AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunications
Department of Telecommunications

Ph.D. Thesis

Andrzej Kamisiński

Methods for Dependability Provisioning in Flow-Oriented Telecommunication Networks

Supervisor:

Prof. dr hab. inż. Andrzej Jajszczyk

Co-Supervisor:

Dr hab. inż. Jerzy Domżał

AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunications
Department of Telecommunications

Al. Mickiewicza 30, 30-059 Kraków, Poland
tel. +48 12 617 39 37
fax. +48 12 634 23 72

`http://www.agh.edu.pl`
`http://www.iet.agh.edu.pl`
`http://www.kt.agh.edu.pl`

Reviewers:

Prof. dr hab. inż. Józef Lubacz[1]
Prof. dr hab. inż. Jacek Rak[2]

[1] Warsaw University of Technology
[2] Gdańsk University of Technology

ISBN 978-83-64189-01-2

Cover and LaTeX template by Rafał Stankiewicz

Printed in Poland
by *Drukarnia UNIDRUK, ul. Bronowicka 117, Kraków*

To my parents

Acknowledgments

First and foremost, I would like to thank my supervisor, Professor Andrzej Jajszczyk, for his support, kindness, and invaluable comments that enabled me to succeed in my work.

During my research stay at Norwegian University of Science and Technology in Trondheim, Norway, I received outstanding support from several members of the Department of Information Security and Communication Technology. In particular, I would like to express my gratitude to Professor Bjarne E. Helvik for his guidance and advice, as well as to my collaborators for their friendliness, constructive comments, and excellent ideas.

Furthermore, I would like to thank the members of the Faculty Board and the Department Board for their assistance which was important at various stages of my graduate career. I am also grateful to my colleagues from the Department of Telecommunications who supported me during my research. I also thank my co-supervisor, dr hab. inż. Jerzy Domżał.

Finally, I would like to thank the reviewers for their insightful comments and suggestions that allowed me to improve the final version of this dissertation.

Abstract

Title: Methods for Dependability Provisioning in Flow-Oriented Telecommunication Networks

This dissertation is focused on dependability provisioning in flow-oriented computer and communication networks. Four different solutions are proposed to improve the dependability of the selected flow-oriented networks with respect to the following issues: failures of network elements, forwarding loops, and link congestions (two solutions). In addition, in the case of Software-Defined Networks, the dependability requirements for traffic flows are explicitly defined, the corresponding measure of decreased dependability is introduced, and a risk assessment scheme is proposed to enable service providers to estimate the risk of violation of Service Level Agreements with respect to the proposed metric. One part of the evaluation of the presented solutions is based on experiments in two laboratory networks containing the custom-built prototype routing devices and off-the-shelf network equipment, while the other part relies on discrete-event flow-level simulation in different scenarios using the specifically-designed and implemented tools. The evaluation results have shown that the impact of failures, forwarding loops, and link congestions on traffic flows in the selected flow-oriented networks can be reduced with the aid of the proposed solutions, improving the overall network dependability perceived by users. In addition, the presented service degradation measure and the risk assessment scheme for Software-Defined Networks have the potential to enable service providers to select the desired recovery mechanisms more effectively with respect to the related expenditures and the estimated risk of violation of Service Level Agreements signed with customers.

Streszczenie

Temat rozprawy doktorskiej: Metody zapewniania niezawodności w sieciach telekomunikacyjnych zorientowanych na przepływy

Przedmiotem rozprawy doktorskiej są zagadnienia związane z zapewnianiem niezawodności w sieciach telekomunikacyjnych zorientowanych na przepływy. W pracy przedstawiono cztery rozwiązania umożliwiające poprawę niezawodności wybranych typów sieci w przypadku występowania uszkodzeń jej elementów, a także w przypadku wystąpienia pętli rutingu oraz przeciążeń łączy. Ponadto, zaproponowano jednoznaczny sposób oceny niezawodności obsługi przepływów w sieciach sterowanych programowo (ang. *Software-Defined Networks*), wprowadzono odpowiednią miarę określającą spadek niezawodności w kontekście zdolności sieci sterowanych programowo do prawidłowej obsługi przepływów, a następnie przedstawiono metodę oceny ryzyka niespełnienia warunków umów SLA (ang. *Service Level Agreement*), zawieranych przez dostawców usług telekomunikacyjnych z klientami, ze względu na wartość zaproponowanej miary. Weryfikacja skuteczności części zaproponowanych w pracy rozwiązań, została przeprowadzona w dwóch sieciach laboratoryjnych, zawierających stworzone przez autora prototypy ruterów sieciowych oraz istniejące komercyjne urządzenia teleinformatyczne. Pozostałe mechanizmy zostały przeanalizowane w ramach różnych scenariuszy, przy użyciu autorskich narzędzi symulacyjnych działających na poziomie przepływów ruchu. Otrzymane wyniki pozwalają na stwierdzenie, że opracowane rozwiązania umożliwiają zwiększenie niezawodności wybranych typów sieci zorientowanych na przepływy, w sytuacjach związanych z występowaniem uszkodzeń, pętli rutingu oraz przeciążeń łączy. Ponadto, przedstawiona miara spadku niezawodności obsługi przepływów oraz metoda oparta na analizie ryzyka w sieciach sterowanych programowo, otwierają nowe możliwości w zakresie optymalnego doboru mechanizmów zapewniania niezawodności w tego typu sieciach, ze względu na koszt

i skuteczność rozwiązań, a także szacowaną wartość ryzyka niespełnienia warunków umów SLA w odniesieniu do zaproponowanej miary.

Contents

List of Figures

List of Tables

List of Symbols

α	The maximum allowed service degradation defined in the SLA
β	The constant determining the relative importance of parameters considered in the proposed congestion control algorithms
τ	The length of the observation period defined in the related SLA
$c\left(t,i\right)$	The availability status of all connections to the logically-centralized SDN controller along the entire path of the i-th flow at time t
$D\left(\tau\right)$	Service degradation in the observation period $[0,\tau]$
d	The number of node/prefix groups of the reference node
dst_g	The destination node/prefix group
E	The set of all network links
e_{ov}	The overloaded link belonging to the set of all network links
F	The flow table
G	The network graph
G_{R}	The relation graph used in the proposed GroupAndReroute solution
k	The number of simultaneous link failures in the network
$l\left(P_{\mathrm{OSPF}}\right)$	The length of the new shortest path configured by the OSPF protocol after its reconvergence

$l\left(P_{\mathrm{RG}}\right)$	The maximum observed length of alternative forwarding paths of a traffic flow, resulting from the operation of GroupAndReroute following one or more link failures in the network
l_{th}	The relative link overload threshold
N	The number of nodes in the network
$n_{\mathrm{a}}\left(t\right)$	Number of all traffic flows of the selected customer at time t
$n_{\mathrm{c}}\left(t\right)$	Number of successfully-delivered traffic flows of the customer at time t
$n_{\mathrm{n}}\left(t\right)$	Number of new traffic flows of the customer at time t
$p\left(t, i\right)$	The availability status of the entire path of the i-th flow at time t
p	An incoming packet that will be processed by a router
P_{current}	The currently used path of a traffic flow
P_{new}	The best alternative path that can be assigned to the considered traffic flow
$S\left(\tau, \alpha\right)$	The SLA success probability with respect to the dependability-related requirements for traffic flows in SDNs
S	The maximum estimated path stretch
src_g	The source node/prefix group
T	The routing table
V	The set of all network nodes
V_{g}	The set of visited node/prefix groups
$W\left(\tau, \alpha\right)$	The SLA violation risk with respect to the dependability-related requirements for traffic flows in SDNs

Abbreviations

AS	Autonomous System
ASON	Automatically Switched Optical Networks
BGP	Border Gateway Protocol
CDF	Cumulative Distribution Function
ESCAP	Efficient SCan for Alternate Paths
FAMTAR	Flow-Aware Multi-Topology Adaptive Routing
FFT	Flow Forwarding Table
FIB	Forwarding Information Base
FID	Flow Identifier
FIR	Failure Insensitive Routing
GMPLS	Generalized Multiprotocol Label Switching
ISP	Internet Service Provider
JSON	JavaScript Object Notation
KF	Keep Forwarding
LSA	Link State Advertisement
LSDB	Link State Database
MTTR	Mean Time To Repair

NTP	Network Time Protocol
OSPF	Open Shortest Path First
PDF	Probability Density Function
QoS	Quality of Service
RG	Relation Graph
RIB	Routing Information Base
SDN	Software-Defined Networking
SLA	Service Level Agreement
SLO	Service Level Objectives
TCP	Transmission Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
VaR	Value-at-Risk

1 Introduction

Modern computer and communication networks consist of several interconnected devices which are configured and maintained to forward traffic associated with network services of different demands. Considering the critical role of the Internet today, high dependability requirements are imposed on the main communication infrastructure. However, the capability of the network to deliver the transmitted messages to the respective destinations may be affected by forwarding loops, link congestions, and inevitable failures of network elements. Thus, various recovery, loop avoidance, and congestion control measures are deployed to avoid service disruption and reduce the related consequences for customers and providers.

In flow-oriented networks, packets representing a single traffic flow are forwarded in a consistent way. It is possible to handle particular flows differently, according to the corresponding service requirements and importance, which is a strong advantage. At the same time, due to the specific way some flow-oriented networks operate, the existing methods used in classical packet networks may not be able to protect traffic flows from forwarding loops, impact of failures, and potential packet losses caused by congested links. Thus, in this dissertation, the corresponding solutions are proposed and evaluated in the context of the selected flow-oriented network types to improve their dependability from the user's perspective. Further, in the case of Software-Defined Networks, an explicit definition of the dependability requirements for traffic flows is provided, the corresponding measure of decreased dependability is introduced, and a complete risk assessment scheme is proposed to enable service providers to estimate the risk of violation of Service Level Agreements with respect to the proposed metric. Based on the estimated risk and the results reported in [37], service providers may differentiate the recovery mechanisms used to protect particular traffic flows more effectively, taking into account the related expenditures. One part of the evaluation of the presented solutions is based on experiments in two laboratory networks including both the custom-built prototype routing devices and off-the-shelf network equipment, while the other part relies on discrete-event flow-level simulation using the specifically-designed and implemented tools.

The evaluation results have shown that the impact of failures, forwarding loops, and link congestions on traffic flows in the selected flow-oriented networks may be reduced with the aid of the proposed solutions, improving the overall network dependability perceived by users. Two of the proposed solutions have been implemented in prototype network devices and successfully evaluated in the laboratory environment containing professional network equipment. In addition, the presented service degradation measure and the risk assessment scheme for Software-Defined Networks have the potential to enable service providers to select the desired recovery mechanisms more effectively with respect to the related expenditures and the estimated risk of violation of Service Level Agreements signed with customers.

The dependability of computer and communication networks is discussed in more detail in Section 1.1, while Section 1.2 familiarizes the reader with the general concept of flow-oriented networks and specifies the corresponding dependability objectives with respect to traffic flows. Finally, Sections 1.3-1.5 present the main contributions of this dissertation, formalize the scope and thesis, and outline the organization of the following chapters. Some parts of this dissertation have been modified based on the comments received from the reviewers.

1.1 Dependability of Computer and Communication Networks

Computer and Communication Networks are usually large and complex systems consisting of several different interconnected devices. Each of the devices may fail due to various reasons and the corresponding system is expected to be able to deal with such failures. To describe the ability of the system to maintain correct operation, the concept of dependability is used. Dependability and its attributes are discussed in Section 1.1.1, while Section 1.1.2 identifies the main issues affecting the dependability of computer and communication networks.

1.1.1 Definitions and Attributes of Dependability

The understanding of dependability and the related concepts in the context of computer and communication networks has been summarized in [10]. The authors provide the following two alternative definitions of network dependability:

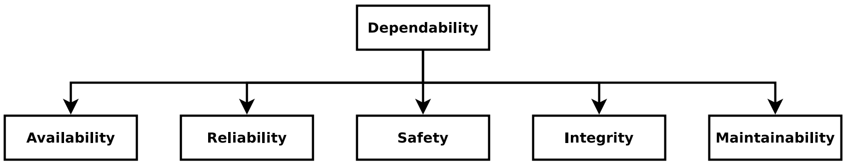


Fig. 1.1: Different attributes of dependabililty, based on [10].

1. “Dependability is the ability to deliver service that can justifiably be trusted.” [10]
2. “Dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable.” [10]

While the first definition is based on the justification of trust related to the delivered network service and generalizes such concepts as availability, reliability, and others, the second definition specifies the criterion that can be used to decide whether the service can be perceived as dependable, even if some failures occurred. Further, the authors also discuss other definitions that have been proposed in the literature, including standards.

Dependability is closely related to its attributes which are summarized in Figure 1.1 based on information provided in [10]. The first attribute, availability, represents the readiness of the system for provisioning of correct service. Reliability emphasizes the continuity of the correct service. Then, safety of the system implies that there will be no catastrophic consequences on the environment and the users as a result of the system’s operation. Integrity means that no improper alterations have been introduced into the system. Finally, maintainability of the system describes its ability to undergo repairs and modifications.

1.1.2 Different Factors Affecting the Dependability of Computer and Communication Networks

According to the definitions provided in Section 1.1.1, as well as the related attributes of dependability, all conditions and events having negative impact on at least one attribute of dependability will also affect the dependability of the entire system. In particular, the following factors may decrease the dependability of computer and communication networks:

– failures of network elements (links, nodes, other devices, software modules) [34, 40, 69];
– maintenance activities [34, 69];
– human errors [69];
– routing loops [69];
– link congestions [43, 69];

- network topology, structural complexity [49, 68];
- attacks on the communication infrastructure (e.g., the attribute of integrity [47]).

The solutions proposed in this dissertation address the dependability issues related to the following three factors: failures of network elements, routing loops, and link congestions.

1.2 The Concept of Flow-Oriented Networks

In the case of the basic Internet architecture, packets are handled with no guarantees related to the Quality of Service (QoS). At the same time, one of the most important advantages of flow-oriented networks is their potential to provide QoS differentiation for particular traffic flows. Considering the existing related proposals, the definition of a traffic flow is not uniform, however. To illustrate different possible approaches, the selected examples are discussed in Section 1.2.1. For an extended discussion, the reader is referred to [26, 87].

The flow-oriented operation of a network may introduce additional requirements with respect to the way the dependability is provisioned in the network. Thus, Section 1.2.2 identifies the dependability objectives in the context of traffic flows.

1.2.1 Definition of a Traffic Flow

Before the transmitted packets can be handled by the network as a single traffic flow, all relevant network devices should recognize flows in the same way. One of the general definitions of a traffic flow is as follows [75]:

> "By flow we mean a flight of datagrams, localized in time and space and having the same unique identifier."

It was proposed in the context of the Flow-Aware Networking architecture. The authors explain that the packets of a single flow are spaced by no more than a specific interval (usually a few seconds), and that they are observed at a specific network interface — hence the localization in time and space, respectively. Further, the authors emphasize that even though the unique identifier may be derived from different IPv4 or IPv6 header fields, it is desired that users have as much flexibility as possible with respect to how the network should recognize traffic flows. An example classification scheme based on IPv4 header fields is the 5-tuple which includes the following descriptors:

- source address;
- destination address;
- source port number;
- destination port number;

– identifier of the transport protocol (e.g., TCP or UDP).

In the case of IPv6, the respective structure might include such header fields as source address, destination address, and the *Flow Label* field [6]. Note that according to the corresponding specification, a *Flow Label* of zero indicates packets that have not been labeled. In addition, it needs to be emphasized that header fields are not protected against unauthorized modifications en route.

The classification scheme based on the header fields of IP packets was also proposed in the context of other flow-oriented architectures, such as those described in [45, 60, 70, 84, 88]. It is worth noting that the *DS* (Differentiated Services) field [73] may also be considered as one of the flow descriptors (for example, see [84]). Further, in Software-Defined Networks (SDNs) and OpenFlow, the flow classification scheme is flexible and allows for the use of different combinations of header fields [60].

1.2.2 Dependability Objectives in Relation to Traffic Flows

The previously-discussed definitions of a traffic flow imply that packets of a single flow are forwarded through the network in a consistent way. It means that to ensure a reliable transmission, the entire path between the source and destination nodes must be available during the time when the flow is active. Furthermore, in the event of failure, the network should be able to move the flow to a backup path between the same pair of nodes. Moreover, the routing of flows should take into account possible forwarding loops and link congestions, to limit packet losses. In addition, in the context of the customer-provider relationship, the formal requirements specified in a Service Level Agreement (SLA) as the dependability Service Level Objectives (SLOs) must also be satisfied by the service provider to avoid the related SLA violation penalty. Thus, the dependability objectives with respect to traffic flows can be summarized as follows:

– successful end-to-end transmission in the presence of failures of network elements;
– avoiding forwarding loops;
– avoiding link congestions;
– meeting the dependability-related SLOs specified for individual flows or predefined groups of flows.

1.3 Scope and Thesis

In this dissertation, four different solutions are proposed to enhance the dependability of the selected flow-oriented network types with respect to possible occurrences of persistent forwarding loops, link congestions (two solutions), and failures of network elements. In addition, in the case of Software-Defined Networks, an explicit definition of the dependability requirements for traffic flows is provided, the corresponding measure of decreased dependability is introduced, and a complete risk assessment scheme is proposed to enable service providers to estimate the risk of violation of SLAs with respect to the proposed metric.

To support the evaluation of the congestion control algorithms and the risk assessment scheme, a discrete-event flow-level network simulator has been designed and implemented. The proposed solutions dealing with forwarding loops and failures of network elements have been implemented in the custom-built prototype routing devices and they were evaluated in two laboratory networks including off-the-shelf network equipment.

The evaluation results have shown that the impact of failures, forwarding loops, and link congestions on traffic flows in the selected types of flow-oriented network may be reduced with the aid of the proposed solutions, improving the overall network dependability perceived by users. In addition, the presented service degradation measure and the risk assessment scheme for SDNs have the potential to enable service providers to select the desired recovery mechanisms more effectively with respect to the related expenditures and the estimated risk of violation of the dependability requirements of SLAs signed with customers. Thus, the following thesis of this dissertation has been proposed and proved:

> **It is possible to improve the dependability of the selected flow-oriented network types using the proposed solutions to deal with failures, forwarding loops, and link congestions, and to estimate the risk of violation of the dependability SLOs related to traffic flows in SDNs with the aid of the proposed risk assessment scheme.**

The related research objectives considered in the dissertation are as follows:

1. Design, implement, and evaluate a method to prevent persistent forwarding loops from occurring in flow-oriented networks in which forwarding decisions are made based on the typical routing table and an independent flow table;
2. Design, implement, and evaluate a method to protect traffic flows against multiple link or node failures in flow-oriented networks in which forwarding decisions are made based on the typical routing table and an independent flow table;
3. Design, implement, and evaluate one or more methods to decrease the overall number of fully-loaded links in centrally-managed flow-oriented networks, such as SDNs, reducing the potential packet losses in the entire network;
4. Clarify the understanding of dependability in the context of traffic flows in SDNs; propose the corresponding measure of decreased dependability in SDNs; design, implement, and evaluate a method to estimate the risk

of violation of the dependability-related SLOs specified for traffic flows or predefined groups of flows in SDNs.

1.4 Previously Published Material

Chapter 2 revises two earlier concepts of loop prevention strategies that have been published in the following document:

[48] A. Kamisiński, A. Jajszczyk, J. Domżał, and R. Wójcik. Sposób usuwania pętli w rutingu pakietów w sieci teleinformatycznej [A method for resolution of routing loops in a telecommunication network], December 2014. Polish patent application, no. P.410390.

Both concepts were designed to solve the problem of forwarding loops in specific flow-oriented network types in which traffic flows are forwarded based on entries stored in an independent flow table, instead of always using the current entries stored in the typical routing table. The foundation of the improved algorithm presented in this dissertation has been implemented in a router prototype together with the solution introduced in Chapter 3 to provide an effective countermeasure against persistent routing loops.

Chapter 4 is based on the following publication:

[52] A. Kamisiński, J. Domżał, R. Wójcik, and A. Jajszczyk. Two Rerouting-Based Congestion Control Algorithms for Centrally Managed Flow-Oriented Networks. *IEEE Communications Letters*, 20(10):1963–1966, Oct 2016. ISSN 1089-7798. doi: 10.1109/LCOMM.2016.2594774.

The main aim of the approach shown in this paper was to decrease the negative impact of link congestions on traffic flows in SDNs. In this case, the considered aspect of network dependability was related to the possibility of packet losses within particular traffic flows due to fully-loaded links, which might be interpreted by some customers as transient unavailability of the service. Two different algorithms based on reallocation of flows were proposed to reduce the number of fully-loaded links in the network, and thus improve network dependability from the perspective of the customers.

Chapter 5 is based on the following research paper that will be presented at the 2017 IEEE Conference on Network Function Virtualization and Software-Defined Networks (NFV-SDN) — Fourth Workshop on Network Function Virtualization and Programmable Networks (NFV-SDN'17-NFVPN):

[51] A. Kamisiński, B. E. Helvik, A. J. Gonzalez, and G. Nencioni. Assessing the Risk of Violating SLA Dependability Requirements in Software-Defined Networks. In *IEEE NFV-SDN 2017 - Fourth Workshop on Network Function Virtualization and Programmable Networks (NFV-SDN'17-NFVPN)*, Berlin, Germany, Nov 2017. Accepted for publication.

The contributions presented in this paper include an explicit definition of the dependability requirements for traffic flows in SDNs, the corresponding measure of decreased dependability, and a complete risk assessment scheme to enable service providers to estimate the risk of violation of SLAs with respect to the proposed metric. The related research has been done in an international research team.

The following publications coauthored by A. Kamisiński are also related to the issues studied in this dissertation:

[49] A. Kamisiński, P. Chołda, and A. Jajszczyk. Assessing the Structural Complexity of Computer and Communication Networks. *ACM Computing Surveys*, 47(4):66:1–66:36, May 2015. ISSN 0360-0300. doi: 10.1145/2755621.

[47] A. Kamisiński and C. Fung. FlowMon: Detecting Malicious Switches in Software-Defined Networks. In *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, SafeConfig '15, pages 39–45, Denver, Colorado, USA, 2015. ACM. ISBN 978-1-4503-3821-9. doi: 10.1145/2809826.2809833.

[33] A. J. Gonzalez, G. Nencioni, B. E. Helvik, and A. Kamisiński. A Fault-Tolerant and Consistent SDN Controller. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Washington, DC, USA, Dec 2016. doi: 10.1109/GLOCOM.2016.7841496.

[71] G. Nencioni, B. E. Helvik, A. J. Gonzalez, P. E. Heegaard, and A. Kamisiński. Availability Modelling of Software-Defined Backbone Networks. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 105–112, Toulouse, France, June 2016. doi: 10.1109/DSN-W.2016.28.

[90] R. Wójcik, J. Domżał, Z. Duliński, G. Rzym, A. Kamisiński, P. Gawłowicz, P. Jurkiewicz, J. Rząsa, R. Stankiewicz, and K. Wajda. A survey on methods to provide interdomain multipath transmissions. *Computer Networks*, 108: 233–259, 2016. ISSN 1389-1286. doi: 10.1016/j.comnet.2016.08.028.

[27] J. Domżał, R. Wójcik, D. Kowalczyk, P. Gawłowicz, P. Jurkiewicz, and A. Kamisiński. Admission control in Flow-Aware Multi-Topology Adaptive Routing. In *2015 International Conference on Computing, Networking and*

Communications (ICNC), pages 265–269, Garden Grove, CA, USA, Feb 2015. doi: 10.1109/ICCNC.2015.7069352.

All the published research papers, except for the patent application [48], have been subjected to a thorough review process before final publication. The work presented in [51] has been accepted for publication and will be presented at the 2017 IEEE Conference on Network Function Virtualization and Software-Defined Networks (NFV-SDN) — Fourth Workshop on Network Function Virtualization and Programmable Networks (NFV-SDN'17-NFVPN).

In [49], the notion of structural complexity was defined and illustrated in the context of computer and communication networks. Different metrics were assessed with respect to their sensitivity to various topology complexifying factors, as well as their potential applicability to dependability-aware planning of network topologies. In addition, the considered indices were compared based on the underlying mathematical concept.

The two solutions presented in [47] address an important issue of how to detect compromised switches in SDNs. A new approach is proposed that is based on the analysis of the statistics reported periodically by switches to the logically-centralized controller. The analysis was focused on the following two types of malicious behavior: packet dropping and packet swapping (i.e., forwarding packets through network interfaces that have not been configured as output interfaces for the affected traffic flows). The simulation results have confirmed that the proposed methods are feasible and can be used to improve the security of SDNs, and thus reduce the potential impact on network dependability. This research has been done in an international research team.

The main contribution of [33] is a new method to design a fault-tolerant Master-Slave SDN controller that is able to balance consistency and performance. As the logically-centralized SDN controller is required to ensure the correct operation of the network, it needs to be protected against failures. At the same time, different recovery mechanisms may lead to inconsistencies in the network state recorded by particular instances of the controller. The related proposals are discussed, and a solution is proposed that takes into account such factors as the consistency of the network state, the latency of the controller, and the number of flows handled by the controller during one second. This research has been done in an international research team.

The logically-centralized control plane of SDNs introduces new challenges influencing the dependability of such networks. Thus, a two-level availability model was proposed in [71] that allows to determine and compare the availability of an SDN with the availability of the typical IP network having the same topology. The structural part of the model is related to the network topology, while the dynamic part deals with the state of particular network elements and is based on the corresponding Markov models. The evaluation is based on two real-world backbone network topologies of different size. It is shown that operation- and management-related failures may have more significant impact on the overall availability of SDN backbone networks, compared to typical IP networks. This research has been done in an international research team.

In [90], different existing methods for the interdomain multipath transmission were surveyed. The selected approaches were described and compared based on various criteria.

Finally, [27] presents two admission control mechanisms for Flow-Aware Multi-Topology Adaptive Routing. Both approaches are based on a strategy that rejects the incoming traffic flows if the preferred forwarding link or path is congested. In addition, an extension to both methods is proposed to handle traffic flows of significant importance, such as emergency calls.

1.5 Organization

The remaining part of this dissertation is divided into six chapters. Chapter 2 considers the importance of forwarding-loop avoidance strategies, discusses the related work, and presents a solution for the selected type of flow-oriented network. Chapter 3 reviews the selected existing IP Fast Reroute strategies and introduces the corresponding solution for the selected type of flow-oriented network, allowing for an effective protection against multiple failures. The proposed method was implemented in a prototype router device together with the persistent-loop avoidance strategy introduced in the previous chapter. The evaluation results have shown that the total number of lost packets in the network can be reduced significantly with the aid of the proposed method, compared to the basic flow-oriented network operation mode and its enhanced variant including a Loop-Free Alternates-based protection scheme [9]. Chapter 4 introduces two congestion control algorithms designed for centrally-managed networks, such as SDNs. Both algorithms rely on flow rerouting based on the obtained values of metrics differentiating the candidate paths. Further, Chapter 5 deals with the dependability of traffic flows in SDNs. First, the corresponding requirements are explicitly defined. Second, the measure of decreased dependability is proposed, and based on this measure, a risk assessment scheme is introduced that allows for the estimation of the SLA violation risk with respect to the proposed measure. Chapter 6 presents the design and the selected implementation details related to the discrete-event flow-level network simulation tools created as an integral part of the research. Finally, Chapter 7 summarizes the results and concludes the dissertation.

2 Combating Routing Loops

In dynamic computer and communication networks, the number of devices and their interconnection scheme may change over time. The state of a network is affected by planned maintenance activities and by unplanned events, such as failures of network elements [69]. If a static routing scheme is used in the network, the recovery options following link or node failures are limited to the use of the preconfigured alternative routes, which may not always guarantee successful transmission, even if the network graph remains connected[1]. Thus, to make sure that all destination nodes are reachable from the selected source nodes in a connected network graph, a dynamic routing protocol is deployed in the network. Whenever a network element fails or becomes available after an inactivity period, the dynamic routing protocol in use determines the new routing scheme based on the selected information about the state of network elements. At the same time, during or after the recomputation of the routing scheme in the selected flow-oriented networks, it may happen that some packets are forwarded along closed paths, which prevents them from reaching the corresponding destination nodes. Such a phenomenon is highly undesirable, as it increases the overall resource utilization in the network and may also cause traffic loss and increased transmission delay. To deal with this issue, a suitable loop prevention mechanism needs to be deployed in the network. The selected existing solutions are discussed in Section 2.1, while the proposed algorithm designed to prevent persistent forwarding loops in some flow-oriented networks is presented in Section 2.2. The algorithm is used and evaluated as an integral part of the solution proposed in Chapter 3. Finally, Section 2.3 summarizes the discussion, outlining the main challenges and open issues.

[1]A graph is connected if and only if each of its nodes is reachable from the other nodes belonging to the same graph.

2.1 Related Work

Transmission of packets between the source and destination nodes implies that the packets will be forwarded along finite paths. Thus, a mechanism to limit the total forwarding time of each packet has already been introduced in the IPv4 Internet Protocol. According to the specification of the IPv4 protocol [77], each packet contains the related header including the *Time to Live* (TTL) field. Originally, the TTL denoted the maximum lifetime of a packet and was expressed in seconds. However, such a definition required that additional measurements be done by network routers and the practical use of this field shifted towards limiting the maximum allowed number of hops (transit routers) along the forwarding path. The initial value of TTL is selected by the sender of the packet, while each of the following nodes on the path decreases the TTL by one. The packet can be forwarded by network devices for as long as the corresponding TTL value stored in the packet is greater than 0. Otherwise, if the TTL value equals 0 and the packet has not reached its destination, the packet is dropped. It is worth noting that although such a mechanism does not prevent routing loops from occurring in the network, it guarantees that packets will not enter infinite routing loops, provided that their TTL values are not increased enroute.

A similar approach is used in the IPv6 Internet Protocol [21]. In this case, the corresponding field in the IPv6 header is called *Hop Limit*, which is an explicit reference to the maximum allowed number of hops a packet can visit on its fowarding path. At the same time, until recently, the specification of the protocol remained ambiguous with respect to the case when a packet having the *Hop Limit* value equal to 0 is received by a network node[2] [20]. In particular, if the 8-bit unsigned *Hop Limit* value is decreased further, it will be interpreted as 255 (instead of -1) and the packet will be forwarded to the next node. Furthermore, it was not clear whether the packet with the *Hop Limit* value equal to 0 should be accepted or dropped if received by a non-forwarding node. Both issues have been resolved in the newest specification of the IPv6 protocol published in July 2017 [21].

The strategy to deal with routing loops based on the TTL-like mechanism was also employed in Flow-Aware Multi-Topology Adaptive Routing (FAMTAR) [88, 89]. Due to the way traffic flows are configured and routed in FAMTAR, an effective solution was needed to eliminate possible infinite routing loops following failures of network elements. In particular, whenever a new flow arrived during the recomputation of the routing scheme in the network, the packets of this flow could be forwarded along a path which contained a loop. Once the route for the flow was configured, it would not be modified later by the routing protocol, which

[2]See further: RFC 2460 Errata, report from February 24, 2015 (Errata ID: 4279), downloaded on: March 25, 2017.

would inevitably result in a persistent routing loop. The solution proposed in [89] stores the first observed TTL value for each new flow received by a node and removes a flow from the internal Flow Forwarding Table (FFT) whenever the TTL value of a received packet of this flow differs from the original one. In such a case, the flow may be registered again with the new TTL value and a different output interface. It is worth noting that the proposed solution is prone to unauthorized modifications of the TTL value stored in forwarded packets, and thus should only be relied on in trusted network environments. In Section 2.2, an alternative strategy is proposed that does not suffer from this issue.

An effective loop prevention strategy in the inter-domain scenario was employed in the fourth version of the Border Gateway Protocol (BGP 4) [79]. According to the specification of the protocol, Autonomous System-level routing loop detection is performed based on the *AS_PATH* attribute of BGP routes and assumes that the identifier of a local Autonomous System (AS) cannot appear in the analyzed AS paths. While this approach requires that an AS path be specified for each advertised BGP route, it is a reasonable trade-off, considering that the inter-domain routing scheme is expected to be stable for a long time. Compared to the previously discussed solutions, this mechanism is an integral part of the routing protocol itself. At the same time, it illustrates a valid approach that could also be used in independent loop prevention modules.

Instead of dropping packets of a traffic flow that entered a routing loop, it is possible to send the packets back towards the source node, so that the preceding node on the active path will detect the routing loop and change the preferred output interface for the related flow. This observation laid the foundation for different solutions proposed for packet networks to mitigate the consequences of network failures, including routing loops. One such solution, Failure Insensitive Routing (FIR) [64], relies on the use of interface-specific forwarding and *backwarding* tables. When a transient link failure occurs in the network and a link state routing protocol is used, FIR prevents the dissemination of the related link state advertisement and performs local rerouting based on a *backwarding* table. Nodes that have not been notified of the failure can detect it by comparing the expected and actual identifiers of network interfaces on which packets are received. This allows the nodes to select an alternative next hop based on the precomputed interface-specific forwarding tables. Further, it is shown in the paper that FIR can successfully deal with all single link failures, provided that the appropriate alternative paths exist in the network. The other related solution, U-turn Alternates for IP/LDP Fast-Reroute [8], extends the IP Fast Reroute [9, 82] concept by proposing the second type of an alternate next hop, together with the corresponding selection method. The main difference from the original Loop-Free Alternates concept is that if an adjacent node U of the local node S forwards packets to destination D via S (S is its primary next hop), then it is possible to deliver packets from

S to D via U if only U has a loop-free node-protecting alternate corresponding to destination D. Thus, in the event of failure of a local network interface or an adjacent node, S uses the precomputed list of alternate next hop nodes for each destination prefix to determine whether the incoming packets can be rerouted to reach the respective destination nodes via suitable adjacent nodes. At the same time, when using Loop-Free Alternates instead of U-turn Alternates, a similar action could lead to a routing loop. U-turn Alternates is a mechanism that is able to deal with single node or link failures, just as the third related solution: Efficient SCan for Alternate Paths (ESCAP) [91]. In ESCAP, backup paths between source and destination nodes are determined in advance. Consequently, for each destination node, the identifier of one backup next hop together with the identifier of the corresponding output port are stored in the routing table next to the fields describing the destination, the primary next hop, and the output port. The decision which of the two ports (i.e., the primary or backup port) to use in the case of failure is made based on the known input port on which a packet was received. In particular, it may happen that packets are forwarded back and ESCAP is able to deal with this case without introducing routing loops in the network, as long as there exists at least one active path to the selected destination node. Another solution closely related to ESCAP was introduced in [7]. The DisPath IP Fast Reroute scheme provides protection against all single link or node failures with the aid of minimum-cost node-disjoint paths. Failures and routing loops are detected whenever packets are received on the corresponding primary output interfaces, which means that the downstream nodes have sent the packets back towards the source nodes. Each node supporting DisPath maintains, in its routing table, an additional pointer to a backup next hop for every destination. The corresponding alternative paths are guaranteed to be loop-free, as long as the original network graph is biconnected, all nodes support DisPath, and no more than one network element has failed.

2.2 Dealing with Routing Loops in Specific Flow-Oriented Network Types

In flow-oriented networks, packets belonging to the same connection are forwarded in a consistent way. To introduce flow-level service differentiation or to respond to the selected network events effectively, traffic flows may be routed according to a scheme that is different from the default scheme based on a routing table. In such a case, additional mechanisms may be required to deal with persistent forwarding loops in the network. FAMTAR is an example flow-aware adaptive routing technique which is known to suffer from persistent forwarding loops following one or more network failures, unless an additional loop prevention mechanism

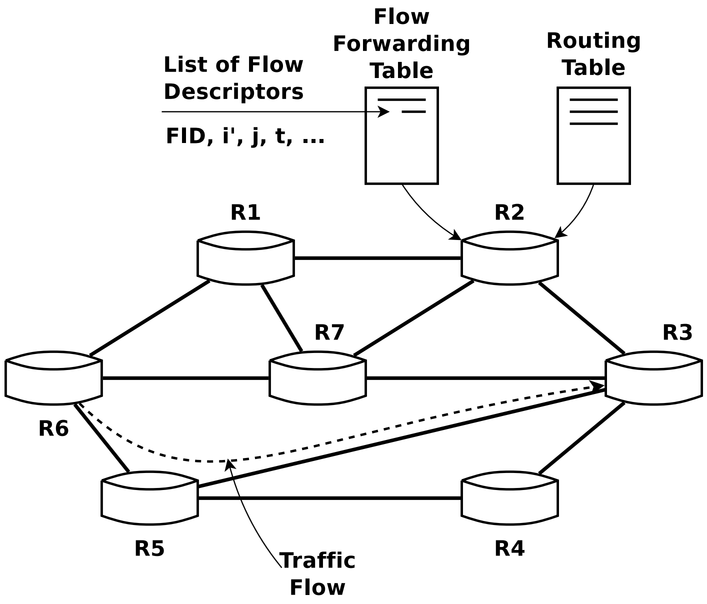


Fig. 2.1: A Flow Forwarding Table (FFT) and the typical routing table maintained on each of the nodes R1-R7 of a flow-oriented network. Both tables are used by the proposed loop prevention algorithm.

is deployed in the network, such as the TTL-based solution presented in [89]. However, the solution recommended for FAMTAR is vulnerable to unauthorized modifications of the TTL value stored in forwarded packets, and thus should only be relied on in trusted network environments. Otherwise, the potential attacker may either modify the TTL value within particular flows to extend the duration of transient routing loops, or inject packets with invalid TTL values into the network to disrupt the multipath transmission of flows and increase routing instability. Both factors can degrade the overall network performance severely, also impacting the dependability of network services. In this section, a new algorithm to prevent persistent forwarding loops is proposed that may be deployed in FAMTAR networks. The algorithm was designed based on the previous experience from the early concepts introduced in [48]. It does not suffer from the TTL modification issues identified above and can also be adapted to meet the specific operation requirements of other flow-oriented networks.

The general concepts related to the considered network model are introduced in Figure 2.1. As presented in the figure, the transmission of packets is organized in traffic flows. Flows are distinguished from each other based on unique Flow Identifiers (FIDs) which may be derived from such parameters as the corresponding source and destination address, the source and destination port number, and the identifier of the transport-layer protocol in use. Whenever a new flow arrives at a router (e.g., router $R2$), its identifier is stored in the local Flow Forwarding Table (FFT) together with additional flow descriptors. Routing decisions at each node are made based on information stored in the FFT and in the local routing table. Expired flows are removed from the FFT based on the observed inactivity period. It is assumed that the capacity of FFTs and routing tables will not be exceeded during normal network operation[3]. Further, without losing generality, it is assumed that network interfaces of each router are assigned locally-unique positive indices (i.e., 1, 2, 3, ...). Whenever a failure is detected, then all nodes adjacent to the failed network element remove entries corresponding to the affected incoming flows from their FFTs. In addition, once the local routing process at each node has finished the recomputation of the routing scheme following a change of the network topology, all FFT entries explicitly marked as temporary are removed from the table.

2.2.1 A New Algorithm to Prevent Persistent Routing Loops

The proposed algorithm is shown in Figure 2.2 and represents a strategy to avoid persistent routing loops in flow-oriented networks in which forwarding decisions are made based on the typical routing table and an independent Flow Forwarding Table (FFT).

In the first step, once a packet of a new traffic flow is received on network interface $i \in \mathbb{N}$, the flow is classified and assigned a unique identifier *FID*. For locally-originated flows, the input interface identifier i is set to 0. As the new flow has not yet been registered in the FFT, its preferred output interface $j \in \mathbb{N}$ is determined based on the routing table. Then, the set of related flow descriptors is added as a new entry to the FFT. The set contains *FID*, i, j, and t, where t is the timestamp of the last received packet of the related flow. In the next steps, the value of t is updated and the packet is forwarded via interface j.

In the case of a previously established traffic flow for which the corresponding *FID* key already exists in the FFT, the expected input and output interfaces (i' and j, respectively) are determined based on the FFT, and then the algorithm compares the values of i and i'. If they match each other, it means that the packet was received on the expected input interface and it can be processed further as discussed in the previous case. Otherwise, the following different cases are considered:

- the packet represents a locally-originated flow, or the corresponding FFT entry has been marked as temporary (in both cases: $i' = 0$);

[3]Note that in real network environments, it is necessary to take into account different limits of numerous hardware devices, as well as the expected load in particular deployments.

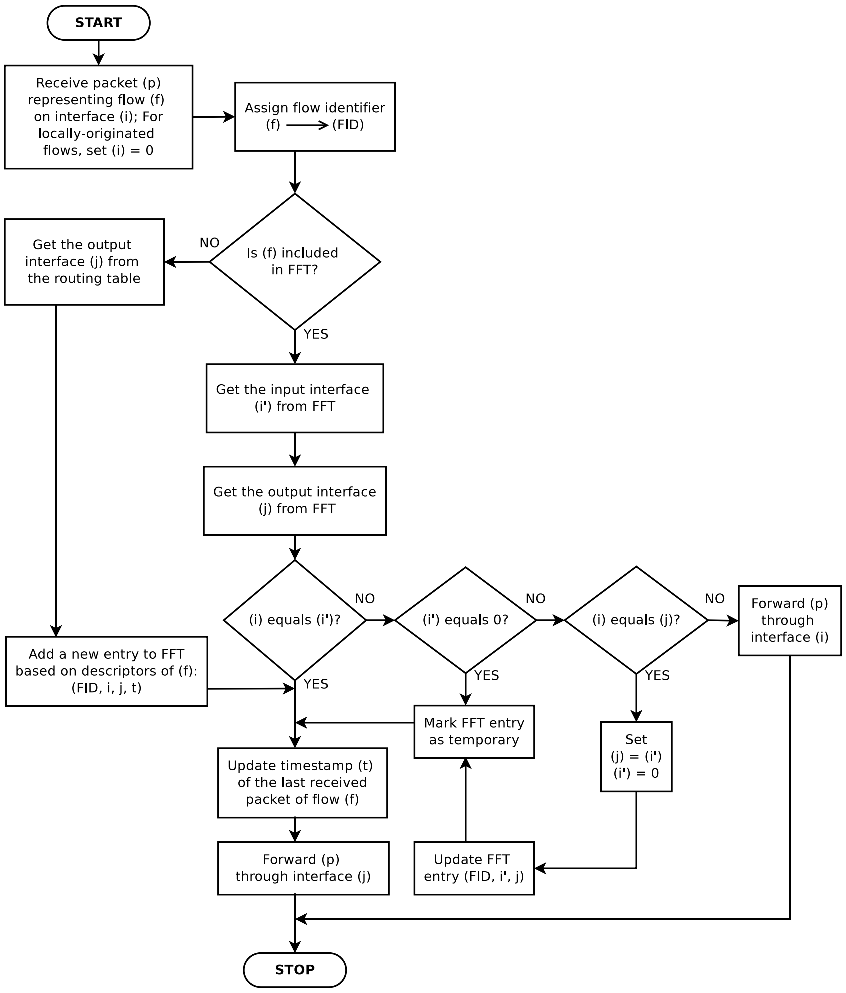

START
Receive packet (p) representing flow (f) on interface (i); For locally-originated flows, set (i) = 0
Assign flow identifier (f) ⟶ (FID)
Is (f) included in FFT?
NO
Get the output interface (j) from the routing table
YES
Get the input interface (i') from FFT
Get the output interface (j) from FFT
(i) equals (i')?
NO
(i') equals 0?
NO
(i) equals (j)?
NO
Forward (p) through interface (i)
Add a new entry to FFT based on descriptors of (f): (FID, i, j, t)
YES
YES
YES
Mark FFT entry as temporary
Set (j) = (i') (i') = 0
Update timestamp (t) of the last received packet of flow (f)
Forward (p) through interface (j)
Update FFT entry (FID, i', j)
STOP

Fig. 2.2: The proposed algorithm designed to prevent persistent forwarding loops in the considered type of flow-oriented network.

– the packet has been received on the preferred output interface, which means that $i = j$;
– the packet has been received on a different network interface $i \notin \{i', j\}$.

In the first situation, the corresponding FFT entry is marked as temporary and the packet follows the typical processing scheme. In the second situation, the received packet must have been diverted by one of the downstream nodes along the original path, possibly due to failure. Thus, the algorithm will forward the packet back towards the source node, so that the upstream nodes on the path can mark the corresponding flow entries as temporary[4]. During this process, the algorithm sets j to i', then i' to 0, and finally, it updates the corresponding FFT entry using the new values and marks it as temporary to ensure that the original route eventually becomes obsolete. In the third situation, the packet is forwarded back via its actual input interface to initiate the backward propagation sequence.

The proposed algorithm is used and evaluated as an integral part of the solution proposed in Chapter 3. The decision not to evaluate a stand-alone implementation of the algorithm was made based on an observation that in real computer networks, whenever the duration of the convergence process of the routing protocol in use is longer than the time needed for the proposed mechanism to redirect traffic flows to the corresponding source nodes, some packets may still be forwarded along closed paths, leading to forwarding loops, because the source nodes have not determined the alternative paths to the destination nodes yet. At the same time, if the proposed solution is combined with a suitable IP fast reroute mechanism and the source nodes can reach the intended destinations via different network interfaces, the source nodes can redirect traffic flows immediately, avoiding forwarding loops. For a detailed discussion of the related experiments, the reader is referred to Sections 3.2.7-3.2.8. It is worth noting that the other methods proposed in Chapters 4-5, focused on centrally-managed flow-oriented networks, do not rely on the algorithm presented above, as it is assumed that the logically-centralized network controller maintains a complete view of the network state and it is able to reroute traffic flows when necessary.

2.2.2 Deployment Considerations and Limitations

The proposed solution is suitable for deployment in such networks in which routing decisions are made based on the typical routing table and an independent flow table. Thus, beyond the related modifications of the routing devices, it is required that the flow table contain all specified fields for the corresponding traffic flow descriptors, as described in Section 2.2.1. In addition, as the proposed algorithm relies on backward propagation of packets to make the upstream nodes mark the flow entries as temporary, it is recommended that nodes supporting the presented algorithm be adjacent to each other, whenever possible.

The proposed loop prevention strategy has one important limitation. If it is deployed only on a fraction of nodes in the network, it may lead to transient forwarding loops, depending on the network topology and paths of particular

[4]Note that this mechanism will also work with asymmetric routing, as each flow entry contains a reference to the input interface for the flow. At the same time, the involved routers that do not support the proposed algorithm may cause transient forwarding loops until the routing protocol reconverges.

flows. In some specific cases (e.g., when several incompatible routing devices appear one after another on the flow's path), more than one update cycle of the routing table may be needed to interrupt all forwarding loops. One of the possible solutions to this issue is to remove the temporary flow entries from the flow table also during a fixed interval after the routing table is updated.

2.3 Summary

In this chapter, an algorithm was introduced that is able to prevent persistent forwarding loops in networks in which routing decisions are made based on the typical routing table and an independent flow table. The operation of the algorithm is discussed in the context of its advantages and disadvantages, as well as the related concepts. In addition, the possible deployment challenges are identified, together with the potential solutions.

3 Responding to Failures of Network Elements

Modern computer and communication networks consist of several interconnected devices which are configured and maintained to forward traffic associated with network services of different demands. Considering the critical role of the Internet today, high dependability requirements are imposed on the main communication infrastructure. However, failures are inevitable and various recovery measures are deployed to reduce the consequences of service disruption for customers and providers.

In flow-oriented networks, packets representing a single traffic flow are forwarded in a consistent way. Depending on the specific way flows are configured and handled in a network, the recovery process may either be started by the forwarding node that has first detected a failure, or it may be managed centrally — for example, in cooperation with a network controller. In the first case, the existing recovery mechanisms designed for classical packet networks may still be able to forward traffic flows along modified routes, so that they reach the corresponding destinations. However, there are two related issues that need to be addressed. Firstly, routing loops may take place due to an inconsistent routing scheme in the network[1]. Secondly, it may happen that no alternative route for a given source-destination pair is known prior to the failure, and such a route will have to be determined before packets of the related flows can be sent further. As this process may take significantly more time than switching traffic to a preconfigured backup path, it may not be acceptable for certain types of network service due to their strict quality requirements. As an example, it is assumed that the real-time voice communication service should not be interrupted or delayed for longer than 50 ms, otherwise the users may start experiencing service degradation [83]. Thus, the capability of the network to provide line-speed recovery to traffic flows is of

[1]For a detailed discussion and the proposed solution for a specific type of flow-oriented network, the reader is referred to Chapter 2.

great importance. Since most of the existing solutions offering such a capability either cannot deal with multiple simultaneous failures in the network in a flexible way[2], or they are incompatible with existing environments through the use of non-standard signaling methods, such as those using additional bits stored in forwarded packets, a new solution is presented in Section 3.2 that aims at solving these issues, while remaining transparent to traffic flows. Due to its relatively simple design, passive operation as an extension to the selected routing protocol in use, low memory requirements, and interoperability with off-the-shelf network equipment, the proposed solution may be deployed gradually in existing computer and communication networks without sacrificing the benefits offered by the routing protocol in use.

In the second case, when the recovery process is coordinated by a network controller, the related delay will depend on the time that is needed to notify the controller about the failure, as well as the time needed by the controller to trigger the appropriate local or global recovery mechanisms [61]. At the same time, if the desired recovery mechanisms can be dynamically configured in forwarding devices in advance, the overall time required to switch traffic flows to the corresponding backup paths will be shorter, because the decision to trigger the appropriate mechanism will be made locally. It is worth noting that such a strategy is closer to the first considered case, but it also has a major advantage: the preconfigured backup paths may be modified automatically by the controller based on the observed network operation conditions. In particular, it might be desired in networks based on the Software-Defined Networking concept. Since the logically-centralized SDN controller collects and maintains complete information about the state of particular traffic flows and the network itself, it can tune the recovery mechanisms frequently to protect the traffic flows better. Further, the solution presented in Section 3.2 can also be employed in this scenario, provided that the relation graphs are determined by the network controller, which then sends the graphs to the corresponding forwarding nodes. On the other hand, an important related issue is the dependability of the network controller itself. For a detailed discussion in the case of SDN networks, the reader is referred to [33].

The general classification of recovery procedures in computer and communication networks is based on the following five principles [15, 16]:

- layer of operation (one or multiple involved network layers [18]);
- recovery path configuration technique (backup paths computed on demand or in advance, before a failure occurs);
- use of network resources (dedicated resources, shared resources, or no reservation of resources);
- scope of the recovery procedure (global, segment, or local);

[2]For example, some techniques are based on backup paths computed in advance that can only protect traffic flows against failures of the selected primary paths [76].

– domain of operation (single domain or multiple involved domains).

According to this scheme, the proposed solution can be classified as operating within the bounds of a single layer (the network layer, as defined in the OSI/ISO model [42]), making local decisions about the preferred backup routes on demand based on some additional information determined in advance, making no resource reservations in advance, and performing global, segment, or local recovery within a single administrative domain. The discussion in this chapter is focused on fast rerouting capabilities of network routers with the goal to reduce the negative consequences (e.g., packet loss, increased length of an alternative path) of one or more simultaneous link failures in wired computer and communication networks. In Section 3.1, the selected related strategies are presented and classified. Their advantages and disadvantages are identified and discussed in the context of the proposed solution which is described end evaluated in Section 3.2. Finally, the discussion is summarized in Section 3.3.

The author would like to acknowledge and thank the Department of Information Security and Communication Technology, the Norwegian University of Science and Technology (NTNU), for using its computing resources in some experiments reported in this chapter.

3.1 Related Work

Whenever one or more links or nodes in a network become unavailable and a dynamic routing protocol is in use, the routing protocol needs to compute a new routing scheme and reach a consistent state across the entire routing domain to ensure that there are no forwarding loops. During the period when computations of the new routing scheme are still in progress, forwarding loops are possible due to inconsistent state of forwarding rules on different devices in the network. In addition, traffic flows are likely to suffer from packet losses and an increased delay, as routers have to determine the preferred output network interfaces for particular destinations before they are able to continue forwarding traffic [31]. To address this issue, different IP Fast Reroute strategies have been proposed. Among the related IP Fast Reroute proposals, there are solutions which are able to handle only one failure at a time [7, 14, 29], whereas the other group of solutions can also deal with multiple simultaneous failures in the network [28, 55, 56, 62, 66, 93].

One example approach belonging to the first group was presented in [14]. The solution provides complete protection against single failures, as long as there are no single points of failure or asymmetric link costs in the network. The idea of tunnels involves redirecting a packet to a node, which in turn will forward the packet further towards the destination. Another solution from the same group is described in [29]. It is based on the precomputation of redundant routing trees for each destination node. Packets are encapsulated to Not-Via addresses which have to be advertised by the routing protocol. At the same time, the authors provided a solution to the major performance- and management-related issues of the original Not-Via concept. Further, one of the most recent proposals in the first group include the DisPath IP Fast Reroute Scheme [7] which is based on the concept of minimum-cost node-disjoint paths. It provides protection against all single link or node failures and assumes a label-free approach. At the same time, the alternative path determined by DisPath may not always be the shortest possible.

The second group of solutions includes the concept of Failure-Carrying Packets [62] which is able to handle multiple simultaneous failures in the network. An interesting and unique feature of this approach is that it removes the convergence period. Instead, the list of unavailable links on the primary path is included in the packet header, which is then used by consecutive routers to compute the preferred forwarding path avoiding the unavailable links. Consequently, the proposed strategy is not compatible with existing network devices. While the solutions presented in [55, 56] provide protection against dual link failures, the technique called Packet Re-cycling [66] allows for handling multiple non-disconnecting link failures at the cost of additional packet header overhead. Assuming that the diameter of a graph is d, the number of required bits in the packet header is on the order of $\log_2 d$. Packet Re-cycling relies on the cellular embedding of the network graph, which provides necessary information to populate the cycle following table used when forwarding packets along their backup paths. While the IP Fast Reroute mechanism presented in [28] also introduces additional packet overhead, it maintains $k + 1$ entries in the routing table per each destination node, where k is the link connectivity of the network. The delivery of a packet is guaranteed for up to $k - 1$ encountered link failures along its path to the destination. The other recent solution, Keep Forwarding [93], provides an interesting approach based on inport-aware forwarding, the new Partial Structural Network model, and a new type of graph traversal. It offers protection against multiple link failures and it does not rely on packet labeling. At the same time, it represents a design of an independent routing strategy, which does not allow for its use together with any other preferred routing protocol.

In the case of centrally-managed networks, such as networks based on the Software-Defined Networking concept [70], the effectiveness of the recovery procedures in use may depend on whether the control traffic is exchanged using the same infrastructure that is also managed by the controller [40]. In particular, failure of network components may interrupt the related control traffic, which can lead to situations when no recovery action is taken in the network, unless the control channels are protected by an additional mechanism [41]. If the network model assumes that forwarding devices can be preconfigured by the controller to perform some recovery actions following a failure, the impact of the failure on the control and data traffic might be less severe. For example, an attempt to use different variants of the Loop-Free Alternates method [9] combined with a new loop detection mechanism in SDNs was discussed in [12].

The other category of network solutions that might be used to respond to failures of network components includes intra- and interdomain routing techniques considered in the context of the multipath transmission of traffic flows. For a review of the selected existing solutions, the reader is referred to [25, 90].

The main limitations of different existing methods can be summarized as follows:

– inability to deal with multiple simultaneous failures in the network in a flexible way;
– reliance on an additional signaling scheme (e.g., passing signaling data in the packet header);
– incompatibility with existing network devices (e.g., through the use of a non-standard format of protocol messages);
– inability to cooperate with different dynamic routing protocols;
– no support for asymmetric link costs.

The new solution presented in this chapter was designed to solve the issues identified above.

3.2 GroupAndReroute: An Effective IP Fast Reroute Scheme for Traffic Flows

In the following sections, a new IP Fast Reroute strategy is introduced that can be gradually deployed in existing computer and communication networks. It is not designed as a replacement of existing routing protocols. Instead, GroupAndReroute acts as an extension to the routing solution in use and requires only minimal changes to the local routing and forwarding process, while keeping the resource requirements on a low level. Further, GroupAndReroute is able to both handle multiple simultaneous failures of network components and provide line-speed recovery to traffic flows. The main contributions of the proposed solution are summarized as follows:

– the design and the corresponding prototype implementation of GroupAndReroute are presented, and its performance is evaluated in a real testbed including professional off-the-shelf communication equipment;
– the proposed solution deals with multiple link failures effectively, offering line-speed recovery to traffic flows, maintaining the average path stretch on a relatively low level, and significantly reducing the total number of lost packets in the network, compared to the basic flow-oriented network

operation mode and its enhanced variant including a Loop-Free Alternates-based protection scheme;
- the proposed approach represents a relatively simple design, it can cooperate with different dynamic routing protocols, and it can be gradually deployed in existing computer and communication networks.

Although the presented solution was designed for flow-oriented networks in which routing decisions are made based on the typical routing table and an independent flow table, successful initial attempts have been made to generalize the approach towards operation in typical IP networks in which routing and forwarding occur in the context of individual packets. The generalized approach is part of the ongoing research.

3.2.1 GroupAndReroute Operation

GroupAndReroute is a transparent and passive mechanism, which means that it does not modify packets forwarded through a local node, it does not send messages to other nodes, and it does not rely on external signaling. It is launched on network nodes next to a routing protocol, collecting and maintaining information about connectivity of particular *node/prefix groups*[3]. The collected information is stored in the so-called *relation graph* and used at a later time to facilitate the quick recovery following one or more link/node failures, whenever it is possible.

Definition 1. *In the context of computation of the routing scheme at a network node, the root node of the routing tree is called the reference node.*

Definition 2. *A node/prefix group of a reference node is meant to be the set of all other network nodes that are reachable via the shortest paths starting at the reference node and leaving it through a common network interface.*

As an example, node 1 in Figure 3.1 is the reference node and there are four node/prefix groups: A, B, C, and D, each corresponding to a different network interface of the reference node.

The node/prefix groups may either be directly connected (groups A-B, B-C in Figure 3.1) or only indirectly connected (groups A-C, A-D, B-D, C-D), depending on the physical network topology and current status of network devices. The directly connected groups share at least one common link, such as link 11-15 between groups B-C, while the indirectly connected groups can exchange traffic only via the reference node or via intermediate groups (e.g., A-B-C). Node/prefix groups are determined separately for each node in the network whenever the routing protocol reconverges. Link or node failures may significantly change the number and size of node/prefix groups for particular nodes.

[3]See Definition 2.

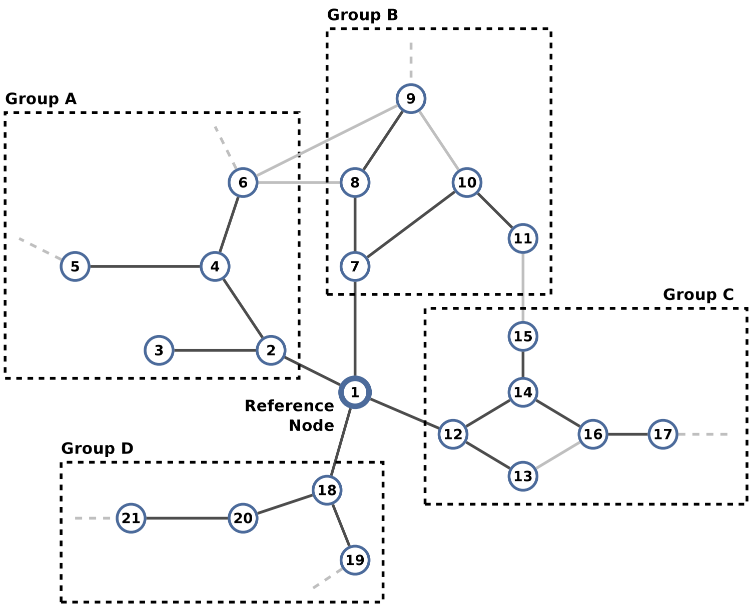


Fig. 3.1: An example assignment of nodes to node/prefix groups of the reference node.

3.2.2 The Initial State of the Relation Graph

As the relation graph maintained by each node reflects the connectivity between different node/prefix groups corresponding to particular network interfaces of the local node, it does not contain arcs in its initial state, i.e., before the first convergence of the routing protocol. It means that the graph is initially disconnected and consists of vertices corresponding to different node/prefix groups, as presented in Figure 3.2(a).

3.2.3 Computation of the Routing Scheme

During the computation of the routing scheme, whenever the involved algorithm (e.g., the Dijkstra's algorithm in the case of the OSPF protocol) examines an alternative path from the local node to the previously considered destination node, it normally selects the shorter path as active, ensuring loop-free routing in the stable state, i.e., after the routing protocol converges. The active paths may also be selected according to the user-assigned link cost values. GroupAndReroute closely follows the steps of the considered routing tree generation algorithm, also called the *primary* algorithm, and adapts an internal copy of the relation graph accordingly. Note that the relation graph must be stable during the convergence time of the

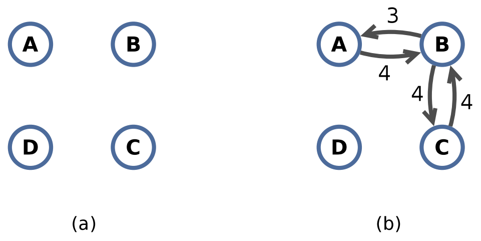


Fig. 3.2: The initial relation graph (a) and its final version after the convergence of the routing protocol (b) for the reference node in Figure 3.1. Letters in circles denote the corresponding node/prefix groups, while the weights assigned to arcs represent distances computed based on the number of hops. Note that distances may also be determined based on the sum of generic link costs.

routing protocol to ensure reliable rerouting. For this reason, modifications are first introduced into an internal copy of the relation graph maintained by the routing process, such as an OSPF daemon, and once the computations have finished, the main instance of the relation graph is updated. Apart from assigning nodes to the appropriate node/prefix groups, GroupAndReroute also detects[4] when an alternative path to a destination node is being examined by the primary algorithm to verify whether it is shorter than the previously found path. In this case, if the destination node belongs to a different node/prefix group than the preceding node on the alternative path, which means that they belong to different branches of the constructed routing tree (for example, consider the following paths in Figure 3.1: 1-7-10-**11** and 1-12-14-**15**-11), GroupAndReroute creates two opposite arcs in the relation graph between the corresponding node/prefix groups and assigns weights to them. The arcs are added to the graph if and only if they have not already been included in the graph. The weight of an arc represents the length of the shortest examined path leading from the reference node through the first node/prefix group (the source vertex of the arc) up to the first encountered node in the second node/prefix group (the destination vertex of the arc). For instance, assuming that distances in the network are computed based on the number of hops along the paths, the weight corresponding to path 1-7-10-11-15 in Figure 3.1 equals 4, which is reflected in Figure 3.2(b) as weight of the arc $B \rightarrow C$. If the weight of an arc is smaller than the previously-stored value, the old value is updated. Figure 3.2(b) shows the complete relation graph for the reference node in Figure 3.1, after all routing computations at this node have finished. Note that in general, distances may also be determined based on the sum of generic link costs.

[4]Several implementation strategies are possible here. For example, GroupAndReroute can wait for an event triggered by the primary algorithm.

3.2.4 Selection of an Alternative Output Interface

Whenever a failure occurs in a network and the routing protocol starts the process of recomputing the routing scheme, new flows are still assigned the corresponding output interfaces based on the routing table, while the existing flows are forwarded based on the flow table. At the same time, when the preferred network interface is inactive, meaning that either the link or the corresponding adjacent node is unavailable, the routing decision is delegated to GroupAndReroute which selects an alternative output network interface based on the relation graph as follows (Algorithms 3.3-3.4).

In the first step of Algorithm 3.3, determine the flow identifier based on the corresponding flow descriptors and check if the preferred output interface is available. If not, then mark the flow as temporary and set a variable that marks that a new output interface should be determined based on the relation graph. In the second condition (Line 6), check whether the current system route entry assigned to the flow is still valid. If not, try to assign a new active route to it, such that leaves the local node via the same network interface. If no suitable route is available, mark the flow as temporary and set a variable that marks that a new output interface should be determined based on the relation graph. Further, check whether packet p arrived on the network interface that is also the preferred output interface assigned to the flow (Line 14). If this is the case, it means that the downstream nodes were not able to deliver the packet to the destination node via the previously selected node/prefix group. Thus, the flow is marked as temporary and if it originates at the local router, it is assigned a new route based on the routing table. External flows are forwarded via the expected input interface towards the upstream nodes.

If the packet arrived on an interface that is different from the expected input and output interfaces, and if the flow is local or the related input and output interfaces are different (Line 23), it means that a forwarding loop has occurred. In this case, mark the flow as temporary and send the packet via the interface on which it has just been received, initiating the backward propagation process. Note that the input and output interfaces assigned to the flow remain unchanged.

At this point, packets for which the route and the output interface are valid are forwarded according to the flow table (Line 47). In the case that the output interface has to be determined based on the relation graph (Line 28), the flow is marked as temporary. Then, a recursive procedure is started (see Algorithm 3.4) to determine the preferred transit node/prefix group on the way to the destination node, assuming that the preferred group is connected with the destination or another transit node/prefix group by an arc having the smallest assigned weight among all the considered options in the relation graph[5]. If such a group is found

[5]Note that this method does not compute the exact distance along the alternative path

```
Input: the routing table $T$; the flow table $F$; the relation graph $G_\mathrm{R}$; incoming packet $p$
 1: $f \leftarrow$ FlowClassifier$(F, T, p)$
 2: $reroute\_based\_on\_rg \leftarrow 0$
 3: if not IsAvailable(OutputInterface$(f)$) then
 4:     MarkAsTemporary$(f)$
 5:     $reroute\_based\_on\_rg \leftarrow 1$
 6: else if not IsRouteValid$(f)$ then
 7:     if RouteViaInterface(OutputInterface$(f)$) $=$ NULL then
 8:         MarkAsTemporary$(f)$
 9:         $reroute\_based\_on\_rg \leftarrow 1$
10:     else
11:         SetRoute$(f$, RouteViaInterface(OutputInterface$(f)))$
12:     end if
13: end if
14: if InputInterface$(p)$ $=$ OutputInterface$(f)$ then
15:     MarkAsTemporary$(f)$
16:     if IsLocal$(f)$ then
17:         SetRoute$(f, T$[Destination$(p)$]$)$
18:     else
19:         SetRoute$(f$, RouteViaInterface(InputInterface$(f)))$
20:         Forward$(p$, InputInterface$(f))$
21:         return
22:     end if
23: else if InputInterface$(p)$ $\neq$ InputInterface$(f)$ and [IsLocal$(f)$ or InputInterface$(f)$ $\neq$
    OutputInterface$(f)$] then
24:     MarkAsTemporary$(f)$
25:     Forward$(p$, InputInterface$(p))$
26:     return
27: end if
28: if $reroute\_based\_on\_rg = 1$ then
29:     MarkAsTemporary$(f)$
30:     $V_\mathrm{g} \leftarrow \emptyset$ {Set of visited node/prefix groups}
31:     $g \leftarrow$ GetPreferredTransitGroup$(G_\mathrm{R}$, InputInterface$(p)$, OutputInterface$(f)$, $V_\mathrm{g})$
32:     if $g \neq$ NULL and IsAvailable$(g)$ then
33:         SetRoute$(f$, RouteViaInterface$(g))$
34:         Forward$(p, g)$
35:         return
36:     else
37:         if IsLocal$(f)$ then
38:             Drop$(p)$
39:             return
40:         else
41:             SetRoute$(f$, RouteViaInterface(InputInterface$(f)))$
42:             Forward$(p$, InputInterface$(f))$
43:             return
44:         end if
45:     end if
46: else
47:     Forward$(p$, OutputInterface$(f))$
48: end if
```

Fig. 3.3: GroupAndReroute: the general packet forwarding scheme.

and the corresponding output interface is available, the packet is forwarded via the new interface. In addition, the reference to the previous output interface in the flow entry is replaced with the new reference to shorten the forwarding time of subsequent packets sent within the same traffic flow. In the other case, when no suitable transit node/prefix group could be found, packets of the locally-generated flows are dropped until the system-wide Forwarding Information Base (FIB) is updated, while the other packets are sent towards the upstream nodes, initiating the backward propagation process. In the second case, the reference to the corresponding output interface is modified accordingly.

The recursive procedure presented in Algorithm 3.4 begins with a set of primary conditions (Lines 1, 33, 35) to determine whether the *dst_g* node/prefix group can be an intermediate group in the chain, the preferred transit group, or if it should not be taken into account as a transit group at all. The result depends on the number of adjacent node/prefix groups, as well as availability of the corresponding network interface of the reference node.

In the first case (lines 1-32), the adjacent node/prefix groups of *dst_g* are taken into account, such that the corresponding output network interfaces of the reference node are available. The preferred transit node/prefix group is selected according to the smallest arc weight in the relation graph. If it is found, the algorithm returns the identifier of the corresponding output network interface. In the other case (the related network interfaces are inactive), the algorithm explores further adjacent node/prefix groups in the relation graph recursively as follows:

– select the node/prefix group s other than the visited ones stored in V_{g};
– start the recursive procedure for the destination group s to get the identifier tg of the network interface associated with the preferred transit group;
– if such a group was found and the related arc weight is smaller than all previously-considered values, update g and w.

Finally, the algorithm returns g (Line 32), which can also contain the NULL value if no suitable node/prefix group was found. Note that if the network graph is disconnected, GroupAndReroute drops packets sent to unreachable nodes to reduce unnecessary traffic in the network. In addition, whenever the FIB is updated by the routing process, the following actions are triggered by the operating system kernel:

– the main instance of the relation graph is updated;
– all temporary or expired traffic flows are removed from the flow table.

Flow entries which are still needed due to packet processing at that time are

between the source and destination nodes. Instead, it considers the distance to reach the second node/prefix group in the chain leading to the destination node/prefix group. If a suitable transit node/prefix group is found in the first iteration of Algorithm 3.4, the algorithm will terminate without entering the recursion, which is advantageous in terms of the overall computational complexity. In this case, the second node/prefix group in the chain is simultaneously the destination node/prefix group.

```
Input: the relation graph $G_\mathrm{R}$; source node/prefix group $src\_g$; destination node/prefix group
    $dst\_g$; set of visited node/prefix groups $V_\mathrm{g}$
    GetPreferredTransitGroup ($G_\mathrm{R}, src\_g, dst\_g, V_\mathrm{g}$)
 1: if InboundArcsCount ($G_\mathrm{R}, dst\_g$) $> 0$ then
 2:    $g \leftarrow$ NULL {The selected transit node/prefix group}
 3:    $w \leftarrow 0$ {Weight assigned to the selected arc ending at $dst\_g$}
 4:    $V_\mathrm{g} \leftarrow V_\mathrm{g} \cup \{src\_g, dst\_g\}$
 5:    $S \leftarrow$ GetAdjacentGroups ($G_\mathrm{R}, dst\_g$)
 6:    $S \leftarrow S \setminus V_\mathrm{g}$
 7:    for all $s \in S$ do
 8:       if not IsAvailable ($s$) then
 9:          proceed to the next iteration
10:       end if
11:       $a \leftarrow$ GetArc ($G_\mathrm{R}, s, dst\_g$)
12:       if $g =$ NULL or $w >$ GetWeight ($a$) then
13:          $g \leftarrow s$
14:          $w \leftarrow$ GetWeight ($a$)
15:       end if
16:    end for
17:    if $g \neq$ NULL then
18:       return $g$
19:    end if
20:    for all $s \in S$ do
21:       $tg \leftarrow$ GetPreferredTransitGroup ($G_\mathrm{R}$, NULL, $s, V_\mathrm{g}$)
22:       if $tg =$ NULL then
23:          proceed to the next iteration
24:       end if
25:       $a \leftarrow$ GetArc ($G_\mathrm{R}, tg, s$)
26:       $tw \leftarrow$ GetWeight ($a$)
27:       if $w = 0$ or $w > tw$ then
28:          $g \leftarrow tg$
29:          $w \leftarrow tw$
30:       end if
31:    end for
32:    return $g$
33: else if IsAvailable ($dst\_g$) then
34:    return $dst\_g$
35: else
36:    return NULL
37: end if
```

Fig. 3.4: GroupAndReroute: selection of the preferred transit node/prefix group based on the recursive examination of the relation graph.

explicitly marked as obsolete and will not be used for future decisions — they will be removed during the next update of the FIB, or at any other time, depending on the specific implementation.

3.2.5 Impact of Failures on the Relation Graph

The relation graph maintained at each node in the network provides additional information that in most cases should allow nodes to forward packets via alternative paths to destination nodes following one or more failures. The main instance of the relation graph, which is maintained by the operating system kernel, is updated just after a successful convergence of the routing protocol to ensure the consistent state of the graph during the transient phase. In the case of failure of an adjacent node or incident link, it is recommended that the arcs going out of the corresponding node/prefix group be preserved in the relation graph. Although the node/prefix group may no longer be directly reachable from the reference node, GroupAndReroute will still be able to use that node/prefix group as a transit group if any other local element of the network fails before the routing protocol updates the state of the main instance of the relation graph.

3.2.6 Memory Requirements

GroupAndReroute is designed to work alongside a routing protocol which updates the content of the system-wide routing table. An entry of the typical routing table usually contains at least two variables that specify the destination and the reference to the corresponding output network interface. If the network graph is connected, the number of entries in the routing table depends linearly on N, where N equals the number of nodes in the network. In such a case, the number of variables used in the routing table is also on the order of $O\left(N\right)$. However, GroupAndReroute requires some additional memory to maintain the relation graph. To estimate the maximum number of variables required by a single instance of the relation graph, the following assumptions are made:

– the relation graph is represented as a list of weighted arcs;
– each arc is described using three variables: index of the source node/prefix group, index of the destination node/prefix group, and weight;
– the considered reference node has d node/prefix groups;
– all node/prefix groups of the reference node are connected with each other.

Then, the number of variables describing the relation graph is on the order of $O\left(d^2\right)$, while the total number of variables required by GroupAndReroute (excluding the flow table) is on the order of $O\left(d^2 + N\right)$. In the case of the recently-proposed Keep Forwarding (KF) inport-aware routing scheme [93] that can also deal with multiple simultaneous failures in the network, the number of variables required to maintain all necessary routing tables, one per each of d network interfaces, is on the order of $O\left(d^3 N\right)$. At the same time, in flow-oriented networks in which routing decisions are made based on the routing table and an independent flow table, the number of required variables is expected to be higher and will depend on the set of descriptors used to identify particular traffic flows, which may significantly affect the maximum estimated number of entries in the flow table. In particular, if traffic flows are classified based on several descriptors, numerous individual flow entries with unique identifiers may be added to the flow table during network operation. As a result, the system may run out of space in the flow table. One of the possible countermeasures is to aggregate traffic flows based on a more general classification scheme involving a reduced set of descriptors. The related trade-off is that the previously-independent traffic flows are handled in larger groups, and thus the opportunities for precise traffic engineering and service differentiation are limited.

3.2.7 Evaluation Environment

To verify and understand the real performance of GroupAndReroute in computer and communication networks, a router prototype was built based on the PC Engines APU2 system board, the OpenBSD 6.0 RELEASE operating system with a modified kernel, and a modified OpenOSPFD routing daemon. Every system board was equipped with three 1 Gbit/s Ethernet network interfaces, 1 GHz quad core CPU, and 4 GB of RAM. To investigate whether GroupAndReroute could be gradually deployed in existing computer and communication networks, two different networks were considered: one containing eleven devices supporting GroupAndReroute (Figure 3.5), and the other one containing eleven GroupAndReroute routers and four professional Cisco 2800 Series routers with two 100 Mbit/s Ethernet network interfaces each (Figure 3.6). The Cisco routers were running the IOS operating system (IOS Software 2801, version 12.4) with one active OSPF process and default protocol-related settings. All routers in both networks relied on the manually-configured values of the *Router ID* parameter to ensure its stability, and all routers were configured as members of one OSPF area (0.0.0.0). In addition, to facilitate the verification of the proposed solution in different environments, all OSPF network interfaces were configured to have an equal OSPF metric of 1[6]. It should be emphasized though that in general, GroupAndReroute estimates the transit distances to particular node/prefix groups based on the metric values determined by the routing process, such as an OSPF daemon.

A high-level implementation diagram of GroupAndReroute in the created network router prototype is presented in Figure 3.7. To ensure that packets are forwarded via appropriate network interfaces, the operating system maintains a Forwarding Information Base (FIB) which is updated by the OSPF routing daemon. The routing daemon collects Link State Advertisement (LSA) messages

[6]Note that in the case of the considered Cisco routers, the default OSPF metric assigned to Fast Ethernet network interfaces equals 1, while the default OSPF metric assigned by the OpenOSPFD routing daemon is 10.

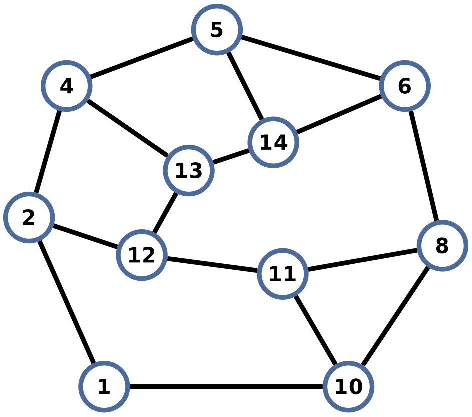


Fig. 3.5: Evaluation network containing eleven custom routers supporting GroupAndReroute (the numbering of nodes is consistent with Figure 3.6). All links in the network had the capacity of 1 Gbit/s.

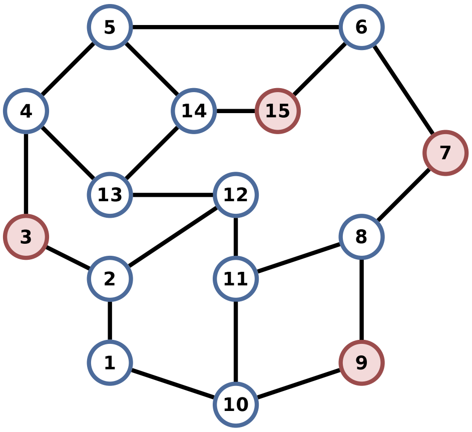


Fig. 3.6: Evaluation network containing four Cisco 2800 Series routers (filled circles) and eleven custom routers supporting GroupAndReroute (empty circles). All links connected to the Cisco routers had the capacity of 100 Mbit/s, while the other links had the capacity of 1 Gbit/s.

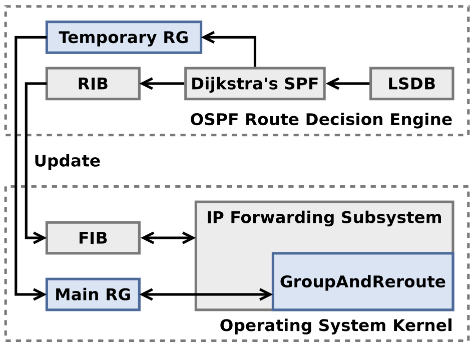


Fig. 3.7: A high-level implementation diagram of GroupAndReroute in the created network router prototype — integration of new components (Main/Temporary RG instances, GroupAndReroute) with the existing subsystems. LSDB: Link State Database, RIB: Routing Information Base, FIB: Forwarding Information Base, RG: Relation Graph.

from other nodes and stores them in the Link State Database (LSDB). Based on the received information, it executes the Dijkstra's shortest path algorithm and updates both the Routing Information Base (RIB) and the temporary instance of the relation graph. Note that GroupAndReroute components are included both in the routing daemon and in the kernel of the operating system. In the first case, the key component is the internal copy of the relation graph which is reconstructed during each execution of the Dijkstra's algorithm. Once the new routing scheme is established, the main instance of the relation graph in the operating system kernel is updated to reflect the new situation. Based on this instance of the graph, GroupAndReroute makes future decisions where to forward packets in the event of failure.

In addition to considering two different network topologies with diverse sets of network devices, the operation of GroupAndReroute was evaluated in different scenarios with respect to the number of simultaneous link failures denoted by k. During a single experiment, one, two, or three different links in the network were selected at random according to the uniform distribution, and they were made unavailable during transmission of user data. An assumption was made that the network with failed links must remain connected, as otherwise, no routing scheme would be able to reduce traffic losses related to the disconnected nodes. Link failures were simulated by physically unplugging network cables from the selected Ethernet ports of network devices. It is worth noting that the Ethernet ports could also be disabled by changing their status in the operating system to *Administratively down* (e.g., with the aid of the *ifconfig* tool), but in such a case, the ports were still active in the physical layer and the adjacent nodes were not aware of the simulated failures until they were discovered by network protocols.

Thus, the first method better reflects real failure scenarios in which links are physically damaged.

To assess the gain GroupAndReroute offers when combined with the OSPF routing protocol in the considered networks, the evaluation was based on the following metrics:

- the total number of lost packets in the network during a single experiment;
- the maximum estimated path stretch defined as follows:

$$S = \frac{l\left(P_{\mathrm{RG}}\right)}{l\left(P_{\mathrm{OSPF}}\right)} \tag{3.1}$$

where $l\left(P_{\mathrm{RG}}\right)$ denotes the maximum observed length of alternative forwarding paths of a traffic flow, resulting from the operation of GroupAndReroute following one or more link failures in the network, and $l\left(P_{\mathrm{OSPF}}\right)$ is the length of the new shortest path determined by the OSPF protocol after its reconvergence.

Note that to reflect the impact of potential transient forwarding loops on the actual length of the forwarding paths, $l\left(P_{\mathrm{RG}}\right)$ and $l\left(P_{\mathrm{OSPF}}\right)$ are determined based on the predefined initial TTL value and the final TTL values stored in the headers of the forwarded packets once they reach the corresponding destination nodes. For example, if the initial TTL value equals $TTL_{\mathrm{I}} = 64$ and the final TTL value is $TTL_{\mathrm{F}} = 61$, then the actual length of the forwarding path P is computed as follows:

$$l\left(P\right) = TTL_{\mathrm{I}} - TTL_{\mathrm{F}} + 1 = 4 \tag{3.2}$$

To extend the evaluation and provide a valuable context for the results, GroupAndReroute was compared against the following two reference network operation modes:

- Basic Flow-Oriented Operation (the baseline mode);
- Basic Flow-Oriented Operation with Loop-Free Alternates.

The first mode is the baseline mode and it assumes that traffic flows can only be rerouted when a failure of a directly connected link is detected by a network node. In this case, the affected flow entries are removed from the flow table and the new entries are created based on the current routing scheme reflected by the system-wide routing table. The second mode introduces an independent protection scheme based on the widely-known Loop-Free Alternates concept [9] and it has also been implemented in the router prototype by the author of this dissertation to provide a valuable comparison for the proposed GroupAndReroute solution. The implemented method follows the assumption that a neighbor node can provide a loop-free alternate if and only if the following condition specified in [9] is satisfied:

$$\text{Distance\_opt}\left(N, D\right) < \text{Distance\_opt}\left(N, S\right) + \text{Distance\_opt}\left(S, D\right) \tag{3.3}$$

where Distance_opt (N, D) is the distance from the neighbor node to the destination, Distance_opt (N, S) is the distance from the neighbor node to the source, and Distance_opt (S, D) is the distance from the source to the destination.

GroupAndReroute is a mechanism that works locally at a network device. In the event of failure, it selects the preferred output interfaces based on the previously collected information about the mutual connectivity of node/prefix groups. Thus, the evaluation scenarios assumed that traffic flows were sent in both directions between each pair of GroupAndReroute nodes in the network. In the performed experiments, traffic flows were identified based on five descriptors (the source and destination IP addresses, the source and destination port numbers, and the identifier of the transport protocol) and they were UDP flows involving 8000 packets of size 1470 B each. The measured rate of a single transmitted flow was approximately 380 kbit/s during its stable activity period[7], which guaranteed that no link was congested during network operation before and after failures. The UDP protocol was used because it does not involve the retransmission of packets if they cannot be delivered to the destination node. The initial value of the IPv4 TTL header field was set to 64, which was the default value configured in the operating system. Traffic flows were transmitted between the client and server applications running on each GroupAndReroute node. In addition, the flows were considered as expired after 60 s of inactivity. Each experiment included the following steps:

1. Synchronize the time in the testbed with the aid of the NTP protocol;
2. Start the server application on every GroupAndReroute node in the network;
3. Schedule the transmission of flows on every GroupAndReroute node in the network (flows are started at the same time);
4. Simulate k link failures during the transmission of traffic flows;
5. At each GroupAndReroute node, wait for 30 seconds after the transmission of locally-generated flows has finished;
6. Stop the server applications and collect the results.

For each considered value of k and the investigated network operation mode (Basic Flow-Oriented Operation, Basic Flow-Oriented Operation with Loop-Free Alternates, Basic Flow-Oriented Operation with GroupAndReroute), the corresponding experiment was repeated eight times with different sets of k randomly selected failed links to get a better understanding of network performance and, in particular, to avoid making conclusions based on arbitrarily selected cases. As the number of trials in which two network interfaces of a single router become inactive may influence the results considerably, making the comparison more difficult, it was assumed for $k \geq 2$ that exactly one out of eight trials involved failures of two links connected to the same router.

[7]The rate was estimated with the aid of the *iftop* monitoring tool [2].

3.2.8 Evaluation Results

The experiments started with the case of a network containing only GroupAndReroute-compatible devices, as shown in Figure 3.5. The evaluation results related to the total number of lost packets in the network are presented in Figures 3.8(a)-(c).

If only one failure occurred in the network ($k = 1$), deployment of GroupAndReroute allowed for a noticeable reduction of the total number of lost packets in the network, compared to the baseline mode. It also eliminated persistent forwarding loops. The second reference mode (the protection scheme based on the Loop-Free Alternates concept) was almost equally effective, but in some cases, it caused persistent forwarding loops.

The highest gain was observed when two or three randomly-selected links were disabled during the transmission of traffic flows (see Figures 3.8(b)-(c); note that the horizontal axes have the logarithmic scale). In both cases, the two reference network operation modes were not able to deal with all simultaneous failures effectively, leading to increased packet losses and persistent forwarding loops. Persistent forwarding loops occurred also in the case of single link failures ($k = 1$), unless GroupAndReroute was used to deal with failed links. An interesting observation is that although the Loop-Free Alternates-based protection scheme has demonstrated reasonable performance when dealing with single link failures, it increased packet losses in some cases involving multiple link failures, compared to the baseline reference network operation mode (Basic Flow-Oriented Operation). The potential cause may be related to the unawareness of other failures in the network at the time when the first rerouting takes place, which may lead to persistent forwarding loops experienced by the downstream nodes. On the other hand, if the forwarding decision is made based on the updated routing scheme, the new path is guaranteed to be loop-free, but the related disadvantage is the time needed for the routing protocol to reconverge.

Theoretically, it might be expected that once a link failure occurs in a network, the physically adjacent routers supporting GroupAndReroute will immediately recognize the change of the link's status and forward the following packets via alternative network interfaces, according to the locally-maintained relation graphs. In real networks, however, it may take some time before a network device actually detects a link failure. It means that in most cases, the device may not be able to take action exactly at the time of failure, which leads to a limited traffic loss. This factor is hardware and software dependent, but in the considered testbed, it was observed that up to $25 - 30$ packets per one forwarded flow could be lost before network traffic was redirected by GroupAndReroute from one network interface to another one. Note that on network devices running an operating system, the detection time involves not only hardware detection (e.g., tracking changes of the signal level), but also detection by the operating system itself (event queuing and

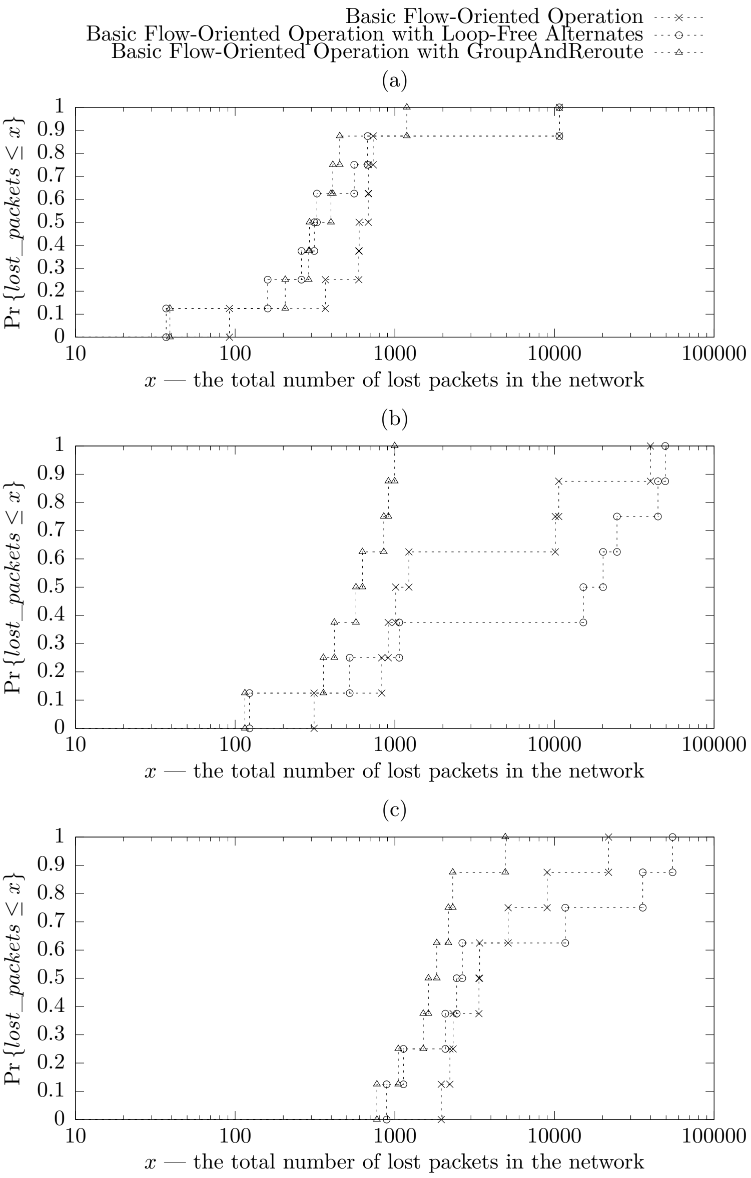


Fig. 3.8: Estimated Cumulative Distribution Function (CDF) of the total number of lost packets in the network shown in Figure 3.5 in the case of (a) one failed link ($k = 1$), (b) two failed links ($k = 2$), and (c) three failed links ($k = 3$).

handling). In addition, packets which are already in the output network buffer at the time of failure may also be lost if the corresponding link becomes unavailable.

The evaluation results related to the maximum observed path stretch corresponding to traffic flows in the network are presented in Figures 3.9(a)-(c). Note that the maximum path stretch represents the worst observed case, i.e., it is computed based on the longest alternative path of a traffic flow, provided that at least one packet has been forwarded along this path. In all three cases related to the value of k, GroupAndReroute was able to maintain the average path stretch on a relatively low level during the recovery phase. Note that the values on the vertical axes start from 0.8.

In the second scenario, the previously analyzed network was extended by connecting four professional Cisco routers to the existing GroupAndReroute-compatible nodes (see Figure 3.6). The related evaluation results are presented in Figures 3.10(a)-(c). As expected, regardless of the selected recovery scheme (Loop-Free Alternates or GroupAndReroute), the number of lost packets in the network was higher than in the first scenario, because only a fraction of nodes in the network were able to redirect traffic flows while the OSPF protocol was still recomputing the routing scheme. At the same time, the observed gain resulting from the deployment of GroupAndReroute was significant in all three cases with respect to the value of k. While the two reference network operation modes often caused persistent forwarding loops, GroupAndReroute allowed for a significant reduction of the total number of lost packets in the network, especially with increasing value of k in the considered range $1-3$. The experiments confirmed that GroupAndReroute can be gradually deployed in existing computer and communication networks, as it remains compatible with off-the-shelf communication equipment and is able to improve the overall network recovery performance, depending on network topology and the number of deployed nodes supporting GroupAndReroute. It is worth noting that in most cases involving multiple failures ($k \geq 2$), the Loop-Free Alternates-based protection scheme slightly degraded network performance, compared to the baseline mode. To improve the performance of the Loop-Free Alternates-based protection scheme in the considered type of flow-oriented network, an independent loop prevention strategy may be employed to reduce packet losses resulting from persistent forwarding loops.

The evaluation results related to the maximum observed path stretch corresponding to traffic flows in the network are presented in Figures 3.11(a)-(c). Again, in all three cases, GroupAndReroute was able to maintain the average path stretch on a relatively low level during the recovery phase.

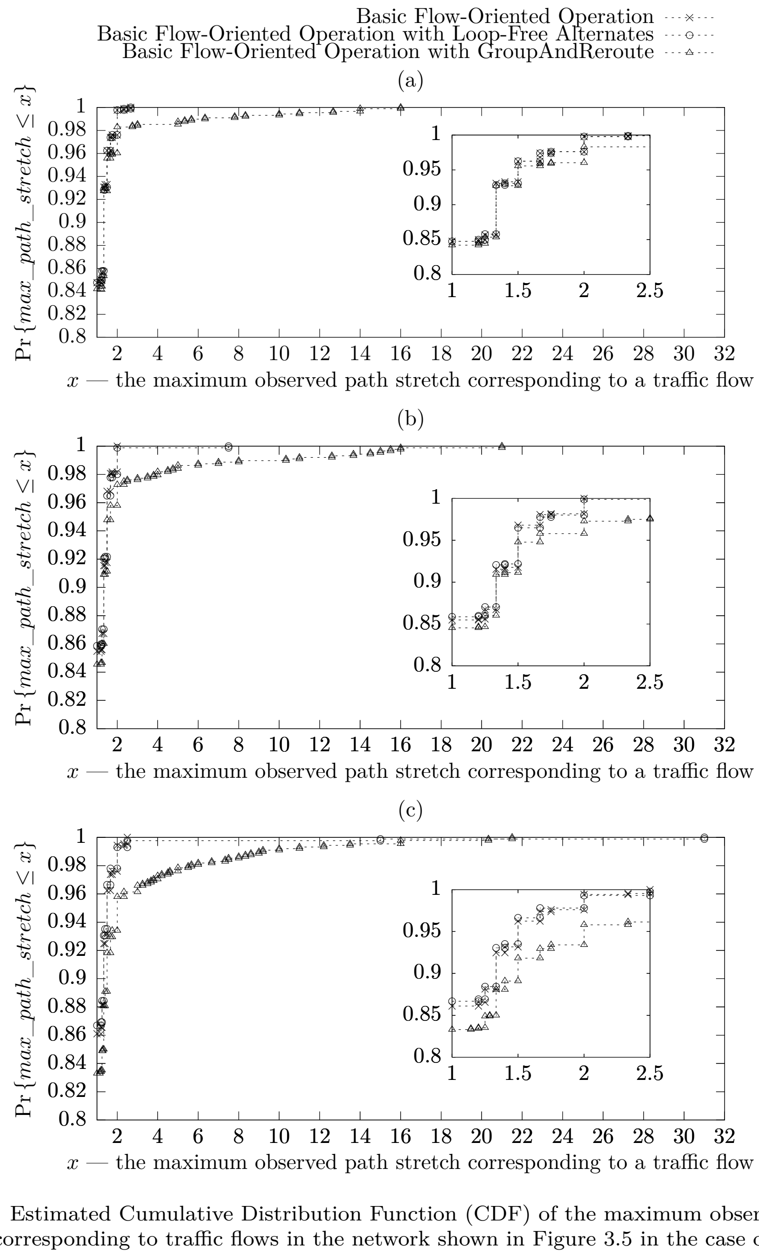


Fig. 3.9: Estimated Cumulative Distribution Function (CDF) of the maximum observed path stretch corresponding to traffic flows in the network shown in Figure 3.5 in the case of (a) one failed link ($k = 1$), (b) two failed links ($k = 2$), and (c) three failed links ($k = 3$).

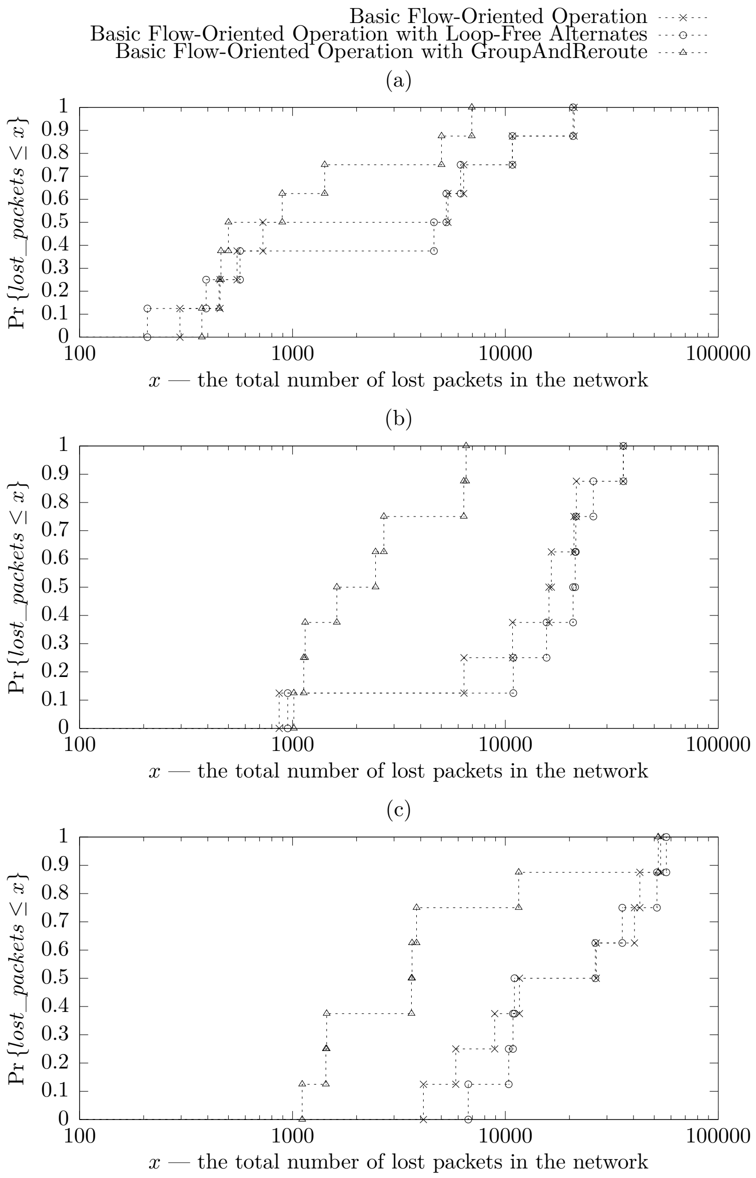


Fig. 3.10: Estimated Cumulative Distribution Function (CDF) of the total number of lost packets in the network shown in Figure 3.6 in the case of (a) one failed link ($k = 1$), (b) two failed links ($k = 2$), and (c) three failed links ($k = 3$).

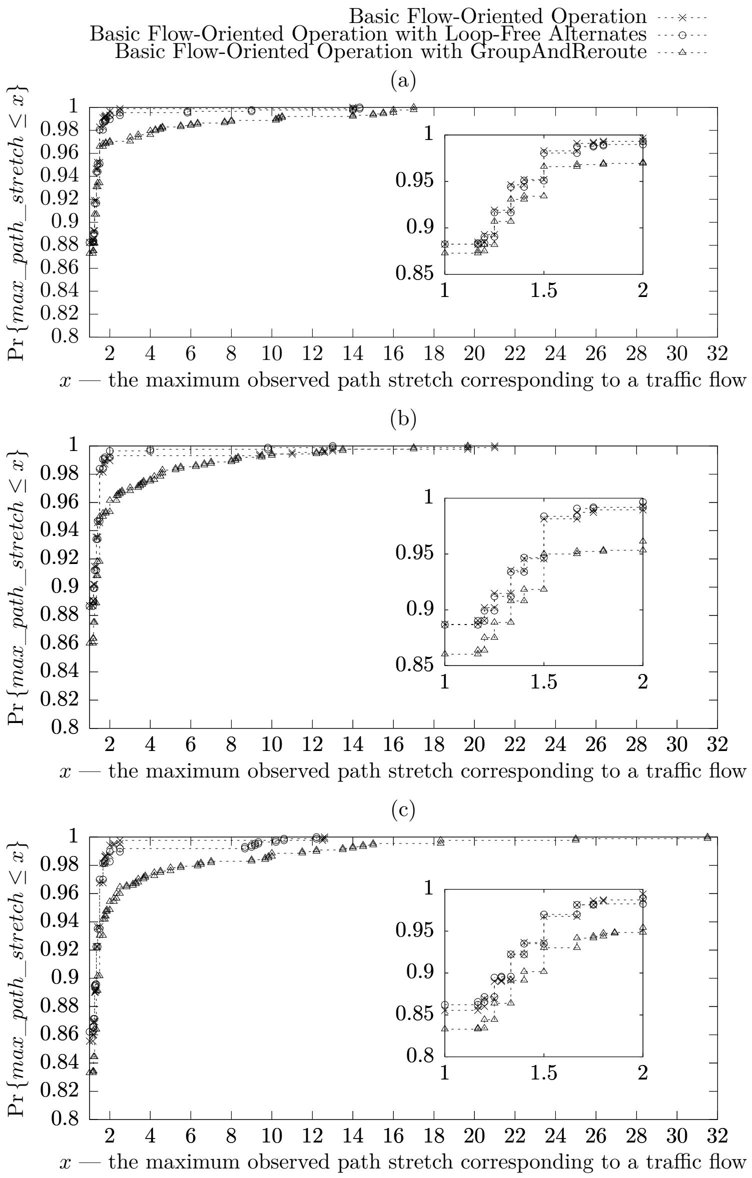


Fig. 3.11: Estimated Cumulative Distribution Function (CDF) of the maximum observed path stretch corresponding to traffic flows in the network shown in Figure 3.6 in the case of (a) one failed link ($k = 1$), (b) two failed links ($k = 2$), and (c) three failed links ($k = 3$).

3.2.9 Dealing with Forwarding Loops

It may happen that one or more failures are followed by transient forwarding loops when using GroupAndReroute in a network, especially such that contains generic communication equipment. Traffic flows entering a forwarding loop will not benefit from line-speed recovery capabilities offered by GroupAndReroute. In particular, forwarding loops can occur after one of the following events:

- one GroupAndReroute device forwards a flow to another GroupAndReroute device, but due to subsequent failures, traffic needs to be sent back to the upstream node;
- a GroupAndReroute device sends a redirected flow to a generic IP router that has not yet finished the recomputation of the routing scheme, and the flow is received on an interface being the preferred output interface for a given destination.

In most cases, transient forwarding loops should be interrupted when the routing table is updated by the routing protocol. Such an event triggers the removal of all temporary or expired flow entries from the flow table and the new entries will contain references to valid routes. Otherwise, if only a fraction of nodes in the network support GroupAndReroute, more than one update cycle of the routing table may be required, depending on the interconnection scheme of GroupAndReroute nodes. However, note that it is possible to deal with this issue by verifying at each GroupAndReroute node that the output interface of every registered flow matches the current routing scheme[8]. If there is a mismatch, the flow entry is removed and the new one will contain a reference to a valid route which is guaranteed by the routing protocol to be loop-free. On the other hand, the related disadvantage is that too many flows may change their paths following a failure. The potential solution might be to enable the proposed mechanism selectively — either on specific nodes or for selected groups of flows. It is worth to note that such a strategy might be advantageous for service providers, as it creates new opportunities in terms of service differentiation.

3.2.10 Deployment Considerations and Limitations

GroupAndReroute is designed to operate as a module which is complementary to the primary routing protocol, such as OSPF. If no failures occur in a network, it does not change (or influence the computation of) the original routing scheme. Furthermore, its operation is passive, which means that no control messages are exchanged with other nodes in addition to what the primary protocol sends. Thus, GroupAndReroute is completely transparent to other network devices and can

[8]It is worth to emphasize that a more general routing policy may be considered instead, such that includes additional factors influencing the decisions about the preferred paths assigned to particular traffic flows.

be deployed gradually in existing network environments. In addition, it has the potential to offer new methods for service differentiation on the level of traffic flows.

At the same time, if generic communication devices that do not support GroupAndReroute are also present in the network, it is recommended that the GroupAndReroute nodes be adjacent to each other whenever it is possible. In this way, the risk of occurrence of forwarding loops is limited, which may further improve network performance in the presence of one or more failures. To enable support for GroupAndReroute in the existing network devices, it is required to modify the implementation of the routing and forwarding engines.

3.3 Summary

In this chapter, the new IP Fast Reroute strategy was presented that is able to handle multiple simultaneous failures of network components in flow-oriented networks, while being suitable for a gradual deployment in existing computer and communication networks. A prototype of the design was built that demonstrated its performance in a real testbed including off-the-shelf network equipment. The evaluation results have shown that the proposed solution allows for a significant reduction of the total number of lost packets in the network, compared to the basic flow-oriented network operation mode and its enhanced variant including a Loop-Free Alternates-based protection scheme, while also maintaining the average path stretch on a relatively low level. Due to its design, GroupAndReroute can also be used as an extension to other routing solutions. Moreover, it can easily be adapted towards service differentiation with respect to separate traffic flows.

4 Dealing with Network Congestions

With the rapid growth of telecommunication networks in recent years, the daily volume of traffic forwarded through backbone networks has risen to an enormous level. Due to high costs of service disruption, especially in the case of services of strategic importance, strong reliability requirements are imposed on the core infrastructure. As limited network resources need to be shared across many users, it may happen that certain network links become overloaded, either blocking the subsequent traffic flows routed via these links or causing packet losses affecting the overall performance.

To reduce the probability of a link overload in the backbone network, Internet service providers have a few options to choose from. For example, the following general strategies may be considered:

- assign more resources to particular network segments than actually needed [22], taking as a reference the results of regularly performed load assessments;
- some new or existing traffic flows may be denied access to the network either for a predefined time or until the congestion is mitigated [24, 58];
- accept new flows and send them via one or more alternative paths [72, 88], thus limiting or reducing the load on the primary path (there are also related proposals for data center networks, such as [23, 46, 53, 54]);
- avoid congestions through the use of Explicit Congestion Notification and Active Queue Management mechanisms (including specific rate control algorithms) [11, 38, 78].

In the third case, the existing flows may stay on their original paths, but it is also possible to use a more flexible approach and reallocate the selected existing flows if necessary.

In this chapter, the opportunities for effective congestion control mechanisms in the third category (i.e., accepting new flows and sending them via one or more alternative paths) are explored with an assumption that the existing traffic flows may be reallocated if needed. The related solutions are discussed in Section 4.1. It is shown that while the new algorithms proposed in Section 4.2 have limitations, they can significantly improve network performance in the presence of link congestions in centrally-managed flow-oriented networks, compared to the strategies discussed in [53, 54] and the case when routing is only managed by the OSPF protocol configured in such a way that it relies on a single path to each destination. In particular, it is demonstrated that in the case of one considered evaluation scenario in which only a few links in the network were fully loaded, the presented algorithms were able to reduce that number down to 0 (or to a close value), provided that enough network resources were available.

The content of this chapter has originally been included in a journal paper [52] and the related online companion [50]. The research was carried out with the support of the project "High quality, reliable transmission in multilayer optical networks based on the Flow-Aware Networking concept" funded by the Polish National Science Centre under project no. DEC-2011/01/D/ST7/03131.

4.1 Related Work

An example of a solution representing the third strategy in the case of centrally-managed networks was presented in [54]. It assumes that new traffic flows are transmitted along the shortest (in terms of the number of hops) paths (candidate paths) that also have the minimum cumulative utilization of links belonging to them. This approach has one major limitation: as it was designed for data center networks which usually offer multiple shortest paths to the selected destination node, it does not consider longer alternative paths.

A similar solution that reroutes traffic flows one-by-one in the case of link congestion is discussed in [53]. In this proposal, rerouting may take place at the point of congestion or one hop before and it is restricted to *elephant* flows (i.e., flows for which the observed size is at least 100 kB). The alternative path is selected based on the following criteria: minimum length (determines the set of candidate paths), the maximum load among links belonging to the path, and whether the new path will be overloaded once the selected flow is moved. While the idea to consider the maximum load among all links of a path is reasonable[1], the proposed solution suffers from the same limitation as the previous approach with respect to the set of candidate paths.

Another related solution relies on counting flows of the same type (e.g., http or ftp flows) for each link individually [72]. Each flow type has the corresponding predefined weight, which may also be determined based on the estimated bit rate.

[1]It is worth noting that the mechanism proposed in [53] compares absolute load values, which may lead to wrong decisions if network links have significantly different capacities.

The main idea is that the more flows of significant types are observed on a link, the more likely the link is to enter the congestion state, which should be avoided. At the same time, the weighted scheme may not reflect the actual situation in the network if the size of flows varies considerably within each category, not to mention the unknown, yet potentially strong influence of unclassified flows.

4.2 New Reallocation-Based Congestion Control Algorithms

In this section, two alternative solutions are proposed to respond to network congestions in centrally-managed flow-oriented networks through the reallocation of traffic flows. Both algorithms do not suffer from the limitations identified and discussed in Section 4.1. The evaluation environment and the simulation results are presented and discussed in Sections 4.2.3-4.2.4, while the deployment considerations and limitations of the proposed strategies are summarized in Section 4.2.5.

4.2.1 Algorithm I: Max Path Load and Path Overload Probability

The first algorithm relies on information about the estimated overload probability of a path and the maximum relative load among all links belonging to the path. As presented in Figure 4.1, the algorithm requires access to the network graph G and must be invoked in the context of an overloaded link e_{ov}. l_{th} represents the global value of the link overload threshold[2]. Before entering the main loop, the algorithm prepares the list of flows forwarded via the overloaded link and selects the one with the highest demand as *the most significant flow.* Then, it enters the loop (lines 3-44) which is executed for as long as link e_{ov} remains congested, or alternatively, until one of the following conditions is satisfied:

- the most significant flow could not be reallocated,
- all flows on link e_{ov} have already been considered.

Distances assigned to particular destination nodes during the execution of the modified Dijkstra's shortest path algorithm on graph G (lines 7-37) depend on the maximum expected relative load among all links belonging to a path and the estimated overload probability of the entire path from the source node to the

[2]The lowest value of the relative link load (link capacity utilization) for which the link is perceived as congested.

```
Input: network topology modeled as a weighted graph $G = (V, E)$; relative link overload
    threshold $l_{th} \in [0, 1]$; overloaded link $e_{ov} \in E$; weight parameter $\beta$
 1: $F \leftarrow \text{GetFlows}(e_{ov})$
 2: $f \leftarrow \text{GetTheMostSignificantFlow}(F)$
 3: repeat
 4:   $f_{\text{previous}} \leftarrow f$
 5:   $P_{\text{current}} \leftarrow \text{GetCurrentPath}(f)$
 6:   $R_f \leftarrow \text{GetDemand}(f)$
 7:   $Q \leftarrow \emptyset$
 8:   for all $v \in V$ do
 9:     $D[v] \leftarrow MAX\_COST$ {distance/cost vector relative to the source node of flow $f$}
10:     $B[v] \leftarrow NULL$ {predecessor nodes according to the computed routing scheme}
11:     $Q \leftarrow Q \cup \{v\}$ {set of nodes to be considered in the while loop (line 14)}
12:   end for
13:   $D[\text{GetSource}(f)] \leftarrow 0$
14:   while $Q \neq \emptyset$ do
15:     $u \leftarrow \text{GetVertexWithMinDistance}(Q, D)$
16:     $Q \leftarrow Q \setminus \{u\}$
17:     for all $v \in \text{GetNeighbors}(u) \cap Q$ do
18:       if $(u, v) = e_{ov}$ then
19:         Proceed to the next iteration
20:       end if
21:       $P_t \leftarrow \text{GetShortestPath}(D, B, \text{GetSource}(f), u)$
22:       $P_t \leftarrow P_t \cup \{(u, v)\}$
23:       $l \leftarrow 0$ {the initial value of the maximum expected relative load among all links of
          path $P_t$}
24:       $p \leftarrow 1$ {the initial value of the estimated probability that path $P_t$ will not be
          overloaded}
25:       for all $e \in E \cap P_t$ do
26:         $p \leftarrow p \cdot (1 - \text{GetEstimatedOverloadProb}(e, l_{th}))$
27:         if $\text{GetExpectedRelativeLoad}(e, R_f) > l$ then
28:           $l \leftarrow \text{GetExpectedRelativeLoad}(e, R_f)$
29:         end if
30:       end for
31:       $d = \beta l + (1 - \beta)(1 - p)$
32:       if $d < D[v]$ then
33:         $D[v] = d$
34:         $B[v] = u$
35:       end if
36:     end for
37:   end while
38:   $P_{\text{new}} \leftarrow \text{GetShortestPath}(D, B, f)$
39:   if $\text{GetCost}(P_{\text{new}}) < \text{GetCost}(P_{\text{current}})$ then
40:     $\text{ReallocateFlow}(f, P_{\text{new}})$
41:     $F \leftarrow F \setminus \{f\}$
42:   end if
43:   $f \leftarrow \text{GetTheMostSignificantFlow}(F)$
44: until $\text{IsCongested}(e_{ov}) = FALSE$ or $f = f_{\text{previous}}$ or $f = NULL$
```

Fig. 4.1: Congestion control algorithm relying on information about the estimated overload probability of a path and the maximum relative load among all links belonging to the path.

currently considered node (line 31)[3], based on the previously computed values[4]. The β parameter defines the importance of each of the two mentioned factors. In particular, it is possible to consider only one factor by setting β to either 0 or 1 (see line 31). Once the distances are computed, it is possible to determine the best alternative path P_{new} for flow f, such that does not include link e_{ov}. If its cost is smaller than the cost of the currently used path P_{current}, the flow is reallocated. In all other cases, the flow remains on the original path[5].

4.2.2 Algorithm II: Max Path Load and Path Length

The second algorithm is very similar to the first one, but instead of the estimated overload probability of a path it considers its length in terms of the number of links. For this reason, only three lines in Figure 4.1 need to be changed as follows:

- line 24: $m \leftarrow 0$,
- line 26: $m \leftarrow m + 1$,
- line 31: $d = \beta l + (1 - \beta)\,\frac{m}{|E|}$,

where m is the number of links belonging to the constructed path.

The motivation for the second algorithm is that less congested alternative paths in a network do not necessarily have to be close to the corresponding primary paths in terms of their lengths. Thus, it is now possible to explicitly assign the importance to the length factor through setting the β parameter to a desired value.

4.2.3 Evaluation Environment

The evaluation of the proposed algorithms was based on a simulation study carried out with the aid of a discrete-event, flow-level network simulator written in the C++ programming language[6]. For the purpose of random data generation (e.g., flow inter-arrival time, flow duration and demand, source and destination nodes), the random number generation engine of the widely-used ns-3 v3.24.1 network simulator [3] was used. The underlying random number generator was the MRG32k3a generator, for which the period length is approximately $3.1 \cdot 10^{57}$ [63].

[3]Note that the overload probability of a path is computed as 1 minus the probability that all links belonging to this path are not overloaded. In other words, the path is overloaded if at least one of its links is. The expected relative load of a link is computed as the capacity utilization (value in the range of $[0, 1]$) involving the cumulative demand of all current flows forwarded via this link and the demand of flow f, as if it was assigned to this link.

[4]Different implementation strategies are possible here, for example, a fixed number of samples may be kept in a circular buffer.

[5]One potentially interesting modification of the algorithm is to try to reallocate the subsequent significant flows if the most significant flow could not be moved. It is expected that this may improve the effectiveness of the algorithm at the cost of an increased runtime.

[6]For a detailed description of the simulator, the reader is referred to Chapter 6.

Following the recommendations provided by the ns-3 project [4], independent simulation replications were obtained by setting the seed to a fixed value, while the simulation run number was different for each replication.

The following three general evaluation scenarios were considered:

– evaluation in a US backbone network topology (similar flow demands),
– evaluation in a Viatel backbone network topology (similar flow demands),
– dealing with network congestions in the presence of large traffic flows.

In each of the considered scenarios, the proposed algorithms were compared with two existing solutions introduced in [53, 54], as well as with the third reference case (OSPF) in which traffic flows were forwarded along the shortest paths and there were no active congestion mitigation mechanisms in the network. Each simulation was repeated 10 times to get credible results. The average values were computed over the period from the 150th to the 650th second of each simulation to avoid the impact of the warm-up and termination phases on the final results.

The objective of the study was to evaluate the effectiveness of the considered flow-level congestion mitigation strategies. Note that the study was focused on different flow allocation strategies and does not extend to congestion control mechanisms introduced on other levels (e.g., in the TCP protocol).

4.2.4 Evaluation Results

In this section, the evaluation results of the proposed algorithms are presented. The first two considered cases are related to traffic flows of similar demands and they also illustrate how different network topologies influence the ability of the algorithms to find alternative paths. Finally, in the third case, the performance of both solutions is evaluated in the presence of some additional traffic flows having relatively high demands.

Evaluation in the case of the US backbone network topology and similar flow demands

In the first simulation scenario, each link of the network shown in Figure 4.2 had the capacity of 1 Gbit/s. The corresponding relative overload threshold was set to 0.7, which means that a link was perceived as *congested* whenever its capacity utilization equaled 70% or more. Further, a network link was perceived as *fully loaded* whenever its capacity utilization reached 100%. The congestion state of links was monitored every 1 s, as this interval was a reasonable trade-off between accuracy and the amount of data to be processed for the considered simulation scenario. In real networks, however, the interval might be set to a greater value (e.g., $5 - 10$ s), as the data from several forwarding devices needs to be transferred to (and processed by) the logically-centralized management unit. The congestion probability of each link was estimated based on 100 most recent samples stored in

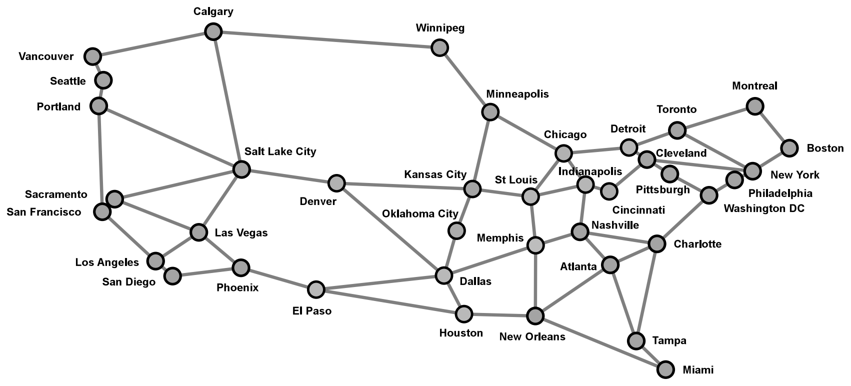


Fig. 4.2: The topology of the US backbone network containing 39 nodes and 122 unidirectional links (based on the data provided by the *SNDlib* project [74]; name of the model: *janos-us-ca*). To maintain clarity, the figure presents an undirected graph.

a circular buffer. 10000 traffic flows were scheduled in each of the 10 simulation trials to make sure that numerous combinations of flows on each link were possible. The flows were generated in such a way that the fraction of overloaded links in the network oscillated around an average value for at least 500 s (the measurement period) during network operation. The source and destination nodes were selected at random according to the uniform distribution. The flow inter-arrival time was selected according to the exponential distribution with the mean value of 0.1 s and the maximum value (bound) of 0.2 s, while the duration of each flow followed the Pareto distribution with the mean value of 120 s, the maximum value (bound) of 240 s, and the shape parameter equal to 1.5. The average flow demand was set to 10 Mbit/s, whereas the actual values were selected at random from range $[7.5; 12.5]$ Mbit/s according to the uniform distribution. The additional bounds on some random variables were used to avoid extreme cases of very large values. The numerical values were selected in such a way that the generated traffic triggered network congestion events for multiple links simultaneously, also ensuring that a fraction of links were fully loaded — about $4-5\%$ of all links on average in the case of the OSPF-like routing with no additional congestion avoidance measures.

The evaluation results are presented in Figures 4.3(a)-(c). In terms of the average fraction of overloaded links, the main point was to observe how the proposed solutions influence the number of congested links in the network, compared to the other considered strategies. According to Figure 4.3(a), only the algorithm presented in [53] rearranged traffic flows in such a way that the total number of overloaded links decreased, while the other methods (except for the algorithm introduced in [54] and the OSPF protocol which was not responding to congestions) often transferred the excessive load to previously uncongested links. However, by

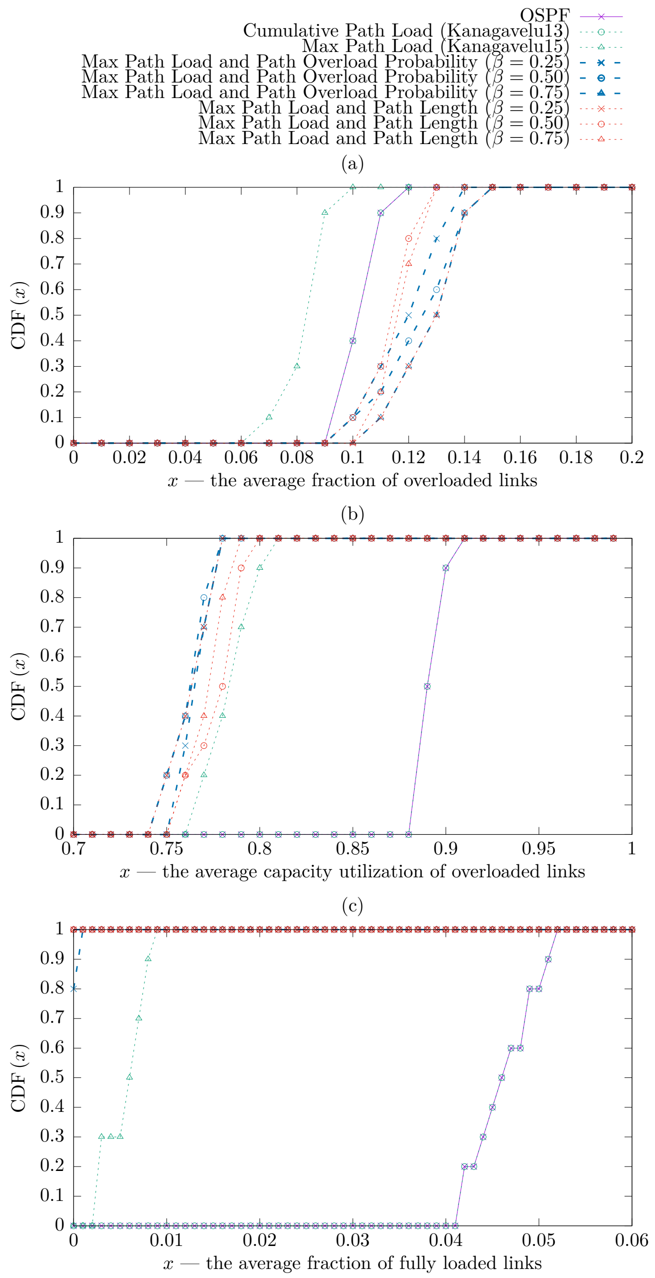


Fig. 4.3: Estimated Cumulative Distribution Function (CDF) of (a) the average fraction of overloaded links, (b) the average capacity utilization of overloaded links, and (c) the average fraction of fully-loaded links between the 150th and 650th second of simulations.

increasing the load of such links, it was possible to reduce the average capacity utilization of the overloaded links in the entire network, thus avoiding traffic losses on the most congested links. This observation corresponds to Figure 4.3(b) in which the proposed algorithms performed better than all three reference solutions in terms of the average capacity utilization of all overloaded links in the network. In particular, the largest difference was observed for OSPF and the algorithm introduced in [54], both of which have demonstrated comparable performance. The results shown in Figure 4.3(c) also confirm the previous statement. While the observed number of fully-loaded links in the case of the algorithm presented in [53] was considerably smaller (on average) than the respective value for OSPF and the third reference solution [54], it needs to be emphasized that the two proposed strategies reduced the number of fully-loaded links down to 0 almost for all of the considered values of β, effectively ensuring that the congested links were not causing traffic losses in the network. It means that except for the case of Algorithm I when β was equal to 0.25, the following relation was true: $\forall_{x\in[0;1]}\ \mathrm{CDF}\,(x) = 1$, as presented in Figure 4.3(c). At the same time, it should be emphasized that the number of fully-loaded links was relatively small during the entire simulation[7].

Until now, we have discussed the performance of the proposed solutions without considering the value of the β parameter. In the case of the second algorithm, the smallest value of β resulted in a considerably larger average fraction of overloaded links in the network, compared to the two other considered values for which the corresponding difference was not significant. This was an expected outcome, as the small weight assigned to the maximum relative path load increased the priority of the load-independent factor (relative path length). In terms of the average capacity utilization of the overloaded links, the smallest gain in relation to the OSPF case was observed for $\beta = 0.50$, while the highest gain was associated with $\beta = 0.25$. One possible explanation of this observation is that, as the algorithm generally preferred shorter alternative paths, fewer links were carrying the offloaded traffic flows.

In the first algorithm, both factors are load-dependent and their mutual importance is also determined by the value of the β parameter. Based on the results shown in Figures 4.3(a)-(c), it was noticed that setting β to different values had impact mainly on the average fraction of overloaded links (the smallest values were observed for $\beta = 0.25$) and the average fraction of fully-loaded links. At the same time, the influence of β is almost not visible in Figure 4.3(c), as the proposed algorithms were effectively preventing all links from entering the fully-loaded state,

[7] In fact, backbone network operators usually prevent even such a small number of links from being fully loaded, as they start to activate additional network resources in advance when the capacity utilization of a link reaches a predefined level, which is typically around 70%.

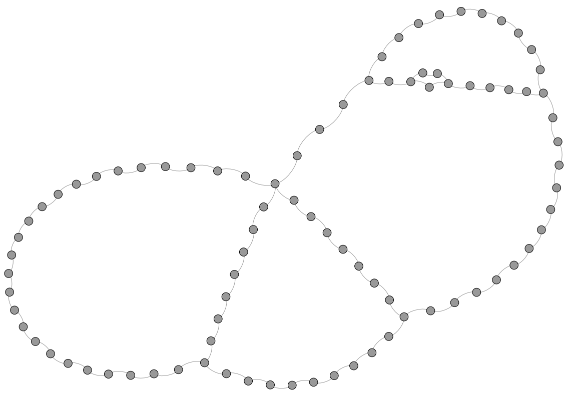


Fig. 4.4: The topology of the Viatel backbone network containing 88 nodes and 184 unidirectional links (based on the data provided by *The Internet Topology Zoo* project [57]; version from 2008, modified layout). To maintain clarity, the figure presents an undirected graph.

except for a very small fraction that was observed only for $\beta = 0.25$ when using Algorithm I.

Evaluation in the case of the Viatel backbone network topology and similar flow demands

The main goal of this scenario was to verify the performance of the proposed algorithms in the case of a different network topology. Figure 4.4 shows the Viatel backbone network topology. Note that compared to the US backbone network considered in the previous section, the Viatel network contains significantly less cycles and the majority of its nodes have degree equal to 2. This means that in the event of network congestion, the number of alternative paths that may be selected to reduce the number of congested links in the network is also much smaller. Thus, it is more difficult to mitigate the congestion.

The simulation strategy was similar to the previously discussed case, with the exception for the link congestion monitoring interval which was increased to 10 s due to the significantly higher frequency of the necessary flow reallocations triggered by the congested links. The corresponding simulation parameters are summarized in Table 4.1.

Figures 4.5(a)-(c) present the corresponding evaluation results. It was observed that the fraction of congested links was higher (for all the considered solutions) than in the case of the US backbone network. In addition, both proposed algorithms maintained more links in the overloaded state than the reference strategies. At the same time, in terms of the average capacity utilization of the congested links and the average fraction of the fully-loaded links, it may be observed that Algorithm I

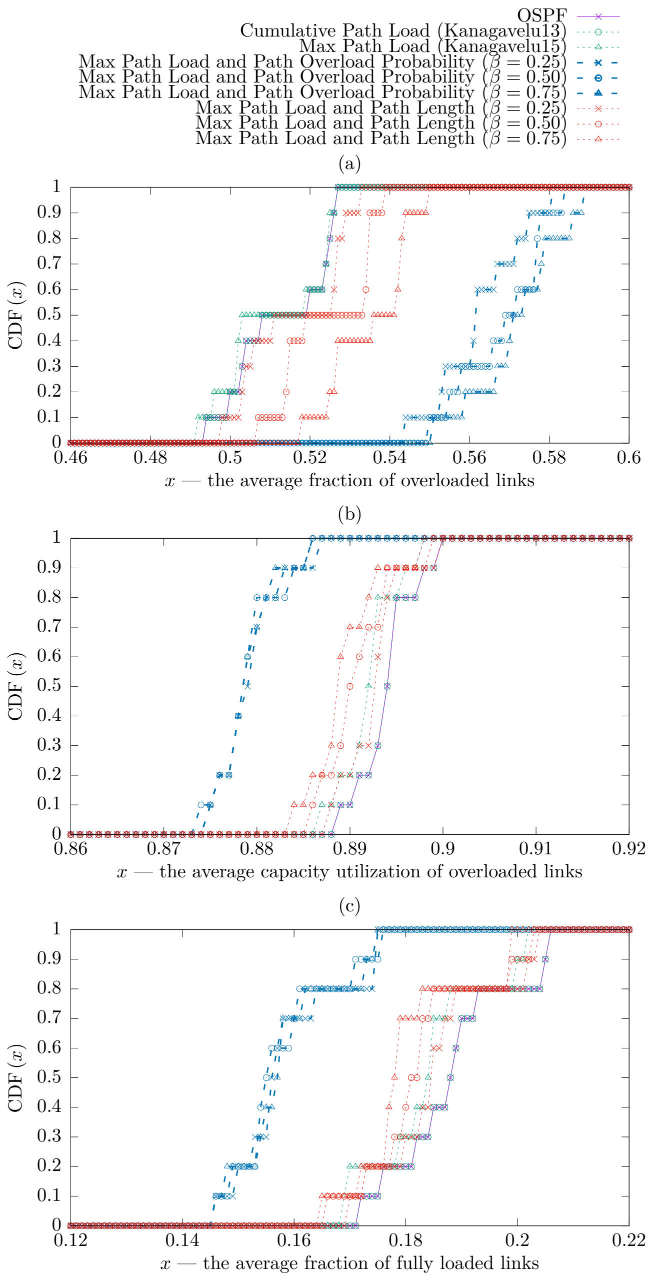


Fig. 4.5: Estimated Cumulative Distribution Function (CDF) of (a) the average fraction of overloaded links, (b) the average capacity utilization of overloaded links, and (c) the average fraction of fully-loaded links in the Viatel network between the 150th and 650th second of simulations.

Table 4.1: Simulation parameters in the case of the Viatel backbone network.

Parameter	**Value**
Network topology	Viatel backbone (version from 2008 containing 88 nodes and 184 unidirectional links)
Link capacity	1 Gbit/s
Relative overload threshold	0.7
Link congestion monitoring interval	10 s
Link overload probability	Estimated based on 100 most recent samples stored in a circular buffer
Number of traffic flows	10000 (source and destination nodes selected at random according to the uniform distribution)
Demand of flows	Selected at random from range [7.5; 12.5] Mbit/s (uniform distribution), 10 Mbit/s on average
Duration of flows	120 s on average, Pareto distribution, upper bound 240 s, shape 1.5
Flow inter-arrival time	0.1 s on average, exponential distribution, upper bound 0.2 s

(Max Path Load and Path Overload Probability) performed significantly better than Algorithm II (Max Path Load and Path Length) and the considered reference mechanisms. This observation is related to the fact that with increasing number of congested links, the probability that the candidate path is overloaded becomes an important factor when selecting an alternative path. Thus, in highly congested networks, it is recommended in this work that Algorithm I be used to reallocate traffic flows.

Dealing with network congestions in the presence of large traffic flows

To provide more information about the performance of the proposed solutions, an additional case has been analyzed in which some traffic flows in the network had significantly higher average throughput than the other flows. The simulation parameters are summarized in Table 4.2.

The corresponding evaluation results are presented in Figures 4.6(a)-(c). Again, the two proposed flow reallocation strategies caused relatively more links to enter the state in which they were perceived as congested, compared to the other algorithms and the OSPF-like routing with no additional congestion avoidance measures. However, the average capacity utilization of such links in the case of Algorithm I was considerably smaller than for the considered reference strate-

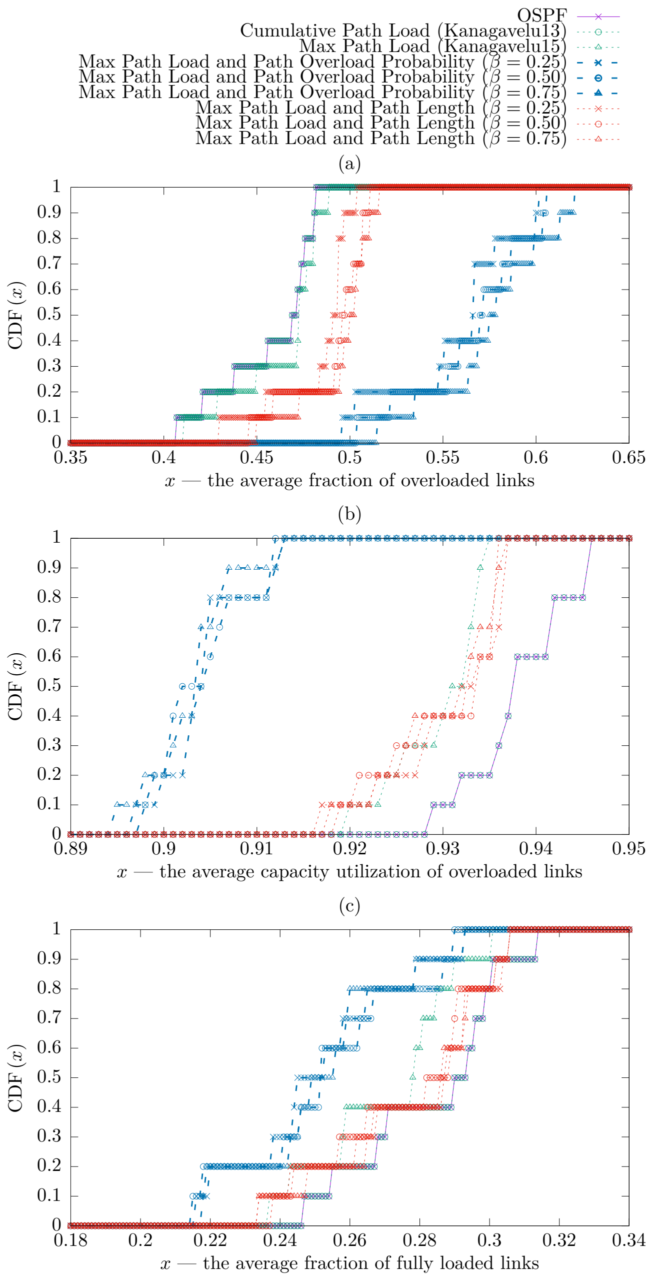


Fig. 4.6: Estimated Cumulative Distribution Function (CDF) of (a) the average fraction of overloaded links, (b) the average capacity utilization of overloaded links, and (c) the average fraction of fully-loaded links in the US backbone network between the 150th and 650th second of simulations.

Table 4.2: Simulation parameters in the case of the US backbone network and flows of diverse demands (small and large).

Parameter	**Value**
Network topology	US backbone (39 nodes and 122 unidirectional links)
Link capacity	1 Gbit/s
Relative overload threshold	0.7
Link congestion monitoring interval	10 s
Link overload probability	Estimated based on 100 most recent samples stored in a circular buffer
Number of traffic flows	10000 (source and destination nodes selected at random according to the uniform distribution)
Probability that a new flow is a large flow	0.05
Demand of large flows	Selected at random from range [75; 125] Mbit/s (uniform distribution), 100 Mbit/s on average
Demand of other flows	Selected at random from range [7.5; 12.5] Mbit/s (uniform distribution), 10 Mbit/s on average
Duration of large flows	300 s on average, Pareto distribution, upper bound 600 s, shape 1.5
Duration of other flows	120 s on average, Pareto distribution, upper bound 240 s, shape 1.5
Flow inter-arrival time	0.1 s on average, exponential distribution, upper bound 0.2 s

gies, which is a strong advantage. The results related to the average fraction of fully-loaded links are even more important (Figure 4.6(c)) — it was possible to reduce the number of fully-loaded links noticeably by using Algorithm I, while Algorithm II demonstrated slightly worse performance than the best considered reference solution. It was an expected behavior, as Algorithm I had been identified to perform better in more congested networks (see the previous simulation scenario).

4.2.5 Deployment Considerations and Limitations

The proposed solutions have some limitations. First, they require a complete view of a network in terms of the load of particular links, including the ability to reallocate traffic flows and make decisions based on the previously collected data (Algorithm I, Section 4.2.1). Considering the number of monitored links and flows in the network, this requirement may lead to scalability issues. Second, none of them is perfect in all scenarios — while Algorithm I performed well in heavily congested networks, it was not the case for Algorithm II. At the same time, both algorithms are compatible with the concept of Software-Defined Networking [70] which is becoming increasingly popular among researchers and network engineers. In particular, the algorithms might be deployed in a logically-centralized network controller that makes routing decisions, configures traffic flows, and monitors the state of flows and network switches.

The related scalability issues may be addressed in the following two ways. The first strategy might be to divide the entire network into several independent domains and use the proposed algorithms within each domain to reduce the number of fully-loaded internal links. The related trade-off involves such factors as the amount of collected data, the processing capacity and speed, and the quality of the result in the context of the entire network. In particular, it should be noted that significant traffic flows traversing multiple domains may not always be reallocated in a way that is advantageous with respect to the total number of fully-loaded links in the network. The second strategy might be to adjust the processing capacity and speed of the logically-centralized network controller on demand, possibly taking full advantage of the available virtualization technology. In this case, the proper synchronization of the controller instances with respect to the current traffic and network state becomes an important challenge.

4.3 Summary

In this chapter, the problem of network congestions and their impact on network dependability was discussed and addressed specifically in the case of centrally-managed flow-oriented networks. The importance of being able to respond to network congestions effectively was emphasized, and the related congestion control strategies were classified and briefly summarized. To deal with the identified shortcomings of the existing solutions, two new algorithms were proposed that are based on the flow reallocation strategy. Both algorithms were evaluated with the aid of a discrete-event flow-level network simulator described in Chapter 6. The evaluation included two recent reference approaches, as well as the case when no congestion control is provided in the network. Based on the results, it was concluded that the proposed algorithms have the potential to reduce the number of fully-loaded links in centrally-managed flow-oriented networks, such as networks based on the Software-Defined Networking concept.

Since the proposed algorithms deal with one overloaded link at a time, they will be triggered separately for each congested link. However, it is also possible that the traffic flows forwarded via the currently-analyzed congested link are reallocated by the considered algorithms in such a way that the congestions on other links are mitigated as well, removing at least some of the future triggers. Thus, determining the best sequence in which the overloaded links should be analyzed by similar algorithms at a certain time during network operation is an interesting research problem for future studies. The selected strategy might also be extended towards parallel analysis of congested links.

5 Risk Analysis and its Role in the Provisioning of Network Services

An increasing demand for diverse network services with different dependability requirements has prompted Internet Service Providers (ISPs) to modernize and expand their national and global infrastructure. To ensure that the services are delivered at the required levels with respect to dependability, ISPs constantly monitor their networks and respond to all events affecting the performance of their services [34, 69]. As part of this task, they also try to estimate the risk of violation of the dependability requirements specified in Service Level Agreements (SLAs) signed with their customers to prepare and deploy adequate protection measures on time, thus avoiding major service disruption and the related penalty [30, 35, 92]. Each violation of the dependability-related Service Level Objectives (SLOs) defined in the corresponding SLA may lead to significant monetary consequences, affecting the ISP's reputation.

With the introduction of Software-Defined Networks (SDNs) [60, 70], in which the control plane is decoupled from the data plane, the research community and telecommunication industry became interested in the flexibility they offer, and in the expected simplification of network management tasks. However, researchers soon realized that it is necessary to identify the potential sources of failures in such networks, as well as their impact on the overall dependability of a system [39–41, 59, 65, 80]. Further, it is not clear how to construct dependability-related SLOs in the case of SDNs, and how to estimate the related SLA violation risk. No previous work has addressed this issue, while it needs to be solved before ISPs start using SDN in their network infrastructure. The currently used metrics in SLAs mainly refer to service downtime which has not been explicitly defined in the case of SDNs. The existing proposals cover non-SDN networks in which an ICT service is either available or not [30, 35, 37, 92]. At the same time, in SDNs, traffic flows established before a failure may still be successfully forwarded through a network, while the new flows between the same pair of nodes may be rejected due to the unavailability of the logically-centralized controller. Thus, the existing solutions that cannot handle this case are not directly applicable to SDNs and should be revisited in this context.

The objective of the research presented in this chapter was to provide ISPs and their customers with a valid, flexible, and easy-to-use solution that allows them to i.) define the dependability-related SLOs for traffic flows in SDNs, and ii.) assess the respective risk of violation of the SLA dependability requirements and take appropriate measures in advance to ensure that this risk remains within an acceptable range. Thus, the main contributions can be summarized as follows:

– the main factors affecting the dependability of SDN networks from the perspective of customers and their ability to utilize their offered services are identified;
– *service degradation* is defined as the key measure of decreased dependability in SDN environments;
– a method for the assessment of the SLA violation risk with respect to the dependability requirements for traffic flows in SDNs is proposed and evaluated by simulation.

The definitions of dependability, availability, and reliability of systems used in this chapter are based on [10].

The remainder of this chapter is structured as follows: in Section 5.1, the related work is discussed, while the general architecture of SDN and the challenges of the SLA-based business relationship model with respect to the ability to assess the dependability of traffic flows in SDNs is discussed in Section 5.2. Section 5.3 introduces the *service degradation* metric used in the assessment of risk of violation of the SLA dependability requirements for traffic flows in SDN. In particular, Sections 5.3.1-5.3.2 describe the evaluation strategy and summarize the results, while the limitations of the proposed solution are discussed together with the related deployment considerations in Section 5.3.3. Finally, Section 5.4 summarizes the research presented in this chapter.

The content of this chapter has originally been included in the research paper that will be presented at the 2017 IEEE Conference on Network Function Virtualization and Software-Defined Networks (NFV-SDN) — Fourth Workshop on Network Function Virtualization and Programmable Networks (NFV-SDN'17-NFVPN) [51]. The research was supported by the collaboration project between Telenor Research and NTNU QUAM Research Lab.

5.1 Related Work

One of the most representative application areas of risk assessment-based techniques is related to safety systems and critical infrastructures needed for the operation of a society, such as power provisioning systems [32, 67, 86]. While the relevant contributions related to computer and communication networks have been limited for a long time, some papers encouraged the use of risk analysis in the assessment of network reliability [13]. With the increasing importance of telecommunication networks over time, service providers started to look at effective tools to estimate the potential impact of failures on their networks, services, and business relationships. The related research evolved towards supporting risk-aware provisioning of network services [17, 19, 30, 35–37, 92].

The use of risk analysis in the context of telecommunication networks was strongly advocated in [19]. The authors analyze the possible approaches, focusing their discussion on the following key aspects: risk framing (including the risk management cycle), risk assessment and modeling techniques, different strategies for an effective response to the risk, and risk monitoring.

The solution presented in [92] allows for the minimization of the SLA violation risk with respect to the availability target in optical WDM networks. The authors analyze single paths and failure arrival rates of the involved links to show that it is possible to achieve lower SLA violation risk by routing according to the adjusted failure arrival rate of the path, compared to a routing scheme in which the routing metric is based on availability. In addition, the authors also propose an extended version of the algorithm that is applicable to shared-path protection scenarios.

In [35], the interval availability and its impact on the SLA violation risk was considered. In particular, different lengths of the observation period were analyzed with respect to the resulting distribution of the interval availability. Furthermore, while the numerical methods used in the previous studies often relied on Markovian models, the authors proposed a new method that enables the analysis of non-Markovian systems.

In the following paper [36], the authors present a method to model the probability distribution of the accumulated downtime. The method is then used in the proposed hybrid management strategy which assumes that multiple resilience provisioning techniques of different effectiveness and cost may be used in a coordinated way to control the SLA violation risk and satisfy the SLOs, while also minimizing the expenditures. Moreover, the authors recommend that the more reliable recovery technology be used near the end of the SLA observation period, because it leads to a smaller variance of the distribution of accumulated downtime, thus allowing for a more accurate control of the SLA violation risk. The presented model is evaluated in the context of a real backbone network in [37].

An interesting approach to the estimation of risk in the context of computer and communication networks is based on the use of widely accepted business risk measures, such as Value-at-Risk (*VaR*). In [17, 81], the authors discuss the applicability of *VaR* to telecommunication networks, and they propose a method that allows to determine the upper bound of the total penalty which must be paid

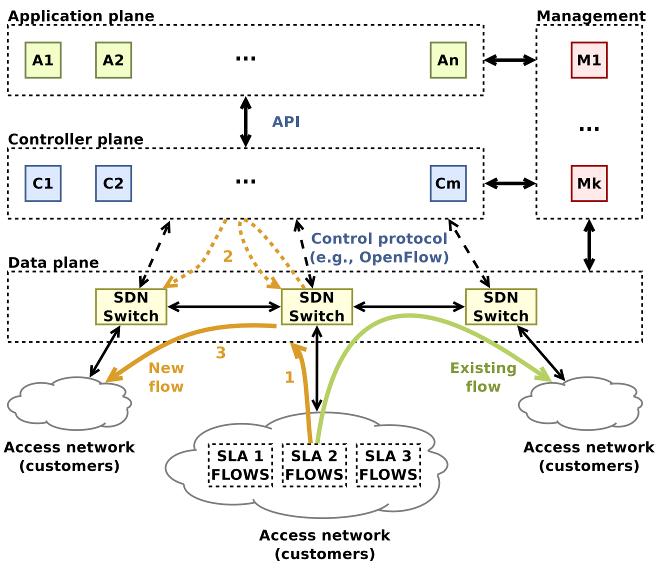


Fig. 5.1: An overview of a Software-Defined Network with different types of traffic flows.

by a service provider in a given period in the case of violation of the terms of one or more SLAs.

From the service provider's perspective, meeting the requirements of all SLAs may not always be the optimal choice. In fact, there may be a trade-off between the cost of constant improvement of the quality of services, and the probability of violation of one or more SLAs. Both aspects are discussed in detail in [30].

Risk analysis is usually based on information received from network monitoring systems. Characterization of failures in real networks was presented in [34, 69].

5.2 SDN Architecture and SLAs

The concept of SDN relies on the assumption that the control plane is decoupled from the data plane. This is a significant design approach that not only simplifies the management of network devices, but it also brings in new challenges in terms of the overall dependability of the network. For example, the survivability of the control plane has been discussed in [44] in the context of the Generalized Multiprotocol Label Switching (GMPLS) or Automatically Switched Optical Networks (ASON) model. To make a clear connection between the considered type of network and the proposed solution described in Section 5.3, the general architecture of SDN is presented in Figure 5.1.

As presented in the figure, customers are located in the access networks. The customers sign SLAs with their service providers for specific contract periods. The contract period formally defines the agreed time for which the SLA is legally binding [30]. Every customer may sign one or more SLAs with a provider, depending on the type and importance of the services it would like to use (e.g., delay-sensitive voice communication, financial transactions, or best effort web traffic), as well as the respective quality and dependability-related requirements, i.e., the SLOs. The service provider is expected to provide the service in such a way that the corresponding SLOs are satisfied. In the opposite case, the provider has to compensate the customer according to the agreed scheme. Typical business relationships that fit this model, and which are considered in this chapter, are established between service providers and either individual or corporate customers.

Customers send traffic flows between pairs of network nodes. At any point in time, the paths for some flows may already have been established in the network, while the other flows might have just arrived at the first SDN switch on their paths and have to be configured by the logically-centralized SDN controller. Note that in SDNs, some flows which have been inactive for a specified time may be marked as expired.

In the considered SDN network model, switches mainly forward customers' traffic. At the same time, it is important to note that control traffic can also be transmitted using the same infrastructure (In-Band control)[1], instead of using dedicated links (Out-of-Band control). In such a case, the reliability of operation of the control plane within an SDN depends on the operation of the same data plane that is to be controlled, which significantly extends the consequences of failures in the data plane and is undesirable with respect to the overall dependability of the system. Switches are connected to one or more logically-centralized controllers and communicate with them using a control protocol, such as OpenFlow [70]. The controllers provide an open interface for custom applications, for example, traffic engineering or network measurement applications. All components in the data plane, the controller plane, and the application plane are managed by network engineers, possibly with the aid of automated tools (note the presence of the respective connections with the management plane in Figure 5.1).

Currently, it is not clear how to construct dependability-related SLOs in the case of SDN networks. One of the main differences between the existing computer and communication networks and SDNs is that for each new flow in an SDN, a working connection from each switch that will belong to the configured flow's

[1]Actually, this is a likely scenario, especially in the case when control messages have to be transmitted over long distances, which means that reserving some links or transmission channels exclusively for control traffic would be much more expensive than sharing the resources with customers' traffic. On the other hand, whenever the reliability of such communication channels is critical to network operation, it might be reasonable to use dedicated channels on short-distance links at the cost of increased capital and operational expenditures.

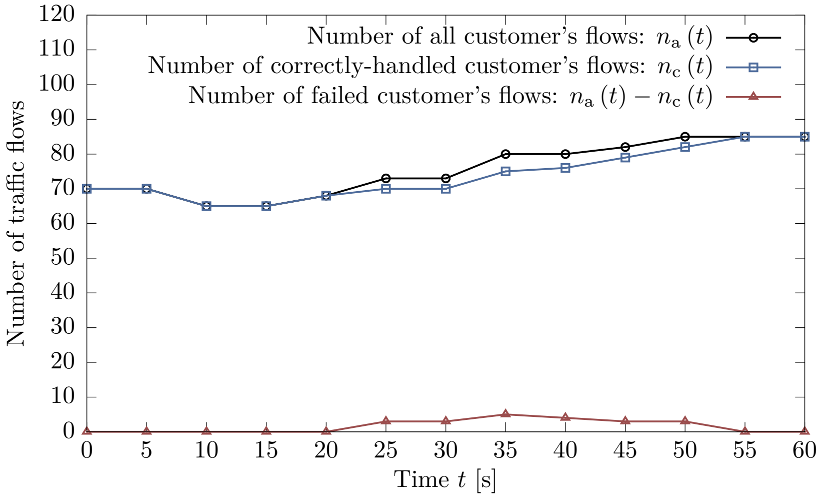


Fig. 5.2: An example showing the number of all traffic flows of a single customer at time t, the number of correctly-handled flows of the customer at time t, and the number of failed flows of that customer at time t.

path to at least one controller is needed. An immediate conclusion is that if no controller is reachable from a switch when a new flow arrives, the customer observes (partial) service downtime, even while the previously-established traffic flows are still transmitted successfully (service uptime). Unlike the existing proposals, the concept presented in Section 5.3 provides a solution to this issue.

5.3 Assessment of the SLA Violation Risk in SDN

In this section, a method is proposed that enables ISPs to estimate the risk of violating an SLA as a result of exceeding the maximum agreed service degradation. The term *service degradation* is an important part of the presented concept and it is defined as the fraction of the total number of traffic flows of a customer that were not successfully delivered to the intended destinations during the observation period. Note that by *observation period*, a predefined time interval is meant over which the service degradation metric is computed. In particular, several observation periods may exist within the SLA contract period [30].

To better illustrate the proposed idea, let us consider the example scenario shown in Figure 5.2. The figure captures an arbitrarily-selected period of 60 seconds during which a single customer sends several traffic flows through an SDN backbone network. The black curve with circular points, $n_a(t)$, reflects the number of flows of that customer at time t. Among these flows are both the new (i.e., unregistered or expired) and the previously-configured flows. The number of correctly-transmitted flows is represented by $n_{\mathrm{c}}(t)$ and the blue curve with square points. Once a failure occurs in the network, some flows may not be delivered to the intended destinations, which is reflected by the red curve with triangular points, $n_{\mathrm{a}}(t) - n_{\mathrm{c}}(t)$. The area below this curve divided by the length of the observation period provides information about the average number of the customer's flows that were not delivered to the intended destinations in the period $[0, t]$. Once this value is computed, the service degradation can be determined for the entire observation period, and then the new estimation of the SLA violation risk with respect to the dependability-related SLOs for traffic flows can be obtained.

While working on the solution, the following general assumptions have been made:

- service degradation must be measurable by both service providers and customers;
- failures affecting the already established flows have the same impact on the estimated service degradation as failures related to new flows (i.e., the service degradation is estimated based on an assumption that all flows are equally important for the customer[2]);
- there is enough network resources to respond to failures through flow rerouting; at the same time, the recovery does not have to be successful.

In addition, the volume of traffic flows is not considered as an indicator of their importance. Note that one flow of very small volume can be far more important than several high-volume flows.

The symbols used in the formulation are presented in Table 5.1. To simplify the explanation (without losing generality), it is assumed that at each time t, new traffic flows receive unique indices in the range of 1 to $n_{\mathrm{n}}(t)$, while the existing flows are assigned higher indices between $n_{\mathrm{n}}(t) + 1$ and $n_{\mathrm{a}}(t)$. In addition, it is assumed that the customer has signed a single SLA with the service provider[3]. In the first step, the customer's flows are counted that can be delivered to the intended destinations at time t, which is reflected by the $n_{\mathrm{c}}(t)$ function as follows:

$$n_{\mathrm{c}}(t) = \sum_{i=1}^{n_{\mathrm{n}}(t)} p(t,i)\, c(t,i) + \sum_{i=n_{\mathrm{n}}(t)+1}^{n_{\mathrm{a}}(t)} p(t,i) \tag{5.1}$$

Based on Equation (5.1), the overall *service degradation* $D(\tau)$ corresponding

[2]Note that the proposed solution may be extended to support different prioritization schemes.

[3]The analysis would follow the same steps individually for each SLA signed with the given customer. Such an approach provides greater flexibility with respect to groups of flows that have different requirements.

Table 5.1: Symbols used in the formulation of the presented risk assessment method.

Symbol	**Description**
τ	The length of the observation period defined in the related Service Level Agreement (SLA); $\tau \in \mathbb{R}^+$
α	The maximum allowed service degradation defined in the SLA; $\alpha \in [0;1]$
$n_\mathrm{a}(t)$	Number of all traffic flows of the selected customer at time t; $\forall_{t\in\mathbb{R}}\, n_\mathrm{a}(t) \in \mathbb{N}$
$n_\mathrm{n}(t)$	Number of new traffic flows of the customer at time t; $\forall_{t\in\mathbb{R}}\, n_\mathrm{n}(t) \in \mathbb{N}$
$n_\mathrm{c}(t)$	Number of successfully-delivered traffic flows of the customer at time t; $\forall_{t\in\mathbb{R}}\, n_\mathrm{c}(t) \in \mathbb{N}$
$c(t,i)$	The availability status of all connections to the logically-centralized SDN controller along the entire path of the i-th flow at time t; $c(t,i) = 1$ if and only if all SDN switches belonging to the path can communicate with at least one active SDN controller, otherwise $c(t,i) = 0$
$p(t,i)$	The availability status of the entire path of the i-th flow at time t; $p(t,i) = 1$ if and only if all network devices (nodes, links, optical amplifiers, ...) belonging to the path are working properly, otherwise $p(t,i) = 0$
$D(\tau)$	Service degradation in the observation period $[0,\tau]$
$S(\tau,\alpha)$	SLA success probability; $S(\tau,\alpha) \in [0,1]$
$W(\tau,\alpha)$	SLA violation risk; $W(\tau,\alpha) \in [0,1]$

to the signed SLA and the related observation period τ is computed as follows:

$$D(\tau) = \begin{cases} 1 - \frac{\int_0^\tau n_\mathrm{c}(t)\,\mathrm{d}t}{\int_0^\tau n_\mathrm{a}(t)\,\mathrm{d}t} & \text{if } \exists_{t\in[0;\tau]}\, n_\mathrm{a}(t) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Note that $\int_0^\tau n_\mathrm{c}(t)\,\mathrm{d}t$ is the number of flows of the selected customer that were delivered to the intended destinations during the observation period τ, while

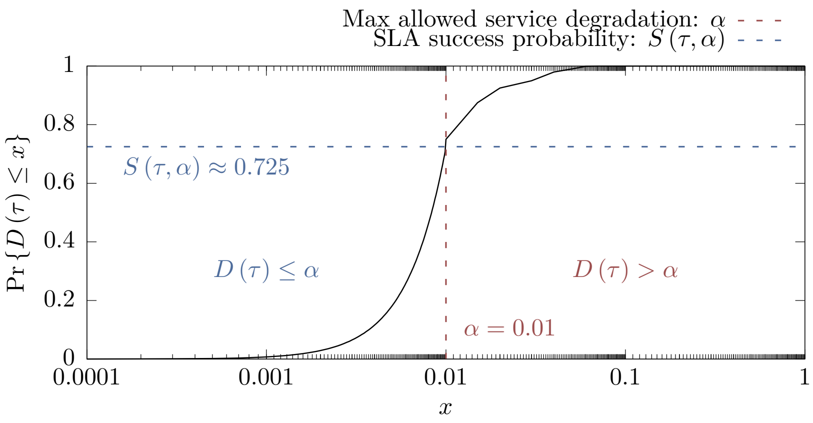


Fig. 5.3: An example Cumulative Distribution Function (CDF) of the accumulated service degradation $D(\tau)$. The corresponding maximum allowed service degradation α was set to 0.01.

$\int_0^\tau n_\mathrm{a}(t)\,\mathrm{d}t$ is the number of all flows offered by the customer during the same period.

Once the service degradation $D(\tau)$ is computed for each of the consecutive observation periods, it is possible to determine the related Cumulative Distribution Function (CDF). The SLA success probability $S(\tau,\alpha)$ and the SLA violation risk $W(\tau,\alpha)$ with respect to the agreed service degradation threshold α and length of an observation period τ are then:

$$S(\tau,\alpha) = \Pr\{D(\tau) \leq \alpha\} \tag{5.3}$$

$$W(\tau,\alpha) = 1 - S(\tau,\alpha) \tag{5.4}$$

See Figure 5.3 for an illustration. Every SLA, for which the risk of violating the service degradation requirement is estimated using the proposed method, should be extended with at least the α and τ parameters agreed on with the customer.

Based on the recent work applicable to typical computer and communication networks [37], it is possible to make risk-aware decisions on the preferred use of different dependability provisioning techniques in the case of SDNs. Although the implementation of specific solutions in real networks is usually complex, the possible directions provided in this chapter may be perceived as the first step, while the simulation study is used to illustrate the potential use cases.

5.3.1 Evaluation Environment

The evaluation of the proposed solution was carried out with the aid of a discrete-event, flow-level network simulator written in the C++ programming language[4]. For the purpose of generation of random data (e.g., flow inter-arrival time, flow duration and demand, source and destination nodes) needed to prepare the input data sets for each simulation, the random number generation engine of the ns-3 v3.24.1 network simulator [3] was used[5]. The main objective of the simulation study was to confirm the capability of the proposed method to estimate the SLA violation risk with respect to the dependability SLOs in Software-Defined Networks in different scenarios. It is shown that based on the collected results, suggestions can be made about the additional protection measures that should be deployed in the network to reduce the risk of violation of the dependability-related SLOs of each individual SLA to an acceptable level.

To illustrate the possible use cases for the proposed solution, the following two evaluation scenarios were considered:

- Scenario I: homogeneous service degradation threshold α across all SLAs (*standard* SLAs);
- Scenario II: differentiated service degradation threshold (*standard* SLAs: α_{s}, *business* SLAs: α_{b}).

Standard SLAs were only signed by customers who were sending traffic between nodes selected at random according to the uniform distribution. However, to reflect the fact that companies often have remote premises in different cities, business SLAs have been introduced in Scenario II. Business SLAs assumed that traffic flows are transmitted within the predefined *business groups*, with higher SLA dependability requirements than traffic associated with standard SLAs. Each *business group* was defined as a set of 4 different nodes selected at random and representing the corresponding company's premises in different cities. In this case, the source and destination nodes were selected at random from the same business group according to the uniform distribution. Further, it was assumed that there are 100 SLAs per backbone node in the network. In the case of Scenario II, each backbone node represented the home location for 70 standard SLAs and 30 business SLAs signed with local customers.

The evaluation was based on the modified version of the real-world backbone network topology shown in Figure 5.4. The considered network included two logically-centralized SDN controllers (C1 and C2). Each link of the network had the capacity of 1 Gbit/s. The flow inter-arrival time was selected according to the exponential distribution with the mean value of 0.1 s, while the duration of each flow followed the Pareto distribution with the mean value of 60 s and the shape

[4]For a detailed description of the simulator, the reader is referred to Chapter 6.
[5]See the related discussion in Section 4.2.3.

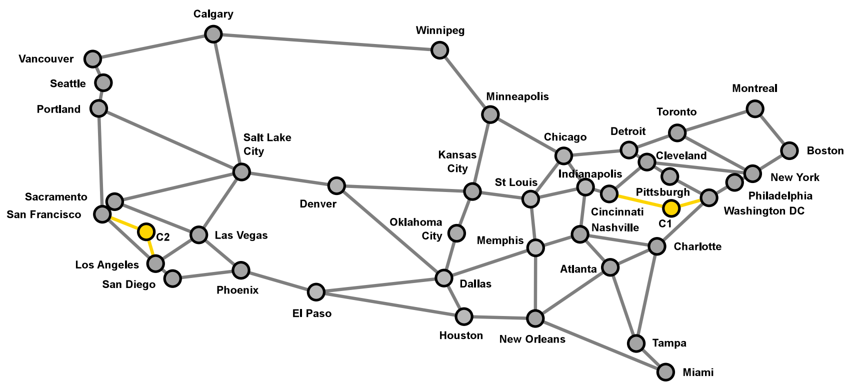


Fig. 5.4: A modified US backbone network topology containing 39 nodes, two SDN controllers (yellow nodes: C1 and C2), and 130 unidirectional links. The topology of the original network was created based on the data delivered by the SNDlib project [74] (name of the model: *janos-us-ca*). To maintain clarity, the figure presents an undirected graph.

parameter equal to 1.5. The average flow demand was set to 1 Mbit/s, whereas the actual values were selected at random from range $[0.75; 1.25]$ Mbit/s according to the uniform distribution. The numerical values were selected in such a way that the generated traffic did not cause link congestions during fault-free network operation. Moreover, it was assumed that the service provider maintains sufficient capacity on links to deal with failures in the network, so that at least a small fraction of each affected flow could still be transmitted, if only the respective reachability requirements were satisfied. During the steady-state period of each simulation run, the total number of traffic flows in the network oscillated around an average value. Further, it was assumed that every 10000 s on average, a node in the network would fail. The time between consecutive failures was selected at random according to the exponential distribution. In the case of links, failures occurred every 1000 s on average. The Mean Time To Repair (MTTR) for nodes and links was set to 1000 s and 100 s, respectively. The actual values were selected at random according to the Pareto distribution with the shape parameter equal to 1.5.

In each simulation run, the service degradation metric was computed every month ($\tau = 1$ month), while the risk of violation of the dependability-related SLOs for each SLA was computed after 5 consecutive months, based on Equation (5.4). For error control, 10 independent simulation runs were executed. Each simulation run consisted of the following phases:

- the transient period of 512 s (estimated using the method described in Appendix A) — to make sure that samples were collected during the steady-state period of the simulation;

– 5 consecutive observation periods, each of length τ;
– the termination phase of at least 40 s.

In each of the considered scenarios, different values of the service degradation threshold α were considered. All traffic flows were forwarded along the shortest paths and it was assumed that the only available resilience provisioning mechanism was flow rerouting.

In real computer and communication networks, rerouting traffic flows may take different amounts of time — usually from tens of milliseconds to several seconds. The duration of this process depends on several factors, such as: the selected rerouting strategy (e.g., convergence of a specific routing protocol, IP Fast Reroute-based schemes), layer of operation, network technology, network topology, and available resources. As the proposed solution is not related to a specific routing protocol or recovery mechanism, it was assumed that rerouting a flow after node or link failure imposes the related service downtime of 1 s.

5.3.2 Evaluation Results

The evaluation results corresponding to Scenario I are shown in Figures 5.5 and 5.6. The first figure represents an example simulation run and shows the Cumulative Distribution Function (CDF) of the SLA violation risk with respect to the service degradation requirement α (further referred to as *SLA violation risk*; see Equation (5.4)). It may be noticed that lower values of α resulted in higher SLA violation risk. Thus, if service providers decide to guarantee low service degradation in their agreements with customers, they should already be prepared to activate enough redundant network resources immediately when such demand occurs. The second figure presents the estimated CDFs of the maximum SLA violation risk and the arithmetic mean of the SLA violation risk computed for all SLAs in each simulation run. Note that the curves in Figure 5.6(a) overlap and have a steep slope for x equal to 1 for all considered values of α, which reflects the fact that the maximum SLA violation risk was always equal to 1. Furthermore, it is clearly visible that the maximum computed SLA violation risk among all SLAs may deviate significantly from the arithmetic mean of the risk values determined for the same set of SLAs. Thus, it is a critical factor that should be monitored.

The impact of the maximum computed SLA violation risk on the decision making process may even be stronger when some SLAs assume different service degradation thresholds than the other SLAs, which was the main focus of Scenario II. In this case, three different combinations of service degradation thresholds for standard and business SLAs were considered. As business communication is usually associated with higher dependability guarantees, it was assumed that the corresponding service degradation threshold α_b should always be lower than the corresponding threshold for standard SLAs, α_s. The simulation results have

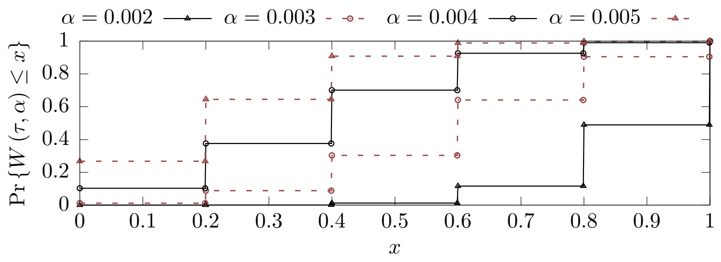


Fig. 5.5: Scenario I: An example CDF of the SLA violation risk with respect to the service degradation requirement α. The results represent an example simulation run and all 3900 standard SLAs (100 per each network node).

shown that in the considered evaluation scenario, the maximum SLA violation risk and the arithmetic mean of the SLA violation risk computed with respect to the exceeded service degradation threshold were much higher for business SLAs, as they had stronger dependability guarantees than standard SLAs (see Figures 5.7 and 5.8; note that there are two overlapping curves in Figure 5.8(a)). Thus, to avoid penalties due to violated dependability-related SLOs of business SLAs, the service provider should either deploy more effective recovery mechanisms for the related traffic flows, or negotiate higher service degradation thresholds with its business customers. However, different recovery mechanisms may have different cost. An idea of how to select the optimal recovery strategy based on the related cost and the estimated SLA violation risk has been presented in [37]. Now, being able to estimate the risk of violation of the dependability-related SLOs for each SLA in SDNs, service providers may plan future expenditures more effectively.

5.3.3 Deployment Considerations and Limitations

Deployment of the proposed solution requires a fully operational SDN network. In addition, due to the way service degradation is estimated, service providers must be able to register both successful and unsuccessful flow configuration and transmission attempts for each of their customers. Thus, in the case of failure of all communication channels between an SDN switch and the logically-centralized controller, the monitoring systems must still be able to record incoming traffic flows reliably. Such information should then be transmitted to the centralized unit which is responsible for the periodic estimation of the service degradation metric. The related algorithm may be implemented within an independent application which communicates with the logically-centralized controller to get additional information (see Figure 5.1). It is desired that the application be able to trigger alarms or predefined actions whenever the estimated SLA violation risk with

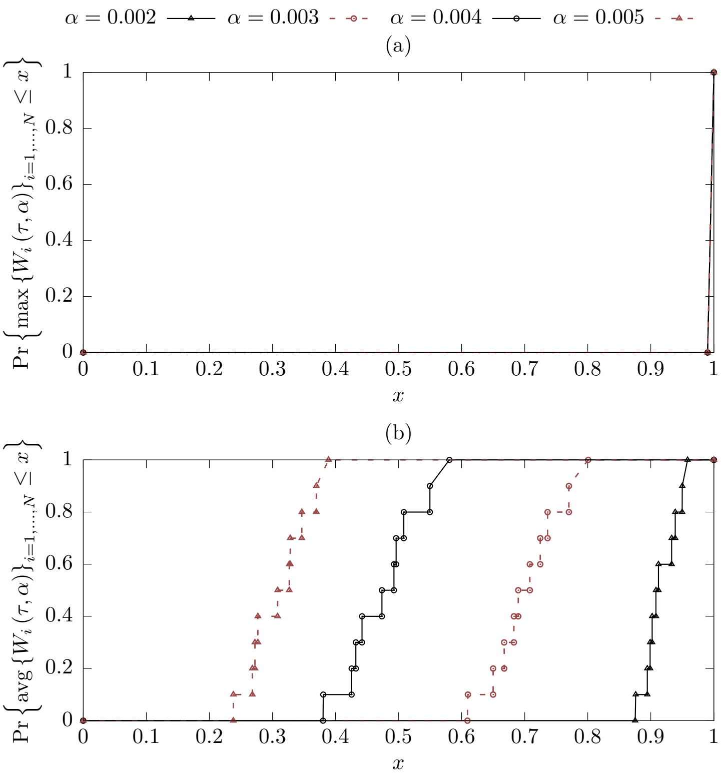


Fig. 5.6: Scenario I: The estimated CDF of (a) the maximum SLA violation risk and (b) the arithmetic mean of the SLA violation risk with respect to the service degradation requirement α. The results represent all simulation runs ($N = 10$) and all 3900 standard SLAs (100 per each network node).

respect to the exceeded service degradation threshold is higher than the maximum allowed value. To avoid possible alarm storms, a mechanism based on different triggering conditions may be applied.

The proposed solution has some limitations. In particular, it assumes that each customer transmits several traffic flows through the network during each observation period. While this condition should generally be satisfied in backbone networks, it may not be the case in some specific scenarios. Further, it is assumed that if a customer is disconnected from the provider's infrastructure (e.g., following a link failure), then both sides will be able to measure the related downtime and decide if a financial compensation is needed according to the SLA. The proposed solution is not directly applicable to such scenarios, as the customer will not

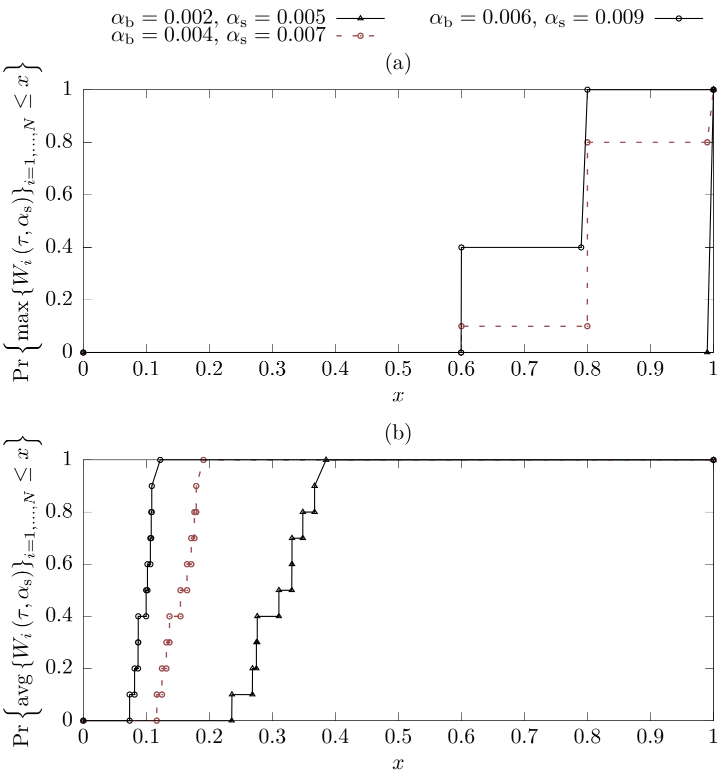


Fig. 5.7: Scenario II: The estimated CDF of (a) the maximum SLA violation risk and (b) the arithmetic mean of the SLA violation risk with respect to the service degradation requirement α_s. The results represent all simulation runs ($N = 10$) and all 2730 standard SLAs (70 per each network node).

be able to send its flows through the provider's network. At the same time, service providers should be able to deal with this issue using the already-deployed monitoring tools.

5.4 Summary

In this chapter, a method was presented that allows for the assessment of the SLA violation risk with respect to the dependability-related SLOs defined for traffic flows in SDNs. To clarify the understanding of dependability in the context of traffic flows in SDNs, the main related factors were identified, and then *service degradation* was defined as the key measure of decreased dependability in SDN

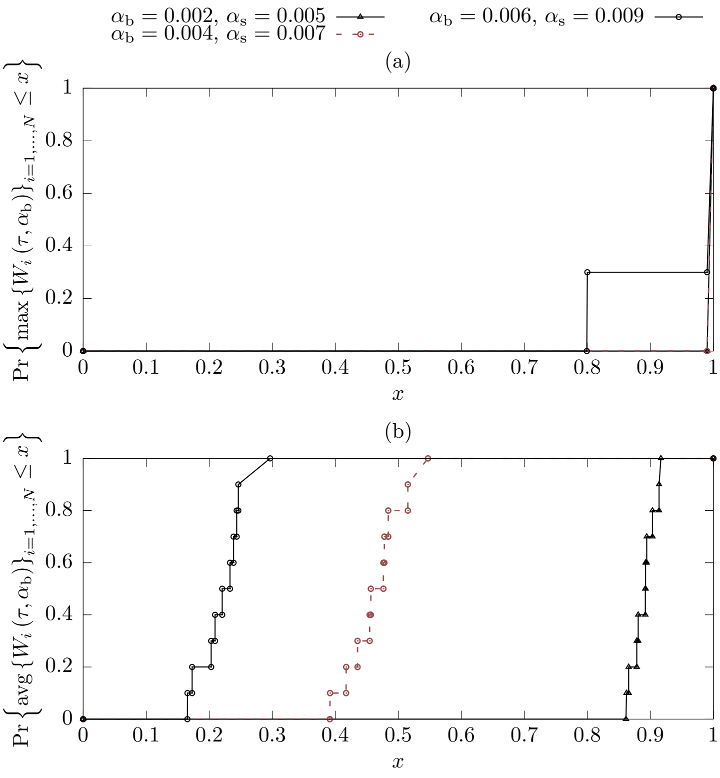


Fig. 5.8: Scenario II: The estimated CDF of (a) the maximum SLA violation risk and (b) the arithmetic mean of the SLA violation risk with respect to the service degradation requirement α_b. The results represent all simulation runs ($N = 10$) and all 1170 business SLAs (30 per each network node).

environments, allowing for the computation of the corresponding SLA violation risk. The simulation results have shown that the proposed solution is feasible and may help service providers to select the preferred recovery technique based on the estimated SLA violation risk related to the known dependability SLOs. The presented work is the first step to understand how to define and assess the SLA dependability parameters in SDNs — a problem that to date was still unsolved.

6 A Flow-Level Discrete-Event Network Simulator for Dependability Research

Some parts of the research presented in this dissertation involved simulation experiments to evaluate the proposed solutions. As the considered solutions addressed specific high-level issues in flow-oriented networks, network operation was analyzed on the level of traffic flows, whereas performance of the verified mechanisms was assessed based on the selected metrics. Thus, a set of simulation tools was designed and implemented to support the evaluation of the selected aspects of network dependability in flow-oriented computer and communication networks. The overall design of the simulator is presented in Section 6.1.

6.1 Design

The proposed simulator is a flow-level, discrete-event simulator implemented in the C++ programming language. It models traffic flows and network elements, such as nodes and links, as objects with different properties (e.g., node name, link capacity, source and destination nodes, flow demand). Its structure is very simple and modular, as presented in Figure 6.1. The simulator is based on the following three independent components:

- Data Generation Module,
- Data Management Module,
- Simulation Module.

The key functionality of the components is briefly discussed in Sections 6.1.1-6.1.3.

6.1.1 Data Generation Module

The Data Generation Module facilitates the preparation of input data related to the simulated network events, such as arrivals of new traffic flows and failures of network elements. By default, for the purpose of random data generation

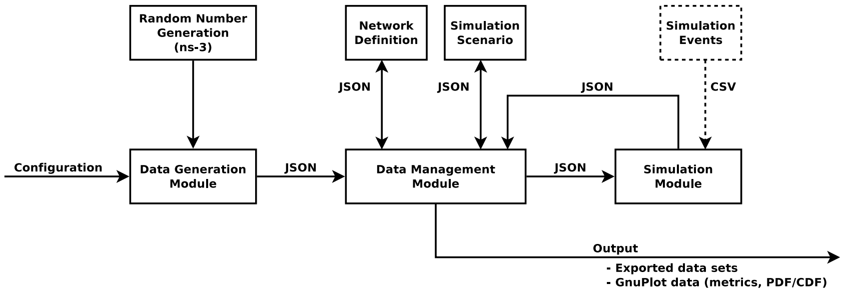


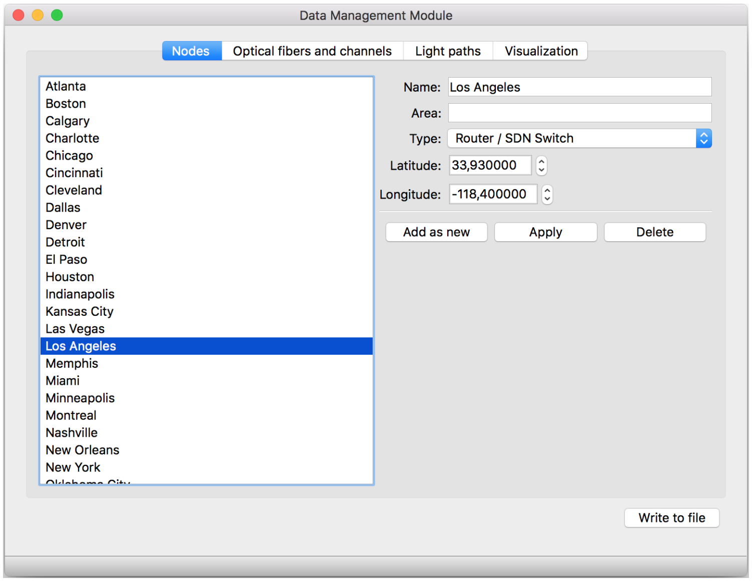
Fig. 6.1: A block diagram showing the main components and data sources of the created flow-level, discrete-event network simulator. Arrows describe the flow of information between particular blocks, input, and output. ns-3 is an existing network simulator [3] that was used solely for the purpose of random number generation according to predefined distributions.

(e.g., flow inter-arrival time, flow duration and demand, source and destination nodes for particular flows), the pseudorandom data generation engine of the widely-used ns-3 [3] network simulator is used. At the same time, it needs to be emphasized that the modular design of the simulator allows researchers to employ other external random generators for this purpose, according to their preferences or specific requirements. The random values are assigned to the selected properties and stored in a text file prepared by a C++ application in the JSON[1] data-interchange format. Multiple data sets, one for each simulation trial, are generated with the aid of a UNIX shell script. The data sets can be directly imported into the Data Management Module.

6.1.2 Data Management Module

The Data Management Module organizes all available information about network topology, simulation scenario, and simulation results. It was developed as a stand-alone C++ application with a graphical user interface based on the cross-platform Qt library [5]. The definition of a network topology or a simulation scenario can either be prepared within the application (see Figures 6.2-6.6), or it can be imported from a text file containing the previously exported data stored in the JSON data-interchange format. While the definition of the network topology includes information about nodes, physical links, light paths, the selected technical parameters, and the physical location of nodes, the simulation scenario defines some general simulation parameters, such as the relative link congestion threshold, network monitoring interval, and network events for each simulation trial with the corresponding occurrence times. Note that although the simulator is designed

[1]JavaScript Object Notation



Atlanta
Boston
Calgary
Charlotte
Chicago
Cincinnati
Cleveland
Dallas
Denver
Detroit
El Paso
Houston
Indianapolis
Kansas City
Las Vegas
Los Angeles
Memphis
Miami
Minneapolis
Montreal
Nashville
New Orleans
New York

Name:	Los Angeles
Area:	
Type:	Router / SDN Switch
Latitude:	33,930000
Longitude:	-118,400000

Fig. 6.2: Data Management Module: an example definition of network nodes.

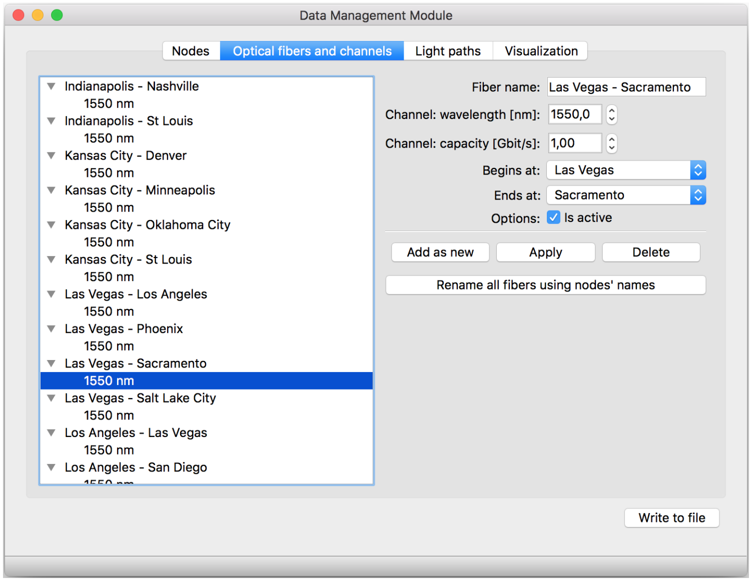
to be able to model the selected elements of optical networks, such as fiber optic links and light paths, it is also possible to model non-optical wired networks which contain generic links. The modeling of network elements is focused on their high-level properties (e.g., link capacity and endpoints), which is sufficient for the purpose of network analyses considering connectivity, reachability, and link congestions.

The post-processing of the simulation results is focused mainly on the capability to estimate the empirical Probability Density Function (PDF) or Cumulative Distribution Function (CDF) of a given performance metric, for the considered type of value (minimum, average, maximum). It is also possible to extract the results acquired for each simulation trial and performance metric without further processing. In both cases, the resulting data set can be exported to a file and visualized with the aid of the Gnuplot graphing utility [1].

6.1.3 Simulation Module

The Simulation Module is the part of the simulator that actually executes the events, maintains the network state, and recomputes the performance metrics periodically.

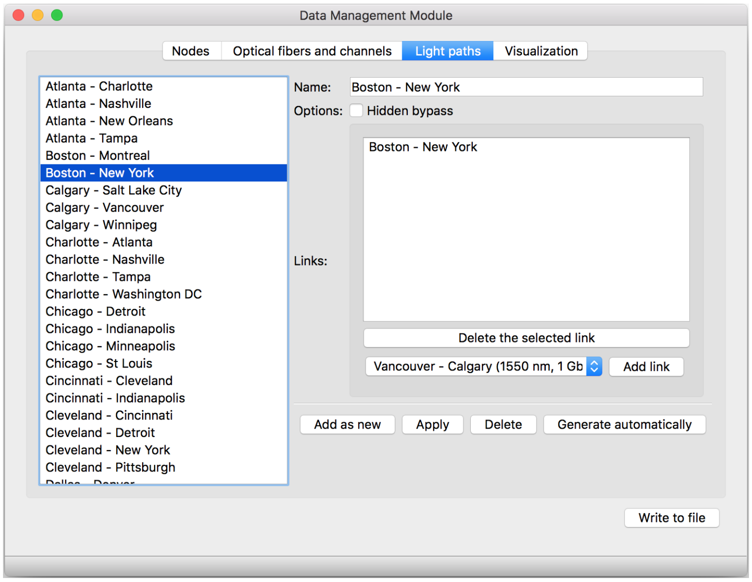
To avoid the complexity involved in the generation of random data reflecting



Indianapolis - Nashville
1550 nm
Indianapolis - St Louis
1550 nm
Kansas City - Denver
1550 nm
Kansas City - Minneapolis
1550 nm
Kansas City - Oklahoma City
1550 nm
Kansas City - St Louis
1550 nm
Las Vegas - Los Angeles
1550 nm
Las Vegas - Phoenix
1550 nm
Las Vegas - Sacramento
1550 nm
Las Vegas - Salt Lake City
1550 nm
Los Angeles - Las Vegas
1550 nm
Los Angeles - San Diego

Fiber name:	Las Vegas - Sacramento
Channel: wavelength [nm]:	1550,0
Channel: capacity [Gbit/s]:	1,00
Begins at:	Las Vegas
Ends at:	Sacramento
Options:	✓ Is active

Fig. 6.3: Data Management Module: an example definition of fiber links and optical channels.



Atlanta - Charlotte
Atlanta - Nashville
Atlanta - New Orleans
Atlanta - Tampa
Boston - Montreal
Boston - New York
Calgary - Salt Lake City
Calgary - Vancouver
Calgary - Winnipeg
Charlotte - Atlanta
Charlotte - Nashville
Charlotte - Tampa
Charlotte - Washington DC
Chicago - Detroit
Chicago - Indianapolis
Chicago - Minneapolis
Chicago - St Louis
Cincinnati - Cleveland
Cincinnati - Indianapolis
Cleveland - Cincinnati
Cleveland - Detroit
Cleveland - New York
Cleveland - Pittsburgh

Name:	Boston - New York
Options:	Hidden bypass
Links:	Boston - New York

Vancouver - Calgary (1550 nm, 1 Gb

Fig. 6.4: Data Management Module: an example definition of light paths.

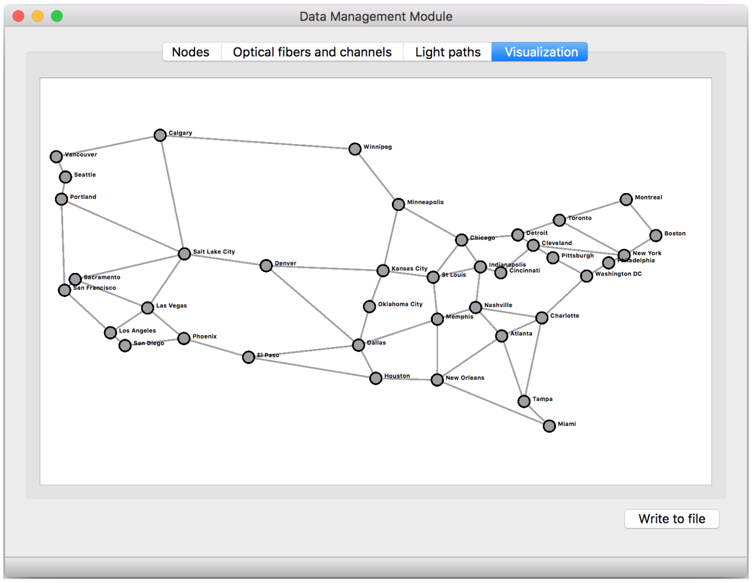
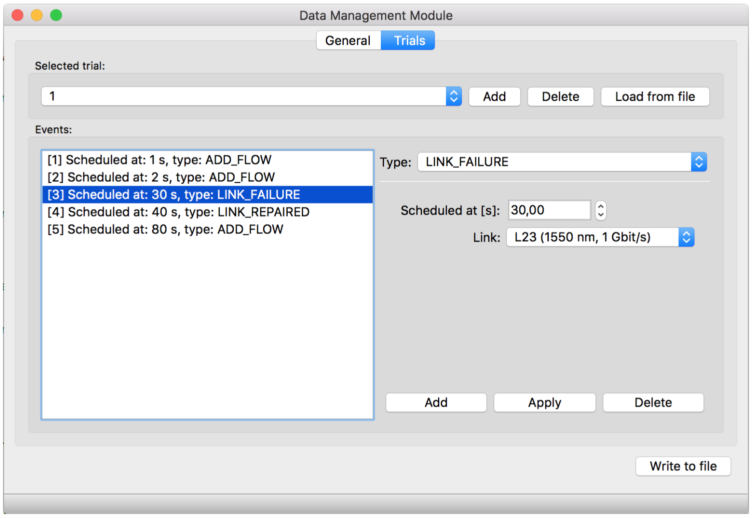


Fig. 6.5: Data Management Module: an example visualization of a network topology.



Selected trial: 1

Events:

[1] Scheduled at: 1 s, type: ADD_FLOW
[2] Scheduled at: 2 s, type: ADD_FLOW
[3] Scheduled at: 30 s, type: LINK_FAILURE
[4] Scheduled at: 40 s, type: LINK_REPAIRED
[5] Scheduled at: 80 s, type: ADD_FLOW

Type: LINK_FAILURE

Scheduled at [s]: 30,00

Link: L23 (1550 nm, 1 Gbit/s)

Fig. 6.6: Data Management Module: an example definition of network events.

the desired distribution, as well as to allow researchers to use different external sources of randomness, the Simulation Module of the simulator by design expects the user to provide a complete simulation scenario on input.

When started, the simulator enters the main loop in which it loads the list of events for the currently considered simulation trial, and then executes the events in the appropriate order, collecting the required data (such as the number of overloaded links) and storing it in the output file. The type of the employed event list is the Henriksen's event list, as described in [85]. Alternatively, in the case of very long simulations with large number of events, a modified version of the simulator has been developed that allows to execute blocks of events loaded sequentially from a predefined text file, thus avoiding the necessity of loading all the events into memory before starting the simulation. For each simulation trial, the simulator saves the results in the predefined output file. The Simulation Module is designed to be portable and can be managed via the command-line interface.

6.2 Summary

In this chapter, the design and the selected implementation details of the proposed simulation tools were presented. The simulation tools have been developed as an integral part of the conducted research on the dependability of flow-oriented computer and communication networks. Using the proposed simulator, it is possible to analyze network operation on the level of traffic flows, avoiding the complexity of different low-level phenomena, which allows to asses the network performance with respect to connectivity, reachability, presence of link congestions, and different custom metrics.

7 Conclusion and Future Work

Considering the increasing reliance of the global society on the communication infrastructure, it is required that modern computer and communication networks be able to deal with forwarding loops, multiple simultaneous failures of network elements, and link congestions effectively. Further, to enable service providers to plan the expenditures related to the development and maintainance of their networks, while meeting the agreed goals with respect to the Service Level Agreements signed with customers, risk estimation-based methods are often used. In this dissertation, the related solutions are proposed for specific types of flow-oriented communication networks. The main contributions are outlined in Section 7.1, while the open issues and future work are discussed in Section 7.2. Finally, Section 7.3 concludes the dissertation.

7.1 Contributions

In this dissertation, four different solutions have been proposed to enhance the dependability of the selected flow-oriented network types with respect to possible occurrences of persistent forwarding loops, link congestions (two solutions), and multiple simultaneous failures of network elements. To support the evaluation of the congestion control algorithms, a set of discrete-event flow-level network simulation tools have been designed and implemented as an integral part of the research presented in this dissertation. The proposed solutions dealing with forwarding loops and failures of network elements have been implemented in the custom-built prototype routing devices and they were evaluated in two laboratory networks including off-the-shelf network equipment. The experimental evaluation demonstrated the potential of the presented methods to be gradually deployed in existing computer and communication networks. To extend the evaluation and provide a valuable context for the results, the proposed methods were compared against two reference network operation modes which have also been implemented in the prototype router devices. The evaluation results have shown that the impact of failures, forwarding loops, and link congestions on traffic flows in the selected types of flow-oriented network may be reduced with the aid of the proposed solutions, improving the overall network dependability perceived by users.

Further, in the case of Software-Defined Networks, an explicit definition of the dependability requirements for traffic flows was provided, the corresponding measure of decreased dependability was introduced, and a complete risk assessment scheme was proposed to enable service providers to estimate the risk of violation of SLAs with respect to the proposed metric. The corresponding evaluation was performed with the aid of the discrete-event flow-level network simulation tools designed and implemented by the author. Based on the evaluation results, it was observed that the presented service degradation measure and the risk assessment scheme for SDNs have the potential to enable service providers to select the desired recovery mechanisms more effectively with respect to the related expenditures and the estimated risk of violation of the dependability requirements of SLAs signed with customers.

Based on the research conducted for this dissertation, it can be stated that the following thesis defined in Section 1.3 has been proved:

> **It is possible to improve the dependability of the selected flow-oriented network types using the proposed solutions to deal with failures, forwarding loops, and link congestions, and to estimate the risk of violation of the dependability SLOs related to traffic flows in SDNs with the aid of the proposed risk assessment scheme.**

7.2 Open Issues and Future Work

Although the proposed solutions have the potential to improve the dependability of the selected types of flow-oriented network, they also have some limitations which are discussed in detail in the corresponding sections. In particular, one open issue related to the GroupAndReroute algorithm is that if generic communication devices that do not support GroupAndReroute are also present in the network, GroupAndReroute may cause forwarding loops. However, a strategy is also proposed to avoid such events. Further, the proposed methods may be extended to support various service differentiation strategies. This area has not been discussed in this dissertation, yet it is an interesting problem for future research.

7.3 Final Remarks

The solutions presented in this dissertation have been designed to address important issues influencing the dependability of the selected types of flow-oriented computer and communication network. The corresponding limitations are discussed in detail in the respective sections. The solutions have the potential to be deployed in real networks.

Appendices

A Estimating the Length of the Transient Period of a Simulation

Simulation experiments are conducted to study the behavior of a system under specific conditions. If the evaluation is based on the assumption that the system is observed during the steady-state operation period, it is necessary to either verify whether the observations during the transient period actually reflect the steady-state distribution, or to estimate the length of the transient period occurring at the beginning of each simulation run and ignore all observations during this period [85]. One of the possible strategies to estimate the length of the transient period is to observe the current estimate and wait until it starts to oscillate around an average value. Although not perfect, this method was used to get the first approximation that could be verified using the following strategy based on the central limit theorem and described in [85]. The strategy assumes that in the case of a stationary distribution, the sample mean of the distribution follows a normal distribution characterized by mean value μ and standard deviation s:

$$s = \frac{\sigma}{\sqrt{n}} \tag{A.1}$$

where n represents the number of samples, while μ and σ denote the true mean and standard deviation of the sampled distribution, respectively. Based on Formula (A.1), the following linear relationship can be obtained:

$$\log_2 s = -0.5 \log_2 n + \log_2 \sigma \tag{A.2}$$

As Formula (A.2) represents $\log_2 s$ as a function of $\log_2 n$, the corresponding plot may be analyzed with respect to the point at which the slope of the curve clearly changes sign to negative and remains close to -0.5. Such a point can be selected as the beginning of the equilibrium.

Further, when consecutive observations are made during a simulation run, the corresponding samples may correlate with each other. Thus, it is recommended

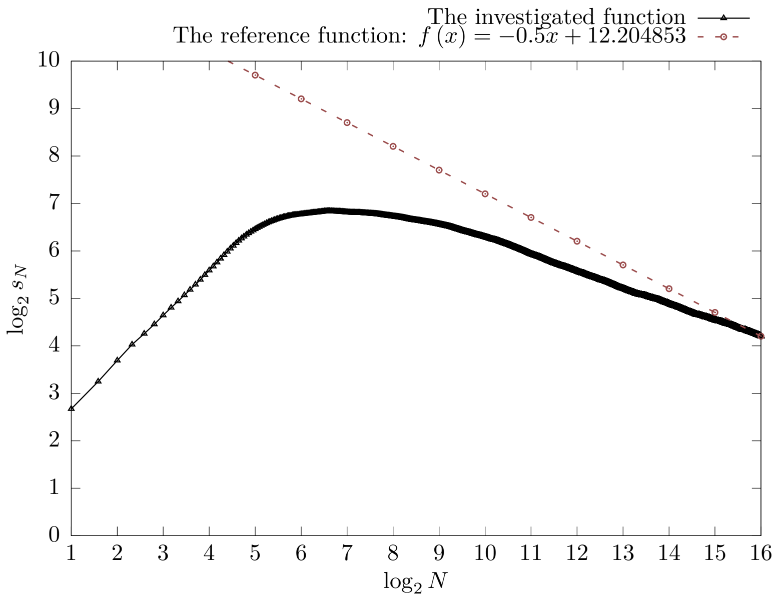


Fig. A.1: An example plot showing $\log_2 s_N$ as a function of $\log_2 N$, as well as a reference function with the slope of -0.5. The plot corresponds to the experiments discussed in Chapter 5 (Scenario I).

that several independent simulation runs be made [85]. Assuming that M denotes the number of simulation runs, each with N observations, the corresponding samples may be represented as X_{nm}, where $n = 1, \ldots, N$ and $m = 1, \ldots, M$. In addition, for each observation n, the mean of the samples across all M simulation runs is as follows:

$$\hat{\mu}_n = \frac{1}{M} \sum_{m=1}^{M} X_{nm} \tag{A.3}$$

The sequence of $\hat{\mu}_n$ values determined based on Formula (A.3) contains N independent normally-distributed random variables [85] and it is possible to compute the corresponding standard deviation s_N as follows:

$$s_N = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} \left(\hat{\mu}_n - \omega_N\right)^2} \tag{A.4}$$

where:

$$\omega_N = \frac{1}{N} \sum_{n=1}^{N} \hat{\mu}_n$$

Finally, based on Equation (A.4), it is possible to create a plot showing $\log_2 s_N$ as a function of $\log_2 N$. An example plot corresponding to the experiments discussed in Chapter 5 is presented in Figure A.1. Note that although the overall slope of the investigated function for arguments greater than or equal 9 does not match exactly the expected value of -0.5, it is relatively close to that value, which agrees with a similar example provided in [85]. In addition, the overall slope has clearly changed sign to negative and it remains stable with increasing value of $\log_2 N$ in the considered range. In this case, it was assumed that the equilibrium started at the point corresponding to $\log_2 N = 9$.

Bibliography

[1] The Gnuplot Project. URL `http://www.gnuplot.info`. Accessed on May 9, 2017.

[2] The iftop monitoring tool. URL `http://www.ex-parrot.com/~pdw/iftop`. Accessed on May 9, 2017.

[3] The ns-3 Discrete-Event Network Simulator, . URL `https://www.nsnam.org`. Accessed on May 9, 2017.

[4] The ns-3.24 Manual, . URL `https://www.nsnam.org/docs/release/3.24/manual/singlehtml/index.html`. Downloaded on May 9, 2017.

[5] The Qt Project. URL `https://www.qt.io`. Accessed on May 9, 2017.

[6] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme. IPv6 Flow Label Specification. RFC 6437, RFC Editor, November 2011. URL `http://www.rfc-editor.org/rfc/rfc6437.txt`. Downloaded on May 9, 2017.

[7] S. Antonakopoulos, Y. Bejerano, and P. Koppol. Full Protection Made Easy: The DisPath IP Fast Reroute Scheme. *IEEE/ACM Transactions on Networking*, 23(4):1229–1242, Aug 2015. ISSN 1063-6692. doi: 10.1109/TNET.2014.2369855.

[8] A. Atlas. U-turn Alternates for IP/LDP Fast-Reroute. Internet-Draft draft-atlas-ip-local-protect-uturn-03.txt, IETF Secretariat, February 2006. URL `https://tools.ietf.org/html/draft-atlas-ip-local-protect-uturn-03`. Downloaded on May 9, 2017.

[9] A. Atlas and A. Zinin. Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286, RFC Editor, September 2008. URL `http://www.rfc-editor.org/rfc/rfc5286.txt`. Downloaded on May 9, 2017.

[10] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1:11–33, 2004.

[11] F. Baker and G. Fairhurst. IETF Recommendations Regarding Active Queue Management. RFC 7567, RFC Editor, July 2015. URL `http://www.rfc-editor.org/info/rfc7567`. Downloaded on May 9, 2017.

[12] W. Braun and M. Menth. Scalable Resilience for Software-Defined Networking using Loop-Free Alternates with Loop Detection. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, April 2015. doi: 10.1109/NETSOFT.2015.7116180.

[13] G. Brush and N. Marlow. Assuring the Dependability of Telecommunications Networks and Services. *IEEE Network*, 4(1):29–34, Jan 1990. ISSN 0890-8044. doi: 10.1109/65.47002.

[14] S. Bryant, C. Filsfils, S. Previdi, and M. Shand. IP Fast Reroute using tunnels. Internet-Draft, IETF, November 2007. URL `https://tools.ietf.org/pdf/draft-bryant-ipfrr-tunnels-03.txt`. Downloaded on May 9, 2017.

[15] P. Cholda, A. Mykkeltveit, B.E. Helvik, O.J. Wittner, and A. Jajszczyk. A survey of resilience differentiation frameworks in communication networks. *IEEE Communications Surveys & Tutorials*, 9(4):32–55, Fourth 2007. ISSN 1553-877X. doi: 10.1109/COMST.2007.4444749.

[16] P. Cholda, J. Tapolcai, T. Cinkler, K. Wajda, and A. Jajszczyk. Quality of Resilience as a Network Reliability Characterization Tool. *IEEE Network*, 23 (2):11–19, March 2009. ISSN 0890-8044. doi: 10.1109/MNET.2009.4804331.

[17] P. Chołda, K. Rusek, and P. Guzik. Upper bound for failure risk in networks. *Electronic Notes in Discrete Mathematics*, 51:31 – 38, 2016. ISSN 1571-0653. doi: 10.1016/j.endm.2016.01.005.

[18] P. Chołda and A. Jajszczyk. Recovery and Its Quality in Multilayer Networks. *Journal of Lightwave Technology*, 28(4):372–389, Feb 2010. ISSN 0733-8724. doi: 10.1109/JLT.2009.2031821.

[19] P. Chołda, E.L. Følstad, B.E. Helvik, P. Kuusela, M. Naldi, and I. Norros. Towards risk-aware communications networking. *Reliability Engineering & System Safety*, 109:160 – 174, 2013. ISSN 0951-8320. doi: http://dx.doi.org/10.1016/j.ress.2012.08.009.

[20] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, RFC Editor, December 1998. URL `http://www.rfc-editor.org/rfc/rfc2460.txt`. Downloaded on May 9, 2017.

[21] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, RFC Editor, July 2017. URL `http://www.rfc-editor.org/rfc/rfc8200.txt`. Downloaded on Aug 5, 2017.

[22] F. Dikbiyik, M. Tornatore, and B. Mukherjee. Exploiting Excess Capacity for Survivable Traffic Grooming in Optical Backbone Networks. *IEEE/OSA Journal of Optical Communications and Networking*, 6(2):127–137, Feb 2014. ISSN 1943-0620. doi: 10.1364/JOCN.6.000127.

[23] J. Domżał. Intelligent routing in congested Approximate Flow-Aware Networks. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 1751–1756, Dec 2012. doi: 10.1109/GLOCOM.2012.6503368.

[24] J. Domżał and A. Jajszczyk. New Congestion Control Mechanisms for Flow-Aware Networks. In *IEEE International Conference on Communications (ICC '08)*, pages 12–16, May 2008. doi: 10.1109/ICC.2008.11.

[25] J. Domżał, Z. Duliński, M. Kantor, J. Rząsa, R. Stankiewicz, K. Wajda, and R. Wójcik. A survey on methods to provide multipath transmission in wired packet networks. *Computer Networks*, 77:18–41, 2015. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2014.12.001.

[26] J. Domżał, R. Wójcik, and A. Jajszczyk. *Guide to Flow-Aware Networking*. Springer International Publishing, 2015. ISBN 978-3-319-24973-5. doi: 10.1007/978-3-319-24975-9.

[27] J. Domżał, R. Wójcik, D. Kowalczyk, P. Gawłowicz, P. Jurkiewicz, and A. Kamisiński. Admission control in Flow-Aware Multi-Topology Adaptive Routing. In *2015 International Conference on Computing, Networking and Communications (ICNC)*, pages 265–269, Feb 2015. doi: 10.1109/ICCNC.2015.7069352.

[28] T. Elhourani, A. Gopalan, and S. Ramasubramanian. IP Fast Rerouting for Multi-Link Failures. In *2014 IEEE Conference on Computer Communications (INFOCOM)*, pages 2148–2156, April 2014. doi: 10.1109/INFOCOM.2014.6848157.

[29] G. Enyedi, P. Szilagyi, G. Retvari, and A. Csaszar. IP Fast ReRoute: Lightweight Not-Via without Additional Addresses. In *2009 IEEE Conference on Computer Communications (INFOCOM)*, pages 2771–2775, April 2009. doi: 10.1109/INFCOM.2009.5062229.

[30] E.L. Følstad and B.E. Helvik. The cost for meeting SLA dependability requirements; implications for customers and providers. *Reliability Engineering & System Safety*, 145:136–146, 2016. ISSN 0951-8320. doi: http://dx.doi.org/10.1016/j.ress.2015.09.011.

[31] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving Subsecond IGP Convergence in Large IP Networks. *ACM SIGCOMM Computer Communication Review*, 35(3):35–44, July 2005. ISSN 0146-4833. doi: 10.1145/1070873.1070877.

[32] A.V. Gheorghe, M. Masera, M. Weijnen, and L. De Vries. *Critical Infrastructures at Risk: Securing the European Electric Power System*, volume 9 of *Topics in Safety, Risk, Reliability, and Quality*. Springer Netherlands, 2006. ISBN 978-1-4020-4306-2. doi: 10.1007/1-4020-4364-3.

[33] A. J. Gonzalez, G. Nencioni, B. E. Helvik, and A. Kamisiński. A Fault-Tolerant and Consistent SDN Controller. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016. doi: 10.1109/GLOCOM.2016.7841496.

[34] A.J. González and B.E. Helvik. Analysis of Failures Characteristics in the UNINETT IP Backbone Network. In *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA)*, pages 198–203, March 2011. doi: 10.1109/WAINA.2011.55.

[35] A.J. Gonzalez and B.E. Helvik. *Advances in Computer Science, Engineering & Applications: Proceedings of the Second International Conference on Computer Science, Engineering and Applications (ICCSEA 2012), May 25-27, 2012, New Delhi, India, Volume 1*, chapter A Study of the Interval Availability and Its Impact on SLAs Risk, pages 879–890. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-30157-5. doi: 10.1007/978-3-642-30157-5_87.

[36] A.J. Gonzalez and B.E. Helvik. Hybrid Cloud Management to Comply Efficiently with SLA Availability Guarantees. In *2013 12th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 127–134, Aug 2013. doi: 10.1109/NCA.2013.32.

[37] A.J. Gonzalez, B.E. Helvik, P. Tiwari, D.M. Becker, and O.J. Wittner. GEARSHIFT: Guaranteeing availability requirements in SLAs using hybrid fault tolerance. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1373–1381, April 2015. doi: 10.1109/INFOCOM.2015.7218514.

[38] J. Gruen, M. Karl, and T. Herfet. Network Supported Congestion Avoidance in Software-Defined Networks. In *2013 19th IEEE International Conference on Networks (ICON)*, pages 1–6, Dec 2013. doi: 10.1109/ICON.2013.6781970.

[39] M. Guo and P. Bhattacharya. Controller Placement for Improving Resilience of Software-Defined Networks. In *2013 Fourth International Conference on Networking and Distributed Computing (ICNDC)*, pages 23–27, Dec 2013. doi: 10.1109/ICNDC.2013.15.

[40] P.E. Heegaard, B.E. Helvik, and V.B. Mendiratta. Achieving Dependability in Software-Defined Networking - A Perspective. In *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, pages 63–70, Oct 2015. doi: 10.1109/RNDM.2015.7324310.

[41] Y. Hu, W. Wendong, G. Xiangyang, C.H. Liu, X. Que, and S. Cheng. Control Traffic Protection in Software-Defined Networks. In *2014 IEEE Global Communications Conference (GLOBECOM)*, pages 1878–1883, Dec 2014. doi: 10.1109/GLOCOM.2014.7037082.

[42] ISO/IEC 7498-1:1994(E). Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. Standard, International Organization for Standardization, International Electrotechnical Commission, CH-1211 Genève 20, Switzerland, November 1994.

[43] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot. An Approach to Alleviate Link Overload as Observed on an IP Backbone. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 1, pages 406–416 vol.1, March 2003. doi: 10.1109/INFCOM.2003.1208692.

[44] A. Jajszczyk and P. Rozycki. Recovery of the Control Plane after Failures in ASON/GMPLS Networks. *IEEE Network*, 20(1):4–10, Jan 2006. ISSN 0890-8044. doi: 10.1109/MNET.2006.1580913.

[45] J. Joung, J. Song, and S. Lee. Flow-Based QoS Management Architectures for the Next Generation Network. *ETRI Journal*, 30(2):238–248, April 2008. doi: 10.4218/etrij.08.1107.0006.

[46] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene. FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 149–160, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3279-8. doi: 10.1145/2674005.2674985.

[47] A. Kamisiński and C. Fung. FlowMon: Detecting Malicious Switches in Software-Defined Networks. In *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, SafeConfig '15, pages 39–45, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3821-9. doi: 10.1145/2809826.2809833.

[48] A. Kamisiński, A. Jajszczyk, J. Domżał, and R. Wójcik. Sposób usuwania pętli w rutingu pakietów w sieci teleinformatycznej [A method for resolution of routing loops in a telecommunication network], December 2014. Polish patent application, no. P.410390.

[49] A. Kamisiński, P. Chołda, and A. Jajszczyk. Assessing the Structural Complexity of Computer and Communication Networks. *ACM Computing Surveys*, 47(4):66:1–66:36, May 2015. ISSN 0360-0300. doi: 10.1145/2755621.

[50] A. Kamisiński, J. Domżał, R. Wójcik, and A. Jajszczyk. Online Appendix, Jul 2016. URL `http://kt.agh.edu.pl/~kamisinski/pub/2016_commltr_congestion_control/`. Downloaded on May 9, 2017.

[51] A. Kamisiński, B. E. Helvik, A. J. Gonzalez, and G. Nencioni. Assessing the Risk of Violating SLA Dependability Requirements in Software-Defined Networks. In *IEEE NFV-SDN 2017 - Fourth Workshop on Network Function Virtualization and Programmable Networks (NFV-SDN'17-NFVPN)*, Berlin, Germany, Nov 2017. Accepted for publication.

[52] A. Kamisiński, J. Domżał, R. Wójcik, and A. Jajszczyk. Two Rerouting-Based Congestion Control Algorithms for Centrally Managed Flow-Oriented Networks. *IEEE Communications Letters*, 20(10):1963–1966, Oct 2016. ISSN 1089-7798. doi: 10.1109/LCOMM.2016.2594774.

[53] R. Kanagavelu and K. M. M. Aung. SDN Controlled Local Re-routing to Reduce Congestion in Cloud Data Centers. In *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, pages 80–88, Oct 2015. doi: 10.1109/ICCCRI.2015.27.

[54] R. Kanagavelu, Bu Sung Lee, R. Felipe Miguel, Le Nguyen The Dat, and L. N. Mingjie. Software Defined Network based Adaptive Routing for Data

Replication in Data Centers. In *2013 19th IEEE International Conference on Networks (ICON)*, pages 1–6, Dec 2013. doi: 10.1109/ICON.2013.6781967.

[55] S. Kini, S. Ramasubramanian, A. Kvalbein, and A.F. Hansen. Fast Recovery from Dual Link Failures in IP Networks. In *2009 IEEE Conference on Computer Communications (INFOCOM)*, pages 1368–1376, April 2009. doi: 10.1109/INFCOM.2009.5062052.

[56] S. Kini, S. Ramasubramanian, A. Kvalbein, and A.F. Hansen. Fast Recovery From Dual-Link or Single-Node Failures in IP Networks Using Tunneling. *IEEE/ACM Transactions on Networking*, 18(6):1988–1999, Dec 2010. ISSN 1063-6692. doi: 10.1109/TNET.2010.2055887.

[57] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The Internet Topology Zoo. *IEEE Journal on Selected Areas in Communications*, 29(9): 1765–1775, October 2011. ISSN 0733-8716. doi: 10.1109/JSAC.2011.111002.

[58] A. Kortebi, S. Oueslati, and J. W. Roberts. Cross-Protect: Implicit Service Differentiation and Admission Control. In *2004 Workshop on High Performance Switching and Routing (HPSR)*, pages 56–60, April 2004. doi: 10.1109/HPSR.2004.1303427.

[59] D. Kreutz, F.M.V. Ramos, and P. Verissimo. Towards Secure and Dependable Software-defined Networks. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 55–60, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2178-5. doi: 10.1145/2491185.2491199.

[60] D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015. ISSN 0018-9219. doi: 10.1109/JPROC.2014.2371999.

[61] K.W. Kwong, L. Gao, R. Guerin, and Z.L. Zhang. On the Feasibility and Efficacy of Protection Routing in IP Networks. *IEEE/ACM Transactions on Networking*, 19(5):1543–1556, Oct 2011. ISSN 1063-6692. doi: 10.1109/TNET.2011.2123916.

[62] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving Convergence-free Routing Using Failure-carrying Packets. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pages 241–252, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-713-1. doi: 10.1145/1282380.1282408.

[63] P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton. An Object-Oriented Random-Number Package with Many Long Streams and Substreams. *Operations Research*, 50(6):1073–1075, 2002. doi: 10.1287/opre.50.6.1073.358.

[64] S. Lee, Y. Yu, S. Nelakuditi, Zhi-Li Zhang, and Chen-Nee Chuah. Proactive vs Reactive Approaches to Failure Resilient Routing. In *2004 IEEE Conference on Computer Communications (INFOCOM)*, volume 1, page 186, March 2004. doi: 10.1109/INFCOM.2004.1354492.

[65] F. Longo, S. Distefano, D. Bruneo, and M. Scarpa. Dependability modeling of Software Defined Networking. *Computer Networks*, 83:280–296, 2015. ISSN 1389-1286. doi: 10.1016/j.comnet.2015.03.018.

[66] S.S. Lor, R. Landa, and M. Rio. Packet Re-cycling: Eliminating Packet Losses Due to Network Failures. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 2:1–2:6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0409-2. doi: 10.1145/1868447.1868449.

[67] W.W. Lowrance. *Of Acceptable Risk: Science and the Determination of Safety*. Kaufmann, William, Incorporated, Los Altos, Calif., 1976. ISBN 9780913232309.

[68] M. Manzano, E. Calle, V. Torres-Padrosa, J. Segovia, and D. Harle. Endurance: A new robustness measure for complex networks under multiple failure scenarios. *Computer Networks*, 57(17):3641–3653, 2013. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2013.08.011.

[69] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, Chen-Nee Chuah, Y. Ganjali, and C. Diot. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking*, 16(4):749–762, August 2008. ISSN 1063-6692. doi: 10.1109/TNET.2007.902727.

[70] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38 (2):69–74, March 2008. ISSN 0146-4833. doi: 10.1145/1355734.1355746.

[71] G. Nencioni, B. E. Helvik, A. J. Gonzalez, P. E. Heegaard, and A. Kamisiński. Availability Modelling of Software-Defined Backbone Networks. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 105–112, June 2016. doi: 10.1109/DSN-W.2016.28.

[72] T. T. Nguyen and D. S. Kim. Accumulative-Load Aware Routing in Software-Defined Networks. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 516–520, July 2015. doi: 10.1109/INDIN.2015.7281787.

[73] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, RFC Editor, December 1998. URL `http://www.rfc-editor.org/rfc/rfc2474.txt`. Downloaded on May 9, 2017.

[74] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. SNDlib 1.0—Survivable Network Design Library. *NET*, 55(3):276–286, May 2010.

[75] S. Oueslati and J. Roberts. A New Direction for Quality of Service: Flow-Aware Networking. In *Next Generation Internet Networks, 2005*, pages 226–232, April 2005. doi: 10.1109/NGI.2005.1431670.

[76] P. Pan, G. Swallow, and A. Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090, RFC Editor, May 2005. URL `http://www.rfc-editor.org/rfc/rfc4090.txt`. Downloaded on May 9, 2017.

[77] J. Postel. Internet Protocol. RFC 791, RFC Editor, September 1981. URL `http://www.rfc-editor.org/rfc/rfc791.txt`. Downloaded on May 9, 2017.

[78] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, RFC Editor, September 2001. URL `http://www.rfc-editor.org/rfc/rfc3168.txt`. Downloaded on May 9, 2017.

[79] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, January 2006. URL `http://www.rfc-editor.org/rfc/rfc4271.txt`. Downloaded on May 9, 2017.

[80] F.J. Ros and P.M. Ruiz. Five Nines of Southbound Reliability in Software-defined Networks. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 31–36, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2989-7. doi: 10.1145/2620728.2620752.

[81] K. Rusek, P. Guzik, and P. Chołda. Effective risk assessment in resilient communication networks. *Journal of Network and Systems Management*, 24 (3):491–515, 2016. ISSN 1573-7705. doi: 10.1007/s10922-016-9370-3.

[82] M. Shand and S. Bryant. IP Fast Reroute Framework. RFC 5714, RFC Editor, Jan 2010. URL `http://www.rfc-editor.org/rfc/rfc5714.txt`. Downloaded on May 9, 2017.

[83] R. Stankiewicz, P. Cholda, and A. Jajszczyk. QoX: What is it really? *IEEE Communications Magazine*, 49(4):148–158, April 2011. ISSN 0163-6804. doi: 10.1109/MCOM.2011.5741159.

[84] The International Telecommunication Union - Telecommunication Standardization Sector. Requirements for the support of flow-state-aware transport technology in an NGN. Recommendation Y.2121, ITU-T, January 2008.

[85] J. Tyszer. *Object-Oriented Computer Simulation of Discrete-Event Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1999. ISBN 0792385063.

[86] I.B. Utne, P. Hokstad, and J. Vatn. A method for risk modeling of interdependencies in critical infrastructures. *Reliability Engineering & System Safety*, 96(6):671 – 678, 2011. ISSN 0951-8320. doi: http://dx.doi.org/10.1016/j.ress.2010.12.006. ESREL 2009 Special Issue.

[87] R. Wójcik and A. Jajszczyk. Flow Oriented Approaches to QoS Assurance. *ACM Computing Surveys*, 44(1):5:1–5:37, January 2012. ISSN 0360-0300. doi: 10.1145/2071389.2071394.

[88] R. Wójcik, J. Domżał, and Z. Duliński. Flow-Aware Multi-Topology Adaptive Routing. *IEEE Communications Letters*, 18(9):1539–1542, Sept 2014. ISSN 1089-7798. doi: 10.1109/LCOMM.2014.2334314.

[89] R. Wójcik, J. Domżał, Z. Duliński, P. Gawłowicz, and P. Jurkiewicz. Loop Resolution Mechanism for Flow-Aware Multi-Topology Adaptive Routing. *IEEE Communications Letters*, 19(8):1339–1342, Aug 2015. ISSN 1089-7798. doi: 10.1109/LCOMM.2015.2439679.

[90] R. Wójcik, J. Domżał, Z. Duliński, G. Rzym, A. Kamisiński, P. Gawłowicz, P. Jurkiewicz, J. Rząsa, R. Stankiewicz, and K. Wajda. A survey on methods to provide interdomain multipath transmissions. *Computer Networks*, 108: 233–259, 2016. ISSN 1389-1286. doi: 10.1016/j.comnet.2016.08.028.

[91] K. Xi and H.J. Chao. IP Fast Rerouting for Single-Link/Node Failure Recovery. In *Fourth International Conference on Broadband Communications, Networks and Systems, 2007. BROADNETS 2007*, pages 142–151, Sept 2007. doi: 10.1109/BROADNETS.2007.4550418.

[92] M. Xia, M. Tornatore, C.U. Martel, and B. Mukherjee. Risk-aware Provisioning for Optical WDM Mesh Networks. *IEEE/ACM Transactions on Networking*, 19(3):921–931, June 2011. ISSN 1063-6692. doi: 10.1109/TNET.2010.2095037.

[93] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng. Keep Forwarding: Towards k-link failure resilient routing. In *2014 IEEE Conference on Computer Communications (INFOCOM)*, pages 1617–1625, April 2014. doi: 10.1109/INFOCOM.2014.6848098.