straightforward; it depends on the position of the failed node. It can be part of none, or very few, of the already established trees, hence its failure will not affect significantly the network operation. Therefore, `Takahashi - Matsuyama` and `MADSWIP` are less vulnerable to node failures, compared to `ReducedCostV`. In order to compare them in more details, we made a simulation analysis to evaluate the probability that a placed pair of trees is affected in case of a random node failure. We did it for cases with 3 and 6 destinations within a multicast group, for the random topology. For both 3 and 6 destinations, the difference in probabilities for these two algorithms is negligible (around 1%). Specifically, we get 25% and 36%, for 3 and 6 destinations respectively.

For the structured topology, failure of an access switch means permanent disconnection for the directly connected source or destination. Failure of an aggregation switch disables one of the trees but the redundancy re-establishment is impossible. This is as we set the level of connectivity between access and aggregation switches to 2 and also because the pods are of size 4, hence, any lower layer aggregation switches is only connected to two upper layer aggregation switches. Therefore, the failure of interest is a failure of one of the core nodes as, in this case, the redundancy re-establishment is possible. Given the topology, almost all of the established trees will be affected and new computations will be needed for all the algorithms.

SDN-rules update-activity in the case of an arrival of a new destination within a multicast group Here we assume that we add a new receiver in the group of multicast receivers. This does not affect `ReducedCostV` as all the nodes are reachable from all the existing sources and spanning trees are constructed and rules are already put in place. For `Takahashi - Matsuyama` and `MADSWIP`, new calculations will be needed and, as we see from the figures that show the CDFs of the number of hops, the effect is similar as up to 20 nodes will be affected in majority of cases, irrespectively of scenario. To conclude, contrary to the previous case, `ReducedCostV` outpeforms the two other algorithms.

SDN-rules update-activity in the case of an arrival of a new source Here we assume that a new source starts sending traffic to already present group of receivers. For all the algorithms, a new computation of tree pairs is needed. In the case of `ReducedCostV`, an installation of new rules will be needed in all the nodes. For `Takahashi - Matsuyama` and `MADSWIP`, the fraction of the nodes that require new rules depends on the exact scenario. As fewer nodes are affected because we are not creating spanning trees, the conclusion is similar to the one in the first scenario: `Takahashi - Matsuyama` and `MADSWIP` are less affected by source arrivals, compared to `ReducedCostV`, and are very close to each other. Concretely, the figures that show the number of rules installed (with and without aggregation) per source placed are the best comparison of the effect of the arrival of a new source, depending on the used algorithm.

6.7 Conclusion

In summary, we can say that `MADSWIP` is the overall winner in this performance comparison. It is robust and it performs the best or comparably well to the best algorithm in a wide range of scenarios and metrics. However, there are two exceptions where it makes sense to consider other algorithms. First, if the topology of interest is truly dual-plane and if it is guaranteed that it will remain so, no matter what the changes in the network are, we might consider applying `Takahashi - Matsuyama` as, under such conditions, its performance is comparable to (or even slightly better than) the one of `MADSWIP`. Second, if the rate of arrivals/departures of new destinations within a multicast group is high, which provokes high SDN-control-plane activity, we should consider `ReducedCostV` as a solution. This, of course, if the higher number of hops between source-destination pairs can be tolerated (this translates to less-strict delay requirements).

7 Benefits of iPRP in Wireless Environment

7.1 Introduction and Motivation

Mission-critical computer applications with very hard-delay constraints were the main focus of the work presented so far in this thesis. In Chapter 5 we presented iPRP - our solution to exploit redundant network-infrastructure with a goal of timely delivery of messages expected by previously mentioned applications. iPRP is a transport-layer solution and, as such, it depends on the proper operation of the network layer, i.e., on each of the redundant networks that are used. Needless to say, in order to reach the stable network operation for mission-critical applications, the natural choice is to design a *wired-based* network infrastructure. However, not all applications used in smart grids (and in other fields) are mission-critical. Hence, this raises the question of what is the effect of iPRP in a *wireless* environment: Would an application perceive fewer losses, if iPRP is used? In other words, we want to quantify the improvement in terms of packet losses brought by iPRP in a wireless environment. An important benefit of such an improvement is that, for some applications, a wireless-based network infrastructure can be considered as a viable alternative to a wired-based infrastructure that is often much more expensive.

In this chapter, we answer this question by presenting results of a measurement campaign performed on our campus testbed. Metrics of interest are loss probabilities over individual networks and loss probability that would be experienced by an application located on top of iPRP. The latter is obtained through an evaluation of the fraction of packets that iPRP is not able to repair due to correlated loss over both of the paths that are used.

In Section 7.2 we describe our experimental setup. The factors that we take into account for this performance evaluation are presented in Section 7.3 and the measurement results in Section 7.4. Concluding remarks are in Section 7.5.

7.2 Experimental Setup

Figure 7.1 shows the map of the EPFL campus with the roof-top measurement sites. The distances between sending and receiving antennas are $180m$ (ELL to INN) and $230m$ (CO to INN), and there is a line-of-sight. We use directional antennas with the $8dBi$ gain, both for transmission and reception. The testbed is based on 2.4GHz Wi-Fi-technology (802.11b standard) and we use different channels and different polarizations for two communication paths to minimize the mutual interference between them.

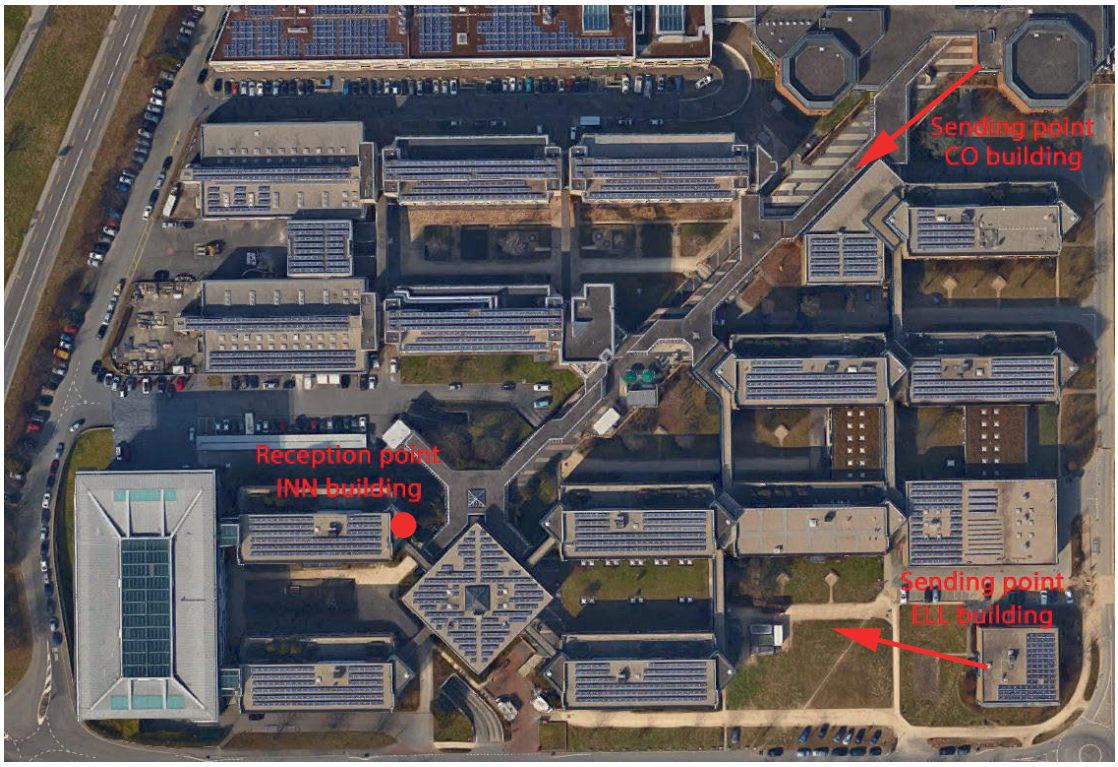


Figure 7.1: Map of the EPFL campus with the antenna locations.

Our measurement system consists of three hosts: two of them are used for traffic generation (located on the roofs of ELL and CO buildings) and one of them for traffic reception (roof of the INN building). The complete setup in the lab environment, during the testing phase, is depicted in Figure 7.2. We use a ruggedized PC as a receiver machine and it is equipped with two Wi-Fi cards to support the two desired wireless channels. On the sender side we have two alix2d2 system boards[1]. The ruggedized PC runs 64-bit Ubuntu OS and alix2d2 runs OpenWrt OS. The software for execution control is done in Python, whereas the applications used for traffic generation and reception are coded in C++. In addition, we use `tcpdump` for traffic logging.

Traffic scenario imitates the streaming of measurements generated by phasor mea-

[1] http://www.pcengines.ch/alix2d2.htm

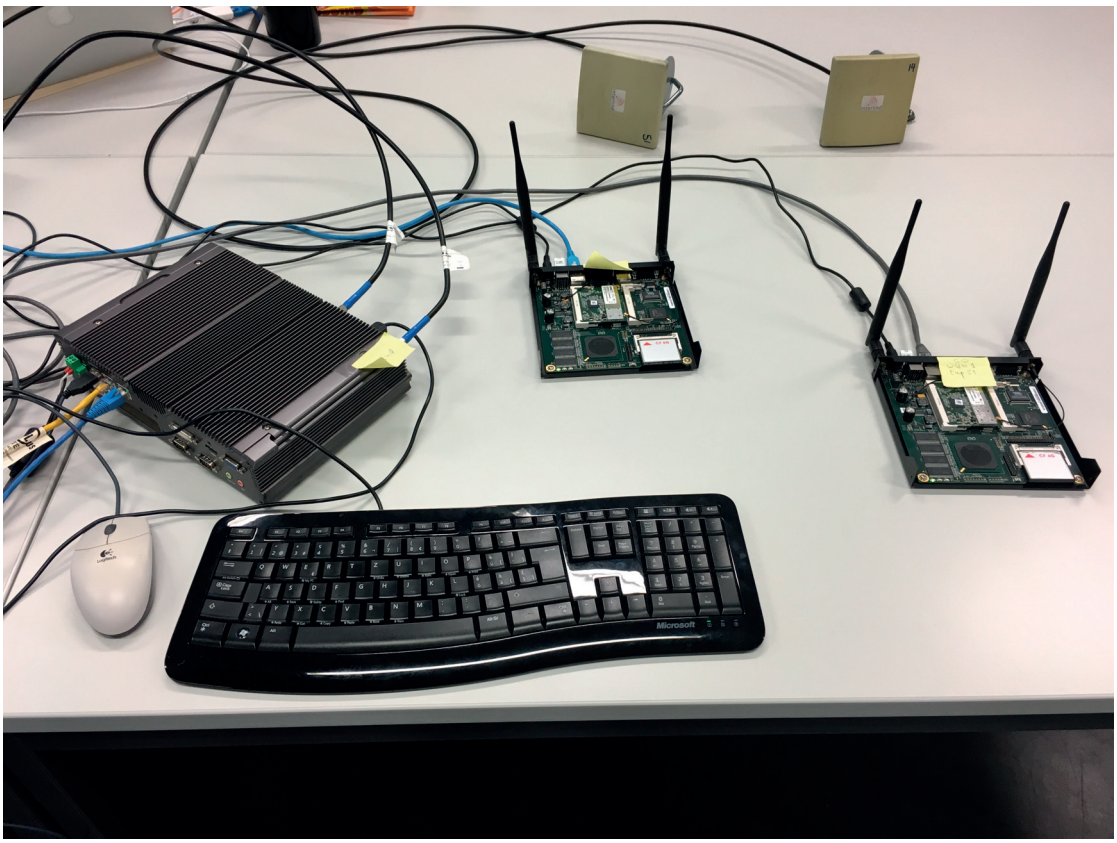


Figure 7.2: Testbed in the lab environment during the testing phase. It consists of the ruggedized PC (on the left) and two alix2d2 system boards. Two directional antennas (out of four used during the experiments) are also visible.

surement units (PMUs), which is a realistic scenario in future smart grids. Concretely, sender applications generate periodically 300-bytes UDP packets, with one packet every $20ms$. We put a packet ID and a timestamp of the moment of packet generation in the payload. On the receiver side, we record packet-reception times and we do this for packets transferred over both communication channels. In post-processing, we analyze recorded log files that contain IDs transferred over two communication channels. From this information we can match pairs of packets that belong to the same "generation" , i.e., they mimic iPRP-replicas of the same original packet. As a result, by analyzing the missing IDs, we can extract the information about losses on different communication channels and about whether the observed losses are correlated (the case when packets with same IDs sent over different channels are missing).

Time synchronization is also part of our measurement setup. To that end, we connect the machines through the campus network (wired) that is used for the network time-synchronization with precision time protocol (PTP). This way, we can also compute one-way propagation times with sufficient accuracy. We take the minimum of the two propagation times to obtain the one-way delay that would be perceived by an

application located on top of iPRP. The necessary packet sending and reception times are already present in the log files, as described in the previous paragraph.

7.3 Factors of Interest and Resulting Measurement Scenarios

We analyze several factors as part of our experiments. Factors such as beacon period or different channel combinations turned out to have no, or very little, effect on the measurement results. Hence we omit these factors from the analysis that follows.

The three factors that we take into account in our analysis are time-of-day of the scenario execution, whether MAC-layer retransmission is used or not and the raw data rate. The time-of-day is taken into account because of the reasoning that interference generated by other users of the non-licensed Wi-Fi spectrum is influenced by human routine (day or night, office hours). MAC-layer retransmissions have an effect on loss probability. With retransmissions, even if the packet is not successfully delivered after the first transmission, MAC layer can repair losses in subsequent transmissions. At the end, we analyze two different raw data-rates supported by the standard: $1Mbps$ and $11Mbps$. In theory, lower raw data-rates should be more robust as they use more redundant channel-coding. Consequently, interference should be less harmful, which should have an effect on the overall packet-loss probability.

This translates to conducting experiments in four different scenarios (combinations of scenarios with and without MAC-layer retransmissions and with rates of $1Mbps$ and $11Mbps$); and we keep track of the time-of-day of scenario execution.

7.4 Measurement Results

We show here the results concerning loss probabilities for the duration of experiment of several days.

We begin the analysis by showing the results as a function of time-of-day. Figure 7.3 confirms our expectation that the interference is the strongest during the working hours, due to human activity. Consequently, the loss probability in general is the highest in that period. We also observe that the link from ELL to INN suffers from fewer losses, compared to the link from CO to ELL. But, more importantly, we see that the losses that are not repaired by iPRP are very rare: loss probability is below 0.002% in all cases.

As for the influence of data rates, in Figure 7.4 we see clearly the expected effect (more losses for higher data rate) in the case of the link from CO to INN, whereas this effect is not so clear for the other link. The conclusion concerning the benefit o iPRP stays the same - loss probability is in the order of 0.001% in both cases.

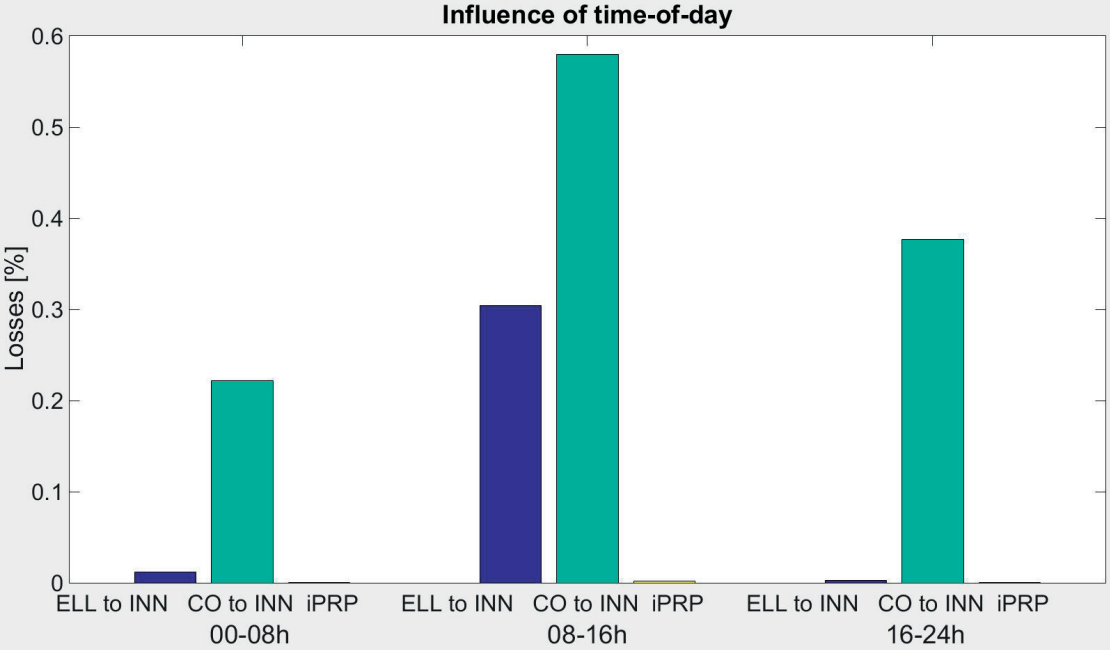


Figure 7.3: Loss probabilities as a function of time-of-day.

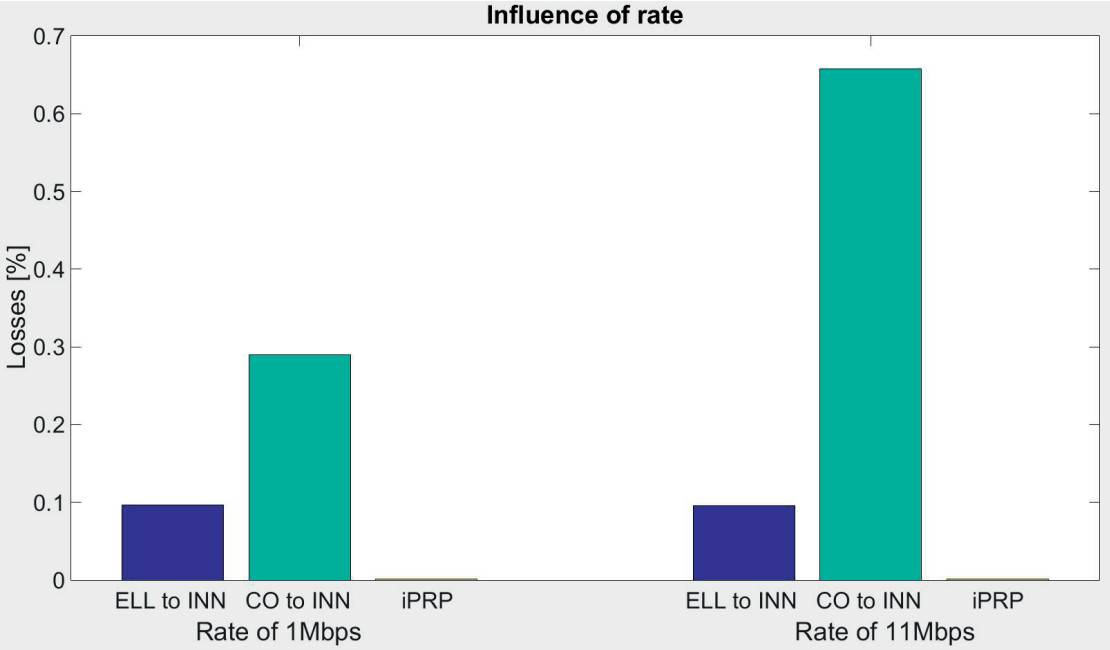


Figure 7.4: Loss probabilities as a function of raw data rate.

In Figure 7.5, we show the results as a function of whether MAC-layer retransmission is used or not. Very few losses are observed on the individual paths when MAC-layer retransmission is used. This results in no losses from the iPRP point of view. Even without MAC-layer retransmission, iPRP brings the loss probability down to below

0.002%. We decided to keep the case without retransmission in our analysis, because such an approach can be of interest for the applications where MAC-layer retransmissions are not desired. For example, very low-delay applications in environments with many Wi-Fi users can benefit from the fact that there are no retransmissions. This is because each retransmission implies additional utilization of time slots, hence, fewer users can be served in one cycle. Nevertheless, iPRP can repair for a good fraction of losses that occur during the only transmission that takes place.

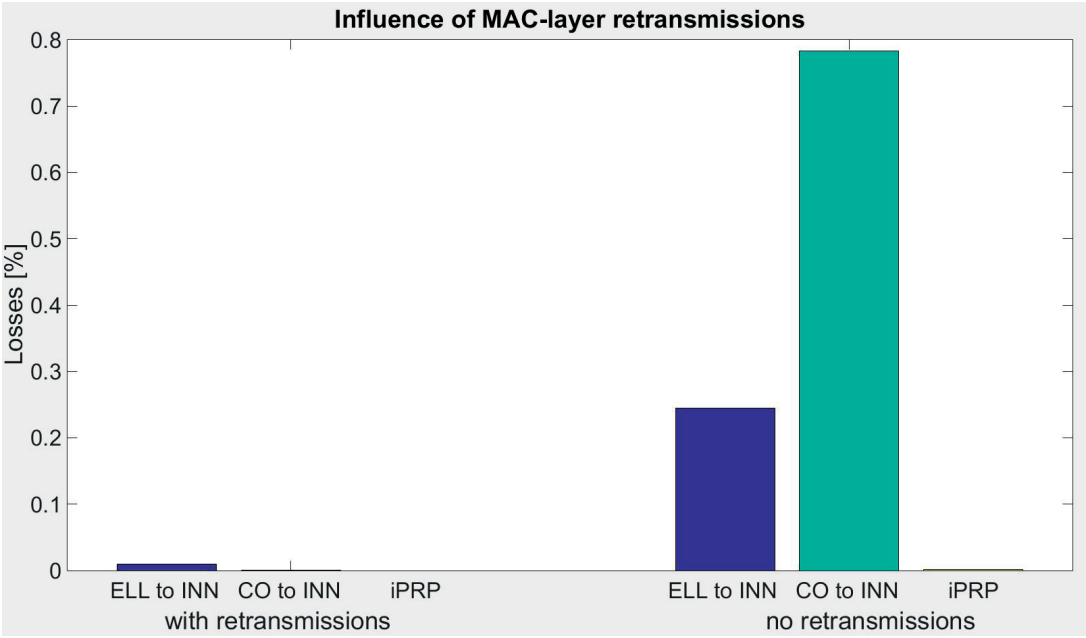


Figure 7.5: Loss probabilities as a function of whether MAC-layer retransmissions are used or not.

7.5 Conclusion and Future Work

In this chapter, we have demonstrated the expected benefit of iPRP in terms of packet losses. From the obtained results we conclude that, if carefully designed, wireless network infrastructure can achieve packet-delivery reliability that is comparable to that expected from the wired infrastructure. Hence, a simple installation of the iPRP software opens doors for cost-effective solutions, given that cabling is often the most expensive part when building wired communication-networks from scratch.

Our future work consists of processing measurement results of the long-lasting experiments that are currently ongoing. In addition to the loss probability analysis, we also expect to obtain results concerning the one-way delays. The goal is to quantify the side-benefit of iPRP in reducing the latency that is perceived by an application that runs on top of iPRP. In other words, we want to reproduce in a wireless environment the experiments for which we show the results in Figure 5.5.

8 Conclusion

In this thesis we study smart-grid communication networks. More specifically, the main focus of our work is on methods that are used to satisfy, through the effective utilization of redundant network-infrastructures, the reliability and timing requirements imposed on those networks. We support our findings with simulation results, as well as with the measurement results obtained from custom-made testbeds. The proposed solutions are tested and evaluated, both in the lab environment and on the production communication network of the EPFL smart-grid pilot.

We analyze the performance of different elements that are part of the EPFL-campus smart-grid communication network that we built. We define the operating region of the whole system and we reveal the bottlenecks that are present. We demonstrate the importance of traffic engineering for the trustworthy operation of networks that support the delivery of critical messages.

In addition, we investigate technologies proposed in the context of smart-grid communication networks, namely MPLS-TP. We concentrate on the security aspects of the protocol and whether the recommendations found in numerous MPLS-TP-related RFCs are correctly implemented in network devices offered by one of the leading manufacturers of networking equipment. The experiments that we conduct on our custom-made testbed reveal security vulnerabilities, even when we followed all the recommendations from the manufacturer. We conclude that an upgrade of traditional electrical grids can make them weaker due to security vulnerabilities, contrary to the expected benefits of smart-grid technologies. The solution can be a cohesive approach to security that should also be unambiguous when relevant RFCs are followed.

We propose also a practical solution to the problem of effective utilization of redundant network-infrastructures. We design, implement and evaluate iPRP: the IP-layer parallel-redundancy protocol. iPRP ensures $0ms$ packet-repair in case of failures or transient problems. It supports the selective replication of UDP flows and does it in a way that is transparent, both to the application and network layers. It requires

only a simple software installation on the end-devices, with no modifications to their operating systems. After the extensive tests performed in the lab environment, iPRP is successfully deployed on the communication network of the EPFL-campus smart-grid testbed. With iPRP, we bridge the identified gap of a missing method that would support the most time-critical applications that require packet communication over WAN IP networks.

We go one step further, by comparing the algorithms that provide node-redundant multicast-distribution-trees that could be used in parallel over a shared network infrastructure. Such multicast-distribution-trees are essential for the optimal operation of iPRP. We study specifically how to adapt these algorithms in an SDN network. The results of our simulation-based study lead to the classification of the evaluated algorithms, based on the metrics that are relevant in the context of smart-grid communication networks.

Finally, we demonstrate the benefits of iPRP in a wireless environment. The measurement results obtained from our roof-top testbed quantify the improvement, in terms of packet losses perceived by an application that runs on top of iPRP. Hence, in cases where target application are not mission-critical but where a certain level of reliability is still desired, iPRP can be an enabler of cost-effective communication networks that are based on wireless technologies.

Bibliography

[1] "Ieee guide for smart grid interoperability of energy technology and information technology operation with the electric power system (eps), end-use applications, and loads," *IEEE Std 2030-2011*, pp. 1–126, Sept 2011.

[2] K. C. Budka, J. G. Deshpande, and M. Thottan, *Communication Networks for Smart Grids: Making Smart Grid Real*. Springer Publishing Company, Incorporated, 2014.

[3] M. Pignati and al., "Real-Time State Estimation of the EPFL-Campus Medium-Voltage Grid by Using PMUs," in *Innovative Smart Grid Technologies Conference (ISGT), 2015 IEEE PES*, 2015.

[4] C. Cheng, R. Riley, S. P. Kumar, and J. J. Garcia-Luna-Aceves, "A loop-free extended bellman-ford routing protocol without bouncing effect," in *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 4. ACM, 1989, pp. 224–236.

[5] C. Hedrick, "Routing Information Protocol," RFC 1058 (Historic), Internet Engineering Task Force, Jun. 1988, updated by RFCs 1388, 1723. [Online]. Available: http://www.ietf.org/rfc/rfc1058.txt

[6] G. Malkin, "RIP Version 2," RFC 2453 (INTERNET STANDARD), Internet Engineering Task Force, Nov. 1998, updated by RFC 4822. [Online]. Available: http://www.ietf.org/rfc/rfc2453.txt

[7] W. Fenner, "Internet Group Management Protocol, Version 2," RFC 2236 (Proposed Standard), Internet Engineering Task Force, Nov. 1997, updated by RFC 3376. [Online]. Available: http://www.ietf.org/rfc/rfc2236.txt

[8] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "OSPF for IPv6," RFC 5340 (Proposed Standard), Internet Engineering Task Force, Jul. 2008, updated by RFCs 6845, 6860, 7503. [Online]. Available: http://www.ietf.org/rfc/rfc5340.txt

[9] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142 (Historic), Internet Engineering Task Force, Feb. 1990, obsoleted by RFC 7142. [Online]. Available: http://www.ietf.org/rfc/rfc1142.txt

[10] M. Seaman, "Link aggregation control protocol," *IEEE http://grouper. ieee. org/-groups/802/3/ad/public/mar99/seaman*, vol. 1, p. 0399.

[11] R. Pallos, J. Farkas, I. Moldovan, and C. Lukovszki, "Performance of rapid spanning tree protocol in access and metro networks," in *Access Networks Workshops, 2007. AccessNets '07. Second International Conference on*, Aug 2007, pp. 1–8.

[12] A. de Sousa and G. Soares, "Improving load balance and minimizing service disruption on ethernet networks with ieee 802.1 s mstp," in *Workshop on IP QoS and Traffic Control*, 2007, pp. 25–35.

[13] J. Gibson *et al.*, "Synchronous optical network (sonet) transport systems: Common generic criteria," GR-1377-CORE, Tech. Rep.

[14] I. Recommendation, "Network node interface for the synchronous digital hierarchy (sdh)," *ITU-T Recommendation G*, vol. 707, 2003.

[15] B. S. Davie and Y. Rekhter, *MPLS: technology and applications*. San Francisco, 2000.

[16] M. Bocci, S. Bryant, D. Frost, L. Levrau, and L. Berger, "A Framework for MPLS in Transport Networks," RFC 5921 (Informational), Internet Engineering Task Force, Jul. 2010, updated by RFCs 6215, 7274. [Online]. Available: http://www.ietf.org/rfc/rfc5921.txt

[17] Y. Weingarten, S. Bryant, E. Osborne, N. Sprecher, and A. Fulignoli, "MPLS Transport Profile (MPLS-TP) Linear Protection," IETF, RFC 6378 (Proposed Standard), Internet Engineering Task Force, Oct. 2011, updated by RFCs 7214, 7271, 7324. [Online]. Available: http://www.ietf.org/rfc/rfc6378.txt

[18] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[20] H. Kirrmann, M. Hansson, and P. Muri, "IEC 62439 PRP: Bumpless Recovery for Highly Available, Hard Real-Time Industrial Networks," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, Sept 2007, pp. 1396–1399.

[21] H. Kirrmann, K. Weber, O. Kleineberg, and H. Weibel, "Hsr: Zero recovery time and low-cost redundancy for industrial ethernet (high availability seamless redundancy, iec 62439-3)," in *Proceedings of the 14th IEEE International Conference on Emerging Technologies & Factory Automation*, ser. ETFA'09. Piscataway, NJ, USA: IEEE Press.

[22] S. S. S. R. Depuru, L. Wang, and V. Devabhaktuni, "Smart meters for power grid: Challenges, issues, advantages and status," *Renewable and Sustainable Energy Reviews*, vol. 15, no. 6, pp. 2736 – 2742, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364032111000876

[23] S. M. Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, 2005.

[24] L. Fang, B. Niven-Jenkins, S. Mansfield, and R. Graveman, "MPLS Transport Profile (MPLS-TP) Security Framework," IETF, RFC 6941 (Informational), Internet Engineering Task Force, Apr. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc6941.txt

[25] N. Sprecher and A. Farrel, "MPLS Transport Profile (MPLS-TP) Survivability Framework," IETF, RFC 6372 (Informational), Internet Engineering Task Force, Sep. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6372.txt

[26] L. Fang, "Security Framework for MPLS and GMPLS Networks," IETF, RFC 5920 (Informational), Internet Engineering Task Force, Jul. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5920.txt

[27] T. Nadeau and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires," IETF, RFC 5085 (Proposed Standard), Internet Engineering Task Force, Dec. 2007, updated by RFC 5586. [Online]. Available: http://www.ietf.org/rfc/rfc5085.txt

[28] A. Durai and V. Varakantam, "Building Smart Grid Core Networks," IEEE Smart Grid Newsletter, October 2012.

[29] F. Ran, H. Huang, T. Wang, and M. Xu, *AsiaSim 2012: Asia Simulation Conference 2012, Shanghai, China, October 27-30, 2012. Proceedings, Part II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Research on Structure of Communication Network in Smart Grid, pp. 135–142. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34390-2_16

[30] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Berlin, Heidelberg: Springer-Verlag, 2001.

[31] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209 (Proposed Standard), Internet Engineering Task Force, Dec. 2001, updated by RFCs 3936, 4420, 4874, 5151, 5420, 5711, 6780, 6790, 7274. [Online]. Available: http://www.ietf.org/rfc/rfc3209.txt

[32] E. Molina, E. Jacob, J. Matias, N. Moreira, and A. Astarloa, "Using software defined networking to manage and control iec 61850-based systems," *Comput. Electr. Eng.*, vol. 43, no. C, pp. 142–154, Apr. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.compeleceng.2014.10.016

[33] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-defined networking for smart grid resilience: Opportunities and challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, ser. CPSS '15. New York, NY, USA: ACM, 2015, pp. 61–68. [Online]. Available: http://doi.acm.org/10.1145/2732198.2732203

[34] D. Gyllstrom, N. Braga, and J. Kurose, "Recovery from link failures in a smart grid communication network using openflow," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, Nov 2014, pp. 254–259.

[35] A. Sydney, D. S. Ochs, C. Scoglio, D. Gruenbacher, and R. Miller, "Using geni for experimental evaluation of software defined networking in smart grids," *Computer Networks*, vol. 63, pp. 5 – 16, 2014, special issue on Future Internet Testbeds - Part II. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004349

[36] N. Dorsch, F. Kurtz, H. Georg, C. Hagerling, and C. Wietfeld, "Software-defined networking for smart grid communications: Applications, challenges and advantages," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, Nov 2014, pp. 422–427.

[37] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takacs, and P. Skoldstrom, "Scalable fault management for openflow," in *Communications (ICC), 2012 IEEE International Conference on*, June 2012, pp. 6606–6610.

[38] https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf, OpenFlow Switch Specification Ver 1.5.1 (April 2015, TS-025).

[39] "Nistir 7628-Guidelines for Smart Grid Cyber Security vol. 1-3," Smart Grid Interoperability Panel Cyber Security Working Group and others, 2010.

[40] G. Dán, H. Sandberg, M. Ekstedt, and G. Björkman, "Challenges in power system information security," *Security Privacy, IEEE*, vol. 10, no. 4, pp. 62–70, July 2012.

[41] T. T. Tesfay, J.-P. Hubaux, J.-Y. Le Boudec, and P. Oechslin, "Cyber-secure Communication Architecture for Active Power Distribution Networks," in *29th ACM Symposium on Applied Computing*, 2014.

[42] C.-. I. S. for Synchrophasor Data Transfer for Power Systems, http://standards.ieee.org/findstds/standard/C37.118.2-2011.html.

[43] http://www.zyxel.com/products_services/p_791r_v2.shtml?t=p, ZyXEL P-791R v2 product details.

[44] ITU-T Recommendation G.991.2, "Single-pair High-speed Digital Subscriber Line (SHDSL) Transceivers."

[45] V. Yodaiken and M. Barabanov, "A real-time linux," *Linux Journal*, vol. 34, 1997.

[46] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing," in *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, June 2007, pp. 160–161.

[47] D. Allan, G. S. Ed., and J. D. Ed., "Proactive Connectivity Verification, Continuity Check, and Remote Defect Indication for the MPLS Transport Profile," IETF, RFC 6428 (Proposed Standard), Internet Engineering Task Force, Nov. 2011, updated by RFC 7214. [Online]. Available: http://www.ietf.org/rfc/rfc6428.txt

[48] Cisco Systems, Inc., "Cisco CSR 1000V Series Cloud Services Router Overview," http://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/configuration/csr1000Vswcfg/csroverview.html, Jul. 2012.

[49] "IP Routing BFD Configuration Guide, Cisco IOS XE Release 3S," http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_bfd/configuration/xe-3s/irb-xe-3s-book.html, Cisco Systems, Inc., 2013.

[50] Philippe Biondi, "Scapy Project," http://www.secdev.org/projects/scapy/, Feb. 2011.

[51] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD)," IETF, RFC 5880 (Proposed Standard), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5880.txt

[52] IEC TC/SC 57, "IEC 61850 Communication networks and systems for power utility automation - Part 8-1: Specific communication service mapping (SCSM) - Mappings to MMS and to ISO/IEC 8802-3," Jun. 2011.

[53] M. Elattar and al., "Using LTE as an Access Network for Internet-Based Cyber-Physical Systems," in *IEEE World Conference on Factory Communication Systems*, 2015.

[54] M. Rentschler and H. Heine, "The Parallel Redundancy Protocol for Industrial IP Networks," in *Industrial Technology (ICIT), 2013 IEEE International Conference on*, Feb 2013, pp. 1404–1409.

[55] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824 (Experimental), Internet Engineering Task Force, Jan. 2013.

[56] IEEE Standards Association., "IEEE Std 802.1AX-2008 IEEE Standard for Local and Metropolitan Area Networks — Link Aggregation." 2008.

[57] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992 (Informational), Internet Engineering Task Force, Nov. 2000.

[58] C. Fragouli and E. Soljanin, "Network Coding Fundamentals," *Foundations and Trends in Networking*, vol. 2, no. 1, pp. 1–133, 2007.

[59] D. J. C. MacKay, "Fountain Codes," *Communications, IEEE Proceedings*, vol. 152, no. 6, pp. 1062–1068, Dec 2005.

[60] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF," RFC 4915 (Proposed Standard), Internet Engineering Task Force, Jun. 2007.

[61] M. Torchia, "Innovative ICT Empower a Better Connected Smartgrid - White Paper," Tech. Rep., August 2014. [Online]. Available: http://enterprise.huawei.com/ilink/enenterprise/download/HW_362734

[62] M. Popovic, M. Mohiuddin, D.-C. Tomozei, and J.-Y. Le Boudec, "iPRP: Parallel Redundancy Protocol for IP Networks," EPFL, Tech. Rep., 2014.

[63] H. Weibel, "Tutorial on Parallel Redundancy Protocol (PRP)," 2003.

[64] J. Nonnenmacher and E. W. Biersack, "Scalable Feedback for Large Groups," *IEEE/ACM Transactions on Networking (ToN)*, vol. 7, no. 3, pp. 375–386, 1999.

[65] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," no. 6347, 2012.

[66] M. Baugher, R. Canetti, L. Dondeti *et al.*, "Multicast Security Group Key Management Architecture," RFC 4046, Tech. Rep., 2005.

[67] S. Hemminger *et al.*, "Network Emulation with NetEm," in *6th Australia's National Linux Conference (LCA 2005), Canberra, Australia (April 2005)*. Citeseer, 2005, pp. 18–23.

[68] S. Salsano, F. Ludovici, and A. Ordine, "Definition of a General and Intuitive Loss Model for Packet Networks and Its Implementation in the Netem Module in the Linux Kernel," Technical report, University of Rome, Tech. Rep., 2009.

[69] S. L. Graham, P. B. Kessler, and M. K. Mckusick, "Gprof: A Call Graph Execution Profiler," in *ACM Sigplan Notices*, vol. 17, no. 6. ACM.

[70] L. Boucher, S. Blightman, P. Craft, D. Higgen, C. Philbrick, and D. Starr, "TCP Offload Network Interface Device," Oct. 16 2007, US Patent 7,284,070. [Online]. Available: http://www.google.com/patents/US7284070

[71] M. Popovic, M. Mohiuddin, D.-C. Tomozei, and J.-Y. Le Boudec, "iPRP: Parallel Redundancy Protocol for IP Networks," in *Factory Communication Systems (WFCS), 2015 IEEE World Conference on*.

[72] T. Pfeiffenberger, J. L. Du, P. Bittencourt Arruda, and A. Anzaloni, "Reliable and flexible communications for power systems: Fault-tolerant multicast with sdn/openflow," in *New Technologies, Mobility and Security (NTMS), 2015 7th IFIP International Conference on*, 2015.

[73] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. ACM Sigcomm '15.

[74] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. ACM CoNEXT '13.

[75] H. Takahashi and A. Matsuyama, "An approximate solution for the steiner problem in graphs," *Math. Japonica*, vol. 24, no. 6, 1980.

[76] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman, "Faster algorithms for construction of recovery trees enhancing qop and qos," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 3, pp. 642–655, June 2008.

[77] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *International Workshop on Quality of Service*, 1999.

[78] F. A. Kuipers, "An overview of algorithms for network survivability," *CN*, vol. 2012, pp. 24:24–24:24, Jan. 2012.

[79] M. Médard, S. G. Finn, and R. A. Barry, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 641–652, Oct. 1999.

[80] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.

[81] Y. Guo, F. Kuipers, and P. Van Mieghem, "Link-disjoint paths for reliable qos routing," *International Journal of Communication Systems*, vol. 16, no. 9, pp. 779–798, 2003.

[82] M. Médard and A. Sprintson, Eds., *Network coding : fundamentals and applications*. Amsterdam, Boston, London: Elsevier, 2012.

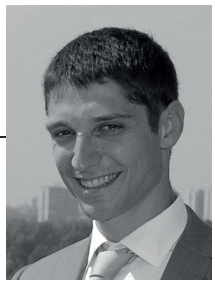
[83] G. Xue, L. Chen, and K. Thulasiraman, "Quality of service and quality protection issues in preplanned recovery schemes using redundant trees," in *IEEE J. Sel. Areas Commun., ser. Optical Communications and Networking series*, vol. 21, 2003, pp. 1332–1345.

[84] N. Singhal, L. Sahasrabuddhe, and B. Mukherjee, "Provisioning of survivable multicast sessions against single link failures in optical wdm mesh networks," *Lightwave Technology, Journal of*, vol. 21, no. 11, pp. 2587–2594, Nov 2003.

[85] R. Nadiv and T. Naveh, "Wireless backhaul topologies: Analyzing backhaul topology strategies," White Paper, Ceragon, August 2010.

[86] M. Howard, "Using carrier ethernet to backhaul lte," White Paper, Infonetics Research, February 2011.

Miroslav POPOVIC



Contact Information

Address: Avenue de la Gare 60, 1022 Chavannes-près-Renens, Switzerland
Phone: +41 78 8321215
E-mail: miroslav.s.popovic@gmail.com
Web: http://people.epfl.ch/miroslav.popovic

Education

2010-2016: PhD in Computer, Communication and Information Sciences, **Swiss Federal Institute of Technology (EPFL)**, Switzerland

2008-2010: M.Sc. in Computer Science, **Swiss Federal Institute of Technology (EPFL)**, Switzerland

2002-2008: B.Sc. in Electrical Engineering (Major in Telecommunications), **ETF, University of Belgrade**, Serbia

Work Experience

Current 2010-2016: **Research Assistant**
Laboratory for Computer Communications and Applications (LCA2), EPFL, Switzerland
Pursuing my PhD thesis under the supervision of Prof. Jean-Yves Le Boudec. My main research interests are various aspects of **smart-grid communication networks** with a focus on **reliability** and **survivability**. This includes design and implementation of a protocol that ensures reliable **packet delivery within stringent delay constraints**, research on the **algorithms for providing redundant paths/trees** for reliable packet delivery and **performance evaluation of the WiFi-based communication** with directional antennas in the context of smart grids.
Furthermore, I contributed in the research on the following topics: active-distribution-network state estimation based on PMU measurements, security of MPLS-TP-based WANs for smart grids and LTE network planning.
This work resulted in **three Best Paper Awards** at e-Energy 2011, CoNext 2012 and WFCS 2015.
Click here for the list of publications.

2009-2010: **Intern**
Skyguide, Geneva, Switzerland
Skyguide is a company based at the Geneva airport that provides air-navigation and related services for civil and military partners. In order to ensure reliable communication between aircrafts and Ground Control there is a need for periodical **control of radio signal quality** (electric field level strength). My responsibility was to come up with the **requirement-specifications document** and then to **design and implement automatic measurement system** that produces measurement campaign reports in the form of easily-readable maps with legend.

Other Professional Activities

2012-2016: **Design, implementation and maintenance of the communication network**
EPFL-campus smart grid project - http://smartgrid.epfl.ch/
Successfully put into operation the communication network that supports PMU-based smart-grid monitoring system within budget constraints of an academic project. My responsibilities included evaluation of the requirements, and then selection, installation, configuration and maintenance of the network equipment.

2014-2016: **Development and maintenance of the project web-site**
EPFL-campus smart grid project - http://smartgrid.epfl.ch/
The project web-site was setup in order to advertise activities and achievements of the laboratories involved in the project. My responsibilities included setting up and maintenance of a hosting server as well as content management.

2011-2015: **Teaching Assistant**
TCP/IP Networking (Master's level course)
Designed several labs with hands-on exercises on socket programming, TCP congestion control, IPv4/IPv6 interworking, tunneling and network security.

2011-2015: **Teaching Assistant**
Performance Evaluation of Computer and Communication Systems (Master's level course)
Responsible for the lab problems that involved performance patterns (bottlenecks, congestion collapse), model fitting and forecasting, discrete event simulation and queuing theory.

2008-2009: **Development and content management of the web-site**
EPFL Communication Systems section, Switzerland - http://ssc.epfl.ch/
The project of the improvement of the EPFL's Communication Systems section web-site was setup as part of a broader effort with a goal of recruiting more students for the section's Master program. My responsibilities included development and content management.

2008: **Completed Cisco CCNA training at the ETF Belgrade Cisco Academy**

2006-2008: **Intern**
Department of Telecommunications, ETF, University of Belgrade, Serbia
The goal of the project was evaluation of the electromagnetic radiation level measurements in the near vicinity of GSM/UMTS base stations for Serbian major telecommunications operator „Telekom Srbija" from the environmental point of view. My responsibilities included development of an automatic measurement system as well as conduction of measurement campaigns. These activities represented a source of financing for the department activities.

Skills

Programming and tools: Python, Matlab, LabVIEW, Shell scripting, PHP, HTML, Drupal, Wireshark, Scapy, NetworkX, Click Modular Router, LaTex, SVN.

Languages: English (fluent), French (fluent), Serbian (mother tongue).

Honors and Awards

2015: Best Paper Award (IEEE), WFCS 2015, Palma de Mallorca,
2014: Teaching Assistant Award, EPFL
2012: Best Paper Award (ACM), CoNext 2012, Nice, France
2011: Best Paper Award (ACM), e-Energy 2011, New York, New York, USA
2008: EPFL "Excellence scholarship" for Master studies
2008: Scholarship of the National Foundation for Scientific and Artistic Youth Development