Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics

Instantaneous Failure Recovery and Localization in Transport Networks

PhD Dissertation

Alija Pašić

Advisor:

Péter Babarczi Ph.D.

MTA-BME Future Internet Research Group
High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics

Budapest, Hungary
2018.

...to my family.

Abstract

Communication networks are considered to be topmost critical infrastructures in today's information society. Owing to the rise of novel mission-critical applications (e.g. telesurgery, stock market), high reliability and the delay requirements gained on importance and became a crucial question. Satisfying the new strict requirements leads to great competitive advantage for the network operators. However, satisfying both requirements i.e. high reliability and low latency at the same time in a bandwidth-efficient manner is unquestionably one of the most challenging tasks for service providers in transport networks.

In today's transport networks, the most widespread protection approach is the so called dedicated $1 + 1$ protection approach, owing to its simplicity and ultra fast recovery time. However, by $1 + 1$ the data is sent parallel on both disjoint paths, fast recovery is provided to the connections at the price of *excessive protection resources*. Besides that, $1 + 1$ usually optimizes only for one cost parameter (e.g. bandwidth or delay) and does not consider multiple constraints at the same time.

In my dissertation, I propose new protection and failure localization frameworks, which not only consider the new trends in transport networks, but are also able to satisfy the new strict multi-constraint requirements. Particularly in the first part of my dissertation, a new, easily deployable and extremely resource efficient single link failure resilient method called Survivable Routing with Diversity Coding (SRDC) is introduced. The method is easily deployable, since the *coding itself can be done in the source and destination nodes, i.e. there is no need to upgrade network nodes*. Several subproblems of SRDC were investigated depending on capacity and node capability constraints in the network. For multiple failures the method called Survivable Routing with Network Coding (SRNC) was introduced.

Several works are dealing with Quality-of-Service (QoS) routing and differential delay (DD) aware survivable routing in transport networks; however all of the methods are designed for disjoint paths only. I generalize these results for diversity coding based survivable routing using Directed Acyclic Graphs (DAG). Hence, I define the before- and after-failure delay of a DAG, and investigate the effect of QoS routing and DD-aware routing delay constraints on these optimal survivable routing structures, ensuring instantaneous recovery.

In the second part of the dissertation the focus is on failure localization. I propose a new all-optical local failure monitoring framework that enables ultra fast (although not instantaneous) restoration even for the shared protection approaches, by reducing the failure recovery time significantly. I introduced a new concept called forbidden link-pairs which generalizes several monitoring trail based problems.

Kivonat

A mai információs társadalomban a kommunikációs hálózatok az egyik legfontosabb kritikus infrastruktúrának számítanak. Az új késleltetés kritikus alkalmazásoknak (távsebészet, tőzsde, VoIP) köszönhetően nem csak a megbízhatóság, de a késleltetési követelmények is a leglényegesebb kérdések egyikévé léptek elő. Ezen új, szigorú követelményeknek való megfelelés nagy versenyelőnyt jelent a hálózati szolgáltatók számára. Azonban a magas rendelkezésre állás és alacsony késleltetés egyidejű biztostása sávszélesség-hatékony módon, vitathatatlanul az egyik legnagyobb kihívást jelentő probléma.

Jelenleg a hozzárendelt védelmek családjába tartozó $1+1$ védelem a legelterjedtebb, melynek lényege, hogy minden összeköttetés igény adatát két diszjunkt útvonalon egyszerre továbbítjuk. Ez igencsak magas erőforrásigényt eredményez, valamint az $1+1$ csak egyszeres hibák védelmére alkalmas és csak egyetlen egy költség paramétert vesz figyelembe. Vagyis az újonnan felmerülő komplex Quality of Service (QoS) igényeknek nem képes eleget tenni.

Az értekezés első részében egy új, rendkívül erőforrás-hatékony és gyakorlatias egyszeres linkhiba ellenálló módszert mutatok be, az úgynezevzett Survivable Routing with Diversity Coding-ot (SRDC). A módszer könnyen telepíthető, mivel a *a kódolás a forrás és a cél csomópontokban elvégezhető, azaz nincs szükség a csomóponti képességek frissítésére*. Az SRDC több alproblémáját vizsgáltam, a hálózat kapacitása és a csomóponti képességek függvényében. A többszörös hibák védelmére a Survivable Routing with Network Coding (SRNC) került bevezetésre.

Számos kutatás foglalkozik a QoS és a differenciális késleltetést (DD) figyelembe vevő megbízható útvonalválasztás kérdésével a transzport hálózatokban, azonban mindegyik munka a diszjunkt utak kérdését veszi szemügyre. Disszertációmban ezen eredményeket általánosítom az irányított körmentes gráf (DAG - Directed Acyclic Graph) struktúrát használó diverzitás kódolás alapú megbízható útvonalválasztási módszerekre. Ehhez meghatározom a DAG-ok meghibásodás előtti és utáni késleltetését, ezután pedig megvizsgálom a QoS és a DD tudatos útvonalválasztás késleltetési kényszereiknek hatását az azonnali helyreállítást biztosító optimális megbízható útvonalakra.

Disszertációm második felében a hibalokalizáció kérdésével foglalkozom. Javaslok egy olyan új lokális hiba monitorozó keretrendszert, amely lehetővé teszi az igen gyors helyreállítást megosztott védelmi módszerek esetén is. Illetve bevezetek egy új koncepciót is, a tiltott linkpárok fogalmat, amelynek segítségével általánosítok több monitorozó út alapú problémát.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Péter Babarczi for all the time and energy he has put into supervising and guiding my research. His encouragement, guidance and support were indispensable to becoming a researcher in the field of telecommunications.

I am also indebted to János Tapolcai, who was my co-supervisor in my M.Sc. years and helped me make my way towards being a PhD student. During my PhD studies, he was always able to help me with new fresh ideas. I would like to thank to my colleagues and roommates.

My work was carried out in the Department of Telecommunications and Media Informatics at Budapest University of Technology and Economics, in the Hungarian research group, called MTA-BME Lendület and in the High Speed Networks Laboratory (HSNLab), therefore I would like to thank to János Tapolcai, Attila Vidács for their support and advices and to Erzsébet Győri for all her help during all these years.

Last but not least, I would like to thank to my family, especially my father and my mother for supporting me through difficult times, studies and life, for teaching me what is important in life and for being real role models in times where most people would have crumbled. I also would like to thank to my siblings and my fiancée for being there for me in all the ways possible. Without of love and patience of such a great family, my dreams would have never come true.

Contents

Chapter 1

Introduction

Today in the cloud era, communication networks are considered to be a topmost critical infrastructure. The relevance of communication networks increased in the past decade owing to the rise of novel mission-critical applications (e.g. telesurgery, stock market) with high reliability and strict delay requirements. Transport network operators face new challenges due to the rise of these applications (e.g. VoIP, 4K/8K video traffic, AR/VR (Augmented and Virtual Reality etc.)) which not only require high resilience from the underlying network but are also highly delay sensitive [40]. Satisfying these strict requirements can lead to a great competitive advantage for the network operators. As [40] highlights a high-frequency trading financial company may increase revenue by 100 million USD a year with only 1 ms speed improvement in each transaction. However, satisfying both requirements i.e. high reliability and low latency at the same time in a bandwidth-efficient way is unquestionably one of the most challenging tasks for service providers in transport networks.

In addition to these challenges, also the nature of transport networks changed. The optical transport networks shifted from opaque to transparent optical (or all-optical) networks [63] to reach higher data rates and to avoid the time and energy consuming O/E/O (optical/electrical/optical) conversion. This means that the lightpaths corresponding to logical links are carried from source to target entirely in the optical transport layer without any intermediate optical-electrical conversion [75], making fault propagation a bigger issue. On the other hand, new technologies like SDN (Software Defined Networking) emerged and introduced an entirely new network management concept decoupling the network control and forwarding functions. SDN both abstracts the infrastructure for applications and network services, and makes the network control fully and directly programmable. It is not surprising that the new failure recovery and localization methods have to consider the needs and requirements of these new technologies and trends (all-optical networks and SDN).

Nonetheless or for that very reason transport networks are highly vulnerable to physical link failures, as they affect a large amount of data [13, 71, 72]. Cable cuts may cause outages disrupting the service for a large number of end users [21, 57, 97]. These cuts may have several causes, like natural disasters [57] construction work, unpermitted digging etc. [21, 57]. For example, in 2001 the Howard Street Tunnel

(Baltimore) fire was caused by a freight train derailment [21], the fire melted away the fiber along the tunnel, leading to a large outage. There are also some bizarre causes like when in 2011 a woman digging for scrap metal damaged a land cable causing an outage for 3 Million users.

Thanks to the efforts in failure mitigation, current Internet has reached the level of reliability where mission-critical applications and cloud services are widely spreading among users. However, this gives an increasing push on the service providers to operate the Internet without any interruption in order to slowly win the trust of most of the potential users. To do so, one of the most important targets of the Internet carriers is to meet the service requirements defined in the Service Level Agreement (SLA) with the subscribers in their backbones, i.e. to provide high Quality of Service (QoS). *In other words providing and maintaining a reliable network (i.e. providing high QoS) has utmost importance for network providers, because business customers are willing to pay significantly more for a reliable service. This means that the providers have to model the network correctly (in hindsight of QoS) and choose the proper failure mitigation techniques (proper failure management techniques), while keeping the CAPEX (CAPital EXpenditures) and OPEX (OPErating EXpenditures) as low as possible.*

Of course there is always a trade-off between the QoS level and the resource requirements, and the providers itself have to decide which QoS level should be enforced.

However, it is a well known fact that transferring a user's data along a single active (or *working*) path in these new transport networks with throughput in order/magnitude of Tbytes per second [30, 102] can not be sufficient to fulfill the service availability requirements in the presence of various network outages and unexpected failure events [70]. This means in such transport networks even the shortest disruptions lead to huge amount of data loss i.e. huge drop in QoS, which is unacceptable. To avoid this, *instantaneous recovery* is required in today's transport networks. The requirement of *instantaneous recovery* is fulfilled if the recovery time is less than 50 ms [92].

This means when planning the recovery process our main goal is to keep the recovery time under 50 ms, in order to ensure seamless operation even when a failure occurs. However, instantaneous recovery is not the only important factor for a protection approach. Besides instantaneous recovery the robustness of the approaches can be crucial. A protection approach is called *robust* if during the recovery process no control plane actions (messaging) are required and no Optical Cross-Connect (OXC) reconfiguration occurs [J1]. With the robustness we ensure simplicity, which is crucial for the practical implementation.

In today's transport networks the most widely spread protection approach is the so called dedicated $1 + 1$ protection approach, owing to its simplicity and ultra fast recovery time. The dedicated $1 + 1$ protection gives the fastest restoration against single link failures, where the failure localization and notification time is zero, and the time setting up the protection path is assumed to be a few milliseconds. The process requires no control plane signaling, as each node can decide whether a failure occurred or not in a distributed fashion. Thus the $1 + 1$ protection provides *robust and instantaneous recovery*. In order to keep the complexity and restoration time low, this property has to be fulfilled. However, as by $1 + 1$

the data is sent parallel on both disjoint paths, fast recovery is provided to the connections for the price of *excessive amount of protection resources*. Besides that, $1+1$ optimizes only for one cost parameter (e.g. bandwidth or delay) and can not consider multiple constraints at the same time. As the *new stricter QoS requirements desire multiple constraints* $1+1$ has to be replaced in the long term. Nonetheless, today's alternatives (i.e. the state-of-the-art protection approaches discussed in details in Section 2.3) could not replace the $1+1$. The new methods either did not ensure instantaneous recovery (e.g. shared protection approaches), or were too complex to be implemented in today's transport networks i.e. the robustness was not ensured. Thus, the trade-off between capacity allocation (bandwidth utilization), recovery time and complexity (computational and operational complexity) was not more appealing than by $1+1$.

The protection approaches introduced in this dissertation reduce the trade-off and provide an appealing alternative in such way that the positive properties of $1+1$ are preserved (i.e. robustness and instantaneous recovery is provided) and in addition the capacity allocation overhead is reduced. Note that although single link failures are the most common ones (more than 70% [61] of all failures affecting links) the backbone networks should be able to support connections with different level of availability (i.e. QoS level). In particular, focusing on these key factors, in the first part of my dissertation I propose new methods with low resource requirements that ensure *instantaneous recovery in a robust manner* while being able to satisfy *delay constraint even after a failure* occurs. I compare the new methods with $1+1$ and other state-of-the-art protection approaches.

In the second part of the dissertation I focus on the failure localization. I propose a new all-optical local failure localization framework, that enables fast restoration even for the shared protection approaches, reducing the failure recovery time significantly. Note that due to the nature of the shared protections optical cross-connect (OXC) reconfiguration is always needed. This means *robust recovery can not be achieved* i.e. the simplicity level of $1+1$ can not be maintained. Since the OXC reconfiguration itself can take up to several tens of milliseconds depending on the technology [53, 60, 101] i.e. instantaneous recovery can not be guaranteed, even if the failure localization and notification time takes only milliseconds.

Of course, network operators can and have to consider the failure mitigation technique according the desired QoS level. If robustness has utmost importance dedicated protection approaches have to be implemented, if resource efficiency is the key factor shared protection approaches can be considered. In my dissertation I propose solutions for several scenarios with different objectives (robustness, capacity efficiency) and QoS level.

Chapter 2

Objectives and Methodology

2.1 Evolution of the Transport Networks

Transport networks are carrying a huge amount of data and hence are relaying on optical communication i.e. light is used to carry information. The optical layer of transport networks started as a set of static point-to-point connections (i.e. pipes) but quickly evolved to a true managed layer with reconfigurable switching nodes.

The first big step in the evolution was the introduction of the so called Synchronous Digital Hierarchy (SDH)/Synchronous Optical NETwork (SONET), which is a word wide standard for transport networks. These networks provide huge amounts of bandwidth by taking advantage of the Wavelength Division Multiplexing (WDM - each optical fiber carries a large number of wavelength channels), ensure low latency and implement a protection scheme called Linear Automatic Protection Switching (APS), which is also known as $1 + 1$. The APS SDH networks are capable of providing fast protection switching (50 ms). This is realized with sophisticated supervisory processes for failure detection, notification and propagation process [92].

However, SDH/SONET networks were still designed and optimized for the more predictable voice traffic and not for data traffic. Thus, these networks were provisioned i.e. no dynamic routing was possible and the granularity was coarse.

In order to enhance the flexibility and improve the coarse granularity of SDH the co called Next Generation SDH (ngSDH/SONET) was introduced, which is basically SDH with some new added functionalities [92]:

- Virtual Concatenation (VCat),
- Link Capacity Adjustment Scheme (LCAS) and
- Generic Framing Procedure (GFP).

However, not only the transport technologies changed but also the topology itself. In order to ensure

higher flexibility and reliability mesh networks became preferable in contrast to ring topologies. Of course this results in more complex networks, leading to more complex routing, protection and signaling problems. Note that by SDH and ngSDN the switching itself in the nodes i.e. OXCs occurs in the electrical domain. This means we need an optical-electircal-optical (O/E/O) conversion in each node, resulting in the so called opaque networks. The O/E/O conversion is not only time and energy consuming but quickly becomes the bottleneck for the communication throughput. On the plus side, this switching technique enables sub-wavelength granularity switching.

As the traffic kept growing, a slow evolution from SDH/SONET to Optical Transport Networks (OTNs) took place. The OTN itself is defined by the International Telecommunications Union-T (ITU-T) G.709: Interfaces for the Optical Transport Network (OTN) and has a structure very similar to SONET/SDH networks. However, the switching in OTN occurs entirely in optical domain, i.e. the OTN is switching complete wavelength channels as a single entity removing the time and energy consuming O/E/O conversion, leading to so called transparent or all-optical networks [92]. Note that as stated in [92] the functionality of the OTN follows the generic principles defined in ITU-T Recommendation (ITU-T Rec.) G.805.

The structure of OTN is the following [92]:

– OTS (Optical Transmission Section): Provides the functionality for the transmission of optical signals between two network elements including amplifiers.

– OMS (Optical Multiplex Section (OMS): Provides the networking functionality for a multiwavelength optical signal. It is the section between the multiplexers and demultiplexers.

– OCh (Optical Channel path): Provides end-to-end networking functionality for the optical channels i.e. encapsulated client signals, allowing a transparent transportation of client signals (of varying format) between 3R regeneration points in the network.

The introduction of OTN allows true optical networking, since there is no need for expensive optoelectrical conversions (O/E/O) and electrical layer processing. As a result the traffic can stay in the optical domain until it reaches its destination, creating the co called *lightpaths*. Of course assuming wavelength granularity, two lightpaths (LPs) can use the same links if and only if they use different wavelengths.

Until 2000 the voice services generated the majority of the traffic. After 2000, the rapid growth of Internet services i.e. data traffic introduced the need to dynamically change the optical layer connectivity. Thus, a control plane (CP) technology called automatically switched optical network (ASON) was introduced [11]. ASON implements automatic topology and resource discovery, automatic service set-up and tear down, and automatic service protection/restoration in the optical network. This way the optical network evolved from manual management to automatic control [40].

Nowadays the technology called Flex Grid is making waves, and is making all-optical networks even more flexible. The technology is improving the current rigid spectrum allocation, using higher-order

and multi-carrier modulation. According to [40] in the future optical transport networks will be able to flexibly select not only the number of carriers but also the optical modulation mode and the spectrum slice in order to achieve the most appropriate solution for a given problem [40].

In these dynamic networks it is a challenge to find the proper protection scheme, taking into account multiple factors like the capacity allocation overhead, recovery time, complexity of the protection scheme, the CAPEX and OPEX and even latency requirements of the connection.

The protection methods and frameworks introduced in the dissertation are designed for transport networks (optical layer protection in WDM and SDN networks), and are taking into account the most recent trends like SDN, Flex Grid and the requirements of cloud computing and 5G mobile network [40]. However, all protection approaches face the trade-off between capacity allocation, recovery time and complexity. The protection approaches introduced in this dissertation reduce this trade-off and provide an appealing alternative (i.e. reduce the capacity allocation overhead while still providing instantaneous recovery in a simple manner) to current protection approaches.

2.2 Graph Representation and Failure Modeling

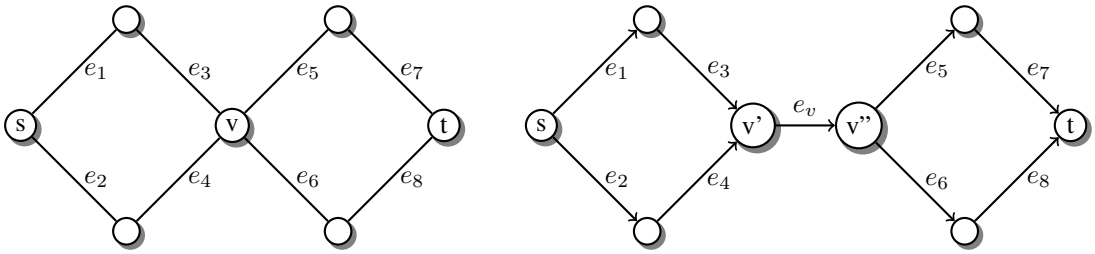
The goal of this section is to introduce the basic concepts used in this dissertation i.e. the graph representation ($G = (V, E)$) of the transport networks and the failure modeling method. In particular in Section 2.2.1 the graph representation is presented, and in Section 2.2.2 the reliability and failure modeling is discussed, introducing the concept of SRLG (Shared Risk Link Group).

2.2.1 Graph Representation and Traffic

Transport networks are usually represented in a layered manner [92]. In my dissertation the physical layer which consists of fibers and optical cross-connects is represented with a directed graph $G = (V, E)$. The physical infrastructure is static i.e. is easily representable with a graph, contrary to the lightpaths (i.e. connections which can be established within milliseconds between arbitrary pairs of nodes in the network) which are built and torn-down dynamically.

In Figure 2.1 an example of a graph representation $G = (V, E)$ is given, with node set V and edge (link) set E. The nodes represent the OXC-s where the connection can enter or leave the networks. The set of edges E represents the bidirectional fiber connections between the OXCs (i.e. Optical Multiplex Section). Since the fiber connections are bidirected edges we can represent the graph with undirected edges. If two OXCs are connected with a fiber i.e. edge then these OXCs are adjacent. The edges can have several attributes, like cost $c(e)$, delay $d(e)$ or free capacity $k(e)$ i.e. the number of wavelengths (channels or carriers) available on the given edge e.

The cost function $c(e)$ corresponds to the cost of allocating a unit of demand (i.e. wavelength) on the given edge e. The cost function can include the number of regenerators needed on the link, the link distance or load on the link etc. We speak about Traffic Engineering (TE) if we use the cost function to



(a) Graph representation (b) Auxiliary Graph

Figure 2.1: The graph representation of the networks and a graph transformation example for modeling node failures.

route traffic i.e. if we set the cost function according to the topology and traffic characteristics. Traffic engineering can be used to ensure load balancing and results in a lower blocking probability and can increase the overall network performance [11]. Another important attribute used in this dissertation is the delay value. The delay value $d(e)$ is most commonly in relation with the link length [km] i.e. propagation delay. Of course for a given path besides the delay of the fiber links i.e. sum of the propagation delays on these links the so called node delay can be added too. As stated in [39] the latency for fiber optics is typically 5 μs/km and the maximum latency per node is 25 μs (in SDH networks, in OTN even less). This means the propagation delay can be treated as the dominant part of the overall delay and is a good approximation for the overall delay itself.

Besides the physical network infrastructure, the traffic itself has to be modeled, too. In the first part of my dissertation for the Survivable Routing (Chapter 3 and 4) a dynamic routing scenario is assumed, where traffic demands arrive one after the other, without any knowledge of future incoming request, i.e., each request is routed independently. Thus, as part of the routing problem input only a single *connection request* $C = (s, t, b, D)$ is given, which consists of the source node $s \in V$, destination node $t \in V$, the number of bandwidth units b requested for data transmission, and if delay requirements are considered a maximal delay bound D which has to be satisfied. If no delay bound D is given, then $D = \infty$ is assumed. Note that the traffic request is directed (from s to t), however we assume that the traffic request from the opposite direction (from t to s) can be routed the same manner. In the second part of my dissertation i.e. in the Failure Localization part (Chapter 5) several traffic scenarios were investigated, a dynamic and a more static version as well.

2.2.2 Reliability and Failure Modeling

As seen in Section 2.2.1 the network is represented with a graph $G = (V, E)$, with node set V and edge set E, where the nodes V represent the OXCs and the edges E represent the bidirectional fiber connections between the OXCs. However, these fiber connections and OXC (i.e. the network elements) are not always functional or are not functioning properly. These failures can be caused by natural disasters (earthquakes, fires, floods, hurricanes, storms), wear out, overload, software bugs, human errors,

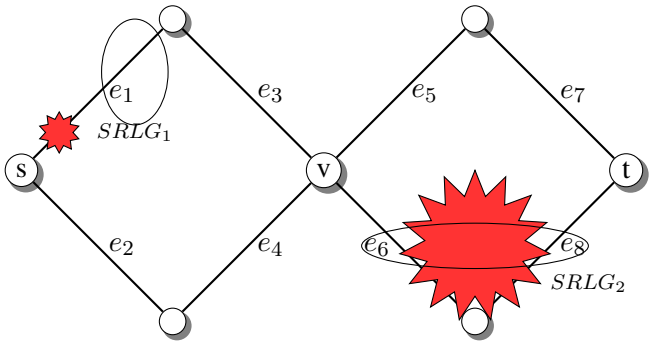


Figure 2.2: An example of different magnitudes of failures and their SRLG representation.

sabotage or even planned or unplanned maintenance [92]. Regardless to the cause of the failure, these can lead to service disruption or degradation. In order to be able to prevent and protect the networks from such events (i.e. service disruption or degradation), we have to be able to model the failures and the network properly and reliably.

The *reliability* of a network element (e.g., a node, link) is defined as the probability of a network element to be fully operational during a certain time frame (i.e. to be fully operational between t_1 and t_2). Another often used measure is the *availability*. The availability is the probability of a network element to be operational at one particular point in time t_1 [92].

When modeling networks, commonly nodes are considered to be completely reliable (i.e. the availability is 1). However, if not, a node failure can be represented with an edge failure in an directed auxiliary graph (Figure 2.1). In this case the given node v is split into two nodes v' and v'' and an edge is added between them. Each incoming edge of v is then directed to v', while each outgoing edge of v is directed from v''. The failure of this edge i.e. single edge failure in the auxiliary graph represents the failure of the node in graph $G = (V, E)$.

Another option is to represent the node failure with a set of edge failures i.e. with the failure of all adjacent links to the node. This can be done with the so called Shared Risk Link Group (SRLG) list $\mathcal{F}$. With the SRLGs we are able to handle dependent multiple failures. In other words the SRLG are capable of expressing the statistical dependencies between failures (e.g., links, nodes). For example if we know that two seemingly unrelated links share a tunnel or a conduit on a given section, then these two links belong to the same SRLG since a single failure event like a tunnel fire (e.g. Baltimore tunnel fire) can disrupt both links [11]. In my dissertation the SRLG method is used to model the failures. Note that an extensive part of the dissertation focuses on providing robust and instantaneous recovery methods for a very practical case i.e. single link failures scenario (the SRLG list contains only all the single link failures) which is the most common one, since more than 70% [61] of all failures affecting links are single failures. Nonetheless several algorithms were presented for the general failures scenario case (i.e. arbitrary SRLG list) in order to complete the study, and to be able to provide different QoS levels.

2.3 State-of-the-Art Protection and Failure Localization Approaches in Transport Networks

Currently, backbone networks are typically designed either without protection or to protect failures that hit just a single network element. However, several papers investigated how to generalize the protection in order to protect multiple link or node failures [15, 37], or failures that affect geographically larger areas [3, 95]. The trend is clear, we expect the services in backbone networks to further improve in the terms of *flexibility and reliability*. In our vision, connections have to be routed promptly after they arrive to the ingress network element, and these connections will operate without interruptions, unless there is a huge catastrophe nearby.

In order to fulfill these requirements, *dedicated protection* is currently the best candidate for improving network survivability. By dedicated protection approaches multiple copies of the traffic flows are launched simultaneously in order to maintain high service availability in the presence of unexpected network outages and failures. Furthermore, these methods provide the desired *robust and instantaneous recovery* for the connections after a failure occurs, i.e. the recovery time is less than 50 ms and no control plane messaging is required (neither packet retransmission nor flow rerouting is required to heal the connections with disrupted working lightpath (also denoted as W-LP)).

Previously reported studies on dedicated protection [17, 28, 44, 94, 100] have manipulated various assumptions on the node capability for the incoming and outgoing flows at each node to improve bandwidth efficiency, such as switching and merging, bifurcated or non-bifurcated, and with or without network coding. Note that when *network coding* [4] is applied, instead of simple forwarding [38], the switches can perform algebraic operations on their incoming packets to construct the outgoing encoded packet.

By shared protection schemes a single protection path (also denoted as P-LP) can be used to protect several working paths i.e. can be shared. This means the user data is carried on the working paths only and the shared backup path is only used when a failure occurs affecting one of those (protected) working paths. Shared protection schemes yield better resource efficiency, however are not able to provide *robust and instantaneous recovery*.

In the next sections we discuss the recovery process (Section 2.3.1) and summarize the state-of-the-art routing and network coding based protection approaches (Section 2.3.3) in optical networks. Furthermore the General Dedicated Protection (GDP) (Section 2.3.4) framework is discussed in a separate subsection because of its theoretical importance. After that the failure localization in optical transport networks is overviewed. The SDN aspects of protection and failure localization are summarized in Section 2.3.6.

2.3.1 Recovery Time

In order to understand and investigate the recovery time (t_R), which is a critical (key) parameter of any protection scheme, we have to understand the recovery process of transport networks. As defined in

GMPLS control plane [69], a restoration process of a working path is composed of a number of real-time tasks after the occurrence of a failure event, until the interrupted flow is completely restored [88]. The restoration time itself is:

$$t_R = t_l + t_n + t_c + t_d + t_s, \tag{2.1}$$

where:

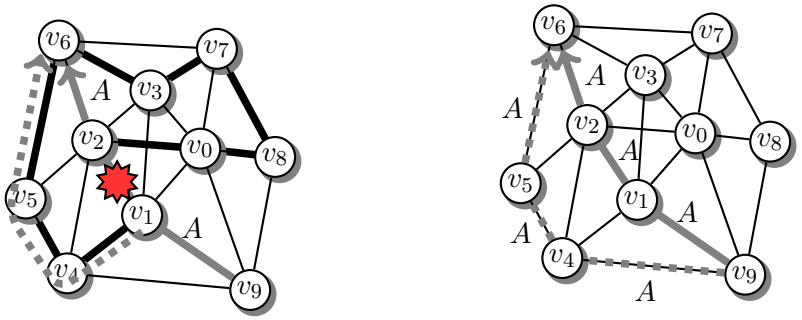
- (failure localization time) t_l is defined as the time to locate the failure at a responsible network entity,
- (failure notification time) t_n is the failure notification time, where the responsible network entity notifies all necessary switching nodes (via control plane signaling) which perform protection switching,
- (failure correlation time) t_c is defined as the time period between the time instant when the switching node receives all the notifications, and the time instant when the failure is identified,
- (decision time) t_d is the time during which the switching nodes select the necessary switching actions (protection paths),
- (switching time) t_s is defined as the time for setting up the protection path i.e. configuration of the nodes, Optical cross-connects (OXC) etc. This step itself can take several tens of milliseconds [60, 101]).

When planning the recovery process our main goal is to keep the recovery time under 50 ms, in order to ensure seamless operation. In order to achieve this the failure localization, notification, correlation and decision time has to be minimized.

2.3.2 Protection Approaches with Routing in Optical Networks

In the traditional ring based systems the first protection approaches were the so called *self-healing rings*, where the network connections were built up as bidirectional rings. In these systems the primary (or working) paths were established via the shortest path method, while the other side of the ring was used as a protection path. If a failure occurred, the two nearest surviving nodes of the working path rerouted the traffic to the protection resources, which made this architecture resilient against single link failures. The restoration speed of these ring networks is $50 - 150$ ms [32], which is used as the benchmark for the restoration time of subsequent protection approaches.

In order to adopt the simplicity of ring-based restoration in mesh networks *p-cycle based protection schemes* were introduced by Grover and Stamatelakis [32]. The p-cycles try to combine ring like speed with mesh-like capacity efficiency, close to that of shared protection approaches with the allocation of pre-configured protection capacity in the shape of cycles (an example on a p-cycle is shown in bold



(a) P-Cycles: The data of working path $v_9 \rightarrow v_1 \rightarrow v_2 \rightarrow v_6$ is rerouted on the pre-configured cycle along $v_1 \rightarrow v_4 \rightarrow v_5 \rightarrow v_6$ upon the on-cycle link failure of (v_1, v_2).

(b) $1+1$ protection: The same copy of user data A is sent along two disjoint paths between source node v_9 and destination node v_6.

Figure 2.3: Routing based single link failure resilient protection approaches [J1]

in Fig. 2.3a). The p-cycles are behaving in the same way as ring based protection for working paths traversing on-cycle links; but they offer more than that, i.e., p-cycles can also be used for restoration of working paths that are not on the p-cycle, but traversing a „straddling" link of the p-cycle. Owing to the mesh property of the topology capacity efficiency was improved compared to ring networks. We note that p-cycles do not rely on control plane signaling, but are still *not able* to provide instantaneous recovery (t_R is still in the range of $40-70$ ms) owing to the slow t_d switching matrix configuration time and to the non-zero failure localization and failure correlation times.

Several shared protection approaches were introduced to improve the resource efficiency of ring based protection approaches. The most resource efficient is the so called Failure Dependent Protection (FDP) [31]. The FDP protects a working path with multiple end-to-end protection paths. These protection paths are typically only partially disjoint from the working paths, and each of these protection paths corresponds to a specific failure event. After the given failure is identified the corresponding protection path is activated. By Shared Segment Protection (SSP) [37] a series of protection domains is established. Each domain contains a working and protection segment. When a working segment fails the corresponding protection segment is activated locally, reducing the recovery time. However, because of the long failure localization, notification and correlation time the shared protection approaches are not able to provide *robust and instantaneous recovery*. Note that these times can be reduced with an all-optical local failure localization scheme (see Section 2.3.5).

As the most widespread protection approach in mesh topologies $1+1$ protection (where data flow is sent parallel on the disjoint working and protection paths, shown in Figure 2.3b) provides sufficient resilience for the connections. The $1+1$ *provides robust and instantaneous recovery* with a single-ended switching at the destination node with $t_R \approx 20$ ms for the price of at least twice as much bandwidth consumption than the working connections. Although Suurballe's algorithm [84] provides a node-disjoint

(and therefore link-disjoint) solution for $1+1$ in the single link and node failure case in polynomial-time, it becomes NP-complete when multiple failures are considered [26]. Despite the fact that $1+1$ *provides robust and instantaneous recovery* it is not a long term solution, because of its bandwidth inefficiency and since $1+1$ optimizes for one cost parameter only (e.g. bandwidth or delay) i.e. $1+1$ can not satisfy multiple constraints as desired by *new, stricter QoS requirements*.

2.3.3 Protection Approaches with Coding in Optical Networks

The idea of in-network modification of user data (i.e., *network coding, NC*) was first introduced in the seminal paper by Ahlswede et al. [4]. Among the several benefits which network coding can provide in wireless networks, capacity efficiency and robustness are the two which can bring benefits to transport networks as well.

Inter-session Network Coding Based Protection Approaches

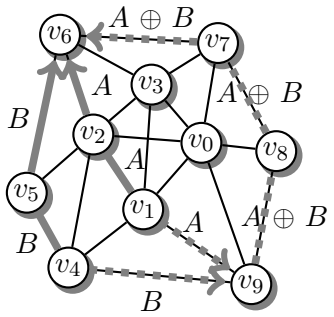
The application of NC for providing protection in transport networks first made an impact on the static traffic scenario [44], where we aim to minimize the capacity consumption for a matrix of traffic demands (i.e., *static routing* using inter-session network coding). In $1+N$ protection the source node performs linear combination operation on the input symbols (inter-session network coding) of the N working paths which is sent along the single protection path (requires $1+N$-connectivity of the topology, which is rarely the case). Thus, the sent data can be recovered at the destination from arbitrary N paths, i.e., the connection survives any single link failure. As neither flow rerouting, nor packet retransmission is required (only the destination node has to change its decoding method), $1+N$ protection clearly satisfies instantaneous recovery. This idea was further generalized into network coding based protection (NC1PP) [12], where inter-session coding of two $1+1$ protected connection is allowed at a single intermediate node (shown in Fig. 2.4a). Furthermore, the high connectivity requirement of $1+N$ protection was relaxed in [17] without highly impairing the capacity efficiency.

In [7] instantaneous recovery is provided with network protection codes by time-sharing the working and protection capacity of different connections to protect each other, which leads to a complex problem. Furthermore, decoding is performed on an additional coding (tree) infrastructure, which introduces additional capacity and complexity in the protection.

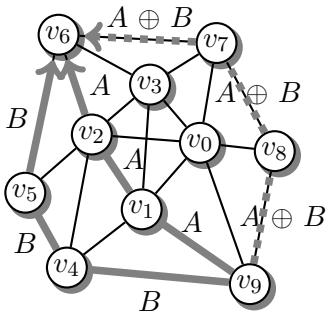
However, knowing the entire traffic matrix in advance is quite a strict requirement in today's dynamic networks. Hence intra-session network coding gained on interest and importance.

Intra-session Network Coding Based Protection Approaches

Intra-session network coding is suited for the dynamic traffic scenario i.e. where the traffic demands are arriving dynamically (sequentially) and the traffic demands are - and have to be - individually routed and protected. This means that no prior knowledge of incoming requests is assumed.



(a) Inter session network coding based protection: The data of connections from v_4 to v_6 and from v_1 to v_6 is sent directly to the common destination node v_6 and to a coding node (v_9) as well.



(b) Intra session - Diversity coding: The user data is split into two parts (A and B) and sent along two disjoint paths between source node v_9 and destination node v_6. Redundancy is added through a third disjoint path carrying $A \oplus B$.

Figure 2.4: Coding based single link failure resilient protection approaches [J1]

The most known protection approach applying intra-session network coding is the *diversity coding* (DC) [10, 66]. By DC redundancy is added at the source by sending a coded data packet ($A \oplus B$) on a third disjoint path constructed from the data of the two working paths (A and B, respectively) belonging to the same connection, presented in Fig. 2.4b. For the single link failure scenario an optimal algorithm for near-hitless network restoration was introduced, and for large arbitrary networks a new integer linear program based non-systematic coding approach and simple design algorithm was presented in [8]. However, also DC has high connectivity requirements, i.e. requires an at least 3-connected network, which is rarely the case for today's transport networks.

2.3.4 Advanced Protection Approaches in Optical Networks: General Dedicated Protection

The General Dedicated Protection (GDP) [J1] framework was introduced for the dynamic routing and includes routing and network coding based protection approaches, too. With network coding the GDP (Section 2.3.4) is able to ensure instantaneous recovery with the minimum resource allocation possible for an arbitrary failures scenario, and is a *theoretical lower bound* for all dedicated protection approaches ensuring instantaneous recovery. However, this method can not be implemented in practice since the GDP with network coding (i.e. *Lower bound* algorithm) is very complex from the network equipment and management point of view, and dealing with arbitrary number of subflows is not possible in nowadays networks.

Note that to resolve the practical implementation issue, in Chapter 3 I introduce a practical version of GDP, in which the flows are divided into exactly two parts [J1].

General Dedicated Protection with Network Coding

Although the *Lower bound* algorithm possibly cannot be implemented in practice owing to the arbitrary many subflows the connection is divided into, it provides us the *theoretical lower bound for resilient capacity allocation among methods providing instantaneous recovery*. This will be used as an indicator of resource efficiency in the rest of the dissertation. Note that as detailed in Section 2.4.1 it is enough to solve the *survivable routing* problem, since if a survivable routing is given a robust configuration for network coding can be found in polynomial time with approaches proposed in [38, 41, 50].

In the following, a linear program is presented which delivers the theoretical *lower bound* in polynomial-time. The goal is to obtain a solution $x \in \mathcal{X}_{\mathcal{I}}$ which minimizes the bandwidth cost in Equation (2.2) while instantaneous recovery is maintained. The notations used are summarized in Table 3.1.

$$\min \sum_{e \in E} c(e) \cdot b(e). \tag{2.2}$$

The following constraints are required:

$$\forall f \in \mathcal{F},\ \forall e \in E\colon 0 \leq b_f(e) \leq \min\{b, k(e)\}, \tag{2.3}$$

$$\forall f \in \mathcal{F}, \forall i \in V\colon$$

$$\sum_{(i,j) \in E_f} b_{(i,j),f} - \sum_{(j,i) \in E_f} b_{(j,i),f} = \begin{cases} -b & , \quad \text{if } i = s \\ b & , \quad \text{if } i = d \\ 0 & , \quad \text{otherwise} \end{cases}, \tag{2.4}$$

$$\forall f \in \mathcal{F},\ \forall e \in E\colon b_f(e) \leq b(e). \tag{2.5}$$

Constraints (2.3) set the range of the variables. The flow conservation constraints for each failure graph ($G_f = (V, E_f)$) is formulated in Equation (2.4). The failure graph $G_f = (V, E_f)$ is obtained by removing the failed edges in $f \in \mathcal{F}$ from E: $E_f = E \setminus f$. Constraint (2.5) sets the capacity in the solution according to the free capacity in the failure graphs.

The resilient solution of the LP above minimizes the consumed bandwidth while being resilient against all possible failure patterns in $\mathcal{F}$, as b units of flow eventually reach the destination in each failure graph. To solve the LPs and ILPs I have used the commercially available solvers GUROBI [34] and CPLEX [1].

General Dedicated Protection with Routing (GDP-R)

It is well known, that the general versions of the resilient capacity allocation problems with integrability constraint on the flow values were proven to be NP-hard for failure patterns $\mathcal{F}$ containing all link sets with up to k links [19,20,22]. In [19,22] it was shown that $1+1$ protection approach is a 2-approximation algorithm for the resilient capacity allocation problem (against single link failures). However, for general failure patterns a disjoint path-pair for $1 + 1$ may not exist, and a more general protection structure is

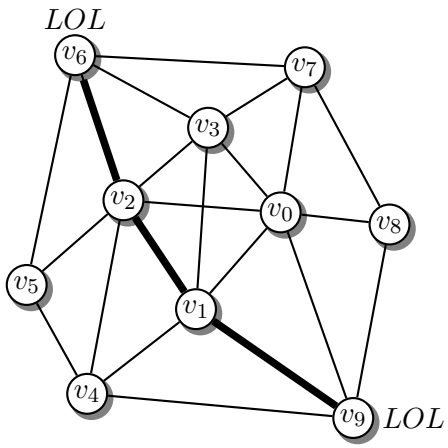


Figure 2.5: The illustration of m-trails.

desired. This is why the so called GDP-R was introduced. This approach is enabling instantaneous failure recovery against general failure patterns listed in $\mathcal{F}$, while providing extremely high connection availability even in sparse network topologies for the price of high computational complexity.

Note that by GDP-R only routing is allowed, i.e., in this case on all links $(v, u) \in E$ with $b(e) > 0$ the data of one of v's incoming interfaces is simply routed in the resilient subgraph. This means we restrict the GDP-R resilience problem for $b(e) = \{0, b\}$ values, i.e., the user data cannot be divided into smaller subflows as by network coding.

Furthermore, in [J1] it was shown that finding an optimal (non-bifurcated) GDP-R solution in terms of bandwidth cost, is NP-complete. In the proof the polynomial time transformation of GDP-R to the Steiner Forest problem (which is proved to be NP-hard [51]) was presented. The detailed proof can be found in [J1]. The method itself also can be used as a baseline to compare our methods, to analyze what is achievable without network coding.

2.3.5 Failure Localization Approaches in Optical Networks

To enable fast restoration for the shared protection approaches fast *failure localization* is crucial. To implement fast *failure localization* in all-optical networks, supervisory lightpaths (S-LP, monitoring trails or m-trails) have been introduced [6, 13, 35, 88, 89, 91]. An m-trail consists of a pair of lightpaths along a common physical route (in opposite directions), and is purely used for monitoring the on/off status of the physical links along the route at the end-nodes. Furthermore, if each traversed node is able to monitor the status of a carefully allocated set of m-trails via lambda monitoring (LOL - Loss of Light detection), *failure notification* time could be reduced to minimal as well, because each node can perform protection switching immediately without waiting for failure notification messages, purely based on local observations. An example is shown in Figure 2.5 with an m-trail between nodes v_9 and v_6. The LOL of this m-trail either can be monitored only at the endpoints i.e. v_9 and v_6 or at all the intermediate nodes

(v_1 and v_2), too (depending on the technology). The status information of the S-LPs either can be utilized locally, by local failure localization schemes or has to be sent to the central failure manager for further processing.

First centralized all-optical m-trail allocation schemes have been introduced [13, 90, 91], where a central failure manager is responsible for unambiguous failure localization based on the alarm messages of the (small set of) disrupted m-trails. Later, the concept of Local Unambiguous Failure Localization (L-UFL) was introduced [6], where the goal was to completely eliminate signaling from the failure localization phase with a properly allocated set of m-trails at a distinguished node of the network where these S-LPs terminate, called *L-UFL capable* node. However, the L-UFL node (or the failure manager) is still needed to notify the switching nodes about the failure with electrical layer messages.

The goal in Network-Wide Local Unambiguous Failure Localization (NWL-UFL) [88] is to enable all-optical signaling-free failure restoration (i.e., eliminate messages in both failure localization and failure notification phases) by making all nodes in the network L-UFL capable under any single link failure. After the failure is localized at each node of the network, the corresponding switching actions can be performed immediately. However, the lengthy m-trails (spanning trees [88] to be specific) cause implementation issues and higher monitoring latencies. In addition, we unambiguously identify all the link failures, although not all link failures have to be unambiguously localized at each node, e.g., if that particular node does not need to perform protection switching action for a failure.

In order to improve these drawbacks of NWL-UFL, Global Neighborhood Failure Localization (G-NFL) has been proposed in [86, 89]. The main idea is to unambiguously localize only a small set of link failures that particular node needs to respond to in the restoration process via short m-trails. These identifiable links are referred to as the *neighborhood* of the node. Hence, G-NFL is currently the most efficient signaling-free unambiguous link failure localization framework, as it allows the *shortest possible W-LPs and P-LPs to be used* in the data plane, as well as short S-LPs (simple or non-simple paths) to be allocated in the control plane. Thus, upon request an S-LP can be turned into a W-LP, while the status of W-LPs can be used for failure monitoring purposes as well.

2.3.6 Protection and Failure Localization Approaches in SDN Networks

By SDN the control and data plane are decoupled. The switches are directly programmed from one or more controllers. In other words SDN both abstracts the infrastructure for applications and network services, and makes the network control fully and directly programmable. Nonetheless optical and SDN networks can be modeled similarly since both can be considered as flow based networks.

In [77, 78] the fast failure recovery in SDN networks was investigated. It was shown that with restoration instantaneous recovery can not be achieved, however with $1 + 1$ protection (not taking into account the propagation delay [76]) the recovery time can be held under 50 ms. Besides the traditional routing problem by SDN the so called, Controller Placement problem arises. In [93] the problem of Reliable Controller Placement (RCP) problem was addressed i.e. how to protect the control plane against

single link and node failures by exploiting the principles of resilient routing.

For failure localization purposes many controller take advantage of the already existing control plane protocols such as Link Layer Discovery Protocol (LLDP) [67]. However, in [55] a new efficient m-trail based solutions for topology verification and failure localization in SDN networks was introduced. The methods minimize the rule number necessary for topology verification and failure localization.

Since SDN networks naturally have a centralized manager (controller), centralized all-optical m-trail allocation schemes [13, 91] could be easily adjusted for the implementation in SDN networks.

2.4 Research Goals and General Assumptions

The objective of this dissertation is to address the new strict QoS (high reliability, low latency) requirements, which rose due to the proliferation of new applications (e.g. VoIP, 4K/8K video traffic, AR/VR (Augmented and Virtual Reality etc.)) which not only require high resilience from the underlying network but are also highly delay sensitive [40]. In particular, in the first part of my dissertation I propose new SDN and all-optical network compatible methods in order to ensure *robust and instantaneous recovery* in transport networks with delay requirements even after the failure occurs and all that in a capacity efficient manner. In the second part of the dissertation a new framework for local failure localization is provided which is essential for the ultra fast (or even instantaneous) recovery for shared protection approaches. Nonetheless, due to the nature of the shared protections optical cross-connect (OXC) reconfiguration is always needed. This means with shared protections *robust recovery can not be achieved* i.e. the simplicity level of $1 + 1$ can not be maintained.

In details:

– In Chapter 3 new dedicated protection approaches with network coding are presented, for the arbitrary and single link failure scenario which provide robust and instantaneous recovery, namely the Survivable Routing with Network Coding (SRNC) and Survivable Routing with Diversity Coding (SRDC), respectively. The user data is split into exactly two parts keeping the protection approaches simple from equipment and management point of view. By SRDC the coding itself can be done in the source and destination nodes (i.e. there is no need for a node capability update) making the method easily deployable. Several subproblems are investigated depending on capacity and node capability constraints.

– In Chapter 4 a new dedicated single link resilient protection approach with network coding is presented which is able to satisfy delay constraints even after the failure occurs. Both delay related problems i.e. the Quality-of-Service (QoS) routing [64] and Differential Delay (DD) aware [39] survivable routing are investigated.

– In Chapter 5 a new framework for local failure localization is provided which is essential for the ultra fast (or even instantaneous) recovery for shared protection approaches. This new framework

called Advanced Global Neighborhood Failure Localization (AG-NFL) focuses on localizing the proper *protection switching actions* rather than on unambiguous link failure localization, as its previous conterparts. We identify the set of links whose failure can be localized together, called *switching link groups*, and demonstrate that localizing link groups instead of single link failures leads to a significantly improved performance.

The main results are analytical but the effectiveness of the methods is validated through simulation results as well.

2.4.1 General Notation

In this section we summarize the main general and operational remarks and assumptions made in this dissertation. First the remarks regarding the node capabilities and the routing solution are presented, then the general assumptions are introduced.

Node Capability Remarks

(a) Forwarding (b) Merging (c) Splitting (d) Coding

Figure 2.6: The basic node capabilities i.e. node roles. Each node v in the networks can be classified into one of this roles.

Each node v in the network can be classified into one of four roles shown in Figure 2.6. These are the basic node capabilities i.e. forwarding, merging, splitting, and network coding.

In particular the splitter and merger nodes as defined as following. Let $\delta^-(v)$ and $\delta^+(v)$ denote the in-degree and the out-degree of a node, respectively.

Definition 2.4.1 *Node $p \in V$ is called **splitter**, if $\delta^-(p) = 1$ and $\delta^+(p) = 2$, i.e., it receives data on a single edge, while forwards the same copy on two outgoing edges. The set of available splitters is denoted as $\mathcal{P} \subseteq V$.*

Definition 2.4.2 *Node $m \in V$ is called **merger**, if $\delta^-(m) = 2$ and $\delta^+(m) = 1$, i.e., it receives* the same data *on two incoming edges, while forwards one of them (or upon failure the intact one) on its single outgoing edge. The set of available mergers is denoted as $\mathcal{M} \subseteq V$.*

We assume that each node in each scenario (single and arbitrary link failure) is capable of forwarding, merging and splitting the incoming data. In Section 6.1.1 a simple implementation of these node

capabilities is shown and discussed in SDN networks. When the single link failures scenario is considered these three node capabilities are sufficient (i.e. Figure 2.6 (a)-(c)). In *network coding* [4] (i.e. Figure 2.6 (d)), instead of simple forwarding [38], the switches can perform algebraic operations on their incoming packets to construct the outgoing encoded packet. Network coding in intermediate nodes is only necessary when a general failure model is assumed i.e. an arbitrary SRLG list is the input.

The Routing Solution

The solution of all methods with network coding consists of two parts.

- The *survivable routing* $R = (V^R, E^R, f)$ which contains the nodes and the edges with the given flow values of the solution.
- And the *robust configuration* $\mathcal{C}$, which includes all the information needed for the OXC configuration, for example the switching matrix settings.

When using network coding, the coded data packet from the data packets of the incoming edges of node v have to be constructed. For this purpose several state-of-the art coding strategies can be applied, e.g., either random network codes proposed in [38] or deterministic network codes presented in [41]. In these approaches it is assumed that zeros are received along the failed links after a failure occurs [50]. Note that these codes are robust, i.e., after an arbitrary failure occurs, the coding operations remain unchanged (only the destination node needs different decoding operation to produce the original packets). This means if *survivable routing* $R = (V^R, E^R, f)$ is given we can construct the *robust configuration* $\mathcal{C}$ for the network coding. With this robust coding approaches the network can be configured accordingly (robust) in polynomial time in order to ensure instantaneous recovery. Thus, in the rest of the dissertation I am only interested in the properties of the allocation of the optimal *survivable routing* $R = (V^R, E^R, f)$ of a connection to address the capacity efficiency issue of the dedicated protection approaches ensuring instantaneous recovery.

General Assumptions

Without a loss of generality we assume that:

- The availability of nodes is 1 i.e. the nodes are fully reliable. As already discussed in Section 2.2.2 the node failures can be modeled by a single link failure in an auxiliary graph.
- The SRLG lists $\mathcal{F}$ are static i.e. the SRLG lists are not changing dynamically over time.
- All connections are bidirectional. Hence the network can be modeled with an undirected graph $G = (V, E)$.

For the m-trail (S-LP) design the following additional assumptions are used:

– The number of wavelengths is not a limiting factor for the failure localization method i.e. there are always enough wavelength channels for the failure localization.

– The S-LP design problem utilizes the (by the shared protection approach) pre-calculated W-LP and P-LP sets as input.

Chapter 3

Survivable Routing in Transport Networks

Several shared protection approaches were introduced to improve the resource efficiency of ring based protection approaches. The most resource efficient is the so called Failure Dependent Protection (FDP) [31]. The FDP protects a working path with multiple end-to-end protection paths. These protection paths are typically only partially disjoint from the working paths, and each of these protection paths corresponds to a specific failure event. After the given failure is identified the corresponding protection path is activated. By Shared Segment Protection (SSP) [37] a series of protection domains is established. Each domain contains a working and protection segment. When a working segment fails the corresponding protection segment is activated locally, reducing the recovery time. However, because of the long failure localization, notification and correlation time the shared protection approaches are not able to provide *robust and instantaneous recovery*. Note that these times can be reduced with an all-optical local failure localization scheme (see Section 2.3.5).

As the most widespread protection approach in mesh topologies $1 + 1$ protection (where data flow is sent parallel on the disjoint working and protection paths, shown in Figure 2.3b) provides sufficient resilience for the connections. The $1+1$ *provides robust and instantaneous recovery* with a single-ended switching at the destination node with $t_R \approx 20$ ms for the price of at least twice as much bandwidth consumption than the working connections. Although Suurballe's algorithm [84] provides a node-disjoint (and therefore link-disjoint) solution for $1+1$ in the single link and node failure case in polynomial-time, it becomes NP-complete when multiple failures are considered [26]. Despite the fact that $1 + 1$ *provides robust and instantaneous recovery* it is not a long term solution, because of its bandwidth inefficiency and since $1 + 1$ optimizes for one cost parameter only (e.g. bandwidth or delay) i.e. $1 + 1$ can not satisfy multiple constraints as desired by *new, stricter QoS requirements*.

In this chapter new network coding and survivable routing based protection approaches are introduced which ensure *robust and instantaneous recovery*. The minimum cost survivable routing optimization problem has been investigated for decades in the literature, and it was shown that finding the optimal survivable routing for multiple edge failures is NP-complete problem [19,26]. However, in today's transport networks the single edge failure is the most relevant failure scenario [61], while dividing user data

into more than two parts is impractical from an operational point of view i.e. leads to practically non-deployable protection methods. Surprisingly, the complexity of this practically relevant special case is an open question. This restriction is motivated by the fact that the minimum cost routing solution in most real word networks can be reached by dividing the input flow into 2 subflows [79].

I investigate the survivable routing problem for both the arbitrary and single link failure scenarios where the user data is split into exactly two parts keeping the protection approaches simple from equipment and management point of view. In particular, in Section 3.1 a new practical (deployable), yet extremely resource efficient method called Survivable Routing with Network Coding (SRNC) is introduced for the arbitrary failure scenario. The method utilizes network coding - like the theoretical lower bound - to ensure resource efficiency. Since the user data is split into exactly two parts the the SRNC can be deployed in today's networks contrary to the *Lower bound*. Nonetheless, the network coding capability has to be added in each node and the method is NP-complete.

In Section 3.2 a very practical scenario is investigated i.e. the single link failure scenario. A new easily deployable and extremely resource efficient method called Survivable Routing with Diversity Coding (SRDC) is introduced. The method is easily deployable since the *coding itself can be done in the source and destination nodes, i.e. there is no need for a node capability update* (contrary to the SRNC). Several subproblems of SRDC are investigated depending on capacity and node capability constraints in the network.

The main notations used in the Survivable Routing part of the dissertation i.e. Chapter 3 and 4 are summarized in Table 3.1.

3.1 Survivable Routing with Network Coding (SRNC)

In this section we introduce the so called Survivable Routing with Network Coding (SRNC) that ensures *robust and instantaneous recovery* for the arbitrary failure scenario in a simple manner. The method has a capacity efficiency close to the theoretical lower bound. However, is much more suitable for the practical implementation, since we divide the user data into exactly two parts and not into arbitrary many, keeping the protection approach simple and deployable.

3.1.1 Problem Formulation

The SRNC problem is formulated as following. Given is the input $\mathcal{I} = \{G, C, \mathcal{F}\}$ instance of the SRNC problem, where:

– Each connection is routed promptly after the request arrives to the edge router i.e. dynamic routing is assumed. In that time instant the actual topology of the network is denoted by $G = (V, E)$. On each edge $e \in E$ a non-negative cost function ($c : E \to R^+$) is defined, and the free capacity ($k : E \to R^+$) is given.

Table 3.1: Notation list for SRNC and SRDC

Notations	Description
$G = (V, E, k, c, d)$	the undirected graph representation V is the node and E is the edge set c is the cost function defined on $e \in E$ k is free capacity along link $e \in E$ d is the delay function defined on $e \in E$
$C = (s, t, b, D)$	source, destination and requested bandwidth and the delay bound of the dynamically arrived traffic demand, respectively
$\mathcal{F}$	failure patterns the connection needs to be resilient against
$G_f = (V, E_f)$	failure graph obtained by removing the failed edges in $f \in \mathcal{F}$ from E: $E_f = E \setminus f$
$R = (V^R, E^R, f)$	the survivable routing for the connection where $V^R \subseteq V$ is the set of nodes , $E^R \subseteq E$ is the set of edges, and $\forall e \in E^R : f(e) \leq k(e)$ are the flow values
$\mathcal{C}$	the robust configuration
$G^* = (V, E^*, c)$	directed multi-graph with edge set E^*, where all edge in $G = (V, E, k, c)$ are replaced by $k(e)$ parallel edges each with cost $c(e)$
$R^* = (V^{R^*}, E^{R^*})$	survivable routing of a connection in $G^* = (V, E^*, c)$, which is a DAG
A, B	the data parts after dividing the user flow
$A \oplus B$	the redundancy data part added with the XOR operation in s
$E_A, E_B, E_{A \oplus B}$	routing DAGs A, B and for $A \oplus B$

– A traffic demand $C = (s, d, b)$ consists of the source node s and sink node d, and the bandwidth requirement ($b \in \mathbb{N}$).

– Finally, the set of failure patterns, denoted as $\mathcal{F}$, is given. For each failure pattern, a failure subgraph is created: $\forall f \in \mathcal{F} : G_f = (V, E_f)$, where E_f is obtained by removing the edges in f from E. We assume that each failure pattern in $\mathcal{F}$ is *protectable*, i.e., each G_f is $s - d$ connected.

– Note that by SRNC no delay values are considered i.e. no delay function or delay bound is considered.

The task is to find a survivable routing $R = (V^R, E^R, f)$ and robust configuration $\mathcal{C}$ with minimal cost, in such way that network coding is allowed, but the user data is split into exactly two parts. Survivable routing is defined as following:

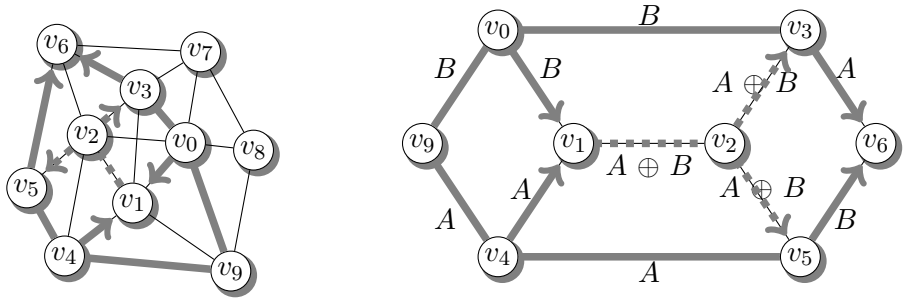
Definition 3.1.1 *[C2] We say that $R = (V^R, E^R, f)$ is a **survivable routing** of a connection $D = (s, t, d)$ in G (where $V^R \subseteq V$, $E^R \subseteq E$, and $\forall e \in E^R : f(e) \leq k(e)$), if there is an $s - t$ flow of value $F \geq d$ in R, even if we delete any single edge of R. On the other hand, a routing is **vulnerable** if it is not survivable.*

Let the set $\mathcal{X}_\mathcal{I}$ contain a set of feasible solutions (referred to as *survivable routing*) $y_\mathcal{I}$ for the problem instance $\mathcal{I}$ satisfying instantaneous recovery. Each resilient subgraph actually contains the reserved bandwidth on the links $\forall e \in E : 0 \leq b(e) \leq b$, where the maximal flow values on $b(e)$ capacities are $\leq b$ in each $G_f = (V, E_f)$. An example of a resilient subgraph is presented in Fig. 3.1, for the failure pattern $\mathcal{F} = ((v_0, v_3), (v_1, v_2), (v_4, v_5))$ i.e. if any of the three middle links can fail. Furthermore, according to the reserved bandwidth the robust configuration $\mathcal{C}$ i.e. the switching matrix configurations can be obtained and appropriate operations can be configured (e.g., how to construct the network code, i.e., the outgoing packet from the incoming packets at node v).

Note that the configuration is robust i.e. the switching matrices (and network codes) are configured only at connection setup accordingly, and this configuration remains unchanged even when an arbitrary failure $f \in \mathcal{F}$ occurs, ensuring robust and instantaneous recovery. Such a robust configuration is presented in Figure 3.1b. Note that the splitting constraint has to be taken into account at any time, i.e. we allow the user data to be split into exactly two parts in order to maintain simplicity.

Note that this problem formulation can be used as a problem formulation for the *Lower bound* if we remove the splitting constraint i.e. if user data can be split into arbitrary many parts and not only into two. Also it can be used for the GDP-R [J1] problem if no network coding is allowed.

Of course by SRNC we have to construct the coded data packet from the data packets of the incoming edges of node v. For this purpose several state-of-the art coding strategies can be applied, e.g., either random network codes proposed in [38] or deterministic network codes presented in [41]. In these approaches it is assumed that zeros are received along the failed links after a failure occurs [50]. Note



(a) A resilient subgraph $y_{\mathcal{I}}$ for general dedicated protection with $b(e) = 1$ on the bold edges, and $b(e) = 0$, otherwise.

(b) A resilient configuration of the GDP solution $y_{\mathcal{I}}$. Without network coding a resilient configuration does not exist for this subgraph.

Figure 3.1: Given an instance $\mathcal{I}$ with the above topology $G = (V, E)$ and link costs $\forall e \in E' = \{(v_9, v_i) \cup (v_j, v_6)\} : c(e) = 3, \forall e \in E \setminus E' : c(e) = 1$. The connection request is $\mathcal{D} = (v_9, v_6, 2)$. The goal is to establish a survivable routing considering the failure pattern $\mathcal{F} = ((v_0, v_3), (v_1, v_2), (v_4, v_5))$. [J1]

that these codes are robust, i.e., after an arbitrary failure occurs, the coding operations remain unchanged (only the destination node needs different decoding operation to produce the original packets).

When single link failures scenario is considered (Section 3.2) the node capabilities represented in Figure 2.6 (a)-(c) are enough [16, 74]. However, if an arbitrary failure scenario is assumed the network coding in intermediate nodes is necessary i.e. by SRNC we can not avoid updating the node capabilities. Nonetheless, in Section 6.1.1 a simple implementation of these node capabilities in SDN networks is shown and discussed.

Note that with these robust state-of-the art coding approaches the network can be configured accordingly in polynomial time i.e. the robust configuration $\mathcal{C}$ can be obtained in polynomial time in order to ensure robust and instantaneous recovery. Thus, in the rest of the dissertation I am interested in the properties of the survivable routing allocation problem, i.e. finding the optimal *survivable routing* $R = (V^R, E^R, f)$ for the given connection, in order to address the capacity efficiency issue of the dedicated protection approaches providing robust and instantaneous recovery.

3.1.2 Computational Complexity of SRNC

SRNC is NP-complete

In the following, we show that in contrast to the polynomial-time complexity of the theoretical *Lower bound* algorithm offered by network coding, finding an optimal SRNC solution in terms of minimizing the bandwidth usage assuming that the flow can be divided into two (or more generally into a finite integer T) parts is NP-complete. In order to prove the NP-completeness of the SRNC we present a polynomial

time transformation of the NP-complete [15] GDP-R to SRNC. Previously the NP-completeness of GDP-R was proven with the polynomial time transformation of the Steiner Forest problem (which is proven to be NP-hard [29]).

Theorem 3.1.2 *To decide whether a solution with $\leq C$ cost exists for the SRNC problem is NP-complete.*

Proof: The SRNC problem is in NP, a solution with $\leq C$ cost is a proof.

Assume we are given an instance $\mathcal{I} = \{G, C, \mathcal{F}\}$ of the GDP-R problem, that is, an undirected graph $G = (V, E)$, edge costs $c : E \to \mathbb{R}^+$, free capacity $k : E \to \{0,1\}$, the traffic demand $C = (s, d,1)$, and the list of failure patterns $\mathcal{F}$ which the resilient subgraph needs to survive.

The polynomial time transformation is given as follows. We add a new edge $e = (s, d)$ with $c(e) = 0$, $k(e) = 1$ to the GDP-R graph, and define the new input of the SRNC problem as $G_2 = (V, E \cup e)$, $C_2 = (s, d,2)$, and $\mathcal{F}_2 = \mathcal{F}$.

Indirectly, we assume that the resulting SRNC problem is solvable in polynomial time. In an SRNC solution, unit capacity is sent from s to d on link e (which is reliable as it is not in $\mathcal{F}$), and a unit capacity is sent through $G = (V, E)$, which is resilient against all failures in $\mathcal{F}$, i.e., it is a solution for the GDP-R problem in G. Furthermore, the SRNC and GDP-R solutions have the same cost, i.e., a solution with cost $\leq C$ exists for both problem instances at the same time. As for the transformation, we have solved the GDP-R problem in polynomial time, which is a contradiction unless P = NP. ■

Note that the proof can be easily generalized to T subflows, using traffic demand $C_T = (s, d, T)$ and adding $T - 1$ new edges during the polynomial time transformation.

3.1.3 Optimal Survivable Routing for SRNC

As the SRNC problem is NP-complete, in the following the corresponding ILP is presented. In SRNC our goal is to obtain a solution $x \in \mathcal{X}_\mathcal{I}$ which minimizes the bandwidth cost

$$\min \sum_{\forall e \in E} c(e) \cdot \frac{b(e)}{b \cdot 2}. \tag{3.1}$$

The following constraints are required (using the same assumptions as in the ILP presented in Section 2.3.4):

$$\forall f \in \mathcal{F},\ \forall e \in E\colon 0 \leq b_f(e) \leq 2 \cdot \min\{b, k(e)\}, \tag{3.2}$$

$$\forall f \in \mathcal{F}, \forall i \in V\colon$$

$$\sum_{(i,j)\in E_f} b_{(i,j),f} - \sum_{(j,i)\in E_f} b_{(j,i),f} = \begin{cases} -2 & \text{, if } i = s \\ 2 & \text{, if } i = d \\ 0 & \text{, otherwise} \end{cases}, \tag{3.3}$$

$$\forall f \in \mathcal{F},\ \forall e \in E\colon b_f(e) \leq b(e), \tag{3.4}$$

$$\forall f \in \mathcal{F},\ \forall e \in E\colon b_f(e) \text{ integer variables.} \tag{3.5}$$

Constraint (3.2) sets the maximal flow value based on the available capacities in each failure subgraph. The constraints in Equation (3.3) formulates the flow conservation in each failure subgraph G_f, and ensures that the connection operates when an arbitrary f failure occurs from $\mathcal{F}$. Finally, Constraint (3.4) sets the flow value in the solution, i.e., if an edge was used in any failure subgraph G_f, we have to include it in the final solution to be resilient against all failures in $\mathcal{F}$. Finally, Constraint (3.5) sets the integer constraint for the flow values.

Note that in the special case when $\mathcal{F}$ contains only single link failures, the simple XOR coding approaches in [74] and [16] can be applied to obtain the robust network codes for the (optimal) survivable routing, which ensures robust and instantaneous recovery. On the other hand, to obtain robust network codes for general (arbitrary) failure patterns, more complex coding operations [38, 41] are required to ensure robust and instantaneous recovery.

3.1.4 Experimental Results

In this section we present a comprehensive bandwidth efficiency and blocking probability analysis. We compare the SRNC to the *Lower bound* and to the state-of-the-art protection approaches i.e. DC and $1 + 1$. Since the $1 + 1$ protection method is the 2-approximation of the resilient capacity allocation problem of GDP-R (when single link failures are considered), we will use it as a benchmark in the simulations.

The data provided in the figures is obtained by averaging the results of 3 different traffic patterns each containing $|\mathcal{D}| = 1000$ traffic demands between randomly chosen s and d pairs with $b = 1$ capacity. The performance is investigated on real world and randomly generated network topologies, too. The following failure patterns are selected, since these are the statistically most probable ones:

– all single link failures (denoted as $p = 0$ *adjacent dual failure density* on the charts),

– all single link and $p\%$ of adjacent dual link failures ($p = 10$,50,90),

– all single link and all adjacent dual link failures ($p = 100$).

For the random topology test we used several random 2-connected planar graphs $G = (V, E)$ as input, which were generated using the graph generator of LEMON [2].

Note that the SRNC is designed to protect any arbitrary failure pattern. The single link failure scenario is only investigated to give a comprehensive picture for all scenarios. If only single link failures are considered the method introduced in Section 3.2 called SRDC should be preferred, since it does not require in-network coding (i.e. a simple XOR ($GF(2)$) coding is enough in the source and destination node).

Blocking Probability Analysis

In this section we demonstrate the advantage of the general protection structure of SRNC approach compared to the rigid structure of DC and the $1+1$ implementations. Since with general failure patterns $\mathcal{F}$ a failure disjoint path-pair may not exist and real network topologies often do not have high connectivity the blocking probability is a real issue for the DC and $1+1$. On the other hand, SRNC can protect arbitrary failure patterns given in $\mathcal{F}$ (if it is protectable at all).

To analyze the blocking probability properly as a function of the adjacent dual failure density, the link capacities were set high, so that blocking of a connection can not occur because of the absence of free capacity (just only because the network topology is not able to support the protection structure of the given protection approach). Furthermore we used several real world ASs from Rocketfuel [80] as input topologies. *Lower bound* denotes the optimal capacity consumption among all protection approaches providing instantaneous recovery, discussed in Section 2.3.4. $1+1$ corresponds to the traditional dedicated path protection approach, where two failure-disjoint paths are reserved, i.e., any failure in $\mathcal{F}$ disrupts at most one of the two paths (Section 2.3.2).

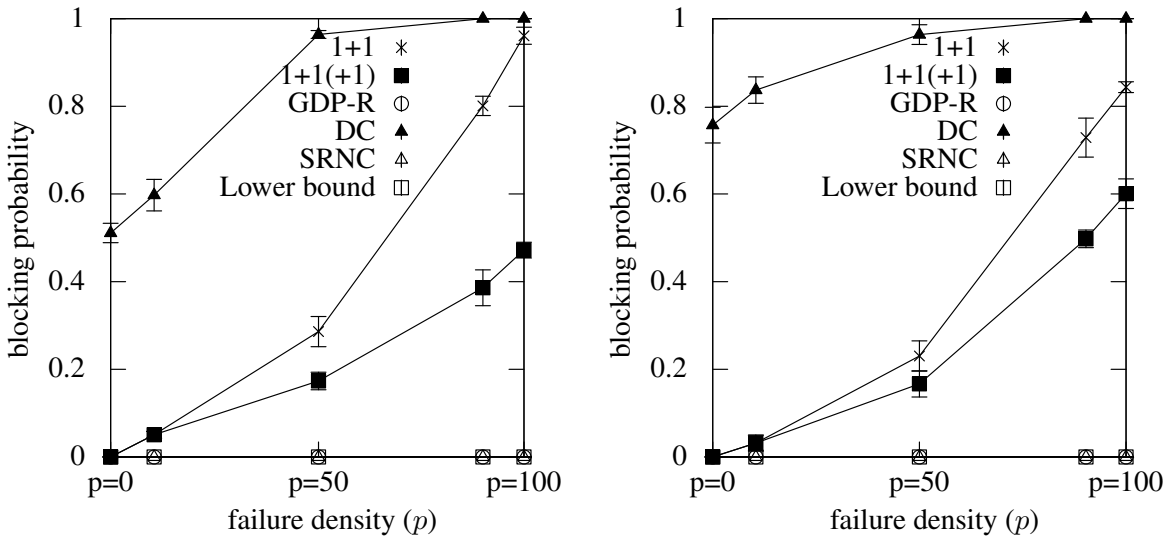
In order to provide a fair comparison of the traditional approaches to our general model, we implemented $1+1(+1)$ protection as well, which runs as follows. If the $1+1$ method fails in first step, it tries to find three link-disjoint paths in the topology (like DC). However, twice as much capacity is reserved on the disjoint paths i.e. the same data is sent along three different paths. If it succeeds, we send the same copy of data on all three paths, thus, this structure survives arbitrary dual link failures, including the ones in $\mathcal{F}$ [J1].

In Figure 3.2 we see that as the number of failure patterns (i.e., adjacent dual failure density) increases, the blocking probability of both DC and $1+1$ solutions increases dramatically, while the blocking probability of the SRNC solution stays zero. This meets our expectation, because SRNC is designed to protect all the failures listed in $\mathcal{F}$. In Figure 3.2 we can observe that the blocking probability of the $1+1(+1)$ implementation is much lower than that blocking probability of $1+1$. However, for the lower blocking probability we pay the price of worse bandwidth efficiency (i.e. the same data is sent along three different paths).

Bandwidth Efficiency Analysis

In this section we investigate the two statistically dominant failure patterns $\mathcal{F}$, i.e. the single link failures ($p=0$), and the one with 10% of adjacent dual link failures ($p=10$). To the dual link failures we refer as multiple failure on the chart.

In Figure 3.3 we present the bandwidth requirement of different protection schemes in random networks with different sizes. The average nodal degree of the investigated networks is around 3.2. In Figure 3.3a the average capacity is shown in the single link failure scenario. As expected, the required



(a) Blocking probability of Abovenet (17 nodes, 37 links, with diameter 4)

(b) Blocking probability of AT&T (22 nodes, 38 links, with diameter 5)

Figure 3.2: Blocking probability of real world networks from [80] [J1].

capacity for the connections grows as the network size increases. The restriction of the maximal division of data into two parts seems to be a stringent requirement, however the proposed SRNC scheme approaches the theoretical lower bound in all the investigated realistic scenarios.

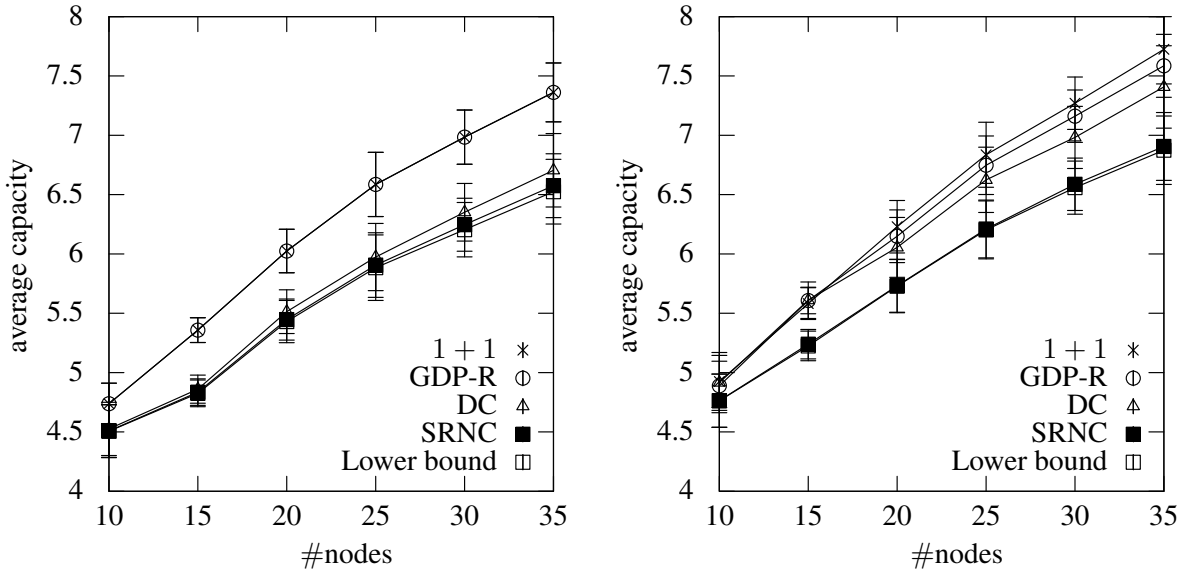
The advantage of network coding is more apparent in the multiple failure scenario, presented in Figure 3.3b. In Figure 3.3b even the coding based DC method uses *one additional link per connection* compered to the lower bound.

3.2 Survivable Routing with Diversity Coding (SRDC)

Single link failures are the most common failures in transport networks, more than 70% [61] of all failures affecting links in the backbone networks are single link failures. Therefore it is not a surprise that today's networks are either single link failure resilient i.e. the capacity inefficient $1+1$ dedicated protection is deployed, or they are unprotected (what is unacceptable for today's high QoS requirements).

As shown previously in Section 2.3.4 the optimal capacity efficiency can be achieved with network coding [4,41,50] while *robust and instantaneous recovery (i.e., $t_R < 50$ ms) is maintained*. Nonetheless, *the application of complex network coding operations sacrifices simplicity, leading to practically non-deployable protection methods*.

On the other hand, there are some simple special cases of coding which could satisfy all three requirements in transport networks, i.e. instantaneous recovery, low capacity requirements and simplicity e.g., *diversity coding* (DC) [9]. However, transport networks are rarely 3-connected which is a requirement for the DC. This is the main reason why the DC could not replace or suppress the $1+1$ dedicated



(a) Single link failures (degree 3.2)

(b) Multiple link failures (degree 3.2)

Figure 3.3: Average capacity reserved for a connection for different $\mathcal{F}$ failure scenarios and in networks $G = (V, E)$ with varying sizes [J1].

protection approach.

Recently, the papers [74] and [16] made important steps to remedy the connectivity problem of diversity coding, and enable it to be an alternative for $1 + 1$ in transport network protection. They formulate the survivable routing problem of $1+1$ and diversity coding more generally, while considering the same routing scenario: single edge failure resilience [61], while the connection data is divided into two parts at the source node. They provide polynomial-time network (and diversity) coding algorithms in a minimum cost survivable routing (called coding subgraph) i.e. *they provide the means for the robust configuration $\mathcal{C}$, if the survivable routing $R = (V^R, E^R, f)$* is given. However, the issue of finding a minimum cost survivable routing was not tackled and addressed in these works.

The last step towards the practical implementation is discussed in this section, namely how to fill the gap and solve the survivable routing problem in order to create a capacity efficient, simple and low-complexity diversity coding based survivable routing method providing robust and instantaneous recovery method, which could replace the $1 + 1$.

Although the general versions of minimum cost survivable routings are shown to be NP-complete [19, 26], surprisingly, the computational complexity of optimal capacity allocation for this practically relevant special case (i.e. two data parts and single link failure) is an open question. However, using the state-of-the-art results [16] it is known that a minimum cost survivable routing with diversity coding can be decomposed into three Directed Acyclic Graphs (DAG), *which preserves one of the most important features of* $1+1$*, i.e., simplicity*. This means if we find the minimum cost survivable routing we only need to perform the network coding in the source and destination nodes. There is no need for any algebraic

operation in the intermediate nodes, hence there is no need for node capability update. Armed with these facts we formulate the minimum cost survivable routing problem (without loss of generality) as finding three appropriate DAGs.

3.2.1 Problem Formulation

The SRDC is a survivable routing (defined and discussed in Section 3.1.1) and diversity coding based method. Note that if a survivable routing is given, after the edge failure is identified any routing method could be adopted to resend the flows on the intact edges of a survivable routing R, clearly resulting in slow recovery. However, several network coding theorems [4, 27, 50] ensure that a sufficient amount of information reaches the destination after a failure occurs without failure identification; but for the price of complex in-network operations. In this dissertation I demonstrate that we can bring these merits (i.e., fast recovery and simplicity) of survivable routing approaches together with the help of current research results on network [74] and diversity coding [16].

From the practical point of view, simplicity i.e. easy deployment is essential. Thus, complex data processing at core nodes (i.e., other than the source and destination) like network coding is not desired. Hence, all complex operations have to be moved to the edge of the network (i.e., to nodes s (source) and t (destination)). We refer to the survivable routing approaches which satisfy this property as diversity coding based methods, or simply *diversity coding*. Luckily, for single edge failures and two data parts, the range of survivable routings providing this simplicity is quite wide:

Theorem 3.2.1 *[16, Theorem 2] Suppose that survivable routing R is critical. Then there are disjoint edge sets $E_A, E_B, E_{A\oplus B}$ of R^*, called **routing DAGs**, such that for an arbitrary edge $e \in E^R$, after removing the corresponding edge(s) from E^{R^*} at least two of the routing DAGs connect s to t.*

Note that a **survivable routing R is critical**, if we are not able to further decrease the flow value $f(e)$ along any edge in $e \in E^R$ without making routing R vulnerable.

This means for a single link failure scenario a very specific structure can satisfy both the survivable routing (Definition 3.1.1) and the diversity coding properties (i.e. satisfying Theorem 3.2.1), giving us the opportunity to create a very practical i.e. *a simple yet very capacity efficient method providing robust and instantaneous recovery*.

The main unsolved problem on the road to achieve this, is to find the minimum cost survivable routing.Hence our goal is to find a survivable routing R for connection D with minimum *bandwidth cost* from the possible set of survivable routings $\mathcal{R}^D$, formally:

$$\min_{R\in\mathcal{R}^D} \sum_{e\in E^R} c(e)\cdot f(e). \tag{3.6}$$

For the sake of easier presentation of our results, the auxiliary graph $G^* = (V, E^*, c)$ is introduced. The node set of G^* is the same as the node set of G, and each $e \in E$ is replaced by $k(e)$ (parallel)

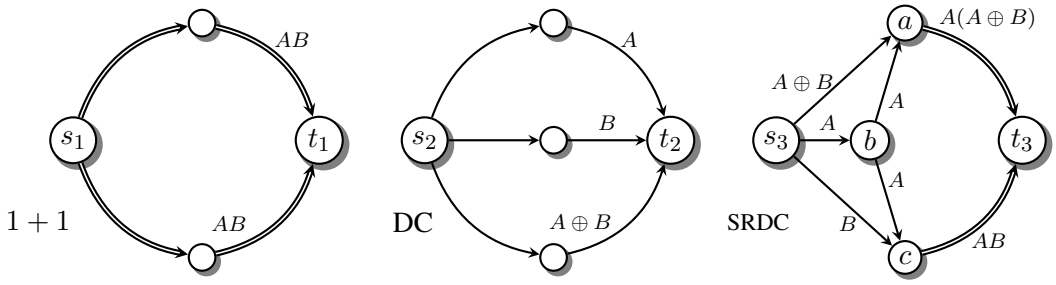


Figure 3.4: The illustration of the structure of the different protection approaches providing robust and instantaneous recovery. Double edges forward the whole user data, single edges the half of it [J3].

edges which have the same tail and head node as e, each with cost $c(e)$. *Note that single edge failure e in G corresponds to the failure of all $k(e)$ edges in G^*.* A critical survivable routing $R^* = (V^{R^*}, E^{R^*})$ forms a directed acyclic graph (DAG) in G^* (discussed in Section 3.2.1), representing the routing of the connection, where $V^{R^*} \subseteq V$, $E^{R^*} \subseteq E^*$, while the objective function in Eq. (3.6) can be rewritten as:

$$\min \sum_{e \in E^{R^*}} c(e) \ . \tag{3.7}$$

The SRDC problem itself can be divided into several subproblems according the given constraints i.e. capacity and/or node capability constraints. These different subproblems are introduced in Section 3.2.1. The main notations are summarized in Table 3.1.

The Structure of SRDC

In order to understand the SRDC it is important to overview its structure. Thus, in this section we will discuss the structure of SRDC in details and compere it to $1+1$ and DC. An illustration of the different structures is presented in Figure 3.4. As already stated in Section 2.3.3 in case of "traditional" diversity coding, for a connection request $D = (s, t, 2)$ the redundancy data $A \oplus B$ is calculated at the source s, and A, B and $A \oplus B$ is sent along three disjoint end-to-end paths between s and t (3-connected graphs are required). The edge sets used by the disjoint paths are denoted as $E_A, E_B, E_{A\oplus B}$, respectively. Also $1+1$ protection could be treated as an implementation of diversity coding: A and B are sent along two disjoint paths E_A and E_B, while the redundancy data $A \oplus B$ is sent along both paths. Note that both of these routings are survivable. By SRDC three routing DAGs carry the user data. Thus the SRDC is able to extend the benefits of diversity coding to 2 connected graphs.

By SRDC with the help of diversity coding the three routing DAGs can be operated independently from each other ($E_A \cup E_B \cup E_{A\oplus B} = E^{R^*}$, $E_A \cap E_B = \emptyset$, $E_A \cap E_{A\oplus B} = \emptyset$, $E_B \cap E_{A\oplus B} = \emptyset$), i.e., they carry the same data part regardless of the failure, while coding is performed only at the end nodes of the connection. However, this general implementation of diversity coding requires splitting and merging of the paths at the core nodes.

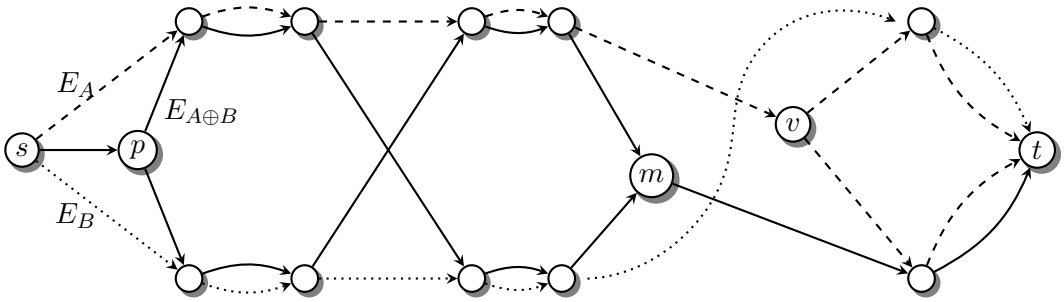


Figure 3.5: A survivable routing $R^* = (V^{R^*}, E^{R^*})$ for connection $D = (s, t, 2)$ with the corresponding routing DAGs E_A, E_B and $E_{A\oplus B}$ [C2].

The splitting and merging operations are simple enough in the sense that every node in the network can perform them without any complicated software or hardware update. Furthermore, in current networking paradigm, such as Software Defined Networking (SDN), a splitter can be easily deployed by applying simple flow rules, while a merger functionality can be implemented as a simple network function as well [J1]. The practical implementation of the survivable routing methods is discussed in Section 6.1.1.

In [16, 74] it was shown that a critical survivable routing has a well defined structure i.e. is created by a maximal series of $s-t$ cuts or by a block decomposition. In a critical solution, we have an alternating series of splitter and merger nodes (if there are any at all). An example of a survivable routing with diversity coding is presented in Figure 3.5. For routing DAG $E_{A\oplus B}$ node p is a splitter and node m is a merger, while for routing DAG E_A node v and node t act as a splitter and merger, respectively.

Definition 3.2.2 *Nodes p and m are splitting and merging node-pair (**or** pm**-pair** shortly) of a survivable routing R^* (given with its routing DAGs $E_A, E_B, E_{A\oplus B}$), if there exists a routing DAG, without loss of generality, E_A, which splits at node p and merges back at m. To the edge set (denoted as $\mathcal{E}^{G^*}_{p,m}$) of the corresponding disjoint path-pair between p and m in the survivable routing is referred to as an **island** in DAG E_A.*

Note that, there always exists an edge disjoint pair of paths between a splitter and its corresponding merger in a critical survivable routing R^* [16, 74]. *It is important to understand that each routing DAG in a critical solution consists of a series of paths and islands from s to t.* Furthermore, a pm-pair could be part of at most one routing DAG [16, 74].

For example, in "traditional" diversity coding all of $E_A, E_B, E_{A\oplus B}$ are $s \to t$ paths. In Figure 3.5 E_A consists of an $s \to v$ path and a $v \to t$ island, E_B is an $s \to t$ path, while $E_{A\oplus B}$ consists of a path $s \to p$, an island $p \to m$, and a path $m \to t$.

Note that, diversity coding corresponds to the case when only the source and destination node can be a pm-pair, which can be solved by Suurballe's algorithm.[1]

[1] Note that the augmenting path technique of Suurballe's algorithm can be used to find 3 edge-disjoint paths.

Subproblems of SRDC

In Section 3.2.1 the general version of SRDC was introduced. However, the SRDC problem itself can be divided into four subproblems according to the technological constraints in the network. We investigate the SRDC under capacity and node capability constraints.

The problem formulations and their practical relevance is the following:

- ICAN (Infinite Capacity and All Node capabilities): In this case there are neither capacity nor node capability constraints in the network. "Infinite" capacity means that each link has the free capacity corresponding to the demand itself i.e. $k(e) \geq b$. In other words, there are no bottleneck links in the network, which can not accommodate the entire demand. Furthermore, we assume that each node is capable to perform the splitting and merging action. In this network scenario the network is not overloaded (i.e. no bottleneck links) and all the nodes received the necessary hardware or software updates i.e. all nodes are able to perform the splitting and merging actions. Note that in SDN networks a software update is enough to achieve the splitter and merger capabilities (see Section 6.1.1).

- ICCN (Infinite Capacity and Constrained Node capabilities): In this scenario there are no capacity constraints, but not all nodes are capable to perform the splitting and merging action i.e. the node capabilities are constrained/restricted. This network scenario belongs to the case where the network is not overloaded (i.e. no bottleneck links) but not all the nodes received the necessary hardware or software updates i.e. not all nodes are able to perform the splitting and merging actions. For example this scenario can occur if the operator decides to prefer incremental deployment. Note that incremental deployment will be discussed further in Section 3.2.7.

- CCAN (Constrained Capacity and All Node capabilities): In this case all nodes are capable to perform the splitting and merging action, but there are bottleneck links in the networks i.e. there are capacity constraints. In this case the operator already updated all the nodes, however the network is overloaded i.e. some links have very limited free capacity.

- CCCN (Constrained Capacity and Constrained Node capabilities): In this scenario there are both capacity and node capability constraints in the network. In other words, there are bottleneck links in the network and not all the nodes received the necessary hardware or software updates. This can be considered as the worst case scenario.

It is important to underline that for the demand $D = (s, t, 2)$ in survivable routing with diversity coding (SRDC) we are searching for three routing DAGs, each forwarding one unit of capacity (either A, B or $A \oplus B$). Thus, in a critical survivable routing one would think that infinite edge capacity means $\forall e \in E : k(e) = 3$, i.e., the case when all DAGs may use the same edge. However, it was shown in [16] (as a consequence of Theorem 3.2.1) that with the application of diversity coding (called resilient

flow decomposition) in a *critical survivable routing* for a connection with $d = 2$ the flow values are $\forall e \in E^R : f(e) \leq 2$. Thus, without loss of generality, we can restrict the available capacities to $k(e) = 2$ for every edge e, without ruling out the minimum cost survivable routing (which is critical, obviously). Hence, in SRDC infinite edge capacities mean $\forall e \in E : k(e) = 2$ (instead of 3). *It is important to understand that by each subproblem each routing DAG in a critical solution consists of a series of paths and islands from s to t.* This knowledge is used when solving the different subproblems.

3.2.2 The Contributions of the Dissertation to SRDC

It is already proven that if there are no capacity nor node constraints in the network (i.e. ICAN) we can find a solution in polynomial time with the help of Algorithm 1 (SRDC-IA) [C2]. The complexity of the capacity constraint or node constraint case i.e. CCAN or ICCN is still an open issue. However, it has been proven that if there are capacity and node constraints too i.e. CCCN (Constrained Capacity and Constrained Node capabilities) then the problem is NP-complete [C2]. These results are the existing results and are highlighted with the gray background in Table 3.2. All other results are the novel contributions of this dissertation (denoted with the white background) and will be discussed in details in the remainder of the chapter. An overview of the problems and results is given in Table 3.2.

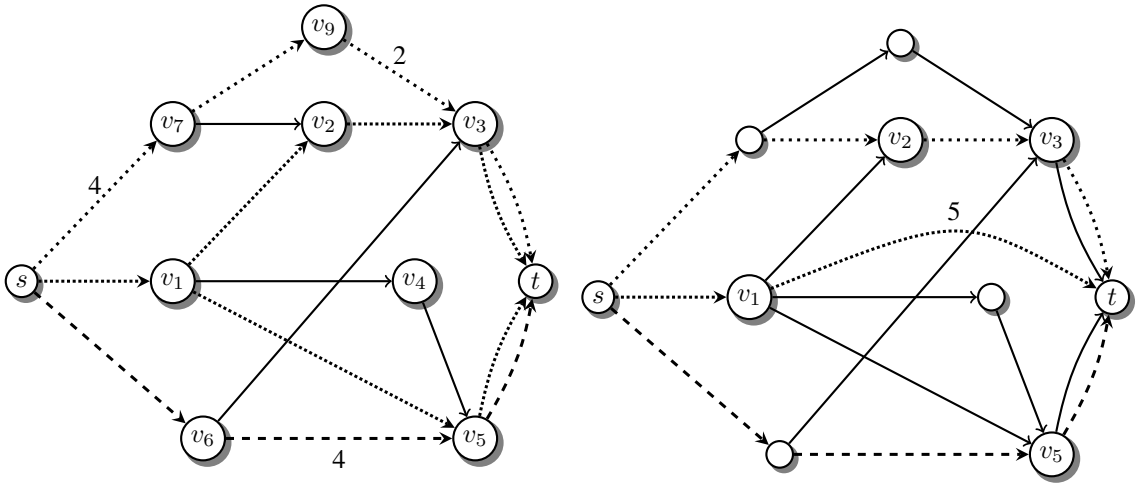
Also note that Section 6.1.1 is devoted to the practical implementation of these new methods in SDN networks. It is shown that the splitting and merging action can be implemented easily with a software update.

3.2.3 Survivable Routing with Infinite Capacities and All Node Capabilities

The SRDC-ICAN subproblem is discussed in details in [C2]. It is proven that this particular subproblem is polynomial time solvable and that without loss of generality the infinite edge capacities mean $\forall e \in E : k(e) = 2$ instead of 3 (as already highlighted in Section 3.2.1).

Table 3.2: SRDC overview. The set of splitters and mergers is denoted as $\mathcal{P}$ and $\mathcal{M}$, respectively. The results with the gray background are existing approaches in the literature [C2], the novel contributions of this dissertation are denoted with the white background.

	No capacity constraints	Capacity constraints
	ICAN	*CCAN*
$\mathcal{P} = V$,	SRDC-IA	ILP + SRDC-CA
$\mathcal{M} = V$	Polynomial time solvable	-
	ICCN	*CCCN*
$\mathcal{P} \subset V$,	ILP + SRDC-IC (Approx. algorithm)	ILP + SRDC-CC
$\mathcal{M} \subset V$	-	NP-complete



(a) The optimal survivable routing with the routing DAGs. The edge costs other than 1 are shown.

(b) The 3 paths in $\widehat{G}$ of Theorem 3.2.4. For the sake of simplicity, only virtual edge (v_1, t) is shown.

Figure 3.6: An example network $G^* = (V, E^*, c)$ with capacity constraint on the edges (remember from the construction of G^* that $k(e) = 2$ edges in G are parallel edges in G^*), where $c(e) = 1$, or otherwise written next to the edge. The edges of the routing DAGs E_A, E_B and $E_{A\oplus B}$ are denoted as dashed, dotted and densely dotted lines, respectively. Here Algorithm 1 fails for connection $D = (s, t, 2)$ [C2].

The polynomial time algorithm which solves the SRDC-ICAN problem is based on finding 3 link disjoint path in an auxiliary ((multi-)graph $\widehat{G} = (V, \widehat{E}, \widehat{c})$). The auxiliary graph is created the following way: Let $\mathrm{cost}(u, v)$ denote the sum of the edge costs of a minimum cost disjoint path-pair between nodes u and v in G. The node set of $\widehat{G}$ is the same as the node set of G, and the edges of $\widehat{G}$ are the edges of G with cost $\widehat{c}(e) = c(e)$ and we add an edge $e_n = (u, v)$ for every pair of distinct node-pairs with cost $\widehat{c}(e_n) = \mathrm{cost}(u, v)$. We refer to the newly added edges as *virtual edges*.

In the algorithm, we search for 3 edge-disjoint paths with minimum bandwidth cost in $\widehat{G}$, which in turn corresponds to three DAGs in G^*. The cost of the minimum cost 3 edge-disjoint paths equals to the cost of the minimum cost survivable routing R^*. The SRDC-IA algorithm itself is presented as Algorithm 1.

The computational complexity is dominated by the creation of graph $\widehat{G}$. Finding the pair of shortest edge-disjoint path from a single source to every destination can be done in $O(|E| \log_{1+|E|/|V|} |V|)$ time [84], which should be launched for every source node, resulting in $O(|V||E| \log_{1+|E|/|V|} |V|)$ steps. Finding minimum cost 3 edge-disjoint paths and saving the survivable routing could be done in $O(|E|)$. This means that the overall complexity of the SRDC-IA algorithm is $O(|E| \log_{1+|E|/|V|} |V|)$.

The Limits of ICAN

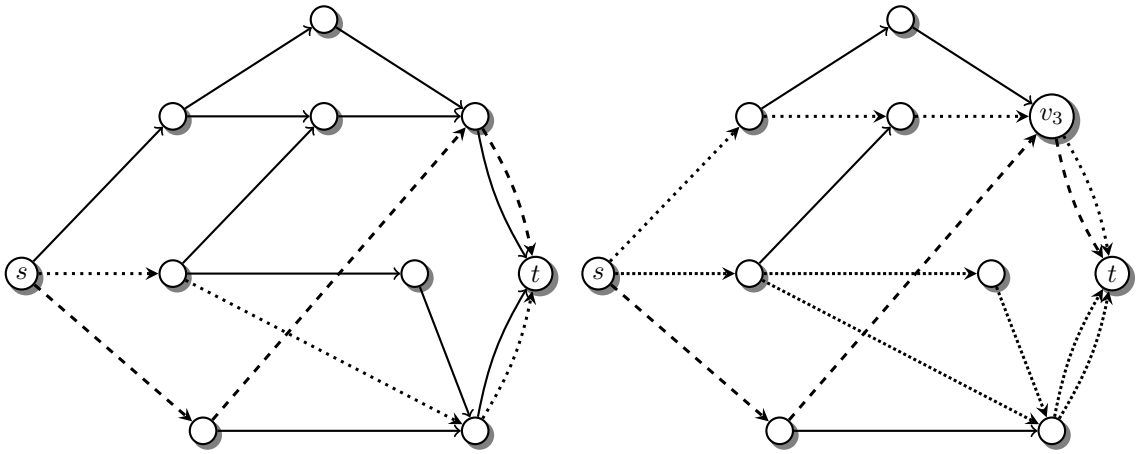
In this section I will show the limits of the ICAN through several examples. A natural question is why the algorithm is not able to cope with capacity constrained networks i.e. where some links have only one

Algorithm 1: Survivable Routing with Diversity Coding - ICAN (SRDC-IA)

Input: $G^* = (V, E^*, c)$, $D = (s, t, 2)$

Result: $R^* = (V^{R^*}, E^{R^*})$, in specific, routing DAGs E_A, E_B, and $E_{A \oplus B}$

begin
  Define cost $\widehat{c} : E \to \mathbb{R}^+$ and edge set $\widehat{E} = \emptyset$, $E_s = \emptyset$;
  `// Create graph` $\widehat{G} = (V, \widehat{E}, \widehat{c})$.
  Add $\forall e \in E$ to $\widehat{E}$ with $\widehat{c}(e) = c(e)$;
  for $u \in V$: **do**
    Find the pair of shortest edge-disjoint paths from source u to all other nodes $v \in V, u \neq v$ in G with Suurballe's algorithm (denote their cost with $\mathrm{cost}(u, v)$);
    Add virtual edge $e_n = (u, v)$ to $\widehat{E}$ with $\widehat{c}(e_n) = \mathrm{cost}(u, v)$;
  end
  `// Find 3 edge-disjoint paths in` $\widehat{G}$.
  Find minimum cost 3 edge-disjoint paths between s and t in $\widehat{G}$ with Suurballe's algorithm;
  Add the traversed edges (i.e., their corresponding edges in G^*) to E_s;
  for $e = (u, v) \in E_s$ **do**
    if *e is a virtual edge* **then**
      Replace virtual edge e with minimum cost island $\mathcal{E}_{u,v}^{G^*}$ in E_s;
    end
  end
  `// Save optimal survivable routing` R^*.
  for $e = (u, v) \in E_s$ **do**
    Add nodes u, v to V^{R^*} (if $u, v \notin V^{R^*}$);
    Add edge e to E^{R^*};
  end
end



(a) The 2 edge-disjoint paths with minimum cost.

(b) A possible vulnerable routing, where the failure of (v_3, t) disrupts two routing DAGs.

Figure 3.7: An example network $G^* = (V, E^*, c)$ with capacity constraint on the edges (remember from the construction of G^* that $k(e) = 2$ edges in G are parallel edges in G^*), where $c(e) = 1$, or otherwise written next to the edge. The edges of the routing DAGs E_A, E_B and $E_{A\oplus B}$ are denoted as dashed, dotted and densely dotted lines, respectively. Here Algorithm 1 fails for connection $D = (s, t, 2)$ [C2].

unit of free capacity ($k(e) = 1$). The problem is that in such a capacity constrained case $\mathcal{E}^G_{p,m}$ (i.e. the island formation) depends on the route of the other two routing DAGs, i.e., another routing DAG may use the single available capacity unit along an edge $e \in \mathcal{E}^G_{p,m}$ of the minimum cost disjoint path-pair. For example, Figure 3.6(a) shows a network with an optimal survivable routing of cost 20. Note that, the virtual edge $e_n = (v_1, t)$ has cost $\widehat{c}(e_n) = 5$ because $\mathrm{cost}(v_1, t)$ is the cost of the shortest path-pair $v_1 \to v_2 \to v_3 \to t$ and $v_1 \to v_5 \to t$ is $3 + 2 = 5$. The minimum cost 3 edge-disjoint paths in $\widehat{G}$ are shown in Figure 3.6(b). Clearly, this is not a valid solution in the capacity constrained case, as edge $e = (v_2, v_3)$ has only $k(e) = 1$ available capacity in G, while two routing DAGs should use it in the optimal solution.

Another possible approach could be to find the minimum cost 3 edge-disjoint paths with Suurballe's algorithm using the augmenting path technique. Applying this technique to SRDC, the virtual edges are only traversed by the 3^{rd} augmenting path, only after 2 edge-disjoint paths were already found. A natural extension of Algorithm 1 may be to run the disjoint path search for each virtual edge (e.g., to (v_1, t)) as a disjoint path-pair between nodes v_1 and t. During this search the reverse edges of the already found 2 edge-disjoint paths can be used (shown in Figure 3.7(a)) similarly as in Suurballe's algorithm, and additionally it can use the reverse edges of the third edge-disjoint path's segment between s and v_1 (which is $s \to v_7 \to v_2 \to v_3 \to v_6 \to v_5 \to v_1$). This could result in an augmenting path between splitter v_1 and merger t of $v_1 \to v_5 \to v_6 \to v_3 \to t$. In this case the second augmenting path between splitter v_1 and merger t would be $v_1 \to v_4 \to v_5 \to t$. This in fact results in a vulnerable routing shown in Figure 3.7(b) with cost 16. Thus, Algorithm 1 is not applicable to the capacity constrained case.

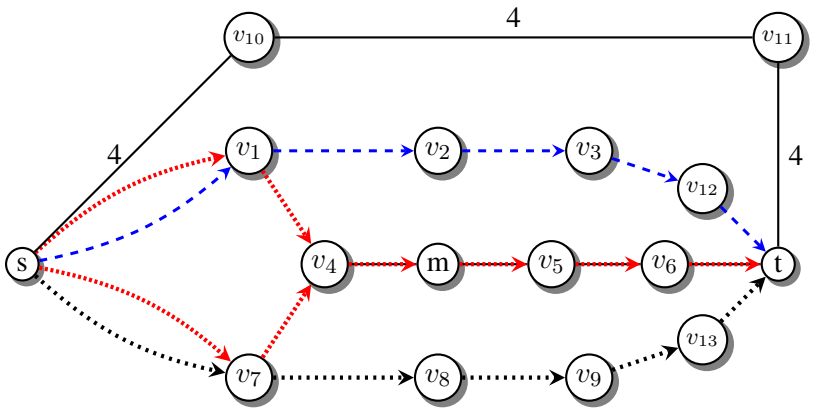


Figure 3.8: The invalid solution of SRDC-IA. Beside the source and destination nodes, m is a potential merger/splitter node. If not displayed otherwise the edge costs are unit.

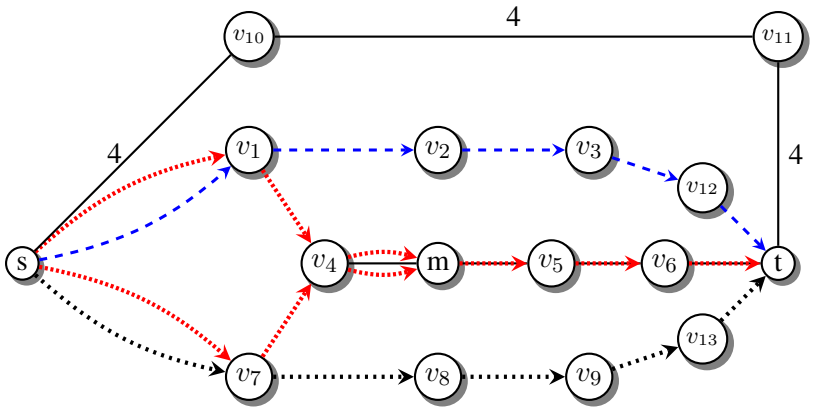


Figure 3.9: The optimal survivable routing solution of ICCN (total cost: 19). Beside the source and destination nodes, m is a potential merger/splitter node. If not displayed otherwise the edge costs are unit.

The other natural question is why this algorithm is not able to cope with node capability constrained networks i.e. where not all nodes are capable to perform the splitting and merging action. We show the difficulties arising when only a given set of nodes can be the splitter or merger through a simple example (Figure 3.9 and 3.8). In this example only node m is upgraded i.e. only node m can be a splitter or merger beside the source (s) and destination (t) node[2].If the polynomial time Algorithm 1 (SRDC-IA) presented in [C2] is used to find the three routing DAGs between s and t, we get an invalid solution (Figure 3.8). The cost of this solution would be 18 but since v_4 can not be a merger node, the solution is invalid (i.e., the solution can not be implemented). On the other hand, if Diversity Coding (DC) is used, the total cost of the solution is 22, since the user data is sent along three edge disjoint paths (i.e. the sparsely dotted (5), the dashed (5) and $s \rightarrow v_{10} \rightarrow v_{11} \rightarrow t$ paths). If $1 + 1$ is used the cost of the

[2] The *source and the destination node is always capable of performing the splitting and merging action*, since the operation can be done in the electrical layer.

solution is 20 (twice the cost of the sparsely dotted (5) and the dashed (5) paths). However, the optimal solution is 19 (dotted path (5), dashed path (5) and densely dotted path (9)), as shown in the Figure 3.9. Note that between nodes v_4 and m two units of data are transferred to get to the merging point in the network. We see that the structure of this subproblem is much more complex than by ICAN.

3.2.4 Survivable Routing with Infinite Capacities and Constrained Node Capabilities

In Section 3.2.3 we already shown through an example (Figure 3.9) how the SRDC problem changes if the node capabilities are restricted in the network. In this section, we present an effective algorithm i.e. the SRDC-IC to solve this subproblem. The SRDC-IC in worst case delivers the solution of $1 + 1$. Furthermore in Section 3.2.4 we prove that the $1 + 1$ is a $4/3$ approximation of SRDC with infinite capacities i.e. ICAN and ICCN. From this follows that our algorithm is in fact an approximation algorithm. Note that in Section 3.2.5 a universal ILP is presented which exactly solves all the SRDC subproblems, however the running time is considerably longer.

Heuristic Approach: SRDC-IC

The algorithm presented in this section i.e. the SRDC-IC solves the ICCN problem improving the capacity efficiency of $1 + 1$. The SRDC-IC contains the $1 + 1$ as a special case. This means if only the source and destination nodes are allowed to be splitter/mergers, the SRDC-IC returns the solution of $1 + 1$. We will use this fact[3] in Section 3.2.4 to prove, that the SRDC-IC is in fact an approximation algorithm.

Our algorithm is based on finding 3-edge disjoint paths in an auxiliary graph $\widetilde{G}$. The auxiliary graph is created the following way: Let $\mathrm{cost}(u, v)$ denote the sum of the edge costs of a minimum cost disjoint path-pair between nodes u and v in G. The node set of $\widetilde{G}$ is the same as the node set of G, and the edges of $\widetilde{G}$ are the edges of G with cost $\widetilde{c}(e) = c(e)$ and we add an edge $e_n = (u, v)$ for every splitter merger node-pair with cost $\widetilde{c}(e_n) = \mathrm{cost}(u, v)$ ($\forall u, v$ such that $u \in \mathcal{P}$ and $v \in \mathcal{M}$: and $u \neq v$ we add $\widetilde{c}(e_n) = \mathrm{cost}(u, v)$). However, if there are no disjoint paths between u and v then no virtual edge is added. Note that the source and destination nodes are always included in the splitter-merger sets, since both actions can be done easily in the electrical domain. The obtained graph $\widetilde{G}$ is similar to $\widehat{G}$ but virtual edges are added only between the possible splitter-merger pairs.

As by SRDC-IA (Section 3.2.3) the computational complexity is dominated by creation of the auxiliary graph resulting in $O(|V||E| \log_{1+|E|/|V|} |V|)$ steps.

The limits of SRDC-IC are shown through the example presented in Figure 3.9. In the illustrated case we are able to find only a disjoint path pair between the source and destination node (i.e. no disjoint path pairs can be found between the source and node m or between m and the destination node). Since the minimum cost disjoint path-pair between nodes s and t are the sparsely dotted (5) and the dashed (5) paths, the SRDC-IC algorithm returns the same solution as the $1 + 1$ and not the optimal one

[3] Note that we assume that the *source and the destination node is always capable of performing the splitting and merging action*.

Algorithm 2: Survivable Routing with Diversity Coding - ICCN (SRDC-IC)

Input: $G^* = (V, E^*, c)$, $D = (s, t, 2)$

Result: $R^* = (V^{R^*}, E^{R^*})$, in specific, routing DAGs E_A, E_B, and $E_{A \oplus B}$

begin
  Define cost $\widehat{c} : E \to \mathbb{R}^+$ and edge set $\widetilde{E} = \emptyset$, $E_s = \emptyset$;
  `// Create graph` $\widetilde{G} = (V, \widetilde{E}, \widetilde{c})$.
  Add $\forall e \in E$ to $\widetilde{E}$ with $\widetilde{c}(e) = c(e)$;
  for $u \in \mathcal{P}$**:** **do**
    Find the pair of shortest edge-disjoint paths from source u to all other nodes $v \in \mathcal{M}, u \neq v$ in G with Suurballe's algorithm (denote their cost with $\mathrm{cost}(u, v)$);
    Add virtual edge between the splitter merger node-pairs $e_n = (u, v)$ to $\widetilde{E}$ with $\widetilde{c}(e_n) = \mathrm{cost}(u, v)$;
  end
  `// Find 3 edge-disjoint paths in` $\widetilde{G}$.
  Find minimum cost 3 edge-disjoint paths between s and t in $\widetilde{G}$ with Suurballe's algorithm;
  Add the traversed edges (i.e., their corresponding edges in G^*) to E_s;
  for $e = (u, v) \in E_s$ **do**
    if e *is a virtual edge* **then**
      Replace virtual edge e with minimum cost island $\mathcal{E}_{u,v}^{G^*}$ in E_s;
    end
  end
  `// Save optimal survivable routing` R^*.
  for $e = (u, v) \in E_s$ **do**
    Add nodes u, v to V^{R^*} (if $u, v \notin V^{R^*}$);
    Add edge e to E^{R^*};
  end
end

(Figure 3.9(a)). However, such a specific optimal solution structure is very rare. The effectiveness of the algorithm is demonstrated through extensive simulation in Section 3.2.7.

Approximability of SRDC with Infinite Capacities

Although $1+1$ is a 2-approximation [19] for the general survivable routing problem, one can observe that the maximum resource saving is below 25% in all investigated topologies (see Section 3.2.7). Thus, it leads us to the conjecture that $1+1$ is a $4/3$-approximation for the special case of two data parts, when no capacity constraints are in the network.

The proof is based on the SRDC-ICAN polynomial time algorithm that can be traced back to finding three paths in an auxiliary graph $\widehat{G}$. Using the inequalities between $1+1$ (2 paths) and SRDC-IA (ICAN) (3 paths), we prove algebraically that $1+1$ is a 4/3 approximation for the SRDC with infinite capacity problems.

Lemma 3.2.3 *$1+1$ is a $4/3$-approximation algorithm for the minimum cost survivable routing in the single edge failure resilient case, when source flow is divided into two parts and there are no capacity constraints in the networks.*

Proof: In the proof, without loss of generality we assume that the length of an edge is the cost of sending a single data unit between its end-nodes.

Let the two edge-disjoint paths of the $1+1$ solution be denoted with P_1, P_2 and $|P_1| \leq |P_2|$. Furthermore, the paths of the $SRDC$ solution are denoted with $P_{E_a}, P_{E_b}, P_{E_c}$ in $\widehat{G}$ and $|P_{E_a}| \leq |P_{E_b}| \leq |P_{E_c}|$. We know that

$$|P_1| + |P_2| \leq |P_{E_a}| + |P_{E_b}|,$$

$$|P_1| + |P_2| \leq |P_{E_a}| + |P_{E_c}|$$

and

$$|P_1| + |P_2| \leq |P_{E_c}| + |P_{E_b}|.$$

We have to prove that:

$$2(|P_1| + |P_2|) \leq \frac{4}{3}(|P_{E_a}| + |P_{E_b}| + |P_{E_c}|). \tag{3.8}$$

Since when using $1+1$ both data parts (A and B) have to be transferred on both paths, we have the total cost of $2(|P_1| + |P_2|)$. On the other hand the SRDC transfers A, B, and $A \oplus B$ on three disjoint paths i.e. the total cost of the solution is $|P_{E_a}| + |P_{E_b}| + |P_{E_c}|$.

From $|P_{E_b}| \leq |P_{E_c}|$ follows, that we can substitute $|P_{E_c}|$ with $|P_{E_b}| + \epsilon$ where $\epsilon \geq 0$. After the substitution we get:

$$2(|P_1| + |P_2|) \leq \frac{4}{3}|P_{E_a}| + \frac{8}{3}|P_{E_b}| + \frac{4}{3}\epsilon.$$

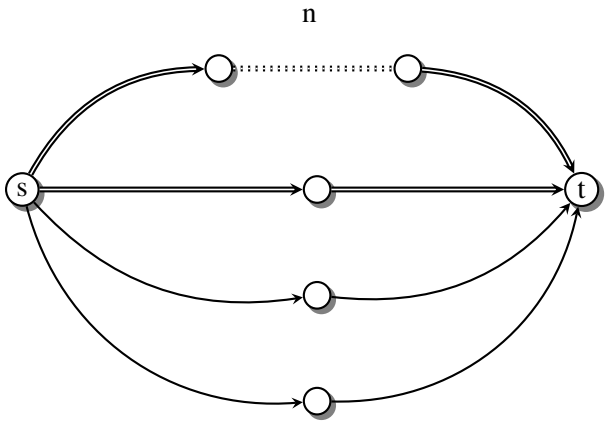


Figure 3.10: The counterexample of the approximability of SRDC-CCAN. The $1+1$ is only capable to use the double edges, while SRDC can use all edges. From this follows that if we increase the number of edges n the difference in cost between the two solutions can be increased arbitrarily.

Furthermore we know that $|P_1|+|P_2| \leq |P_{E_a}|+|P_{E_b}|$ which means that $|P_1|+|P_2| = |P_{E_a}|+|P_{E_b}|-\theta$ where $\theta \geq 0$. Subtracting $\frac{4}{3}(|P_1|+|P_2|) = \frac{4}{3}(|P_{E_a}|+|P_{E_b}|-\theta)$ from equation 3.8 we get that:

$$\frac{2}{3}|P_1|+\frac{2}{3}|P_2| \leq \frac{4}{3}|P_{E_b}|+\frac{4}{3}\theta+\frac{4}{3}\epsilon.$$

And from $|P_1|+|P_2| \leq |P_{E_a}|+|P_{E_b}|$ we know that $|P_1|+|P_2| \leq 2|P_{E_b}|$ since $P_{E_a} \leq P_{E_b}$. From here it follows trivially that the $\frac{2}{3}|P_1|+\frac{2}{3}|P_2| \leq \frac{4}{3}|P_{E_b}|+\frac{4}{3}\theta+\frac{4}{3}\epsilon$ i.e. inequality (3.8) always holds. ■

From the fact that $1+1$ is a $4/3$-approximation algorithm for SRDC with infinite capacities, follows that the SRDC-IC algorithm is a 4/3 approximation algorithm for the SRDC-ICCN problem.

Theorem 3.2.4 *The SRDC-IC algorithm is a 4/3 approximation algorithm for the SRDC-ICCN.*

Proof: We already proved that the $1+1$ is a 4/3 approximation for the SRDC-ICAN. Since even in the worst case the SRDC-IC algorithm returns the solution of $1+1$ (since the source and destination is always a splitter-merger pair). From that and from the fact that even in the worst case the SRDC-ICCN solution has the cost of a relaxed ICAN problem, follows that the SRDC-IC is a 4/3 approximation algorithm for the SRDC-ICCN. ■

In order to show that in a capacity constraint case, the $1+1$ does not approximate the SRDC problem, we present a graph construction (Figure 3.10). Note that in Figure 3.10 only the double edges have the capacity of 2. In other words, these edges can accommodate the entire demand ($D = (s,t,2)$) i.e. only these edges can be used by $1+1$. Furthermore, the single edges have a free capacity of 1 and can be only used by SRDC. This results in graph where the cost of $1+1$ is dependent on n (number of edges) i.e. the cost is $4+n$, and the cost of SRDC is always 6 (is independent from n), proving that the $1+1$ does not approximate the SRDC in a capacity constraint case.

3.2.5 Survivable Routing with Capacity Constraints and All Node Capabilities

In this section, we will investigate the capacity constrained scenario, i.e., when there are some bottleneck links with just one unite free capacity ($k(e) = 1$). Although a polynomial time algorithm exists for the infinite capacity case (Section 3.2.3), the survivable routing with diversity coding (SRDC) problem is more complex in this scenario. However, we present in this section an ILP and an efficient heuristic solution (SRDC-CA) for the problem when all nodes are able to perform the splitting and merging action.

Note that the SRDC-CA heuristics always provide a feasible solution (if a solution exists) for the CCAN subproblem, however if there are node capability constraints too in the network (i.e. CCCN subproblem), then this heuristics is no longer applicable. In this case the heuristic presented in Section 3.2.6 i.e. SRDC-CC should be used. The SRDC-CC does not guarantee a solution (since even deciding if there exists a solution for CCCN is already NP-complete as shown in [C2]), however it is extremely capacity efficient (the capacity consumption is near to the optimum i.e. *Lower Bound*) and has a low blocking probability.

The effectiveness of both heuristics is investigated through extensive simulations (see Section 3.2.7).

Optimal Solution

In this section, we present an ILP to obtain an optimal survivable routing R in terms of bandwidth cost. The ILP formulation provides the three routing DAGs for SRDC even in the capacity constraint case.

To do so, we need to introduce the so called *reduced capacity function* [74]: $k'(e) = \begin{cases} 1.5 & \text{if } k(e) \geq 2 \\ 1 & \text{if } k(e) = 1. \\ 0 & \text{otherwise} \end{cases}$

Theorem 3.2.5 *In [74, Theorem 2] the authors show that a survivable routing exists in a given graph $G = (V, E, k, c)$ if and only if there is a flow of value three in $G = (V, E, k', c)$.*

The Theorem 3.2.5 will be used in both our ILP formulation and in the heuristic approach described in Section 3.2.5. Note that given a routing DAG E_A in a critical survivable routing, a function x^A which is half on the edges of an island and 1 on all other (path) edges in E_A, forms an $s - t$ flow of value 1, according to [C2, Claim 2].

Our goal is to obtain the (critical) flow values $f(e)$ in the input graph $G = (V, E, k, c)$ which minimize the bandwidth cost in terms of Equation 3.6 for the connection $D = (s, t,2)$ i.e. to find the minimum cost survivable routing. The three flows are denoted as $w \in \{A, B, A \oplus B\} = \mathcal{W}$, respectively, with corresponding (real) flow variables $x^w(e)$ and indicator variables $f^w(e)$. Based on Theorem 3.2.5 the reduced capacity values $k'(e)$ ensure that the failure of an arbitrary edge e disconnects at most one routing DAG, thus, at least two routing DAGs remain which connect s and t, i.e., the data can be decoded

at the destination. Our objective is to minimize the bandwidth cost of the SRDC problem in terms of Equation 3.6:

$$\min \sum_{e \in E} c(e) \cdot f(e).$$

The following constraints are required:

$$\forall w \in \mathcal{W}, \forall i \in V\text{:}$$

$$\sum_{(i,j)\in E} x^w(i,j) - \sum_{(j,i)\in E} x^w(j,i) = \begin{cases} 1 & \text{, if } i = s \\ -1 & \text{, if } i = t \\ 0 & \text{, otherwise} \end{cases}, \tag{3.9}$$

$$\forall e \in E\text{: } \sum_{w \in \mathcal{W}} x^w(e) \leq k'(e), \tag{3.10}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{: } x^w(e) \leq f^w(e), \tag{3.11}$$

$$\forall e \in E\text{: } \sum_{w \in \mathcal{W}} f^w(e) \leq f(e) \leq k(e), \tag{3.12}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{: } 0 \leq x^w(e) \leq 1, \tag{3.13}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{: } 0 \leq f^w(e) \text{ are integer variables.} \tag{3.14}$$

The constraint in Equation (3.9) formulates the flow conservation for each routing DAG w. Constraint (3.10) set the maximal flow value based on the reduced capacity function, while Constraint (3.11) set the indicator variables $f^w(e)$ of edge usage for the routing DAGs in G. Constraint (3.12) sets the flow value in $G = (V, E, k, c)$, i.e., if edge e was used in an arbitrary routing DAG w, we have to include it in the final solution with value $f(e) = \sum_{\mathcal{W}} f^w(e)$. Constraints (3.13)-(3.14) sets the bounds for the flow variables, and sets the integer constraint for the indicator variables $f^w(e)$. Note that the $f^w(e)$ variables correspond to the edge set w in the solution, i.e., provide the three end-to-end DAGs. Since $x^A + x^B + x^{A\oplus B}$ gives an $s - t$ flow of value 3 in G, from Theorem 3.2.5 we get that f is indeed survivable.

Heuristic Approach - SRDC-CA

As the ILP presented in Section 3.2.5 is NP-hard, the running time can be long, especially for large networks. Hence in this section, we present a fast heuristic approach as an alternative for finding a survivable routing in the capacity constrained case (CCAN).

The algorithm described in details in Algorithm 3 (SRDC-CA) always returns a feasible solution. The input of the SRDC-CA is a graph $G = (V, E, k, c)$ and the connection request $D = (s, t, 2)$. The output is the survivable routing $R = (V^R, E^R, f)$. In Steps (4)-(7) the auxiliary graph $G' = (V, E', k', c')$ is created using the reduced capacities ($k'(e) = \min\{1, k(e)\}$). In the transformation, the $e \in E$ edges of G are added to E' with their original cost $c(e)$. Based on its capacity $k(e)$ of $e \in E$, we add a parallel

Algorithm 3: Survivable Routing with Diversity Coding - CCAN (SRDC-CA)

Input: $G = (V, E, k, c)$, $D = (s, t, 2)$, α

Result: $R = (V^R, E^R, f)$

begin

  Define capacity $k', k'' : E \to \mathbb{R}^+$, cost $c' : E \to \mathbb{R}^+$ and edge sets $E_n = \emptyset$; $E' = \emptyset$; $E_s = \emptyset$;

  `// Create graph` $G' = (V, E', k', c')$.

  Add $e \in E$ to E' with $k'(e) = \min\{1, k(e)\}$ and $c'(e) = c(e)$;

  for $e = (u, v) \in E : 2 \leq k(e)$ **do**

    Add extra edge $e_n = (u, v)$ to E_n with $k'(e_n) := 0.5$ and $c'(e_n) := c(e) \cdot \alpha$;

  end

  $E' := E' \cup E_n$;

  `// Phase 1: Find flow in` $G' = (V, E', k', c')$.

  Find a minimum cost $s - t$ flow f' of value 3 in G' with respect to the reduced capacity function k';

  for $e = (u, v) \in E$ **do**

    Set $k''(e) := 0$;

    Set $e_n = (u, v) \in E_n : k''(e_n) := 0$;

    if $0 < f'(e)$ **then**

      if $e_n = (u, v) \in E_n : 0 < f'(e_n)$ **then**

        Add edge e to E_s and set $k''(e) := 1$;

      end

    end

  end

  for $e^1 = (u^1, v^1), e^2 = (u^2, v^2) \in E_s, e^1 \neq e^2$ **do**

    if *no pair of edge-disjoint* $s - t$ *paths in* $G \setminus (e^1 \cup e^2)$ **then**

      $e_n^1 = (u^1, v^1), e_n^2 = (u^2, v^2) \in E_n : k''(e_n^1) = 0.5; k''(e_n^2) = 0.5$;

    end

  end

  `// Phase 2: Find flow in` $G'' = (V, E', k'', c')$.

  Find a minimum cost $s - t$ flow f'' of value 3 in G'' with respect to the capacity function k'';

  `// Save survivable routing` R.

  for $e = (u, v) \in E : 0 < f''(e)$ **do**

    Add node u, v to V^R (if $u, v \notin V^R$);

    Add edge e to E^R;

    Set flow value $f(e)$ to 2 if $e_n = (u, v) \in E_n : 0 < f''(e_n)$, and to 1 otherwise;

  end

end

edge $e_n = (u, v)$ (called *extra edge*) to E' if $2 \leq k(e)$ with reduced capacity $k'(e_n) = 0.5$ and with cost $c'(e_n) = c(e) \cdot \alpha$.

The main idea of the algorithm is to find a minimum cost $s - t$ flow with value of 3 in the reduced capacity graph $G' = (V, E', k', c')$ in Steps (8)-(14), in such way that "creating islands" (i.e., using the extra edge e_n with $k'(e_n) = 0.5$ capacity) is penalized via a scaling factor (i.e., the extra edges have a higher cost $c'(e_n) = c(e) \cdot \alpha, \alpha \geq 1$). Note that based on [74, Theorem 2] this is equivalent with finding a survivable routing in $G = (V, E, k, c)$

In order to avoid "false" island creation (resulting from an imprecise selection of α), based on the results $f'(e)$ of the first phase, we prepare the input of the second phase of Algorithm 3 (SRDC-CA). First, we initialize the reduced capacity for the next stage $\forall e \in E : k''(e) = 0$. We create an edge set E_s, so that the original edge $e = (u, v)$ is in E_s, if both $e = (u, v)$ and the corresponding extra edge $e_n = (u, v)$ is used in the solution of the minimum cost flow, in other words if $0 < f'(e)$ and $e_n = (u, v) \in E_n : 0 < f'(e_n)$. Additionally, we set the capacity of all used original edges ($e = (u, v) \in E_s : 0 < f'(e)$) to $k''(e) = 1$. In Iteration (18)-(20) we check for every edge-pair $e^1 = (u^1, v^1), e^2 = (u^2, v^2) \in E_s, e^1 \neq e^2$, whether there are two edge-disjoint $s - t$ paths in $G \setminus (e^1 \cup e^2)$ or not. If there is no such flow, then we consider the corresponding extra edges as potential islands in the survivable routing, and we set their capacity values to $k''(e_n^1) = 0.5$ and $k''(e_n^2) = 0.5$, respectively. Note that, the cost $c'(e)$ remains the same.

In Phase 2 in Step (23), we search again for a minimum cost flow with value of 3 between s and t, now in graph $G'' = (V, E', k'', c')$. Finally, the flow values $f''(e)$ of the solution give the survivable routing R. Hence, in Steps (24)-(27) the survivable routing is saved in a way that if both $e = (u, v)$ and $e_n = (u, v)$ were used in the solution ($f''(e) > 0$ and $f''(e_n) > 0$), then we set $f(e) = 2$. If $f''(e) > 0$ and $f''(e_n) = 0$, then $f(e) = 1$, and else $f(e) = 0$, which gives $R = (V^R, E^R, f)$. The routing DAGs in R can be constructed as shown in [16].

The computational complexity of creating the auxiliary graph (G') is $O(|E|)$. The problem of finding a minimum cost flow can be solved with the dual network simplex [54] approach in $O(|V||E| \log_2 |V|(|E| + |V| \log_2 |V|))$ steps. Iteration (18) leads in worst case to $O(|E|^2)$ steps and saving the survivable routing can be done in $O(|E|)$.This means that the overall complexity of the SRDC-CA is $O(|V||E| \log_2 |V|(|E| + |V| \log_2 |V|))$.

Selection of Scaling Factor α ♦ We have seen that choosing a proper α for a network is essential. For this purpose, intuitively we used the ratio $\alpha = \frac{|E|}{|V|}$, which is corresponding to the density (to be specific, it is half of the average nodal degree) of the network. In a denser network it is more likely to have 3 edge-disjoint paths with a relatively short third path, which might be the optimal routing DAGs. Therefore, the cost of the extra edge is relatively high (α is high) in order to avoid creation of islands. Meanwhile in sparse networks, it is more likely that the third path (if it exists at all) is really long, and not beneficial to use. Note that, if any of the edges $e \in E'$ is used ($f(e) > 0$), we have to reserve $f(e) = 1$ flow on its

corresponding edge in G (despite the fact that we used only 0.5 in the reduced capacity graph G'), as the flow values of the routing DAGs are integers in the optimal survivable routing R. Note that, theoretically this could result in a higher capacity consumption than $1+1$ for some connections in special topologies. However, as it is shown in Section 3.2.7, in real-like networks with such a carefully chosen α it performs much better than $1+1$.

As stated the SRDC-CA heuristic always provides a feasible solution (if a solution exists) for the CCAN subproblem, however if there are node capability constraints in the network too (CCCN), then this heuristics is no longer applicable. In this case the heuristic presented in Section 3.2.6 i.e. SRDC-CC should be used. The SRDC-CC does not guarantee a solution (since even deciding if there exists a solution for CCCN is already NP-complete as shown in [C2]), however is extremely capacity efficient (the capacity consumption is near to the theoretic *Lower bound*) and has a low blocking probability.

3.2.6 Survivable Routing with Capacity Constraints and Constraint Node Capabilities

In this section we examine the most demanding subproblem i.e. the Capacity and Node restricted subproblem (CCCN). To solve the CCCN subproblem an ILP and a heuristic is presented. The SRDC-CC heuristic solves the CCCN problem very fast in an efficient manner.

Optimal Solution

It was shown in [C2] that if only a given set of nodes is capable to perform the splitting and merging action (i.e. not all nodes are updated) and there are capacity constraints in the network, then the survivable routing problem (i.e. the CCCN) is NP-complete. The proof is based on the well-known 2-Disjoint Path Problem (2-DPP). From the proof follows that even deciding if a solution exists is NP-complete. Thus we present an ILP which solves all the subproblems including CCCN and a very fast and effective heuristic i.e. the SRDC-CC. However, the SRDC-CC can not guarantee a feasible solution (since the question itself is NP-complete), nonetheless it is extremely capacity efficient (the capacity consumption is near to the theoretic *Lower bound*) and has a low blocking probability.

The integer linear program, which solves all the SRDC subproblems is formulated the following way.

$$\min \sum_{e \in E} c(e) \cdot f(e).$$

The following constraints are required:

$$\forall w \in \mathcal{W}, \forall i \in V\text{:}$$

$$\sum_{(i,j)\in E} x^w(i,j) - \sum_{(j,i)\in E} x^w(j,i) = \begin{cases} 1 & \text{, if } i = s \\ -1 & \text{, if } i = t \\ 0 & \text{, otherwise} \end{cases}, \tag{3.15}$$

$$\forall w \in \mathcal{W}, \forall i \in \mathcal{P} \setminus \mathcal{M}\text{:}$$

$$\sum_{(i,j)\in E} f^w(i,j) \leq \sum_{(j,i)\in E} f^w(j,i), \tag{3.16}$$

$$\forall w \in \mathcal{W}, \forall i \in \mathcal{M} \setminus \mathcal{P}\text{:}$$

$$\sum_{(i,j)\in E} f^w(i,j) \geq \sum_{(j,i)\in E} f^w(j,i), \tag{3.17}$$

$$\forall w \in \mathcal{W}, \forall i \in \mathcal{F}\text{:}$$

$$\sum_{(i,j)\in E} f^w(i,j) = \sum_{(j,i)\in E} f^w(j,i), \tag{3.18}$$

$$\forall e \in E\text{:} \sum_{w\in\mathcal{W}} x^w(e) \leq k'(e), \tag{3.19}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{:}\ x^w(e) \leq f^w(e), \tag{3.20}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{:}\ 2x^w(e) \geq f^w(e), \tag{3.21}$$

$$\forall e \in E\text{:} \sum_{w\in\mathcal{W}} f^w(e) \leq f(e) \leq k(e), \tag{3.22}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{:}\ 0 \leq x^w(e) \leq 1, \tag{3.23}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{:}\ 0 \leq f^w(e) \text{ are integer variables.} \tag{3.24}$$

The constraint in Equation (3.15) formulates the flow conservation for each routing DAG w. Additionally the (3.16) (3.17) (3.18) formulates the constraints necessary to represent the different node capabilities. Namely (3.16) represents the set of nodes ($\mathcal{P} \setminus \mathcal{M}$) that are able to perform the splitting operation. Constraint (3.17) is needed for the nodes ($\mathcal{M} \setminus \mathcal{P}$) that are only capable of merging the data stream and Constraint (3.18) is for the forwarding only nodes ($\mathcal{F}$) that perform no operation on the incoming flow. However, we do not need extra constraints for the nodes that can both split and merge the data ($\mathcal{P} \cup \mathcal{M}$). The other constraints are the same as the constraints presented in Section 3.2.5 i.e. Constraint (3.19) sets the maximal flow value based on the reduced capacity function, while Constraint (3.20) sets the indicator variables $f^w(e)$ of edge usage for the routing DAGs in G. Constraint (3.22) sets the flow value in $G = (V, E, k, c)$, i.e., if edge e was used in an arbitrary routing DAG w, we have to include it in the final solution with value $f(e) = \sum_{\mathcal{W}} f^w(e)$. Furthermore constraints (3.23)-(3.24) set the bounds for the flow variables, and set the integer constraint for the indicator variables $f^w(e)$. Note that the $f^w(e)$ variables correspond to the edge set w in the solution, i.e., provide the three end-to-end DAGs. Since $x^A + x^B + x^{A\oplus B}$ gives an $s - t$ flow of value 3 in G, from Theorem 3.2.5 we get that f is indeed survivable.

Algorithm 4: Survivable Routing with Diversity Coding - CCCN (SRDC-CC)

Input: $G = (V, E, k, c)$, $D = (s, t, 2)$,

Result: $R = (V^R, E^R, f)$

begin

Define capacity $k' : E \to \mathbb{R}^+$, cost $c' : E \to \mathbb{R}^+$ and edge sets $E' = \emptyset$; $E_s = \emptyset$;

`// Create reduced capacity graph` $G' = (V, E', k', c')$.

Add $e \in E$ to E' with $k'(e) = \min\{1.5, k(e)\}$ and $c'(e) = c(e)$;

`// Find two paths in` $G' = (V, E', k', c')$.

Find a pair of shortest edge-disjoint paths from source u to v, in $G' = (V, E', k', c')$ with Suurballe's algorithm;

for $e = (u, v) \in PATH$ **do**

  $k'(e) - 1$, Add the traversed edges to E_s;

end

for $u \in \mathcal{P}$: **do**

  Find a pair of shortest edge-disjoint paths in $G' = (V, E', k', c')$ with Suurballe's between u and v such that $v \in \mathcal{M}$ and $u \neq v$ (denote their cost with $\text{cost}(u, v)$);

  Add virtual edge $e_n = (u, v)$ to E' with $c'(e_n) = \text{cost}(u, v)$ and $k'(e) = 1$;

end

Delete edges with $k'(e) < 1$;

`// Find 3rd path in` G' `using Dijkstra.`

Find minimum cost 3rd path with Dijkstra in G';

Add the traversed edges to E_s;

for $e = (u, v) \in E_s$ **do**

  if e *is a virtual edge* **then**

    Replace virtual edge e with minimum cost island $\mathcal{E}^G_{u,v}$ in E_s;

    Check capacity constraints;

  end

end

`// Save survivable routing` R.

for $e = (u, v) \in E_s$ **do**

  Add nodes u, v to V^R (if $u, v \notin V^R$);

  Add edge e to E^R;

end

end

Heuristic Approach: SRDC-CC

The input of the algorithm is a graph $G = (V, E, k, c)$ representing the transport network and the connection request $D = (s, t,2)$. The output is the survivable routing $R = (V^R, E^R, f)$.

In Step (3) the auxiliary graph $G' = (V, E, k', c')$ is created using the reduced capacity function [74]. In(4)-(7) we find two disjoint paths and store them in E_s. We also subtract one unit capacity along these edges ($\forall e \in PATH$). After the subtraction in graph $G' : \forall e \in E : k'(e) = \{0, 0.5,1, 1.5\}$. Note that we get 0 free capacity on the edges $e = (u, v) \in PATH$ that initially had 1 unit capacity in the reduced capacity graph i.e. had $1 \geq k(e) < 2$ in G, and 0.5 where the edges had initially 1.5 free capacity in the reduced capacity graph i.e. had $k(e) \geq 2$ in G. This means if we find an island, in the worst case scenario there are two parallel paths with 0.5 capacity i.e. $k(e) \geq 1$ in G. Hence we can be sure that our virtual edges have enough capacity in G. In Steps (8)-(10) we find the "islands" between the potential splitter and merger pairs.

After that in Step (12) we erase the edges that have less than 1 unit capacity e.g the edges with 0.5 free capacity, to ensure that in the next step the path that we find is capable to transfer at least half of the demand i.e. 1 unit. Finally in Steps (13)-(14) we find the path and store the corresponding edges in E_s. In (15)-(18) we process the virtual edges, map them back on the original graph G and check if all capacity constraints are satisfied. It is necessary to check the validity of the solution because there are some rare instances where we could use an edge twice in the path, once as a "normal" edge and once as a part of a virtual edge. In this case we have to check if this edge had enough capacity to ensure the required throughput. If the edge does not have the required free capacity, we block the connection, otherwise in (21) we save the routing solution.

The reduced capacity graph can be calculated in $O(|E|)$ steps. Finding two disjoint paths can be done Suurballe's algorithm in $O(|E|+|V| \log_2 |V|)$ steps [84] and the shortest path can be found with Dijkstra in $O(|E| + |V| \log_2 |V|)$, too. The computational complexity of SRDC-CC is dominated by the graph transformation (Step (8)) which could be done in $O(|V||E| \log_{1+|E|/|V|} |V|)$. Saving the survivable routing and the checking function could be done in linear time. This means that the overall complexity of the SRDC-CC is $O(|V||E| \log_{1+|E|/|V|} |V|)$ i.e. the SRDC-CC has a lower complexity than the SRDC-CA. Note that the SRDC-CA guarantees a solution if it exists, however the SRDC-CC does not.

3.2.7 Experimental Results

In this section we investigate the bandwidth cost (in terms of Eq. (3.6)) of the different survivable routing approaches through simulations. We compare our methods to the theoretical lower bound [J1] and the most common deployed survivable routing scheme, the $1+1$ protection, which is the 2-approximation of the survivable routing problem [19] with assuming two data parts[4] in general and is a 4/3-approximation

[4] Note that the 2-approximation algorithm for feasible coding graphs presented in [74] gives $1 + 1$ for most practical scenarios.

of the SRDC-ICCN problem.

The following notations are used in the charts:

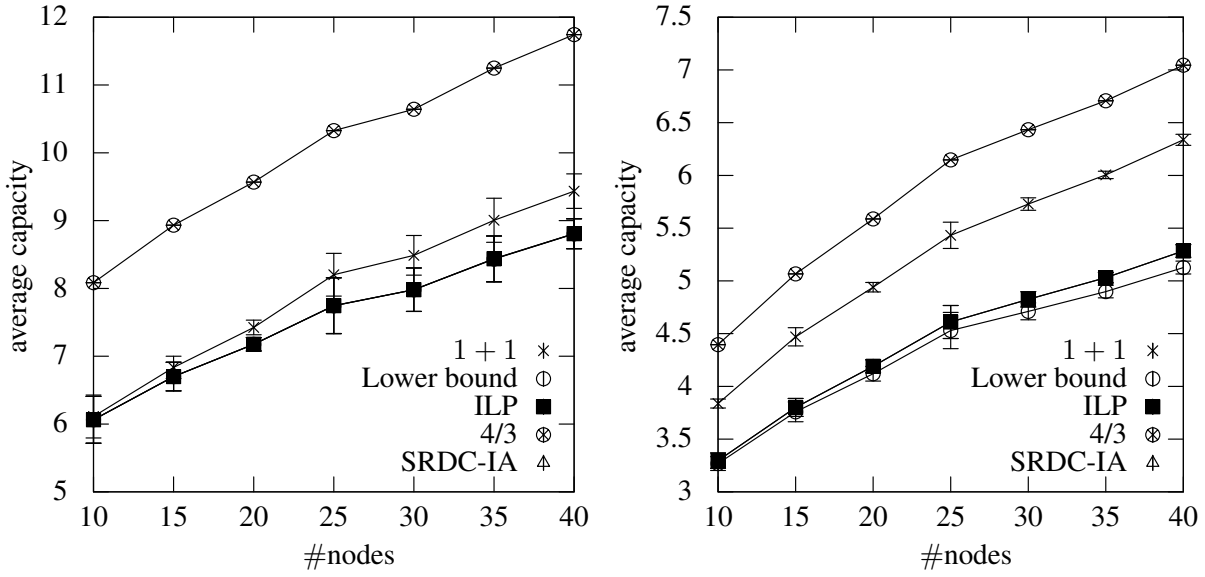
- SRDC-IA refers to Algorithm 1,
- SRDC-IC refers to Algorithm 2,
- SRDC-CA refers to Algorithm 3,
- SRDC-CC refers to Algorithm 4,
- 4/3 refers to the baseline of the approximation,
- and ILP refers to the Integer Linear Program presented in Section 3.2.6.
- The number in the parenthesis beside the algorithms i.e. (X) refers to the percentage of splitter merger nodes e.g. (10) means 10% of the nodes can be splitter/mergers. The letter "S" in the parenthesis indicates the smart node selection of the splitter/merger nodes (see Section 3.2.7. For example (S10%) means 10% of the nodes can be splitter/mergers and the nodes are selected according the smart selection.

We investigate random generated real-like planar $G = (V, E, k, c)$ topologies with different sizes and densities, and some real world transport network topologies, too. By the real-like topologies the simulation results are obtained by averaging several results from the topologies with the same properties, the confidence interval is 95%. When bottleneck links are introduced to the network, we identified a certain number of edges which are most prone to congestion based on their betweenness centrality value. We considered these edges as bottlenecks in the simulations (i.e., only a single capacity unit $k(e) = 1$ is available on these edges), thus, $1 + 1$ and SRDC-IA cannot use them in the survivable routing. Nonetheless the *two-connectedness of the network is preserved*, i.e., no blocking of the connection occurs. In other words only those edges are added to the set of bottleneck links which do not compromise the two-connectedness.

In addition, note that the different algorithms solve different problems, hence we can not compare all the SRDC subproblems i.e. algorithms fairly. It depends on the technological constraints which algorithms can be used. Of course where a comparison is possible and makes sense from the technological point of view we present it.

ICAN Experimental Results

First the simulation results for ICAN in sparse networks are presented in Figure 3.11a. The x-axis represents the node numbers of the networks, the y-axis shows the *average capacity allocation per connection*.



(a) Optimality gap in sparse graphs

(b) Optimality gap in maxplan graphs

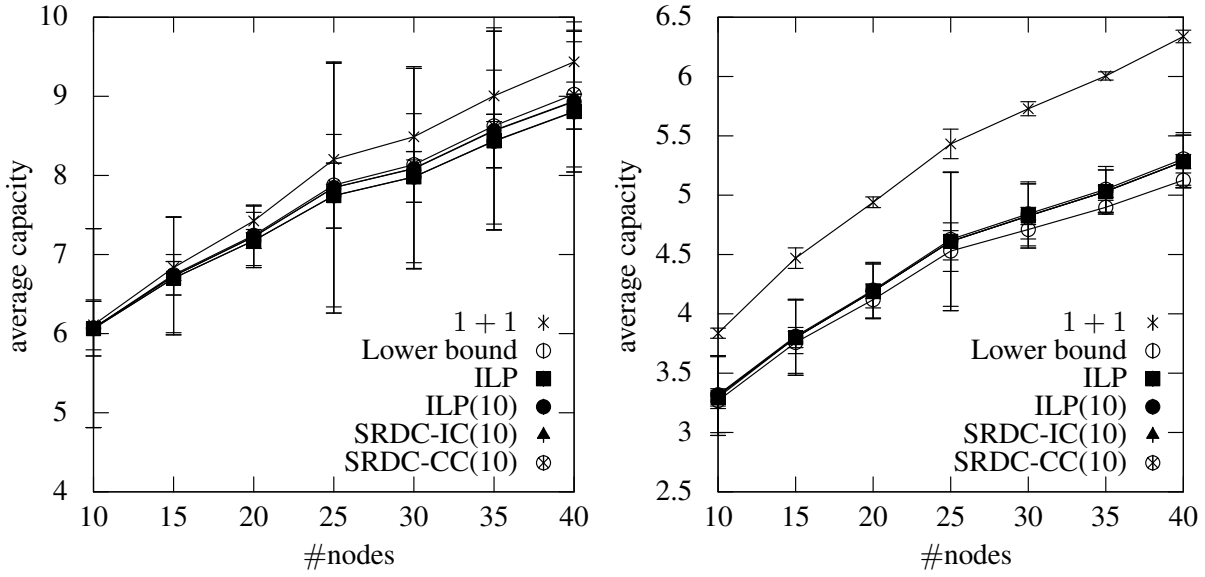
Figure 3.11: Bandwidth cost in sparse (average nodal degree between 2.4 and 3.2) and maximal planar (maxplan) graphs (average nodal degree between 4.2 and 5.7) with infinite capacities and no node capability constraints.

This is an excellent example why $1 + 1$ is still the most often deployed protection scheme, as the gap between the bandwidth cost of $1 + 1$ and the theoretical lower bound for survivable routing is small. However, our method outperforms $1 + 1$ even in this scenario, when $1 + 1$ performs well. In fact, the SRDC-IA and the ILP reach the theoretical lower bound, as optimal survivable routing can be reached with dividing connection data into two parts.

Figure 3.11b shows the simulation results in maximal planar graphs. It can be observed that all of our methods are approaching the theoretical lower bound. However, in these topologies the theoretical lower bound requires that connection data is divided into more than two data parts, which is not always feasible from a practical point of view. The gap between $1+1$ and our SRDC-IA is significant, especially if we take into consideration the fact, that the bandwidth cost reduction of 1 unit means that *every SRDC connection in the network uses one less bandwidth unit than* $1+1$, while still maintains its simplicity. As expected the SRDC-IA and the ILP deliver exactly the same results in both figures, since the SRDC-IA solves the problem optimally.

ICCN Experimental Results

By ICCN there are no capacity constrains in the network, however not all nodes are splitter/merger capable. In Figure 3.12 we show that just by upgrading 10% of the nodes we can achieve great improvement compared to the $1 + 1$, both in sparse (Figure 3.12a) and dense networks (Figure 3.12b). In this scenario



(a) Optimality gap in the node constraint case in sparse graphs (b) Optimality gap in the node constraint case in maxplan graphs

Figure 3.12: Bandwidth cost in sparse (average nodal degree between 2.4 and 3.2) and maximal planar (maxplan) graphs (average nodal degree between 4.2 and 5.7) in the infinite capacity, node capability constraint case.

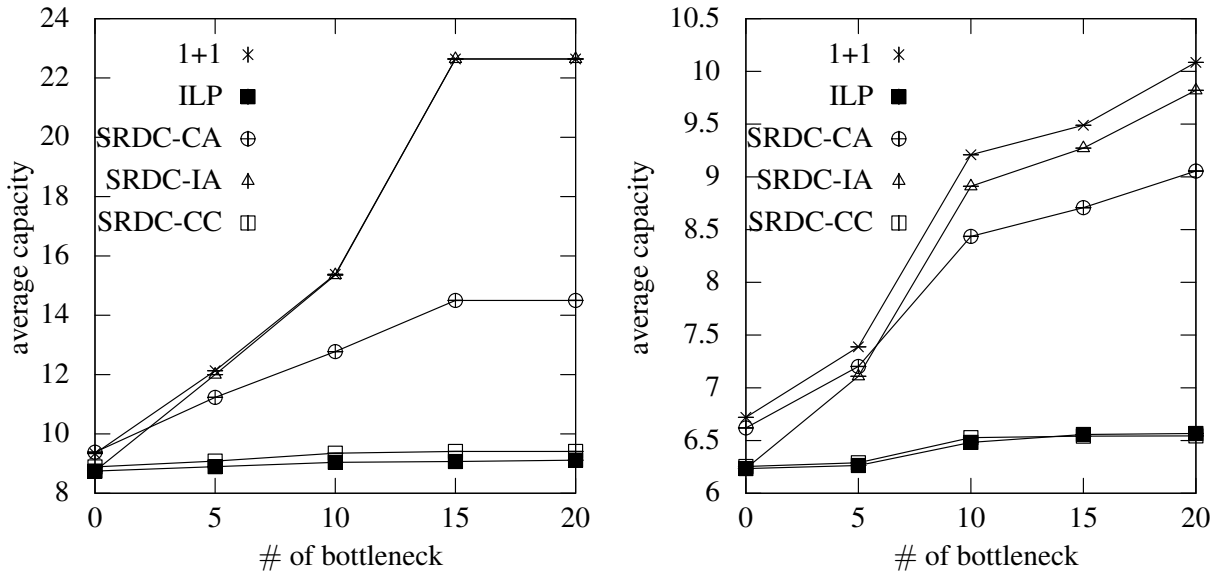
we consider 10% since it can be a realistic target value for the network operators for the first stage of the incremental deployment.

We can observe that our heuristic algorithm i.e. the SRDC-CC and the approximation algorithm i.e. the SRDC-IC performs very well, it is near to the optimal solution i.e the ILP(10). Of course with further upgrade of node capabilities we can better approximate the optimal solution i.e. ILP and *Lower bound*.

Note that the source and destination nodes are always considered to be splitter/merger capable and in addition nodes are upgraded randomly. Thanks to that in the dense networks (Figure 3.12b) the results are even better, since there are almost always 3-disjoint paths.

CCAN Experimental Results

In this section we investigate the CCAN subproblem i.e. capacity constraint all node capability case through the performance of our methods in real network topologies (SNDLib and Rocketfuel ASs [65, 80]). As alredy stated, in this scenario the edge capacities are constrained, i.e., we identified a certain number of edges which are most prone to congestion based on their betweenness centrality value. We considered these edges as bottlenecks in the simulations (i.e., only a single capacity unit $k(e) = 1$ is available on these edges), thus, $1 + 1$ and SRDC-IA cannot use them in the survivable routing. The x-axis represents the number of bottlenecks link in the networks, and y-axis shows the *average capacity allocation per connection*.



(a) Bandwidth cost in Cost 266 (37 nodes, 57 links with diameter 8) [65]

(b) Bandwidth cost in Sprintlink (AS1239) (30 nodes, 59 links, with diameter 6) [80]

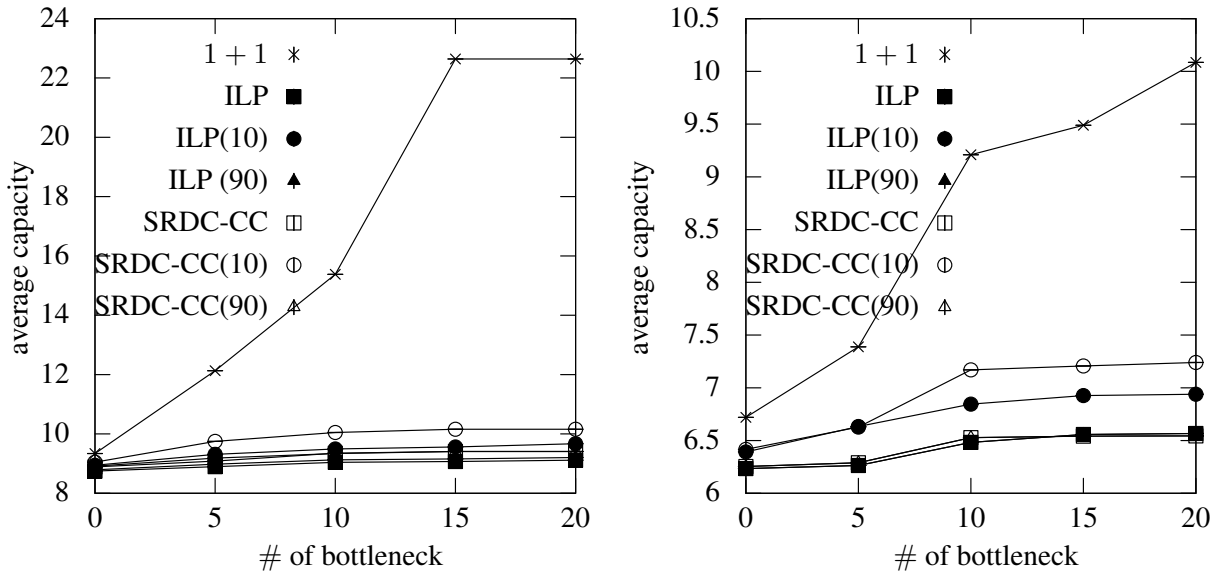
Figure 3.13: The bandwidth cost in real-world topologies in the capacity constraint all node capability scenario, depending on the number of bottleneck links.

One can observe in Figure 3.13 that as the number of bottleneck edges increases, the average bandwidth cost of $1+1$ and SRDC-IA increases dramatically, while the average bandwidth cost of SRDC-CA scales better in terms of the number of bottleneck links. However the SRDC-CC scales even much better, and provides in most cases the optimal solution. As expected when the number of bottleneck links is large the SRDC-CC blocks few connections (even in the worst case less than 2%). Note that the SRDC-CA guarantees a solution if it exists, however the SRDC-CC does not i.e. the SRDC-IC has a justification, too.

CCCN Experimental Results

By CCCN both the capacity and the node capabilities are constrained. This is the most demanding subproblem. In this case the splitter/merger nodes gain on importance. As in previous section, the CCCN subproblem is investigated through simulations in real network topologies (SNDLib and Rocketfuel ASs [65, 80]).

In Figure 3.14 we see both the effects of the bottleneck links and the effect of the incremental deployment. We show that with the increase of the percentage of the splitter/merger nodes the average capacity consumption improves i.e. decreases. We can also observe that the ILP and the SRDC-CC scales very well. The gap between the ILP and the SRDC-CC is not significant. In this case as by CCAN the $1+1$ performs poorly compared to the SRDC algorithms i.e. sometimes the capacity consumption of the $1+1$ is the double of the SRDC algorithms.



(a) Bandwidth cost in Cost 266 (37 nodes, 57 links with diameter 8) [65]

(b) Bandwidth cost in Sprintlink (AS1239) (30 nodes, 59 links, with diameter 6) [80]

Figure 3.14: The bandwidth cost in real-world topologies in the capacity and node capability constrained scenario.

Incremental Deployment

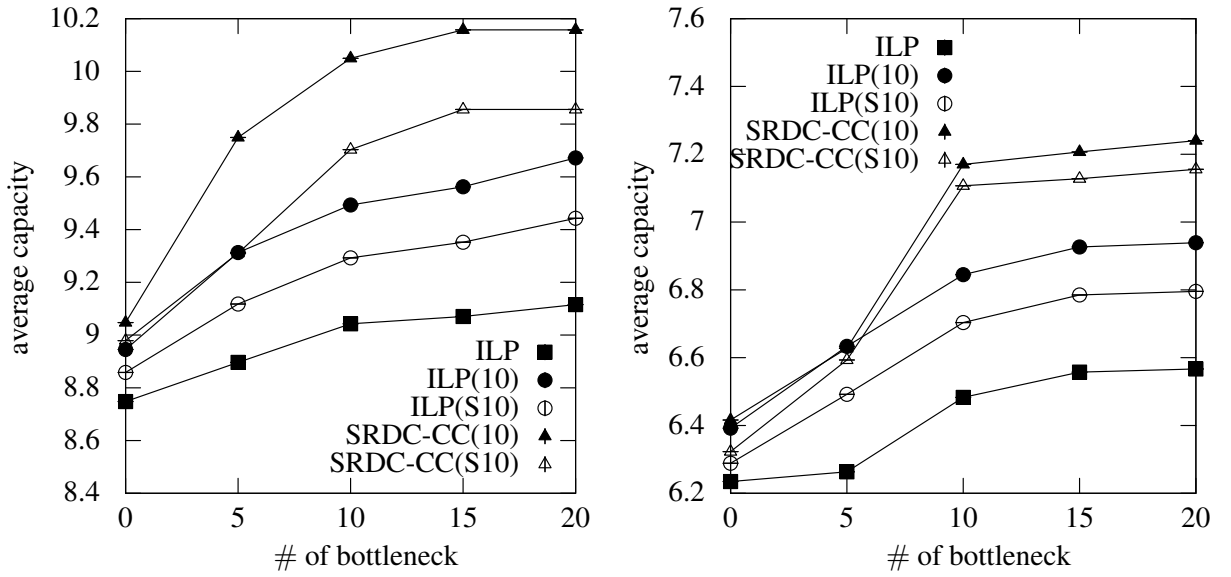
When discussing the SRDC subproblems we divided the problem according to the technological constraints in the network. We investigated the SRDC under capacity and node capability constraints. The free capacity on the edges is a dynamic parameter and depends on the traffic (and routing). However, the choice which nodes are upgraded i.e. which nodes have the ability to perform the splitting and merging action is static. This choice has to be made by the network operators.

In this section we investigate how the choice of the upgraded nodes influences the overall performance. As input given is a percentage of nodes we are allowed to upgrade. We compare two methods:

– Random method: The given percentage of nodes which are upgraded are selected randomly.

– Smart method: We run the SRDC-IA algorithm and for each node we count how many times the given node was utilized as a splitter/merger. We sort the nodes according the counter value and upgrade the first ones according the given percentage. Of course the nodes with the higher value are upgraded first.

In Figure 3.15 we can observe the impact of the smart selection. We can observe, that with the smart selection we are able to utilize the potential of SRDC more efficiently. In Figure 3.15 we show that also by real-word topologies we can achieve great improvement compared to the $1 + 1$ by only upgrading 10% of the nodes.



(a) Bandwidth cost in Cost 266 (37 nodes, 57 links with diameter 8) [65]

(b) Bandwidth cost in Sprintlink (AS1239) (30 nodes, 59 links, with diameter 6) [80]

Figure 3.15: The bandwidth cost in real-world topologies in the capacity and node capability constrained scenario. The smart and the random selection of nodes is compared.

3.3 Summary of Results

In this chapter we introduced two methods. First the so called Survivable Routing with Network Coding (SRNC) was introduced, a new practical (deployable), yet extremely resource efficient method for the arbitrary failure scenario. The method utilizes network coding to ensure resource efficiency. The user data is divided into exactly two parts keeping the protection approach simple from equipment and management point of view. However the network coding capability has to be added in each node and the method is NP-complete. Then the so called Survivable Routing with Diversity Coding (SRDC) was introduced, a new easily deployable and extremely resource efficient single link failure resilient method. The method is easily deployable since the *coding itself can be done in the source and destination nodes, i.e. there is no need for a node capability update* (contrary to the SRNC). Several subproblems of SRDC were investigated depending on capacity and node capability constraints in the network.

Note that Section 6.1.1 is devoted to the practical implementation of these new methods in SDN networks. It is shown that the splitting and merging action can be implemented easily with a software update. We show that even if a failure occurs, using our SRNC and SRDC approaches the source flow can be recovered instantaneously, without packet retransmission or flow rerouting. The recovery time only depends on the delay characteristics of the underlying topology. The high resilience of SRDC was also demonstrated in [56] through a video streaming application scenario, but it was also noted that the end-to-end delay and the delay difference should be considered in the routing problem. This observation inspired the Delay Aware Routing with Coding (DARC) problem discussed in Chapter 4.

3.3.1 Theses Summary

Thesis 1 *[C1, C2, J1] I proposed a new network coding based method called SRNC (Survivable Routing with Network Coding) which ensures* robust instantaneous recovery *for the arbitrary failure scenario. I proved that the problem is NP-complete at any finite user data splitting. According to that I proposed an Integer Linear Program (ILP) to solve the problem. For the single link failures scenario I proposed a capacity efficient, simple and low-complexity diversity coding based survivable routing method providing robust instantaneous recovery called SRDC (Survivable Routing with Diversity Coding). I investigated the method under capacity and node capability constraints (Table 3.2). I proposed several ILPs and heuristics to solve the different subproblems. I proved that the* $1+1$ *dedicated protection is a* $4/3$ *approximation of the subproblems without the capacity constraints (ICAN and ICCN). I showed through an example that the* $1+1$ *does not approximate the problem with capacity constraints (CCAN and CCCN). For the ICAN subproblem I proposed an approximation algorithm.*

Thesis 1.1 (SRNC complexity) *[C1, J1] I proved that by an arbitrary failure scenario the SRNC problem is NP-complete at any finite user data splitting. According to that I proposed an ILP to solve the problem.*

Thesis 1.2 (SRDC Integer Linear Programs) *[C2] I proposed a fast running ILP in order to solve the CCAN problem. I extended this ILP to solve the node capability constrained problem i.e. the CCCN problem, too.*

Thesis 1.3 (SRDC approximation) *I proved that the* $1+1$ *dedicated protection is a 4/3 approximation of the subproblems without the capacity constraints (ICAN and ICCN). I showed through an example that the* $1+1$ *does not approximate the problem with capacity constraints (CCAN and CCCN). For the ICCN subproblem I proposed an* $4/3$ *approximation algorithm.*

Thesis 1.4 (SRDC heuristics) *[C2] I proposed two heuristics for the CCCN and CCAN problems, i.e., the SRDC-CA which always provides a feasible solution (if exists) and the SRDC-CC which provides a faster and more efficient solution for most practical problem instances.*

Chapter 4

Delay Aware Survivable Routing

In Chapter 3.2 we introduced the Survivable Routing with Diversity Coding, a capacity efficient, simple and low-complexity diversity coding based survivable routing method providing robust and instantaneous recovery. The SRDC is easily deployable in SDN networks (see Section 6.1.1). However, the SRDC does not consider any end-to-end delay characteristics, which are getting more and more into spotlight [23] with the proliferation of multi-media and streaming applications. These new applications e.g. VoIP, 4K/8K video traffic, AR/VR (Augmented and Virtual Reality etc.) not only require high resilience from the underlying network, but are highly delay sensitive. Satisfying both constraints at the same time in a bandwidth-efficient way is unquestionably one of the most challenging tasks of service providers in transport networks.

This high resilience of SRDC was demonstrated in [56] through a video streaming application scenario. However, it was also noted that the end-to-end delay and the delay difference should be considered in the routing problem [C3] since it greatly influences the performance. Although several works are dealing with Quality-of-Service (QoS) routing [64] and differential delay (DD) aware [39] survivable routing in optical networks, all of the methods are designed for disjoint paths only. We generalize these results for diversity coding-based survivable routing using DAGs. Hence, we define the before- and after-failure (single link) delay of an end-to-end DAG, and investigate the effect of QoS routing and DD-aware routing delay constraints on these optimal survivable routing structures ensuring instantaneous recovery.

The rest of the chapter is organized as follows. In Section 4.1 the state-of-the-art i.e. the related work is presented focusing on previous QoS and Differential Delay (DD) aware routing results. In Section 4.2 the preliminaries and problem formulation for survivable routing is discussed in details, and the delay of a DAG is defined. We show a graph transformation to an equivalent survivable routing problem, too, which trace back our problem to finding disjoint paths with delay constraints in a special graph. Section 4.3 introduces our complexity analysis, integer linear program and approximability result for the survivable QoS routing problem, while the same is presented together with a heuristic solution for the differential delay aware routing problem in Section 4.4. Experimental results are shown in Section 4.5, and the chapter is concluded in Section 4.6.

4.1 Delay Awareness in Transport Networks

In this section we give an overview of the two most common delay related problems in transport networks, namely the so called QoS and the Differential Delay Aware Routing. These particular problems will be addressed in the rest of the chapter.

4.1.1 QoS Routing

Finding a minimum cost (or shortest) path while minimizing a single metric (e.g., cost or length) can be solved in polynomial time with Dijkstra's algorithm. On the other hand, satisfying an additional constraint (e.g., delay or jitter) along the path while minimizing its cost is a fundamental problem and arises in several application scenarios, referred to as Quality-of-Service (QoS) routing [96]. Note that, this problem is already NP-hard [29], called the shortest weight-constrained path problem (or restricted shortest path problem), where a minimum cost path is required between the source s and destination t, such that the delay of the path is lower than a pre-defined bound D. Several papers proposed heuristics in order to address the issue, in [43] a lagrange relaxation based method for the QoS routing problem was presented, in [73] a distributed heuristic solution, called the delay-constrained unicast routing (DCUR) algorithm was introduced. An exact solution can be found by pseudo-polynomial algorithms, thus, if the input parameters are bounded it can be solved in polynomial time through a standard dynamic programming approach [42]. Furthermore, ϵ-optimal fully polynomial approximation schemes (FPTAS) exist for the problem both for acyclic [36] and general graphs [59].

The problem of finding two link-disjoint paths while minimizing the total cost of the paths is solvable in polynomial-time, e.g., with the Suurballe-Tarjan [85] algorithm. In order to extend this work to QoS routing, in [64] the 2-Restricted Link Disjoint Paths (2DP) problem was introduced where the total cost of the paths was minimized while both paths have to obey a specific delay bound D. Note that, even if we assume that each link has zero cost, it is NP-hard to find a solution that does not violate the delay constraint D of at least one of the paths [64] (following from the fact that finding a disjoint path-pair while minimizing the delay of the longer path is NP-hard [58]). Furthermore, no polynomial-time algorithm exists which can approximate the delay within a factor of $1 \leq \alpha < 2$ [64]. However, a 2-approximation on the delay-bound can be provided with minimizing the total delay of the disjoint path-pair, e.g., with the Suurballe-Tarjan [85] algorithm. Thus, in [64] four bifactor approximation algorithms were presented, i.e., both the total cost and the total delay of the path-pair can be bounded with a constant factor. The first approximation algorithm 2DP-1 uses standard flow techniques, i.e., employs the path augmenting approach to find the restricted shortest paths [59] in the residual network with a delay bound of D and $2D$ for the first and second path, respectively. Thus, the approximation ratio is 1.5 on the delay, and it is a slightly larger factor on the cost. The subsequent approximation algorithms reduce the computational complexity and the delay violation at the expense of higher total cost.

The problem of finding a set of k link-disjoint paths from s to t, such that the total cost of these paths

is a minimal and that the delay for each path is not greater than a specified bound D was introduced in [99]. Of course, this problem contains the problem of 2DP [64], thus, it is also NP-hard. The approximation ratios of [64] were generalized for k paths and improved as well in [33]. Besides obeying a delay bound for each individual path, in [99] a more general network programming based approach was presented for finding k constrained shortest link-disjoint paths, such that the *overall delay of these paths* should be lower than a specified bound (kD). Note that, the approximation algorithms in [64] also relax the problem of 2DP with delay constraint D on both paths to a minimum constrained flow problem with a total delay constraint of $2D$ on the solution.

4.1.2 Differential Delay Aware Routing

Although the delays of individual paths in QoS routing is an important question, from a practical point of view the difference between the path delays could be a more serious issue in some application environments. For example, with the deployment of next-generation SONET/SDH technology virtual concatenation (VC) enabled service providers to split the traffic of a single circuit into multiple finer granularity parts, and route these parts along multiple paths. However, beside the several advantages the application of VC provides, it introduces *differential delay (DD)* among the diversely routed paths as well, which boiled down to the issue of increased buffer size at the destination node for DD compensation. In order to avoid service degradation, differential delay of the paths should be considered in the routing problem, as in optical networks the maximal DD compensation is about 125 ms with off-chip SDRAM technology [39].

The authors in [5] introduced the Two-Sided Constrained Path (TSCP) problem, where the task is to decide whether a new VC can be added to a VC group. Formulating with the DD constraint, the task is finding a path with overall delay of D between a given minimum and a maximum bound, i.e., $D_{min} < D < D_{max}$. It was proved that the TSCP problem is NP-hard [5]. In [82], the DD is defined as the difference between the delay of the highest and smallest delay paths. In their Differential Delay Routing (DDR) problem the task is to find a given number of paths, while their DD is lower than a pre-defined delay bound. It was shown that minimizing the delay difference of paths in DDR is not only NP-hard but provably hard to approximate within a constant factor. In [81] the same authors introduce the cumulative differential delay, which is the sum of the differences of delays of all the paths of a solution compared to the highest delay path. Also heuristic approaches were introduced like the Sliding-Window Algorithm and the Cost-Based Algorithm, which are based on finding k-shortest paths.

In the previous works the objective function was to minimize the differential delay, while neither link costs nor the disjointness of the paths were considered in the optimization. The study in [39] extended DD aware multi-path routing with survivability. Contrary to the differential delay problems above, their goal is to minimize the total cost of the paths while the disjointness of these paths is required in order to ensure single link failure resilience. The mathematical formulation of the survivable multi-path DD constrained routing problem has been presented in [39], where a DD bound has to be satisfied between

each pair of the k paths. The NP-completeness of this problem follows from the DDR problem [82]. Thus, two heuristic approaches were introduced based on k-shortest link-disjoint paths, both of them following the Shared Protection of the Largest Traversed link (SPLIT) approach.

4.2 Delay Aware Routing with Coding: The Problem Formulation

In the *Delay Aware Routing with Coding (DARC)* problem the network is represented by a directed graph $G = (V, E, k, c, d)$ with node set V, link (arc) set E, and three additional attributes i.e. free capacity $k(e)$, cost $c(e)$ and the delay value $d(e)$.

As by SRDC a dynamic routing scenario is considered, where traffic demands arrive one after the other, without any knowledge of future incoming request, i.e., each request is routed independently. Thus, as part of the input of the delay aware survivable routing problem a single *connection request* $C = (s, t, b, D)$ is given, which consists of the source node $s \in V$, destination node $t \in V$, the number of bandwidth units b requested for data transmission, and a maximal delay bound D which has to be satisfied by the routing of the request to ensure a given QoS or differential delay for the connection.

4.2.1 Defining the Delay of a Routing DAG

We have already discussed that the minimum cost SRDC solution has a very strict structure (also demonstrated in Figure 4.1), i.e., each routing DAG consists of series of paths ($\mathcal{P}$) and disjoint path-pairs, called **islands** ($\mathcal{I}$). Furthermore, each island is part of at most one routing DAG. As the same routing structure must be satisfied by DARC, we assume that the routing DAGs in a DARC solution can be decomposed into subsequent paths and islands, too. In this section we will define the delay of a routing DAG and we will introduce a graph transformation in Section 4.2.2 which trace back the survivable routing problem to finding minimal cost routing DAGs consisting of paths and islands obeying specific delay bounds.

An example is presented in Fig. 4.1 to demonstrate the effect of a failure regarding the connectivity and the end-to-end delay. If a link with $f(e) = 1$ fails, e.g., (s, v_1), it disrupts only a single routing DAG (E_B). Further note, that the failure of a link with $f(e) = 2$, e.g., (v_1, v_3) affects two routing DAGs, but in different ways: while routing DAG E_B is disrupted, only the end-to-end delay increases for the other routing DAG $E_{A\oplus B}$. This is because E_B is simple path $s \to v_1 \to v_3 \to v_5 \to v_7 \to t$ in Fig. 4.1, and when link (v_1, v_3) fails, the path is disrupted. On the other hand, $E_{A\oplus B}$ is not a simple path, it contains a segment between v_0 and v_9 which is a link-disjoint path-pair. In the operational state (i.e., no failure occurs) the data is transmitted along the lower delay path between v_0 and v_9, that is $v_0 \to v_1 \to v_3 \to v_6 \to v_8 \to v_9$ with total delay of 5 units. If (v_1, v_3) fails this path can not be used anymore, but the data flow of $E_{A\oplus B}$ still reaches destination t on the other path of the segment between v_0 and v_9, that is $v_0 \to v_2 \to v_4 \to v_5 \to v_7 \to v_9$ with total delay of 7 units. Thus, $s - t$ connectivity is still ensured without any reconfiguration of the network. Note that, in this example the end-to-end delay

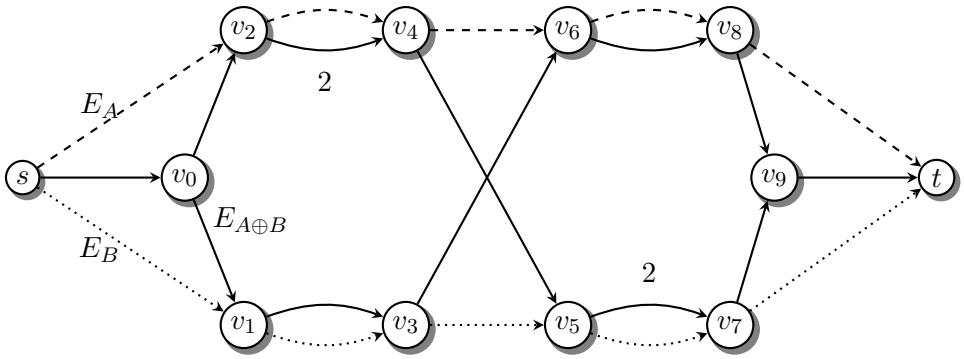


Figure 4.1: A survivable SRDC solution for request $C = (s, t, 2, -)$ with the corresponding routing DAGs E_A, E_B and $E_{A\oplus B}$. Links with $f(e) = 2$ are duplicated. Link costs are unit. The link delays are $d(e) = 1$, otherwise written next to the arc [J2].

between s and t increased from 7 to 9 units on routing DAG $E_{A\oplus B}$ in Fig. 4.1 upon the failure of link (v_1, v_3).

It is important the emphasize that the implementation of the merger node (i.e., which selects among the two identical copies of the same data part arriving on the two paths of an island, e.g., v_9 of $E_{A\oplus B}$ in Fig. 4.1) is crucial to ensure instantaneous recovery [J1]. In order to eliminate signaling from the recovery process, a link failure should be oblivious to the merger node, i.e., the merger has to switch from the failed path autonomously to the operating path. The merger node implementation (see Section 6.1.1) solves this issue by tracking of the highest sequence number of the forwarded packets. A packet is only forwarded on the merger's outgoing link if its sequence number is larger than the current (highest) sequence number. Of course upon forwarding the highest sequence number is updated. As a result, a merger forwards the packets from the "faster" path from the two disjoint paths of an island (I_{min}) in a failure-less state, and discards the duplicates that arrive on the "slower" path of the island (I_{max}). On the other hand, if a failure occurs on the faster path (as we have seen in Fig. 4.1 after the failure of (v_1, v_3)), the merger will forward the packets arriving on the slower path on its outgoing link automatically. Note that, at switching some jitter could occur on the routing DAG owing to the delay difference between the last packet arrived on I_{min} and the first forwarded packet from I_{max}.

In order to capture the before- and after-failure delay characteristics of the routing DAGs, we introduce two delay values for each island I: $d^I_{min} = \sum_{e\in I_{min}} d(e)$ corresponding to the delay of the island in the failure-less state (i.e., the faster path); and the delay difference between the two disjoint paths $\Delta^I = \sum_{e\in I_{max}} d(e) - \sum_{e\in I_{min}} d(e)$ corresponding to the delay increase between the splitter and merger node of the island upon a failure occurs on the faster path. Thus, the *end-to-end delay of a routing DAG* can be modeled as

$$\delta_{E_j} = \sum_{P\in\mathcal{P}_{E_j}} \sum_{e\in P} d(e) + \sum_{I\in\mathcal{I}_{E_j}} d^I_{min} \tag{4.1}$$

in the failure-less state, while it increases to

$$\Delta_{E_j} = \delta_{E_j} + \max_{I \in \mathcal{I}_{E_j}} \Delta^I \tag{4.2}$$

in worst case upon a failure along the island with the largest delay difference between its two paths[1]. In the example in Figure 4.1, $\delta_{E_{A\oplus B}} = 7$ because the delay of $s \to v_0 \to v_1 \to v_3 \to v_6 \to v_8 \to v_9 \to t$ is 7 units. For $E_{A\oplus B}$ the $\max \Delta^I = 2$ because the delay difference between the two disjoint paths $v_0 \to v_1 \to v_3 \to v_6 \to v_8 \to v_9$ and $v_0 \to v_2 \to v_4 \to v_5 \to v_7 \to v_9$ between the splitter node v_0 and the merger node v_9 of the single island in $\mathcal{I}_{E_{A\oplus B}}$ is 2 units. Of course this means $\Delta_{E_{A\oplus B}} = 9$.

Further note that, besides instantaneous recovery diversity coding provides additional benefits to the connections in the failure-less state as well. On one hand, in multi-path routing (or shared path protection approaches [39, 45]) we have to wait for the highest delay path to reconstruct user data. On the other hand, the additional redundancy provided by diversity coding ensures that the two lower delay paths determine the delay of the connection (as we can *reconstruct user data from arbitrary two of the three routing DAGs* with low additional coding complexity owing to the applied simple XOR codes [J1]). This may lead to lower overall end-to-end delay for the applications in the failure-less state, and also results lower jitter and differential delay increase caused by protection switching.

4.2.2 An Equivalent Delay Aware Survivable Routing Problem

Using the polynomial graph transformation proposed in Section 3.2.3 the minimum cost survivable routing problem of SRDC-ICAN was traced back to finding three link-disjoint $s-t$ paths with minimum total cost in an auxiliary graph G^*. In this section we extend the transformation with the delays of the routing DAGs, which enables the incorporation of additional delay constraints for survivable QoS routing and DD aware routing.

The input of DARC is the graph $G = (V, E, k, c, d)$ with delay values $d(e)$ for every link e. An auxiliary (multi-)graph $G^* = (V, E^*, c^*, d_{min}, \Delta)$ is created, where:

- node set V in G^* is the same as in G, while
- link set E^* contains the *original links* of G. Additional *virtual links* $e_{(u,v)}$ are added between every pair of distinct node-pairs for which a link-disjoint path-pair exist, i.e., representing a potential island I with splitter node u and merger node v.
- The cost of $c^*(e_{(u,v)})$ is set to the cost of a minimum cost link-disjoint path-pair between nodes u and v in G (calculated with Suurballe's algorithm [84]). The original links of G have the same cost ($\forall e \in E : c^*(e) = c(e)$).
- In addition to the previous transformation for SRDC, in DARC we have to capture the routing DAG delays in Eq (4.1)-(4.2). Thus, two variables d^I_{min} and Δ^I are introduced for each virtual

[1] Note that, $\Delta_{E_j} - \delta_{E_j}$ gives also the largest jitter we might expect along the routing DAG caused by protection switching.

link (island) $I = e_{(u,v)}$ (i.e., the delay of I_{min} and the delay difference between I_{min} and I_{max}). For original links $e \in E$ we define $d^e_{min} = d(e)$, and $\Delta^e = 0$.

For example, 10 virtual links are added in Fig. 4.1, e.g., the virtual link $e = (v_0, v_9)$ representing potential island with splitter v_0 and merger v_9 has $c^*(e) = 12$, $d^e_{min} = 5$, $\Delta^e = 2$. Note that, if the original graph G was 2-link-connected, then G^* contains the original links and a full mesh of virtual links.

We assume that there are no capacity or node capability constraints in the network i.e. $\forall e \in E$: $k(e) = 2$ (or greater) and all nodes are updated. In this case the optimal survivable routing in G^* – that is, three link-disjoint paths with minimum total cost – either with or without obeying delay bounds can be easily transformed back to the routing DAGs in G. The links of the routing DAGs are the links of the three paths in G^*, with the difference that the virtual links $e_{(u,v)}$ in G^* are replaced with the original links of the corresponding disjoint path-pairs (islands).

We already discussed that the SRDC problem (without capacity or node capability constraints i.e. ICAN) is polynomial-time solvable [84]. However, this will not be true for DARC with additional delay constraints, as we have seen that finding link-disjoint paths while obeying some additional delay constraint is already an NP-complete problem [29, 39, 64, 82]. We note here that, we can use the transformed graph to run algorithms, but owing to the correlation between the links and link parameters of G^* *the hardness results can not be directly transformed back to our original survivable routing problem*. Thus, further investigation is required to show the complexity of survivable QoS routing and DD aware routing problems, done in Section 4.3 and Section 4.4, respectively.

4.3 Computing Routing DAGs for QoS Routing

In this section we investigate the survivable routing problem when delay constraints are given for the individual DAGs to ensure QoS routing (DARC-QoS). Formally, the task is to minimize the total cost in terms of Eq. (3.6), while the after-failure delay for each routing DAG is less than a given bound D [64]:

$$\forall j \in \{A, B, A \oplus B\} : \Delta_{E_j} \leq D \, .$$

First, the complexity of DARC-QoS is discussed in Section 4.3.1, and it is shown that DARC-QoS is NP-complete. Thus, in Section 4.3.2 we present an Integer Linear Program (ILP), while in Section 4.3.3 the approximability of the problem is discussed.

4.3.1 Complexity Analysis of DARC-QoS

In this section we show that DARC-QoS is NP-complete. The proof is based on the reduction from the Three-Way Partition problem [29], which is known to be NP-hard (following also from the fact that in an optimal three-way partition the items in arbitrary two subsets are partitioned into two equal parts).

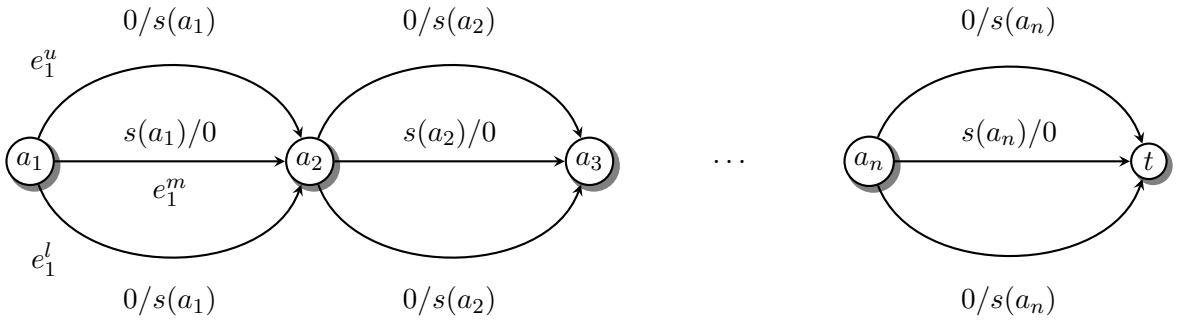


Figure 4.2: Transformation of Three-Way Partition [29] to DARC-QoS. Link delays/link costs are shown next to the links. Each link has $k(e) = 2$ units of free capacity [J2].

Theorem 4.3.1 *To decide whether a $\leq Z$ cost solution for DARC-QoS exists is NP-complete.*

Proof: DARC-QoS is in NP, a solution with $\leq Z$ cost with delay $\leq D$ on each routing DAG is a proof.

Assuming we are given an instance of the Three-Way Partition Problem [29], [52], that is, a finite set A of items with size $s(a) \in \mathbb{Z}^+$ for each $a \in A$. Let us denote $T = \sum_{a \in A} s(a)$. Is there a perfect three-way partition of set A, i.e., partitions P_1, P_2, P_3 such that $P_1 \cup P_2 \cup P_3 = A$, $\sum_{a \in P_1} s(a) = \sum_{a \in P_2} s(a) = \sum_{a \in P_3} s(a) = T/3$ and $P_1 \cap P_2 = \emptyset$, $P_1 \cap P_3 = \emptyset$, $P_2 \cap P_3 = \emptyset$?

The polynomial time transformation for Three-Way Partition with $|A| = n$ to DARC-QoS is given as follows (shown in Figure 4.2). We construct a graph with $n + 1$ nodes using the following gadget for each a_i: we add three links (upper, middle and lower) $e_i^u = e_i^m = e_i^l = (a_i, a_{i+1}), i = 1,2,\ldots,n$ (with $t = a_{n+1}$). We define link delay and link cost values according to $s(a_i)$, as shown in Figure 4.2. We define $Z = 2T$ and $D = T/3$ for the connection request $C = (s = a_1, t,2, D)$. Next, we show that the two instances are solvable at the same time.

($\Rightarrow$) First, we show how to convert a DARC-QoS solution with $\leq Z$ cost to a Three-Way Partition of A. From the survivability aspect we know that for every gadget $f(e_i^u) + f(e_i^l) \geq 2$, i.e., the remaining flow should be at least two after the failure of e_i^m. If we sum up this for all gadgets, this leads to $\sum_{e \in E} f(e_i^u) + f(e_i^l) \geq 2T$. However, the total cost is $\leq 2T$, thus, $\forall i : f(e_i^u) + f(e_i^l) = 2$ follows. Furthermore, $f(e_i^m) \geq 1$ follows as well because of the survivability aspect (i.e., the remaining flow should be at least two either e_i^u or e_i^l fails). On the other hand, from $D = T/3$ on the individual routing DAGs we know that the total delay suffered by the three routing DAGs is $\leq T$. Thus, $f(e_i^m) \leq 1$ follows, because otherwise this total delay bound, hence, the delay bound of the individual routing DAGs could not be satisfied. So at the end $f(e_i^m) = 1$, which leads to $f(e_i^u) = 1$ and $f(e_i^l) = 1$ to maintain survivability. This means that a DARC-QoS solution with $\leq Z$ cost, i.e., the three routing DAGs E_A E_B and $E_{A \oplus B}$ are three $s-t$ paths. As the total delay of these routing DAGs is T (as e_i^m is traversed exactly by one routing DAG in each gadget), their delay is exactly $T/3$. As the delay of e_i^m was defined as the weight $s(a_i)$, the partition of a_i is defined by the routing DAG which traverses e_i^m.

($\Leftarrow$) In the other direction, it is easy to convert a Three-Way Partition into three routing DAGs with $\leq D$ delay, i.e., in this case to three $s-t$ paths. Let assume the three partitions P_1, P_2 and P_3 are given. Without a loss of generality, we assume that

- the single $s-t$ path in $\mathcal{P}_{E_A}$ is using $\forall a_i \in P_1 : e_i^m$, and e_i^u otherwise,
- the single $s-t$ path in $\mathcal{P}_{E_B}$ is using $\forall a_i \in P_2 : e_i^m$, and e_i^l otherwise,
- and the single $s-t$ path in $\mathcal{P}_{E_{A\oplus B}}$ is using $\forall a_i \in P_3 : e_i^m$, and otherwise the remaining available link of the gadget.

Thus, a Three-Way Partition gives a DARC-QoS a solution with $\leq Z$ cost. This concludes the proof as the two problems are solvable at the same time. ■

4.3.2 Integer Linear Program for DARC-QoS

Here, we present an ILP formulation for DARC-QoS based on [C2, C3] for finding three minimum cost link-disjoint paths in $G^* = (V, E^*, c^*, d_{min}, \Delta)$ with additional delay bounds required for QoS routing. The connection request is $C = (s, t, b = 2, D)$. The *three paths corresponding to the routing DAGs* are denoted as $w \in \{A, B, A \oplus B\} = \mathcal{W}$, respectively. Binary variables $x^w(e)$ are used to indicate the paths for each routing DAG. Our objective is to minimize the total bandwidth cost in terms of Eq. (3.6):

$$\min \sum_{w \in \mathcal{W}} \sum_{e \in E} c^*(e) \cdot x^w(e). \tag{4.3}$$

The following constraints are required to find a survivable routing:

$$\forall w \in \mathcal{W}, \forall i \in V\text{:}$$

$$\sum_{(i,j)\in E} x^w(i,j) - \sum_{(j,i)\in E} x^w(j,i) = \begin{cases} 1 & \text{, if } i = s \\ -1 & \text{, if } i = t \\ 0 & \text{, otherwise} \end{cases}, \tag{4.4}$$

$$\forall e \in E\text{:} \sum_{w \in \mathcal{W}} x^w(e) \leq 1, \tag{4.5}$$

$$\forall w \in \mathcal{W},\ \forall e \in E\text{:}\ x^w(e) \cdot \Delta^e \leq y^w. \tag{4.6}$$

Constraint (4.4) formulates the flow conservation for each path w. Constraint (4.5) ensures the disjointness of the paths. Constraint (4.6) gives a lower bound for the integer variable y^w, which captures the worst case delay increase of path (routing DAG) w upon a single link failure formulated in Eq. (4.2). For DARC-QoS, we have to add Constraint (4.7), which ensures that the maximal delay of the routing DAGs is less than the specified bound:

$$\forall w \in \mathcal{W}\text{:} \sum_{e \in E} x^w(e) \cdot d_{min}^e + y^w \leq D. \tag{4.7}$$

From the paths defined by the binary variables $x^w(e)$ in G^* the routing DAGs in G can be easily obtained by replacing the virtual links by the original links of the corresponding link-disjoint path-pair.

Table 4.1: The end-to-end delay difference on the two fastest routing DAGs upon a single link failure (wlog $\delta_{E_A} \leq \delta_{E_B} \leq \delta_{E_{A\oplus B}}$) [J2].

disrupted \ inc. delay	$\emptyset$	E_A	E_B	$E_{A\oplus B}$
$\emptyset$	$\lvert\delta_{E_A} - \delta_{E_B}\rvert$	$\lvert\delta_{E_B} - \min\{\Delta_{E_A}, \delta_{E_{A\oplus B}}\}\rvert$	$\lvert\delta_{E_A} - \min\{\Delta_{E_B}, \delta_{E_{A\oplus B}}\}\rvert$	$\lvert\delta_{E_A} - \delta_{E_B}\rvert$
E_A	$\lvert\delta_{E_B} - \delta_{E_{A\oplus B}}\rvert$	–	$\lvert\Delta_{E_B} - \delta_{E_{A\oplus B}}\rvert$	$\lvert\delta_{E_B} - \Delta_{E_{A\oplus B}}\rvert$
E_B	$\lvert\delta_{E_A} - \delta_{E_{A\oplus B}}\rvert$	$\lvert\Delta_{E_A} - \delta_{E_{A\oplus B}}\rvert$	–	$\lvert\delta_{E_A} - \Delta_{E_{A\oplus B}}\rvert$
$E_{A\oplus B}$	$\lvert\delta_{E_A} - \delta_{E_B}\rvert$	$\lvert\Delta_{E_A} - \delta_{E_B}\rvert$	$\lvert\delta_{E_A} - \Delta_{E_B}\rvert$	–

4.3.3 Approximability of DARC-QoS

It was shown that it is NP-hard to find two link-disjoint paths that does not violate the delay constraint D of at least one of the paths [64], and this problem even can not be approximated within a factor of 2 on the delay-bound. However, Suurballe's [84] algorithm which minimizes the total delay of the two paths provides a 2-approximation on the delay. On the other hand, for our survivable routing problem – finding three link-disjoint paths which minimizes the average (or total) delay of the routing DAGs – the same algorithm [84] provides only a 3-approximation on the delay, without considering the solution cost in the optimization problem. Although the solutions of the bifactor approximation algorithms [33, 64] gives bound on the total cost of the solution, the two delay parameters on the links in DARC-QoS can not be easily incorporated in the path augmentation approach for finding restricted shortest paths. Thus, we cannot hope an efficient algorithm which obeys the delay bound in G^* on each of the three paths. Hence, if strict delay bounds have to be satisfied in DARC-QoS, we suggest the usage of the ILP in Section 4.3.2, which runs in reasonable time for several instances owing to the small number of constraints on the path delays.

4.4 Computing Routing DAGs for Differential Delay Aware Routing

In this section we investigate the survivable routing problem when delay bounds are given on the end-to-end delay differences of the routing DAGs (DARC-DD). The task is to minimize the total cost in terms of Eq. (3.6), while all possible DD bounds in Table 4.1 have to be satisfied. These constraints ensure that the delay difference between the two fastest routing DAGs is under a specific bound $\leq D$ [39] corresponding to the maximal buffer size when an arbitrary single link failure occurs. Note that, a single link failure can cause the disruption of a routing DAG (if a link in a path is failed) or increase its end-to-end delay (if the link failure affects one of its islands). Furthermore, if a link with $f(e) = 2$ fails (e.g., (v_1, v_3) in Figure 4.1), it could happen that one routing DAG is disrupted while the end-to-end delay is increased for another routing DAG. First, the complexity of the problem is discussed in Section 4.4.1, and it is shown that DARC-DD is NP-complete. Thus, in Section 4.4.2 we present an Integer Linear Program (ILP), while in Section 4.4.3 its approximability is discussed. Finally, in Section 4.4.4 heuristic approaches

are proposed to solve the DD-aware survivable routing problem on the transformed graph introduced in Section 4.2.2.

4.4.1 Complexity Analysis of DARC-DD

In this section the NP-completeness proof of DARC-DD is provided. The proof is based on the polynomial reduction from the Longest Path Problem [29], which is known to be NP-complete.

Theorem 4.4.1 *To decide whether a $\leq Z$ cost solution for DARC-DD exists is NP-complete.*

Proof: DARC-DD is in NP, a solution with $\leq Z$ cost with $\leq D$ delay difference between the routing DAGs is a proof. Assuming we are given an instance of the Longest Path Problem [29], that is, a directed graph $G = (V, E)$ with length $l(e) \in Z^+$ for each $e \in E$, a positive integer K, and specified vertices s and t. It should be decided whether there is a simple path in G from s to t of length K or more? We use the version of the problem when $\forall e \in E : l(e) = 1$, which is still NP-complete [29].

The polynomial time transformation for Longest Path Problem to DARC-DD is given as follows (shown in Figure 4.3). We add two nodes s' and t' to graph G, and connect them to node s and t with links $e_s = (s', s)$ and $e_t = (t, t')$, respectively. Also a link $e_n = (s', t')$ is added between s' and t'. The transformed graph is denoted as $G' = (V', E')$. The cost of all links in G' is $\forall e \in E' : c(e) = 1$. The delay of the links in G' is set to the length of the links in G (i.e., $\forall e \in E : d(e) = 1$), while the delay of additional links e_s and e_t is set to zero. The delay of $e_n = (s', t')$ is set to value $n = |V|$, which is larger than the length of the longest simple path in G (if the graph is Hamiltonian the longest path is $n-1$). The DARC-DD problem can be formulated in $G'(V', E')$ as follows: Does there exist a DARC-DD solution with $\leq Z = 2n + 4$ cost and $\leq D = n - K$ differential delay bound for all possible failure scenarios in Table 4.1 for the connection request $C = (s', t', 2, D)$? We show that the two problems have a positive answer at the same time.

($\Rightarrow$) First, we show how to obtain a longest path in G from a DARC-DD solution in G'. Because of the survivability aspect we know that the links e_n, e_s and e_t have the flow value of $f(e_n) = f(e_s) = f(e_t) = 2$ in any DARC-DD solution (as upon the removal of either of them there should remain an $s' - t'$ flow with at least value 2). Thus, s' is a splitter node of an island and a starting node of two path segments. Hence, a routing DAG, without loss of generality $E_{A \oplus B}$ consists of a single path segment $\mathcal{P}_{E_{A \oplus B}} = \{e_n\}$ with $\delta_{E_{A \oplus B}} = \Delta_{E_{A \oplus B}} = n$. Note that, $E_{A \oplus B}$ has the highest δ and Δ delay among the routing DAGs (the delay of simple paths not containing e_n is at most $n - 1$). Thus, in order to satisfy all constraints in Table 4.1 it is enough if both $\delta_{E_A} \geq K$ and $\delta_{E_B} \geq K$.

The second routing DAG, e.g., E_A is a single island $\mathcal{I}_A = \{s' \rightarrow t'\}$ with link-disjoint path-pair I^A_{max} and I^A_{min}, where $\Delta_{E_A} = n$ (delay of $I^A_{max} = e_n$). In order to satisfy the DD bound the delay δ_{E_A} of the simple $I^A_{min} = s' \rightarrow s \rightarrow t \rightarrow t'$ path in G' should be at least K. This is exactly the delay of the $s \rightarrow t$ segment of I^A_{min} in G as both $d(e_s) = d(e_t) = 0$. The third routing DAG E_B has path segments e_s and e_t, and could have an arbitrary structure of paths and islands inside G. However, we only have to

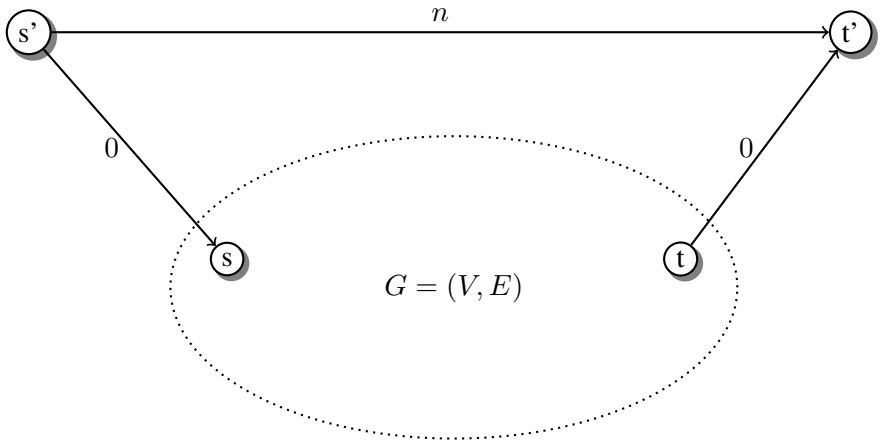


Figure 4.3: Polynomial-time reduction of the Longest Path problem to DARC-DD. Link delays are depicted next to the additional links, while it is set to the length of the links in $G = (V, E)$ [J2].

show that a routing DAG with $\delta_{E_B} \geq K$ exists. In fact, path $\mathcal{P}_{E_B} = \{I^A_{min}\}$ is always a feasible option as routing DAG E_B. Thus, all constraints in Table 4.1 are satisfied with the above routing DAGs with a maximal cost of $2n + 4$. Furthermore, the $s \to t$ segment of I^A_{min} is a path in G with at least length K.

($\Leftarrow$) In the other direction, it is easy to convert the longest path solution to three routing DAGs. Let the longest path between s and t be denoted with P_{s-t}. The three routing DAGs are the following: $\mathcal{P}_{E_{A\oplus B}} = \{e_n\}$, $\mathcal{P}_{E_B} = \{s' \to P_{s-t} \to t'\}$ and the third DAG is an island $\mathcal{I}_{E_A} = \{s' \to t'\}$ with link-disjoint path-pair e_n and $s' \to P_{s-t} \to t'$. This solution has $\leq Z$ cost and satisfies delay bound D for all routing DAGs, which concludes the proof. ■

4.4.2 Integer Linear Program for DARC-DD

Here, we present an ILP for DARC-DD based on the mathematical formulation of [39] for finding three minimum cost link-disjoint paths in G^* with additional delay bounds required for DD-aware routing on DAGs. The connection request is $C = (s, t, b = 2, D)$.

The notations are the same as in Section 4.3.2. Our objective is to minimize the total bandwidth cost as formulated Eq. (4.3). Similarly to DARC-QoS, Constraint (4.4) formulates the flow conservation for each path w. Constraint (4.5) ensures the disjointness of the paths. Constraint (4.6) gives a lower bound for the integer variable y^w, which captures the worst case delay increase of path (routing DAG) w upon a single link failure formulated in Eq. (4.2). However, to ensure that no loops or node-disjoint cycles are built, we need additional Constraints (4.8)-(4.13). For this purpose the so called voltage analysis is used [98], where the main idea is that for each next node in the flow the sum of the voltages on the outgoing arcs has to be higher than on the incoming arcs (Constraint (4.12)). To enable these properties we have to introduce several new variables:

– d_i indicates if node i is the target node, i.e. if yes $d = 1$, otherwise zero.

– z_i^w is a binary variable, z_i^w has the value one if node i is traversed by flow w, otherwise zero (Constraints (4.8-4.9)).

– $q^w(i,j)$ is a non-negative continuous fractional variable (Constraints (4.10-4.11)), i.e. the voltage assigned to a given arc (i,j) and flow w.

– ϵ is predefined small positive constant, which determines the minimum step of the voltage. In our case it is set to $|1/V|$, to ensure that even the longest path can be found.

$$\forall w \in \mathcal{W}, \forall i \in V:$$

$$\forall (i,j) \in E : z_i^w \geq x^w(i,j), \tag{4.8}$$

$$\forall (j,i) \in E : z_i^w \geq x^w(j,i), \tag{4.9}$$

$$\forall (j,i) \in E : q^w(j,i) \leq x^w(j,i), \tag{4.10}$$

$$\forall (j,i) \in E : q^w(j,i) \leq x^w(j,i), \tag{4.11}$$

$$d_i + \sum_{(i,j)\in E} q^w(i,j) - \sum_{(j,i)\in E} q^w(j,i) \geq \epsilon \cdot z_i^w, \tag{4.12}$$

$$\sum_{(i,j)\in E} x^w(i,j) \leq 1, \tag{4.13}$$

In order to capture the delay difference between the two fastest (i.e., lowest delay) routing DAGs in DARC-DD, we formulate the order of the DAGs (wlog $\delta_{E_A} \leq \delta_{E_B} \leq \delta_{E_{A\oplus B}}$) in Constraints (4.14)-(4.15).

$$\sum_{e\in E} x^A(e) \cdot d_{min}^e \leq \sum_{e\in E} x^B(e) \cdot d_{min}^e, \tag{4.14}$$

$$\sum_{e\in E} x^B(e) \cdot d_{min}^e \leq \sum_{e\in E} x^{A\oplus B}(e) \cdot d_{min}^e, \tag{4.15}$$

The differential delay bound D corresponds to paths x^A and x^B in the failure-less state, formulated in Constraint (4.16).

$$\sum_{e\in E} [x^B(e) - x^A(e)] \cdot d_{min}^e \leq D. \tag{4.16}$$

In order to formulate all possible delay differences between the two fastest paths upon a single link failure occurs, we have to formulate all possible situations in Table 4.1. For example, Constraints (4.17)-(4.18) formulate when E_B is disrupted while the end-to-end delay of E_A is increased (as we don't know which one has lower delay we need two constraints):

$$y^A + \sum_{e\in E} [x^A(e) - x^{A\oplus B}(e)] \cdot d_{min}^e \leq D, \tag{4.17}$$

$$-y^A + \sum_{e\in E} [x^{A\oplus B}(e) - x^A(e)] \cdot d_{min}^e \leq D. \tag{4.18}$$

4.4.3 Approximability of DARC-DD

The NP-hardness of finding paths with obeying a given differential delay bound (DDR problem) was proven in [82], and in Section IV C of [82] its approximability was investigated. It was shown that the DDR problem can not be approximated within a factor of n^ϵ for any $\epsilon < 1$ for Hamiltonian graphs, where $n = |V|$ is the number of vertices in graph G. Note that, the DDR problem is defined strictly for paths (and not DAGs). Furthermore, it does not require the disjointness of paths, thus, it does not provide any survivability for the connection. Meanwhile, DARC-DD is defined for the three routing DAGs which have to satisfy survivability and delay requirements simultaneously. Despite the relevant differences of DDR and DARC-DD, their NP-completeness proofs follow similar reasoning. Thus, we can claim that constant factor approximation of DARC-DD is NP-hard as well based on [82]:

Theorem 4.4.2 *The DARC-DD problem can not be approximated within n^ϵ for any $\epsilon < 1$ for Hamiltonian graphs.*

Proof: See the reasoning of [82][Section IV C] for DDR built on the hardness of finding paths longer than a given constant in Hamiltonian graphs [46]. ■

Hamiltonian graphs are special cases of general graphs, which contain paths of length $|V| - 1$. Hence it is sufficient to show the in-approximability in Hamiltonian graphs to prove it generally. The proof presented in Section IV C [82] assumes that given is an approximation algorithm which is able to approximate the DDR problem. This algorithms is run on a transformed graph similar to one presented in Figure 4.3. This way the approximability question boils down to the question of finding path longer then $n - n^\epsilon$ in Hamiltonian graphs, which is already proven to be NP-hard in [46]. The same argumentation can be used to prove that the DARC-DD problem can not be approximated within n^ϵ for any $\epsilon < 1$.

4.4.4 Heuristic Approach for DARC-DD

Since the problem is hard to approximate, it is legitimate to use a heuristic approach, which leverages the computational complexity of the presented ILP. In [39] two heuristic approaches were introduced based on k-shortest link-disjoint paths, both of them following the SPLIT approach. These methods are based on finding the so called merger nodes (two or more paths crossing the given node), dividing the paths into segments (starting and ending at these merger nodes), and creating all possible combinations of $s-t$ paths by cascading the path segments. Finally, they distract all maximum disjoint sets of these paths and check whether they satisfy a delay constraint or not. Of course, the SPLIT approach could be modified for finding three disjoint paths in the transformed graph G^* for DARC-DD, which satisfies the delay constrains in Table 4.1. However, it is obvious that owing to the huge problem space introduced by the investigation of all possible combination of segments, the SPLIT approach would not be a good fit in G^*, where original links and a full mesh of virtual links are present, leading to a large k in the link-disjoint

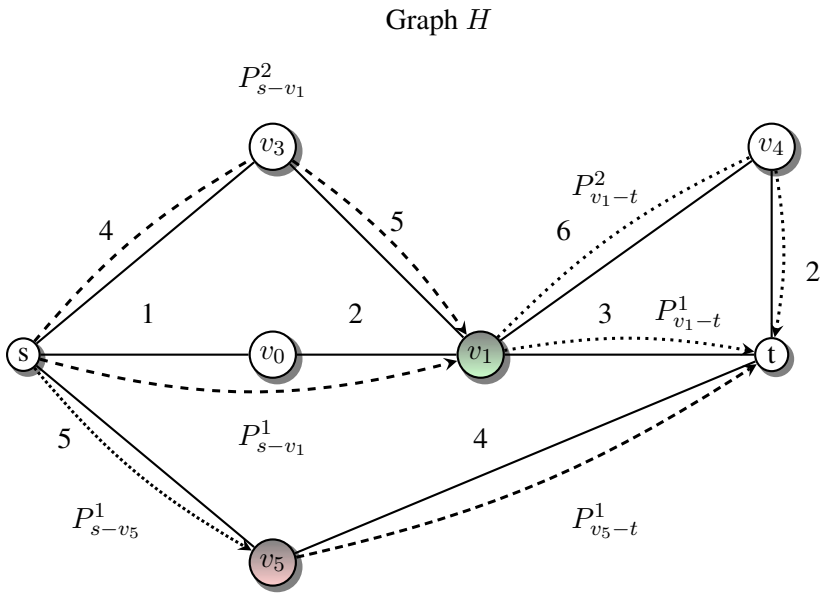


Figure 4.4: Path generation in the DARC-DD heuristic of Algorithm 5 (illustrating the differences between SPLIT [39] and our approach). The numbers next to the links represent the d_{min} value of the link [J2].

path problem. This motivated us to crate new heuristic in Algorithm 5, which reduce the problem space of SPLIT with a smart and intuitive way of path generation instead of enumerating all possible paths.

Our path generation method is illustrated in Fig. 4.4 on graph H, which is obtained with a run of the Suurballe algorithm for finding $h = 3$ link-disjoint paths in G^* (Step 4 of Algorithm 5). First the merger node v_1 is processed in Step 7 (this node has the maximum in-degree $m = 2$). In Step 9 the paths $P^1_{s-v_1}$ and $P^2_{s-v_1}$ are found in Fig 4.4 between s and merger node v_1 with minimum delay 3 and 9, respectively. Next, paths $P^1_{v_1-t}$ and $P^2_{v_1-t}$ are found in Step 13 between nodes v_1 and t with minimum delay 3 and 8, respectively. Next, in Step 16 $s-t$ paths are created from the lowest delay segment between $s-v_1$ and the highest delay segment between v_1-t, from the second lowest and second highest delay segment, etc. Hence, in Fig 4.4 paths $P^1_{s-t} = \{s \to v_0 \to v_1 \to v_4 \to t\}$ and $P^2_{s-t} = \{s \to v_3 \to v_1 \to t\}$ are created with total delay of 11 and 12, respectively. The second merger node which is processed in Step 7 is v_5. Paths $P^1_{s-v_5}$ and $P^1_{v_5-t}$ are found, and glued together in Step 16 to crate $P^3_{s-t} = \{s \to v_5 \to t\}$ with delay minimum of 9. As P^1_{s-t}, P^2_{s-t} and P^3_{s-t} is the only triplet which can be created for $h = 3$ paths, we save it to $\mathcal{P}$ as the minimal cost solution which satisfies the delay bound for $h = 3$ in Step 17. After repeating this procedure while h-link-disjoint paths exist, we select from $\mathcal{P}$ the minimum cost solution in G^*, and obtain the routing DAGs from it.

Note that, the rearrangement of paths in Step 16 of Algorithm 5 is based on $d_{min}(e)$. However, we have two other attributes on the links in DARC which can be used in Step 4 to calculate H, i.e., the initial minimum „cost" h-link-disjoint paths. If the $c(e)$ is used as the metric, then we speak about the DD-CH (*Differential Delay - Cost Heuristic*). However, if the delay maximum, i.e., $d_{min}(e) + \Delta(e)$ is

Algorithm 5: DARC-DD Heuristic

Input: $G = (V, E, k, c, d)$, $C = (s, t, b, D)$

Result: Routing DAGs E_A, E_B and $E_{A \oplus B}$

begin
  Initialize the number of searched link-disjoint paths $h = 3$;
  `// Create graph` $G^* = (V, E^*, c^*, d_{min}, \Delta)$.
  Create graph G^* according to Section 4.2.2;
  `// Finding link-disjoint paths in` G^*.
  while *h-link-disjoint paths with minimum total cost exist between s and t in G^** **do**
    Let $H = (V^H, E^H)$ denote the subgraph defined by the link-disjoint paths;
    while *E^H is not empty* **do**
      Find the merger node u with the largest in-degree m in $V^H \setminus s, t$ (or t if no merger with $m > 0$ exists);
      while *u has incoming arcs in H* **do**
        Find a path P_{s-u} with minimal total d_{min} between s and u;
        Remove the arcs of P_{s-u} from E^H;
      Sort the P_{s-u} paths from lowest delay to the highest (*increasing order*). Denote this sorted set as $\mathcal{P}_{s-u}$;
      while *u has outgoing arcs in H* **do**
        Find a path P_{u-t} with minimal total d_{min} between u and t;
        Remove the arcs of P_{u-t} from E^H;
      Sort the P_{u-t} paths from highest delay to the lowest (*decreasing order*). Denote this sorted set as $\mathcal{P}_{u-t}$;
      Take the ith path segments from $\mathcal{P}_{s-u}$ and $\mathcal{P}_{u-t}$ and **create path** P^i_{s-t} from them. Add P^i_{s-t} to set $\mathcal{P}_h$;
    Generate all combination of triplets in $\mathcal{P}_h$, and add the one which satisfies D with minimal cost to $\mathcal{P}$;
    Increase h by 1;
  `// Save routing DAGs in` G.
  Select the solution with minimal cost from $\mathcal{P}$ (if $\mathcal{P} \neq \emptyset$);
  Build routing DAGs from the paths in G^* by replacing virtual links with their corresponding islands in G;
  Save the routing DAGs E_A, E_B and $E_{A \oplus B}$;
end

considered as the cost metric on the links in Step 4, then we talk about the DD-DH (*Differential Delay - Delay Heuristic*). Note that, DD-CH minimizes the cost on expense of no control over the delay. Meanwhile DD-DH minimizes the maximal delay, which may result in lower blocking probability but without any control on the cost of the solution. With rearranging the paths at merger nodes based on the delay minimum (d_{min}) value we get a feasible solution with a higher probability than purely using the link-disjoint paths found in Step 4 based on either metric, as the delay differences are reduced to the lowest possible value.

The computational complexity is dominated by the graph creation (in Step 3) which can be computed in $O(|V||E|\log_{1+|E|/|V|}|V|$ and by finding h shortest edge-disjoint path ($O(|h|(|E|+|V|\log_2|V|))$, where $|h|$ is the maximum number of h shortest edge-disjoint paths). As in [39], since the number of simple paths and set of link-disjoint paths are explicitly bounded with $O(|h|(|E|+|V|\log_2|V|))$ the Steps (6)-(18) do not impact the overall complexity. This means that the overall complexity is $O(|V||E|\log_{1+|E|/|V|}|V|+|h|(|E|+|V|\log_2|V|))$.

4.5 Experimental Results

In this section we investigate the total bandwidth cost of diversity coding based survivable routing DARC, i.e., when not only bandwidth cost is considered but also additional QoS routing and differential delay bounds are given for every connection request. We compare the DARC with SRDC-ICAN (Section 3.2.3), i.e., with the polynomial-time algorithm SRDC-IA (Algorithm1). Of course the cost increases by introducing additional bounds, but in exchange for that we can guarantee a certain level of QoS, not only in terms of reliability but also in terms of end-to-end and differential delay. This could dramatically improve the user experience of video streaming, which leads in long term to higher revenue and competitive advantage (compared to other providers).

We investigated random generated real-like planar $G=(V,E,k,c,d)$ *network topologies* with different sizes and densities, and some real world optical backbone topologies, too. All of the arcs have unit cost ($\forall e\in E: c(e)=1$). The arc capacities were set high enough so that no blocking occurs due the capacity deficit (i.e., $\forall e\in E: k(e)\geq 2$ satisfied for all requests). Furthermore, the delay of the arcs $d(e)$ is a function of the distance between its adjacent nodes, and scaled into range of 1 and 25 ms. These values are based on the measurements taken in optical transport networks [39,J1]. Note that, the different random networks are generated on the same (unit) square, this means that the average delay parameters (on arcs) for the smaller networks are higher than by larger ones, because the average distance is greater between two adjacent nodes. 200 *connection requests* $C=(s,t,2,D)$ were generated randomly with a given delay bound (while the same demands are considered for SRDC without the delay parameter). Note that, the limited request number is a consequence of the high computational complexity of our ILPs. With 200 demands we enabled our ILP to run for middle-scale topologies in a reasonable time.

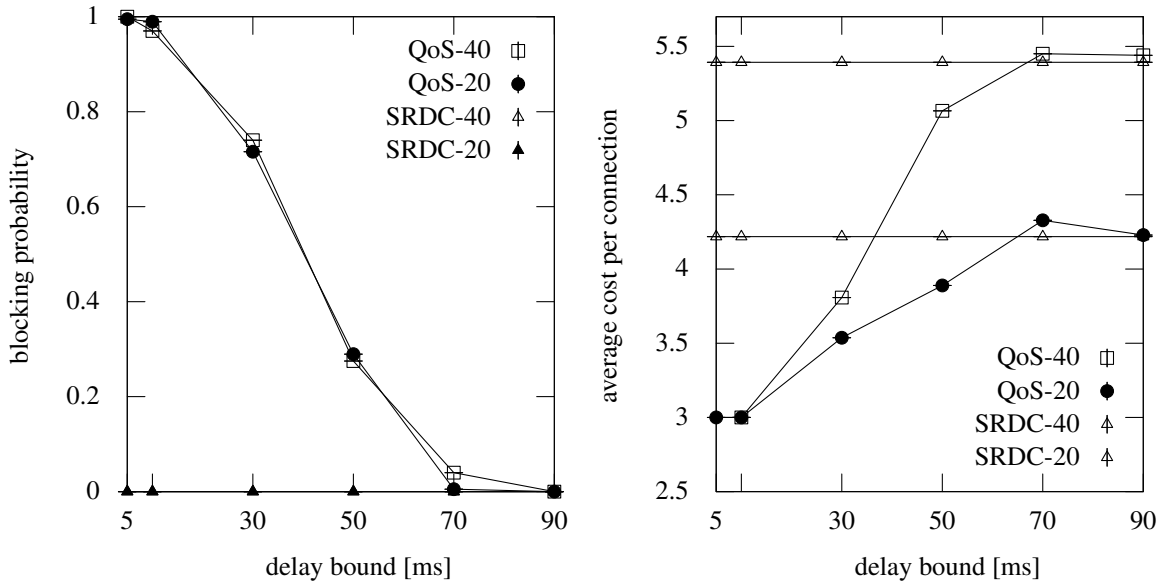


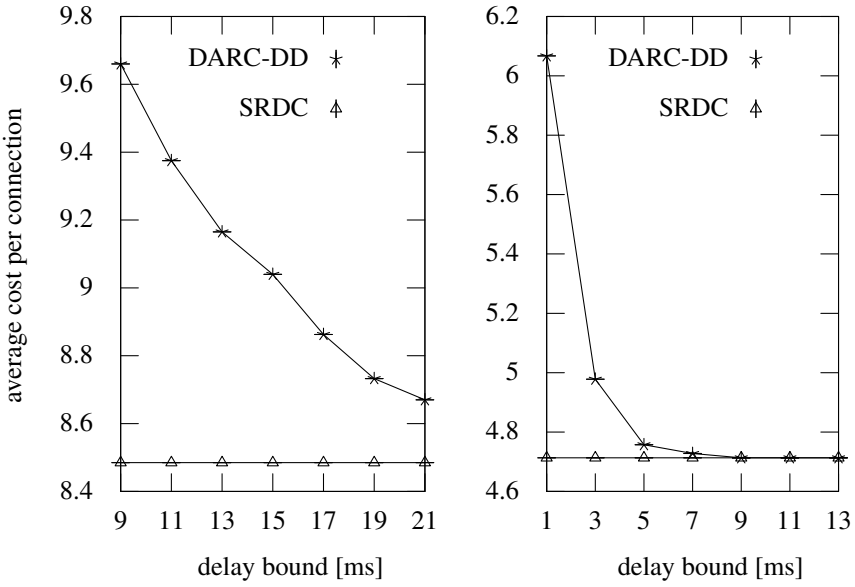
Figure 4.5: Blocking probability and average cost per connection of the DARC-QoS problems as a function of the delay bound in maximum planar graphs with node number 20 and 40 [J2].

4.5.1 Performance Evaluation of DARC-QoS

Optimizing for differential delay there are a lot more options to explore as a survivable solution. On the other hand, in QoS routing where a strict bound is given on path-length might be too strict to satisfy depending on the physical topology. Thus, if there is no path obeying this bound in the network, then the connection request has to be blocked. This leads us to the recognition that in the QoS routing problems the blocking probability characterizes the problem better than total cost. This value could be an indicator for the network operator, i.e., what percentage of the request can be satisfied with a given QoS level. According to that, in Fig. 4.5 the blocking probability of DARC-QoS is shown for different QoS bounds. One can observe that for both QoS cases the blocking probability increases rapidly after a given delay bound is reached. As mentioned before, this is due the fact that if there are no paths shorter than a given bound, then the request gets blocked.

4.5.2 Performance Evaluation of DARC-DD

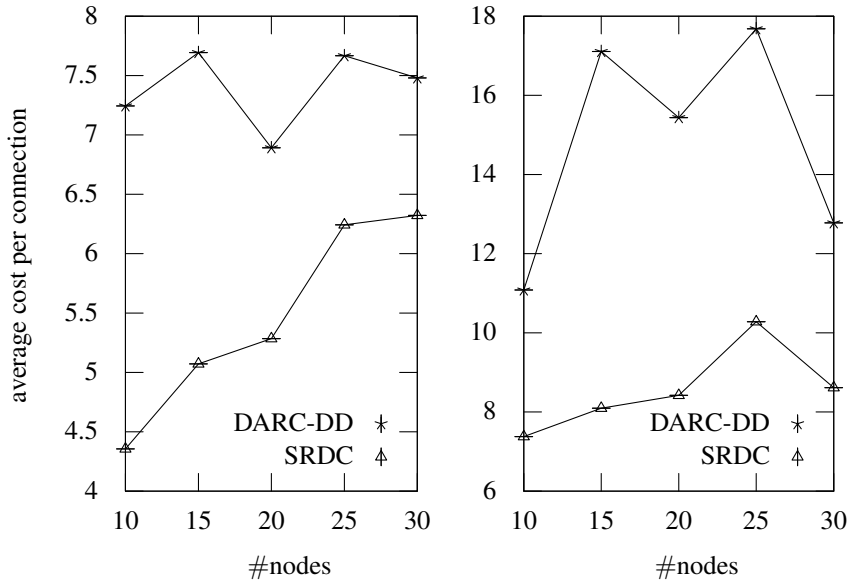
In Figure 4.6 and Figure 4.7 the simulation results of the DARC-DD methods are presented. Fig. 4.6a-4.6b present the total bandwidth cost of real-world topologies depending on the delay bound. It can be observed that as the delay bound decreases the total cost of DARC-DD increases dramatically. It is foreseeable because DARC-DD takes all possible failure scenarios into account. This means that if any single link failure event occurs, this method provides a solution within the given differential delay bound. In the end, it sacrifices the cost efficiency in order to find three routing DAGs approximately with the same delay.



(a) Cost 266 (37 nodes, 57 links, with diameter 8) [65]

(b) Abovenet (17 nodes, 37 links, with diameter 4) [80]

Figure 4.6: Bandwidth cost of the DARC-DD problem as a function of the delay bound in real-world [J2].



(a) Dense networks (average nodal degree between 3.2 and 4)

(b) Sparse networks (average nodal degree between 2.4 and 2.8)

Figure 4.7: Bandwidth cost of the DARC-DD problem as a function of the delay bound in real-world like topologies [J2].

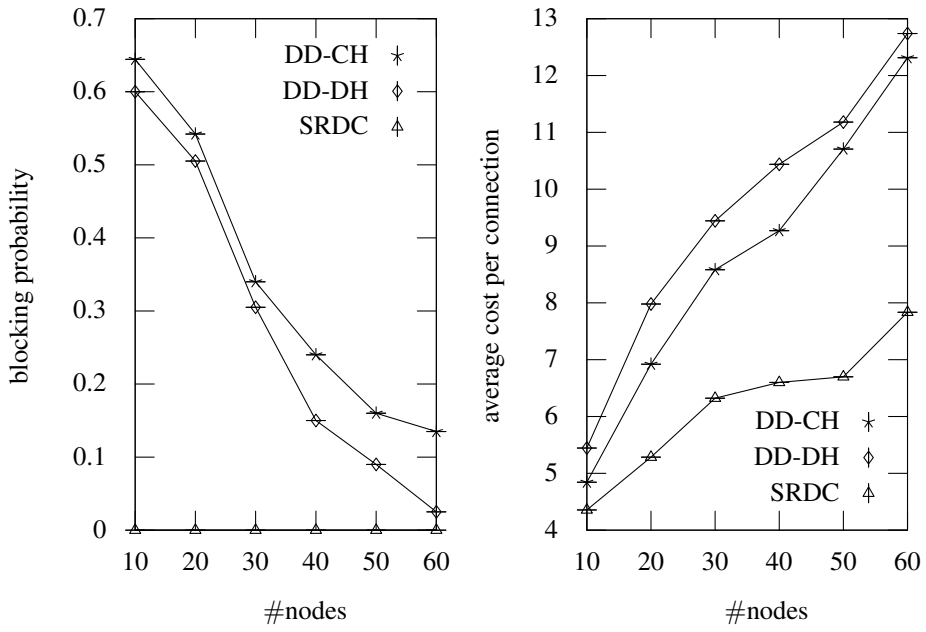


Figure 4.8: Blocking probability and average cost per connection of the DARC heuristics in a dense network [J2].

Fig. 4.7a-4.7b plot the total bandwidth cost in networks with different sizes and densities for a given delay bound (9 ms). The tendencies are the same as in real world topologies, i.e., DARC-DD needs much more resources than SRDC to satisfy all of the constraints related to all single link failures, independently from network size. Note that, the randomly generated traffic demands can cause some bandwidth cost fluctuation (as in larger networks not all $s - t$ pairs are considered as requests).

In Figure 4.8 the simulation result of the two DD heuristics are presented, namely the blocking probability and the average cost in dense networks. We can see that there is a trade off between the two heuristics, either we minimize the blocking probability on the cost of the higher average cost per connection (if we use the heuristic DD-DH). Or we can use the DD-CH in this case we get a lower average cost per connection but the blocking probability increases. The blocking probability is higher by smaller networks because there are not so many paths between the source and target node. Note that, for example, in the 30 node sparse network the average running time of the heuristics is less than a second per demand, while on the other hand, the ILP needs an average time of about 40 minutes for one single demand. In other words, the heuristic is about 2400 times faster than the ILP.

4.6 Summary of Results

In this chapter we introduced Delay Aware Routing with Coding (DARC) which captures several QoS routing and differential delay aware bounds of diversity coding-based survivable routing. Our work is built on the fact that the optimal solution of this routing problem has a well-defined structure , i.e., can

be decomposed into three routing DAGs between the source and target node (Section 3.2). In order to reach the benefits provided by this redundancy, we defined the delay of routing DAGs capturing all possible single-link failure scenarios. Although DARC inherits several results from previous works on finding three link-disjoint paths between the communication end-points obeying some additional delay bounds, the two delay parameters (i.e., before- and after-failure end-to-end delay) of the routing DAGs leads to a more complex routing problem (even if the problems were already hard for paths both for QoS and differential delay bounds). Thus, we gave NP-completeness proofs for these problems, and demonstrated that although minimizing the total bandwidth cost of routing DAGs is polynomial time solvable (see Section 3.2.3), it leads to a hard problem when an additional delay bound introduced on the routing DAGs. We gave an Integer Linear Program for DARC-QoS and DARC-DD to find a minimal cost solution while obeying additional QoS and differential delay bounds, respectively. An efficient heuristic approach was proposed for DARC-DD to select paths with appropriate delay differences. Through simulations on small- and medium-scale network topologies we demonstrated the effect of the different delay bounds on the total bandwidth cost of the optimal and heuristic solution.

4.6.1 Theses Summary

Thesis 2 *[C3, J2] I defined the delay of routing DAGs. I introduced the DARC-QoS and DARC-DD problems. I proved that both problems, namely the DARC-QoS and the DARC-DD, are NP-complete. Furthermore, I have proved that the DARC-DD problem can not be approximated within n^{ϵ} for any $\epsilon < 1$ in Hamilton graphs. I presented ILPs to solve both problems, and introduced two effective heuristics to solve DARC-DD.*

Thesis 2.1 (QoS complexity) *[C3, J2] I proved that the DARC-QoS problem is NP-complete.*

Thesis 2.2 (DD complexity) *[C3, J2] I proved that the DARC-DD problem is NP-complete.*

Thesis 2.3 (Integer Linear Programs) *[C3, J2] Based on the novel definition of routing DAG delays I presented ILPs to solve the QoS and DD problem.*

Thesis 2.4 (Heuristics) *[J2] Because of the slow running time of the ILP, I suggested two fast heuristics (cost and delay based) by modifying the SPLIT approach to be able to solve the DD problem.*

Thesis 2.5 (DD approximability) *[C3, J2] I have proved that the DARC-DD problem can not be approximated within n^{ϵ} for any $\epsilon < 1$ in Hamilton graphs.*

Chapter 5

Local Failure Localization in Transport Networks

In the first part of my dissertation I proposed new methods with low resource requirements that ensures *instantaneous recovery in a robust manner* (Chapter 3) while being able to satisfy *delay constraint even after a failure* occurs (Chapter 4). In other words, the first part of my dissertation was all about the survivable routing under different technological constraints i.e. capacity, node capability and delay constraints. This means we focused on protection and not on failures localization, we reserved and provisioned the necessary backup resources in advance in order to protect the connection in case of a certain failure pattern (e.g. single link failures or arbitrary SRLG lists).

In the second part of the dissertation I focus on the failure localization. I propose a new all-optical local failure localization framework, that enables fast restoration even for the shared protection approaches, reducing the failure recovery time (t_R) significantly. Of course, due the nature of the shared protections, optical cross-connect (OXC) reconfiguration is always needed i.e. *robust failure recovery is not an option*. Note that the OXC reconfiguration itself can take up to several tens of milliseconds depending on the technology [60, 101] i.e. instantaneous recovery can not be guaranteed even the failure localization and notification time takes only several milliseconds. Of course network operators can and have to consider the failure mitigation technique according the desired QoS level.

5.1 Motivation

As already stated in Section 2.3, dedicated protection approaches like $1 + 1$ and p-Cycle achieve 50 ms or shorter restoration time on the expense of 70% or higher redundancy of required wavelength resources [31]. Lately, trends point toward that providers are slowly but surely facing the capacity cap of optical transport networks [83], and have to consider more capacity efficient restoration methods, like SRNC and SRDC.

Also shared protection approaches could be an alternative. However, approaches like shared segment

protection (SSP) [37] or failure dependent protection (FDP) [31] trade control simplicity for better capacity efficiency, in which one or multiple *protection lightpaths* (P-LPs) are pre-computed for each *working lightpath* (W-LP). As the P-LPs share wavelength resources for better capacity efficiency, the switches can be set up only after a failure occurs and is identified using extensive control plane signaling [68] leading to slow recovery.

However, in the recovery process of the disrupted W-LPs *failure localization* and *failure notification* can not rely on any time consuming (electrical) signaling protocol to reach fast failure-restoration.

As already mentioned in Section 2.3.5 in order to enable fast *failure localization* in all-optical networks, supervisory lightpaths (S-LP, monitoring trails or m-trails) have been introduced [6, 13, 35, 88, 89, 91][1]. If each traversed node is able to monitor the status of a carefully allocated set of m-trails *failure notification* time could be reduced to minimal as well (Section 2.3.5). This means the S-LPs enable the recovery of an arbitrary shared protection scheme [31, 37] to be performed completely signaling-free [88, 89].

The goal of previous m-trail allocation methods was unambiguous localization of single link failures to support both protection switching and link maintenance [88, 89], i.e., each link failure has to correspond to a different set of alarms/disrupted m-trails. We argue that this strict requirement is only mandatory for link maintenance; hence, often leads to an unnecessary amount of S-LP transponders and wavelength links for protection switching. Thus, we aim to reduce the number of S-LPs and their used wavelength links by relaxing the requirement of unambiguous link failure localization, focusing not on link failures but rather on the identifiable *protection switching actions*. Maintenance of the failed link can be performed afterwards on a different time-scale even involving the electrical layer if necessary, without delaying the restoration of the disrupted connections. An example is presented in Fig. 5.1 where e.g., the *loss of light (LOL)* (referred to as the "darkness" of the S-LP) of monitoring trail T_1 indicates at node v_6 that one of the links failed form $(v_6, v_5), (v_5, v_1), (v_1, v_2), (v_2, v_3)$, i.e., T_1 clearly does not provide unambiguous single link failure localization at v_6 for link maintenance. However, node v_6 is still able to perform the proper protection switching action by activating P-LP $P_{\lambda_1}^{(v_1 \to v_3)}$.

In this part of the dissertation we first identify the set of links whose failure can be localized together, called *switching link groups*, and demonstrate that localizing link groups instead of single link failures leads to a significantly improved S-LP performance. We prove that the resulting failure localization problem, called Advanced Global Neighborhood Failure Localization (AG-NFL) is NP-complete, and a yet efficient heuristic is proposed. We investigate the performance of four different AG-NFL m-trail scenarios with different extent of reliance on data plane information in order to avoid continuous m-trail reconfigurations in dynamic traffic scenarios with frequent W-LP setup and teardown events.

The rest of the chapter is organized as follows. Section 5.2 provides related work and summarizes the background of the study. Section 5.3 summarizes the observations AG-NFL switching link groups are

[1] An m-trail consists of a pair of lightpaths along a common physical route (in opposite directions), and is purely used for monitoring the on/off status of the physical links along the route at the end-nodes.

based on and details the improved group formation process by introducing the notion of forbidden link-pairs. In Section 5.5 the m-trail scenarios are introduced which can reduce the data plane dependency of AG-NFL. In Section 5.6 the complexity of AG-NFL is discussed and a heuristic approach is proposed. Section 5.7 presents the simulation results, while Section 5.8 concludes our work.

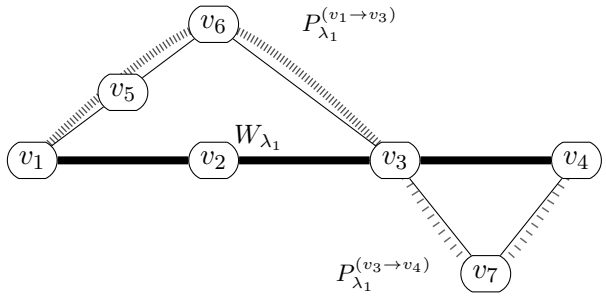
5.2 State-of-the-Art Failure Localization in Transport Networks

In Generalized Multi Protocol Label Switching (GMPLS) [69] the fault management – including failure localization and notification – is based on electrical layer messages, i.e., each downstream node of the working path detects the loss of light and sends an alarm to its upstream node. Each node that receives an alarm, checks the proper upstream link, if it is also subject to LOL the alarm is sent further upstream, etc. On the other hand, if the link is intact, then the faulty link is localized and failure notification is triggered. An example is presented in Fig. 5.1(a), where W-LP W_{λ_1} and two P-LPs $P_{\lambda_1}^{(v_1 \to v_3)}$ and $P_{\lambda_1}^{(v_3 \to v_4)}$ are given. If (v_1, v_2) or (v_2, v_3) fails then $P_{\lambda_1}^{(v_1 \to v_3)}$, and if (v_3, v_4) fails then $P_{\lambda_1}^{(v_3 \to v_4)}$ should be activated, respectively. By GMPLS if link (v_1, v_2) fails all the nodes along the W-LP v_1, v_2, v_3, and v_4 have to perform the messaging procedure explained above (*W-LPs assumed to be bidirectional*). After the failure of (v_1, v_2) is localized, in the failure notification phase the corresponding control messages are sent to v_1, v_3, v_5 and v_6 to activate $P_{\lambda_1}^{(v_1 \to v_3)}$. The messaging procedure at both failure localization and failure notification leads to a long recovery time which can cause upper layer protocols to react, i.e., serious performance degradation of the transport service.

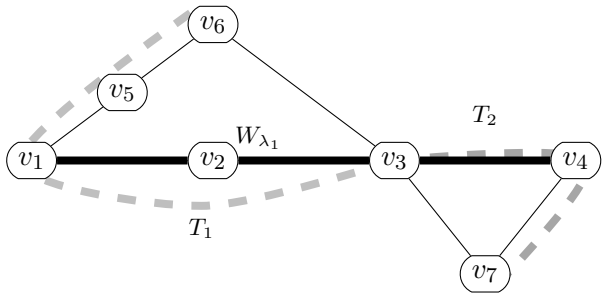
To reduce the reliance of fault management on the electrical layer, thus, reduce the failure localization time, several all-optical m-trail allocation schemes have been introduced as summarized in Section 2.3.5. The evolution of m-trail based failure localization started with the centralized all-optical methods, however to completely eliminate signaling from the failure localization phase local failure localization methods have to be introduced. Currently the G-NFL is the most efficient signaling-free unambiguous link failure localization framework.

We introduced the concept of identifying switching link groups rather than unambiguously localizing link failures at each node in [C4]. The method called Advanced Global Neighborhood Failure Localization (AG-NFL), which kept all the benefits of G-NFL (i.e., provides ultra-fast all-optical restoration for any shared protection scheme) with significantly improved monitoring resource usage (e.g., number of transponders). However, the proposed switching link group formation was too restrictive in some scenarios, and also required frequent reconfiguration of m-trails at each W-LP setup and W-LP teardown event to be able to perform the proper protection switching action.

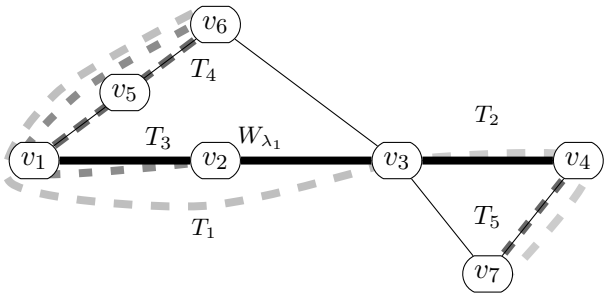
In [J4] we presented the refined version of the AG-NFL. We introduce four different m-trail scenarios [86, 89] with decreasing data plane dependency for the price of increased monitoring resource usage (Section 5.5) in order to reduce the dependency of AG-NFL on the data-plane dynamics. More importantly we improved the switching link group formation process by considering and excluding conflicting



(a) W-LP W_{λ_1} and the corresponding P-LPs.



(b) S-LPs T_1 and T_2.



(c) G-NFL solution (needs three extra S-LPs).

Figure 5.1: Monitoring the status of W_{λ_1} and S-LPs T_1, T_2 provides enough information for each node to perform proper protection switching action, while G-NFL needs additional m-trails $T_3 - T_5$ for unambiguous link failure localization. Intermediate nodes are able to monitor the status of the traversing m-trails and W-LPs.

switching actions in order to make all of these m-trail scenarios applicable for AG-NFL with an improved m-trail performance. Section 5.3 presents the details of the revised and refined AG-NFL framework.

5.3 Switching Link Groups

M-trail allocation methods before G-NFL focused on identifying all link failures unambiguously. G-NFL made the first step towards reducing the number of identifiable links, but still left a lot of room for further improvement owing to the strict requirement of unambiguous localization of the links in the nodes' neighborhood. An example is presented in Fig. 5.1, where AG-NFL needs only two m-trails – T_1 and T_2 in Fig. 5.1(b) – to enable all nodes included in the recovery process of W_{λ_1} to *perform the proper switching action* (e.g., if T_1 and W_{λ_1} are down then $P_{\lambda_1}^{(v_1 \to v_3)}$ is activated at nodes v_1, v_3, v_5, and v_6). However, G-NFL would require further S-LPs (shown in Fig. 5.1(c)) to unambiguously identify neighborhood links $(v_1, v_2), (v_2, v_3)$ at v_6. This advantage of AG-NFL is based on the following two observations:

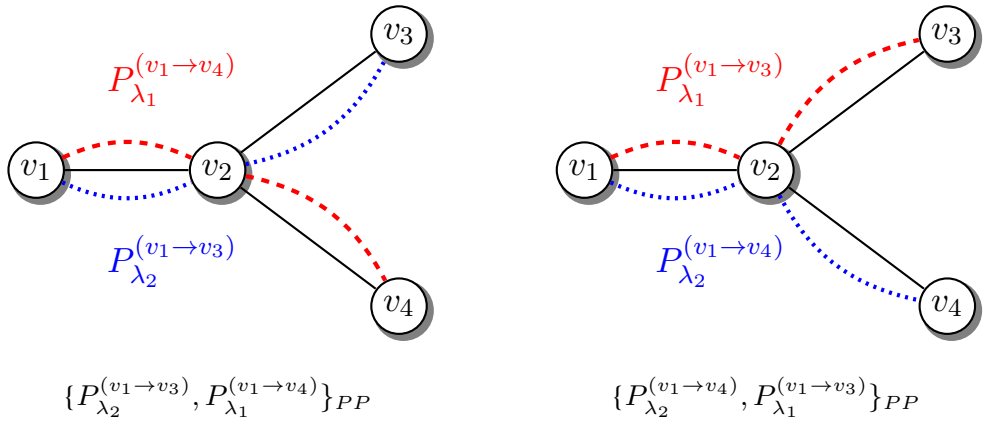
- **Observation (i):** If exactly the same switching action belongs to two links in the neighborhood at node v, then we do not have to distinguish their failures from each other to perform the proper switching action. In other words, these links are a *working segment* of the corresponding W-LP (e.g., links $(v_1, v_2), (v_2, v_3)$ at v_6 in Fig. 5.1(a)).
- **Observation (ii):** Link failures in the neighborhood of a node with only *"non-disruptive" switching actions* do not have to be distinguished from links outside of the neighborhood (i.e., links to which no switching action belongs). For example, links $(v_1, v_5), (v_5, v_6)$ do not need to be distinguished from neighborhood links $(v_1, v_2), (v_2, v_3)$ at v_6 in Fig. 5.1(a), as v_6 is an intermediate node of $P_{\lambda_1}^{(v_1 \to v_3)}$. Thus, false switching on λ_1 at v_6 will not disrupt W-LP W_{λ_1}.

Based on these two observations we form *switching link groups*, i.e., links whose failures do not need to be distinguished from each other in AG-NFL with S-LPs.

5.3.1 Switching Action Sets

After applying an arbitrary shared protection scheme to route a connection request (e.g., SSP [37] or FDP [31]), the W-LP and corresponding P-LPs protecting different links or segments of the W-LP are given. Based on this information all of the protection switching actions upon link failure e along the W-LP can be divided into the following two categories at nodes v along the P-LPs:

Definition 5.3.1 *Activation of P-LP P at node v upon link failure e is called working-to-protection switching action (*WP-action*) if v is one of the end-nodes (switching or merging) of P-LP protecting the failure of W-LP link e.*



(a) At failure of e_1

(b) At failure of e_2

Figure 5.2: Forbidden link-pairs e_1 and e_2 at node v_2. Their failures have to be distinguished from each other in order to perform the proper switching action on both wavelengths λ_1 and λ_2 [J4].

Definition 5.3.2 *Activation of P-LP P at node v upon link failure e is called protection-to-protection switching action (*PP-action*) if v is an intermediate node of the P-LP protecting the failure of W-LP link e.*

For example, in Fig. 5.1(a), activation of $P_{\lambda_1}^{(v_1 \to v_3)}$ when link failure (v_2, v_3) occurs (and disrupts W_{λ_1}) is a WP-action for v_1 and v_3 (end-nodes of the P-LP), while it belongs to the PP-actions for nodes v_5 and v_6 (intermediate nodes of $P_{\lambda_1}^{(v_1 \to v_3)}$). Note that multiple WP- and PP-actions might be performed at a given node v upon a link failure e (disrupting multiple W-LPs) to activate all the traversing P-LPs, at most one at each wavelength (WL). These actions can be divided into the *WP-action set* ($\{P_i, \ldots, P_j\}_{WP}$) and *PP-action set* ($\{P_i, \ldots, P_j\}_{PP}$) at each node v.

5.3.2 Forbidden Link-Pairs

Although our goal is to form as large switching link groups as possible to decrease the number of S-LPs required for their localization, we have to make sure that the union of the switching actions in the action sets of the links in a given switching link group are realizable at the same time at an arbitrary node v. Hence, we have to define/identify link-pairs which failure have to be distinguished from each other at v, e.g., because they have *conflicting switching actions*. We define these link-pairs as *forbidden link-pairs* at v.

In Figure 5.2 the PP-action sets at v_2 is shown in case of link failures e_1 and e_2 (not shown in the figure) with two switching actions that need to be performed. It is clear that e_1 and e_2 cannot belong to the same switching link group, as we need to switch λ_1 towards v_4 and λ_2 towards v_3 upon link failure e_1, and vice versa upon the failure of v_2, i.e., the switching actions are conflicting and are not realizable at the same time. In general, two switching actions are in conflict if they order the optical switch to switch an incoming wavelength onto two different outgoing links (or onto two different wavelengths if

wavelength conversion is available). In the running example, the union of the PP-action sets of e_1 and e_2 contains conflicting switching actions on both λ_1 and λ_2. Hence, we define e_1 and e_2 as a forbidden link-pair at v_2, as they failures have to be distinguished from each other, i.e., they have to be in different switching link groups.

Note that, in NWL-UFL [88] at every node each link-pair in the network was defined as a forbidden link-pair regardless of the corresponding switching actions, i.e., each link failure had to be distinguished from the failure of every other link which resulted in unambiguous link failure localization. G-NFL [89] relaxed this requirement by defining the neighborhood links at each node (i.e., links to which failure arbitrary switching action belongs) as forbidden link-pairs with every other link. Therefore, the failures of neighborhood links had to be unambiguously localized, but two don't care links could have the same code as they are not forbidden pairs. In AG-NFL, *we further refine the set of forbidden link-pairs* based on the conflicting switching actions which have to be performed. With this approach only those link-pairs are considered as forbidden, which switching actions are not satisfiable at the same time at an arbitrary node v, i.e., based on the W-LPs and P-LPs we identify the exact set of forbidden link-pairs at every v.

5.3.3 Link Failure Identification Requirements

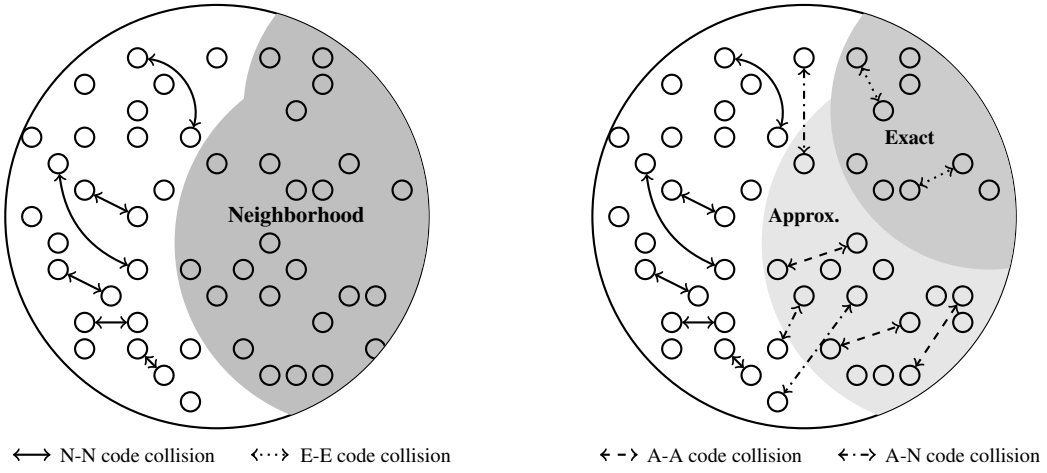
From a failure localization perspective, the corresponding WP- and PP-action sets partition the links e into three types:

- **Exact identification required (E)**: If there is at least one P-LP activation in the WP-action set of link e at node v, then the failure of e has to be exactly identified, as an erroneous P-LP activation at v would lead to the disruption of the corresponding W-LP. For example, in Fig. 5.1(a) at v_3 for links (v_1, v_2) and (v_2, v_3) the WP-action set is $\{P_{\lambda_1}^{(v_1 \to v_3)}\}_{WP}$. Hence, they are *exact links* at v_3.

- **Approximate identification required (A)**: If the WP-action set is empty, but the PP-action set contains P-LP activations, then an approximated identification of link failure e can be allowed, as an erroneous P-LP activation at the intermediate nodes of the P-LP is non-disruptive, i.e., will not cause the disruption of any W-LP. In other words, we must activate the P-LPs when e fails, but there might be some false positive activations as well. For example, in Fig. 5.1(a) at node v_6 for link (v_1, v_2) WP-action set is $\{\}_{WP}$, while PP-action set contains $\{P_{\lambda_1}^{(v_1 \to v_3)}\}_{PP}$. Thus, (v_1, v_2) is an *approximate link* at v_6.

- **No identification required (N)**: To these links both the WP- and PP-action sets are empty (*don't care links* in [89]). For example in Fig. 5.1(a) at node v_6 link (v_3, v_4) does not need to be identified, as neither of the P-LPs protecting (v_3, v_4) traverses node v_6.

We note here that G-NFL [89] partitioned the links at each node into neighborhood (if arbitrary switching action belonged to the link) and no identification required (don't care) types. With the above definition of identifiability based on WP- and PP-action sets, we refined this strict categorization of



(a) G-NFL link alarm codes

(b) AG-NFL link alarm codes

Figure 5.3: Possible alarm code collisions for neighborhood links of G-NFL versus links in AG-NFL with different identification requirements at node v. For the sake of simplicity two alarm codes collide, but there could be multiple links with the same code [J4].

G-NFL and further divided the neighborhood into exact and approximate links (shown in Fig. 5.3). Although it seems to be a minor modification, it opens up the way towards an efficient design of forbidden link-pair sets for AG-NFL.

5.4 Advanced Global Neighborhood Failure Localization

5.4.1 Alarm Code Collisions in AG-NFL

If a working path W_{λ_i} that traverses link or segment $s = (v_a \rightarrow v_b)$ is interrupted due to the failure of s, the corresponding protection path $P^s_{\lambda_i}$ has to be activated (in both directions) at each node lying on $P^s_{\lambda_i}$. Thus, these nodes have to localize the failure of s to perform the proper switching action and make the ultra-fast restoration happen based on the on-off status of the traversing m-trails $\mathcal{T}^v$ and working paths $\mathcal{W}^v$. This set $\mathcal{T}^v \cup \mathcal{W}^v$ defines the *alarm code table (ACT)* $\underline{\underline{A}}^v$ at node v. The alarm codes are built in the following way: For each link e there is a row in ACT (its *alarm code*), and let the i^{th} bit of e be denoted by $a^v_{e,i}$ for $1 \leq i \leq |\mathcal{T}^v \cup \mathcal{W}^v|$ where $|\mathcal{T}^v \cup \mathcal{W}^v|$ is the number of m-trails and working paths traversing (or terminating at) node v. We set $a^v_{e,i}$ to one ($a^v_{e,i} = 1$) if and only if the i^{th} m-trail or working path contains link e, otherwise $a^v_{e,i} = 0$. An example of W-LPs, m-trails and the corresponding ACT is presented in Fig. 5.4.

As mentioned before, at G-NFL all the links in the neighborhood of v have to be localized unambiguously, i.e., each link has to have a unique alarm code in the ACT (traversed by a unique set of S-LPs and W-LPs). On the other hand, for the links outside of the neighborhood (don't care links) alarm code

uniqueness is not a requirement, i.e., the alarm code of two don't care links could be the same, as no P-LP activation belonged to them:

– ***N-N* code collision**: Alarm code collision is possible between links with empty WP- and PP-action sets (don't care links at v). This is the only collision type that is allowed in G-NFL. No P-LP activation is required.

However, even unambiguous localization of neighborhood links requires an excessive number of S-LPs [86, 89]. Dividing the neighborhood links in AG-NFL into exact and approximate links (Fig. 5.3 (b)) allows other alarm code collisions as well without affecting the nodes' ability to perform the proper switching action when an arbitrary single link failure occurs:

– ***E-E* code collision**: Alarm code collision is allowed between two (or more) exact links with *exactly the same WP-action sets* if they are not a forbidden link-pair based on their PP-action sets[2]. These links form a working segment of the corresponding W-LPs (e.g., in Fig. 5.1(a) at v_3 links (v_1, v_2) and (v_2, v_3)). The failed segment should be unambiguously localized to perform proper switching action, but we do not need to distinguish the failures of the segment links from each other. The P-LPs in the PP-action sets of all links in the group have to be activated as well.

– ***A-A* code collision**: Alarm code collision is allowed between two (or more) approximate links if they are not a forbidden link-pair, i.e., their PP-action sets does not contain conflicting switching actions. For example in node v_2 in Fig. 5.4 we are not able to switch from link (v_5, v_2) wavelength λ_1 to two outgoing links, namely to (v_2, v_1) and (v_2, v_3) at the same time (parallel activation of $P_{\lambda_1}^{(v_5 \to v_1)}$ and $P_{\lambda_1}^{(v_3 \to v_5)}$ is not possible). This means we have to know which failure occurred, i.e., either link (v_1, v_5) or segment $(v_3 \to v_5)$ failed. Hence, (v_1, v_5) will form a forbidden link-pair with both (v_3, v_4) and (v_4, v_5) at node v_2, and their failures have to be distinguished. On the other hand, alarm codes of the working segment $(v_3 \to v_5)$ links could be the same.

– ***A-N* code collision**: Similarly to the *A-A* collision, an approximate link (segment) can have the same alarm code as don't care links, as don't care link have empty PP-action sets, thus, can not be a forbidden-pair of an A-A link. The P-LPs in the PP-action set of the approximate link(s) have to be activated if the failure is localized. For example, in Fig. 5.1(a) the approximate working segments' $(v_1, v_2), (v_2, v_3)$ code collision does not need to be resolved with don't care links $(v_1, v_5), (v_5, v_6)$. However, if this group is localized when one of the don't care links (e.g., (v_1, v_5)) fails, then activation of $P_{\lambda_1}^{(v_1 \to v_3)}$ is unnecessary. Although we are wasting energy on protection switching, this action will not cause the disruption of any W-LP.

By allowing these collisions (*E-E* and *A-A* based on Observation (i), and *A-N* built on Observation (ii)), the number of S-LPs required to identify the switching link groups – thus to perform the proper

[2] Because the WP-action sets are exactly the same for the two link failures, and WP-actions never conflict with the PP-actions of the same link failure (they use different wavelengths as they are activated at the same time), only the PP-actions can conflict.

switching action – can be significantly reduced compared to the unambiguous link failure localization requirement of G-NFL. Further note that, in [C4] the *A-A* (and *E-E*) alarm code collision for AG-NFL was defined based on *dominant PP-action sets*, i.e., where one of the links' PP-action set in a switching link group contains the PP-actions of all other links in the set. The *A-A* (and *E-E*) collision we defined in [J4] is more accurate, as it only keeps track of the minimal set of forbidden link-pairs which failures have to be distinguished from each other. For example, $\{P_{\lambda_1}^{(v_3 \to v_5)}\}_{PP}$ and $\{P_{\lambda_3}^{*}\}_{PP}$ in Figure 5.5 are allowed to be placed in the same switching link group if required, as links (v_3, v_4), (v_4, v_5) and link (v_6, v_3) are not forbidden link-pairs (the corresponding switching actions belong to different wavelengths). On the other hand, neither of the switching actions dominated by the other, so they were placed into different switching link gropus by [C4].

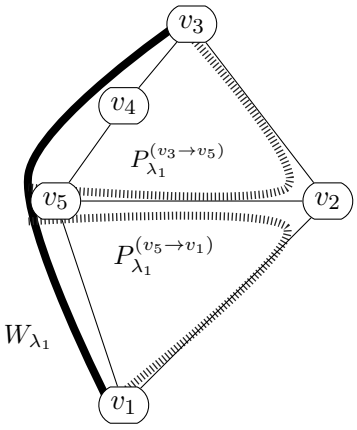
Furthermore, with the usage of forbidden link-pairs the working segments (i.e., code collisions based on Observation (i)) are not fixed in advance based the output of the shared protection scheme as it was in [C4] using dominant PP-action sets. This additional freedom lets the m-trail allocation process better explore the design space by deciding the switching link group of a link during S-LP allocation rather than having it as an input[3], which further improves the AG-NFL performance of [C4].

5.4.2 Illustrative Example

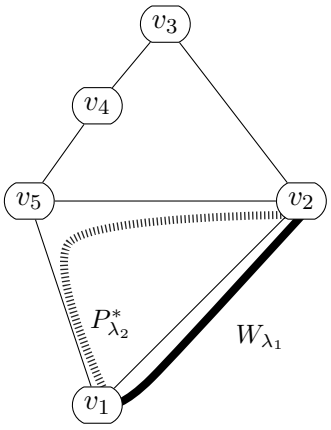
An example is presented in a five node topology in Fig. 5.4 to demonstrate the benefits of the code collisions introduced in Section 5.4.1 at node v_2. Two W-LPs are assumed, denoted as W_{λ_1} and W_{λ_2}, each being bidirectional. Working path W_{λ_1} is protected by P-LPs $P_{\lambda_1}^{(v_3 \to v_5)}$ and $P_{\lambda_1}^{(v_5 \to v_1)}$ (Fig. 5.4(a)) against link failures (v_3, v_4), (v_4, v_5) and (v_5, v_1), respectively. The second working path, W_{λ_2} is meanwhile protected by $P_{\lambda_2}^{*}$ as shown in Fig. 5.4(b). To ensure signaling-free restoration, v_2 must be able to localize the failures of (v_3, v_4), (v_4, v_5), (v_5, v_1) and (v_2, v_1) in order to perform the proper protection switching actions. Thus, the neighborhood of v_2 contains: (v_4, v_5), (v_3, v_4) with PP-action $P_{\lambda_1}^{(v_3 \to v_5)}$; (v_5, v_1) with the PP-action $P_{\lambda_1}^{(v_5 \to v_1)}$; and (v_2, v_1) with the WP-action $P_{\lambda_2}^{*}$. In Fig. 5.4(c) and Fig. 5.4(d) the corresponding m-trail solution of G-NFL and AG-NFL is presented. Furthermore, in Fig. 5.4(e) the *alarm code table* at node v_2 is given for both G-NFL (complete table) and AG-NFL (table without the gray box). Each row of an ACT on top of the separator corresponds to the links in the neighborhood of v_2, while the rows below the separator are the links outside of the neighborhood (i.e., without a corresponding switching action). In Fig. 5.4(f) the AG-NFL view of the ACT is presented, where the alarm codes of the switching link groups ensure unambiguous protection switching identification.

Note that, the status information of S-LPs T_1 and T_2 together with the status of W-LP W_{λ_2} is enough to perform the proper switching action at node v_2 in AG-NFL, as the forbidden link-pairs (v_5, v_1) from (v_4, v_5), and (v_5, v_1) from (v_3, v_4) can be distinguished. However, approximate links (v_3, v_4) and (v_4, v_5) have an *A-A* collision, while don't care link (v_3, v_2) has the same code as the working segment

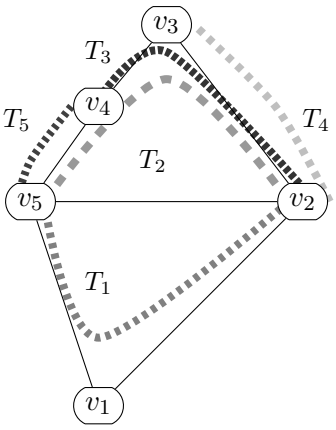
[3] Owing to the same WP-action set of links in switching link group with *E-E* code collision defines the group quite straightforward, this freedom really brings benefits only for *A-A* collisions.



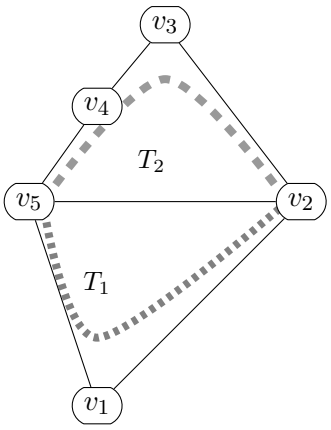
(a) Connection 1



(b) Connection 2



(c) G-NFL m-trails



(d) AG-NFL m-trails

Link	T_1	T_2	W_{λ_2}	T_3	T_4	Action Sets
(v_4, v_5)	0	1	0	0	0	$\{\}_{WP}, \{P_{\lambda_1}^{(v_3 \to v_5)}\}_{PP}$
(v_3, v_4)	0	1	0	1	0	$\{\}_{WP}, \{P_{\lambda_1}^{(v_3 \to v_5)}\}_{PP}$
(v_5, v_1)	1	0	0	0	0	$\{\}_{WP}, \{P_{\lambda_1}^{(v_5 \to v_1)}\}_{PP}$
(v_2, v_1)	1	0	1	0	0	$\{P_{\lambda_2}^{*}\}_{WP}, \{\}_{PP}$
(v_3, v_2)	0	1	0	1	1	$\{\}_{WP}, \{\}_{PP}$
(v_2, v_5)	0	0	0	0	0	$\{\}_{WP}, \{\}_{PP}$

(e) (A)G-NFL ACT at v_2

Action Sets	Type	T_1	T_2	W_{λ_2}	Switching Link Groups
$\{P_{\lambda_2}^{*}\}_{WP}$	E	1	0	1	(v_2, v_1)
$\{P_{\lambda_1}^{(v_3 \to v_5)}\}_{PP}$	A	0	1	0	$(v_4, v_5), (v_3, v_4), (v_3, v_2)$
$\{P_{\lambda_1}^{(v_5 \to v_1)}\}_{PP}$	A	1	0	0	(v_5, v_1)
$\{\}_{WP}, \{\}_{PP}$	N	0	0	0	(v_2, v_5)

(f) AG-NFL view of the ACT at v_2

Figure 5.4: Difference between AG-NFL and G-NFL using the status information of both S-LPs and W-LPs [J4].

of (v_3, v_4) and (v_4, v_5) (i.e., *A-N* collision). The failure of this switching link group is indicated by the darkness (i.e., disruption) of T_2 while both T_1 and W_{λ_1} operates (alarm code [0,1,0]). When the failure is localized, then v_2 has to activate the P-LP $P_{\lambda_1}^{(v_3 \to v_5)}$, the only action in the PP-action set. On the other hand, in G-NFL additional m-trails have to be allocated (gray box in Fig. 5.4(e)), as link failures (v_3, v_4) and (v_4, v_5) have to be unambiguously localized even if the same P-LP $P_{\lambda_1}^{(v_3 \to v_5)}$ is activated at both failures (T_3 has to be added). Furthermore, link (v_3, v_2), which is not in the neighborhood of v_2 has to have a different alarm code than the neighborhood links, thus, m-trail T_4 has to be allocated.

We conclude that with the novel refinement of failure localization requirements G-NFL efficiency can be improved significantly. The G-NFL solution consists of 5 m-trails while AG-NFL needs only 2, and the total links used by the S-LPs for G-NFL is 9 links and it is 5 for AG-NFL. This means that AG-NFL improves G-NFL in all relevant performance metrics [89], i.e., the number of S-LPs (i.e., transmitters) is decreased from 5 to 2, while the *cover length* (used wavelength links by the S-LPs) from 9 to 5 even in this toy example.

5.5 Data Plane Information Requirements

Defining forbidden link-pairs allows an improved switching link group formation compared to the one based on dominant action sets [C4] as with the novel definition of *A-A* code collisions we can merge more approximate links into the same switching link group. Furthermore, it also allows different m-trail designs with different data plane dependencies. In this section we summarize these m-trail designs [86, 89] focusing on the novel forbidden link-pair definition, and discuss its implications for the AG-NFL problem.

5.5.1 On-Demand Identification of Forbidden Link-Pairs

Until this point we assumed that the m-trails are designed based on the currently deployed W-LPs and calculated P-LPs in the network. Although it requires only the minimal number of S-LPs for unambiguous switching link group failure localization, it requires recalculation of action sets, and thus, reconfiguration of m-trails at each W-LP setup event, which could be a nightmare for the network operator. Thus, we propose two different traffic matrices (current and maximal) to calculate forbidden link-pairs as follows:

– *Strictly On-Demand (SOD)*: A given node v only needs to localize the link failure e if node v is involved in the restoration process of e according to the *current traffic matrix*. In other words v is on a P-LP which protects link e along an *active W-LP*. This method enables the most efficient allocation of monitoring resources at the expense of frequent m-trail reconfiguration at each newly setup W-LP. This means the dynamic W-LP and P-LP route information has to be considered and switching action sets need to be updated all the time, yielding in a *strong dependency between monitoring and data planes*.

Table 5.1: Summary of the need of m-trail reconfiguration upon data-plane changes [J4] [89]

	SOD-IO	SOD-O	LOD-IO	LOD-O
W-LP setup	X	X		
W-LP teardown	X		X	

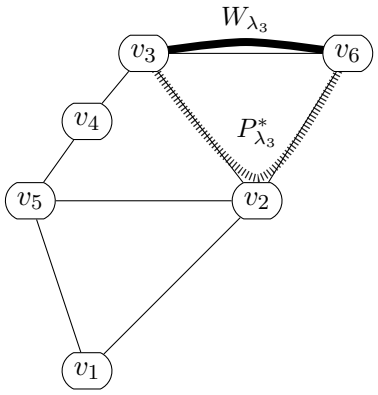
– *Loosely On-Demand (LOD)*: In this case all the possible future traffic (i.e., a maximal traffic matrix e.g., with connection request between all node-pairs or node-pairs with spare transponders) is considered when forbidden link-pairs are identified and S-LPs are allocated at a given node v. In other words v is on a P-LP which protects link e along an *active or possible future W-LP*. According to this, the worst case design of LOD significantly reduces the need for reconfiguration of m-trails by *separating the monitoring and data planes* for the price of decreased m-trail efficiency owing to the excessive number of forbidden link-pairs.

The strictly-on-demand AG-NFL (and G-NFL [86, 89]) scenario is more beneficial from a supervisory bandwidth perspective (and requires less transponders for S-LPs as well), but requires to respond each W-LP setup in the data plane. In order to incorporate the loosely-on-demand concept for AG-NFL to reduce its data plane dependency, we have to use the novel *A-A* (and *E-E*) collision definition based on forbidden link-pairs introduced in Section 5.3.2, as the dominant PP-action sets in [C4] do not ensure monotonicity in the number of switching link groups as the number of active W-LPs increases. Thus, dominant action sets do not allow the worst case design of m-trails for LOD. On the other hand, the most forbidden link-pairs will occur if all W-LPs are present in the network. Hence, we will investigate the AG-NFL performance in Section 5.7 only for the group formation definition built on the forbidden link-pairs.

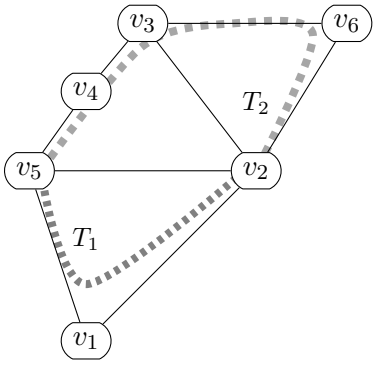
5.5.2 In-Band or In- and Out-of-Band Failure Localization

In order to further reduce the data plane dependency of AG-NFL, we can consider two failure localization scenarios. First, if AG-NFL only utilizes the status information of the S-LPs for switching link group failure localization we call it out-of-band (-O) failure localization. Second, when both the status information of W-LPs and S-LPs are considered we refer to it as in- and out-of-band (-IO) monitoring, as in-band status information of W-LPs carrying data plane traffic is also considered in the failure localization.

Note that, forbidden link-pairs were defined either by SOD or LOD, AG-NFL could build on both -O and -IO monitoring to find an unambiguous switching link group failure localization solution. However, for the -IO schemes upon a W-LP teardown event it needs to be checked whether the W-LP has been used for in-band monitoring purposes or not. If it was, then the status information of that particular W-LP is needed for the unambiguous switching link group failure localization, and thus, the W-LP should



(a) Connection 2



(b) AG-NFL m-trails

Action Sets	Type	T_1	T_2	W_{λ_2}	Switching Link Groups
$\{P^*_{\lambda_2}\}_{WP}$	E	1	0	1	(v_2, v_1)
$\{P^{(v_3 \to v_5)}_{\lambda_1}, P^*_{\lambda_3}\}_{PP}$	A	0	1	0	$(v_4, v_5), (v_3, v_4),$ $(v_2, v_6), (v_3, v_6)$
$\{P^{(v_5 \to v_1)}_{\lambda_1}\}_{PP}$	A	1	0	0	(v_5, v_1)
$\{\}_{WP}, \{\}_{PP}$	N	0	0	0	$(v_2, v_5), (v_2, v_3)$

(c) AG-NFL view of the ACT at v_2

Figure 5.5: An illustrative example of AG-NFL where forbidden link-pairs are defined based on full WL and routing information [J4].

be transformed into an S-LP instead of being torn down immediately.

5.5.3 Data Plane Dependency – Performance Trade-Off

We summarized the above m-trail scenarios and their dependency on the data plane in Table 5.1. As from an operator's perspective minimizing the number of changes in the network might be desirable instead of frequent reconfigurations to reach slightly better monitoring resource efficiency, we only require reconfiguration of an S-LP solution if it is not able to further satisfy its goal, i.e., unambiguous switching link group failure localization. Note that, throughout the dissertation, we mainly considered the SOD-IO m-trail scenario, which requires S-LP reconfiguration at both W-LP setup and W-LP teardown for the price of the best S-LP performance [89]. Based on the operational needs, the network operator can choose the m-trail scenario which is most suitable for the managed network from both perspectives.

In the example in Figure 5.5 the network presented in Figure 5.4 is extended with node v_6 and links (v_6, v_3) , (v_6, v_2). We assume that an S-LP solution was already calculated for the W-LPs and P-LPs in Figure 5.4 (SOD-IO scheme), and a new working path W_{λ_3} is setup with protection path $P^*_{\lambda_3}$. In SOD we have to adjust the AG-NFL solution, as the switching link groups were changed and now we have to identify the link failure of (v_6, v_3) in node v_2. We see that the working path between v_6 and v_3 utilizes the wavelength λ_3. This means that the switching action $P^{(v_3 \to v_5)}_{\lambda_1}$ and $P^*_{\lambda_3}$ can be put in the same switching

link group, as they can be performed at the same time. To achieve this, we need to redirect m-trail T_2 through (v_6, v_3) as shown in Figure 5.5. This results in a new switching link group $\{P_{\lambda_1}^{(v_3 \to v_5)}, P_{\lambda_3}^{*}\}_{PP}$ containing four links $(v_4, v_5), (v_3, v_4)$, (v_3, v_6) and (v_2, v_6) with the alarm code [0,1,0]. The two other switching link groups remain the same.

In a LOD design T_2 would be already deployed as shown in Figure 5.5 using more bandwidth resources as in the SOD design, but the new W-LP setup event would not need additional adjustment of the m-trail solution. Furthermore, if LOD-O is considered, the AG-NFL solution would be independent of W-LP teardown events as well, i.e., completely separates failure localization from the data plane dynamics for the price of increased monitoring costs.

5.6 The AG-NFL M-trail Allocation Problem

The input of the AG-NFL m-trail allocation problem is an undirected graph $G = (V, E)$ with node set V and link set E. As the result of the shared protection method a set of bidirectional W-LPs, denoted as $\mathcal{W}$, is given, as well as the set of forbidden link-pairs $\forall v \in V : \mathcal{F}_v$ can be constructed (see Section 5.6.1). The AG-NFL m-trail allocation problem is to establish a set of (bidirectional) S-LPs, which together with the status information of the traversing W-LPs (in case of an -IO design) helps each node to localize the switching link groups and to perform the proper switching actions when an arbitrary single link failure occurs, i.e., all the necessary P-LPs are activated for the disrupted W-LPs. The set of m-trails is denoted by $\mathcal{T} = \{T_1, \ldots, T_b\}$ where b is the number of m-trails, and $|T_l|$ denotes the number of (wavelength) links in m-trail T_l. The objective is to minimize the cover length:

$$\|\mathcal{T}\| = \sum_{i=1}^{b} |T_i|. \tag{5.1}$$

5.6.1 Constructing the Set of Forbidden Link-Pairs

Intuitively, the best AG-NFL performance is provided when the optimal set of forbidden link-pairs can be identified, i.e., the switching action sets contain the least conflicting switching actions. Although with a joint W-LP, P-LP and S-LP design with the goal to find a wavelength assignment minimizing the number of forbidden link-pairs could be optimal, it cannot be used in practice [87] owing to its high computational complexity. In our approach shared protection and m-trail allocation are solved independently one after the other. Hence, full *routing and wavelength assignment* information is given as the output of the shared protection approach, which defines the WP-action and PP-action sets for every link failure at each node. Thus, based on the WP-action and PP-action sets (calculated either for strictly-on-demand or loosely-on-demand traffic matrix), at each node the links can be partitioned into exact (E_e), approximate (E_a) and don't care links (E_n), and based on the allowable code collisions discussed in Section 5.4.1 the minimal set of forbidden link-pairs $\forall v \in V : \mathcal{F}_v$ can be constructed, and serves as the input of the m-trail allocation heuristic in Algorithm 6.

5.6.2 M-trail Allocation Heuristic for AG-NFL

As integer linear programs for m-trail allocation only allow to solve very small instances [87], we propose an efficient AG-NFL heuristic approach. Note that, all different m-trail designs in Section 5.5 can be solved with Algorithm 6 (and solves NWL-UFL and G-NFL as special cases, too), explained step by step as follows. The set of forbidden link-pairs $\forall v \in V : \mathcal{F}_v$ is given as the input of the m-trail allocation algorithm, and the investigated on-demand scenario depends only whether it was calculated based on a SOD (current) or LOD (maximal) traffic matrix. The W-LPs $\mathcal{W}$ are given as an input as well, and if their on/off status can be used for failure localization at the traversed nodes (-IO design), it is set in Step (2), otherwise we do not consider them (-O design).

Algorithm 6: Advanced Global Neighborhood Failure Localization M-trail Allocation

Input: $G = (V, E)$, W-LPs $\mathcal{W}$, $\forall v \in V : \mathcal{F}_v$

Result: $\mathcal{T}$ set of m-trails

1 **begin**
2 Set W-LP visibility from $\mathcal{W}$; `/* -IO design */`;
3 **for** $v \in V$ **do**
4 Construct current ACT $\underline{\underline{A}}^v$ at node v;
5 **for** $[e_1, e_2] \in \mathcal{F}_v$ **do**
6 **if** $A^v_{e_1} = A^v_{e_2}$ **then**
7 Identify working segment $s_1 = (u_1, v_1)$ of e_1 and $s_2 = (u_2, v_2)$ of e_2;
8 Use Dijkstra's algorithm to get the shortest paths from v to $\{u_1, v_1, u_2, v_2\}$;
9 Set $\mathcal{P}_1$ and $\mathcal{P}_2$ to the shortest path to $\{u_1, v_1\}$ and $\{u_2, v_2\}$, respectively ;
10 **if** $|\mathcal{P}_1| \leq |\mathcal{P}_2|$ ***or*** $e_2 \in E_n$ **then**
11 Add m-trail $\forall e \in \mathcal{P}_1 \cup s_1$ to $\mathcal{T}$;
12 **else**
13 Add m-trail $\forall e \in \mathcal{P}_2 \cup s_2$ to $\mathcal{T}$;
14 Refresh $\underline{\underline{A}}^v$;
15 **end**

In Step (3) each node $v \in V$ is considered one after the other, and the current ACT $\underline{\underline{A}}^v$ is constructed based on the W-LPs and already allocated S-LPs traversing through node v in Step (4). In Iteration Steps (5)-(14) we check whether forbidden link-pair e_1 and e_2 have the same alarm code seen at v or not. If yes, then we identify working segments in Step (7) if they exist (only for E_e and E_a links), i.e., consecutive links (path) with the same WP- and non-conflicting PP-action sets at v, and run Step (8) for the end-nodes of these segments instead of the end-nodes of its individual links. In Step (8) we calculate the cost of allocating an m-trail starting from v and traversing segment links of either s_1 or s_2, but disjoint

from the others' links, and the shorter path from $\mathcal{P}_1$ and $\mathcal{P}_2$ (extended with the segment links) is added to $\mathcal{T}$ in Step (11) or Step (13). Note that, in Step (10) the m-trail traversing s_1 is selected and added to $\mathcal{T}$ in Step (11) if $e_2 \in E_n$ even if $\mathcal{P}_1$ is the longer path. This is because an exact link must belong to a switching link group with non-zero alarm code for exact identification ($\mathcal{P}_1 \cup s_1$ already satisfies that), while a don't care link can have zero alarm code. After that, we refresh the ACT of v in Step (14) with the new m-trail.

As the result we have allocated m-trails to distinguish the failure of each forbidden link-pair from each other, while the alarm code collisions detailed in Section 5.4.1 were allowed in the switching link group formation. The collision of approximate link E_a alarm codes with don't care link E_n codes automatically follows from the the definition of $\mathcal{F}_v$, thus, *A-N* collisions (Observation (ii) of AG-NFL) and *N-N* collisions are not explicitly distinguished in Algorithm 6. Identifying working segments (Observation (i) of AG-NFL) in Step (7) ensures that the link failures within s are distinguished together from a given forbidden link-pair e_2. Hence, there is no need of additional m-trails in further iterations for forbidden link-pairs $[e_1 \in s, e_2]$, as they were already distinguished. However, the segment links can belong to several different switching link groups as the result of Algorithm 6, as distinguishing only the forbidden link pairs in $\mathcal{F}_v$ gives larger freedom in the possible *A-A* (and slightly in the *E-E*) code collisions in the m-trail design than the dominant action sets gave [C4].

The G-NFL m-trail allocation algorithm [89] has $O(|V|\cdot|E|\cdot|I|\cdot(|E|+|V|\log|V|))$ steps in the worst case, where $|I|$ is the maximum size of the neighborhoods (i.e., identification required $|I| = |E_e| + |E_a|$, typically $|I| << |E|$) and $O(|E| + |V|\log|V|))$ is the running time of the shortest path algorithm. However, the size of $\mathcal{F}_v$ is decreased to $O(|E_e| \cdot |E| + |E_a|^2)$ from $O(|E| \cdot |I|)$ in AG-NFL owing the introduction of exact and approximate links (the same if $|E_a| = 0$), reducing the overall complexity to $O(|V| \cdot (|E_e| \cdot |E| + |E_a|^2) \cdot (|E| + |V|\log|V|))$.

5.6.3 Computational Complexity of AG-NFL M-trail Allocation

Complexity of some m-trail allocation problems is known for multiple link failure localization [14], but the complexity of single link failure localization with m-trails is a largely unexplored area. We make the first contribution to this field, and settle the question of AG-NFL complexity for the SOD-IO and LOD-IO (denoted jointly as the AG-NFL -IO design problem) as follows.

Theorem 5.6.1 *To decide whether an AG-NFL solution with $\leq Z$ cover length exists for switching link groups and for -IO design (both SOD and LOD) is NP-complete.*

Proof: AG-NFL -IO is in NP, a solution with $\leq Z$ is a proof. Assume we are given an instance of the Hamiltonian $s-t$ Path Problem [29, GT39], the task is to decide whether a Hamiltonian path exists between nodes s and t in graph $G = (V, E)$. The polynomial time transformation of the Hamiltonian $s-t$ path problem to AG-NFL-IO is given as follows (shown in Figure 5.6). We add two dedicated nodes s' and t' to the graph, and connect these nodes to s and t with direct links $e_s = (s', s)$ and $e_t = (t, t')$.

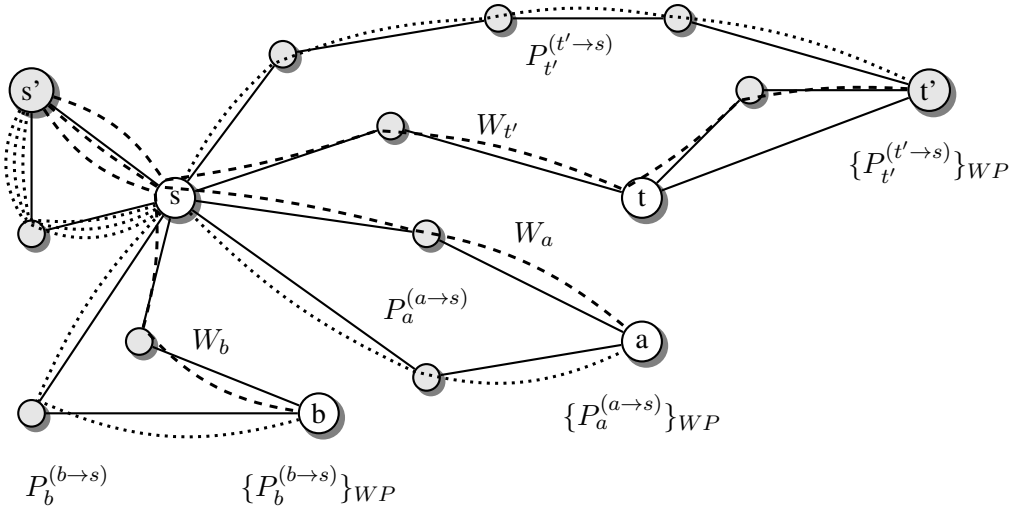


Figure 5.6: Polynomial-time reduction of Hamiltonian path to AG-NFL. Original nodes of $G = (V, E)$ are white, the added nodes are gray. Only added links are displayed, and links E are not shown. As the input of AG-NFL, the W-LPs (dashed) and P-LPs (dotted) and relevant action sets are shown [J4].

Furthermore, we add two-hop paths between s' and s, t and t', s and t; a four-hop path form s to t'; and 2 two-hop paths $\forall v \in V \setminus \{s, t\}$ between s and v, each with distinct new intermediate nodes.

A W-LP W_x starts at $\forall x \in \mathcal{X} = V \setminus \{s, t\} \cup \{t'\}$, and terminates at s' using only added links ($|\mathcal{X}| = |V| - 1$), as shown in Figure 5.6. Each W_x is protected by two P-LPs $P_x^{(x\to s)}$ and $P_x^{(s\to s')}$ – using only added links, too –, which defines WP-action set $\{P_x^{(x\to s)}\}_{WP}$ at each x. On the one hand, all nodes other than $\mathcal{X}$ in the transformed graph have sufficient information to perform the proper switching actions, either because it is an intermediate node of a W-LP or P-LP (gray nodes and t), or monitoring the status of the traversing W-LPs provides enough information (nodes s and s'). On the other hand, $\forall x \in X$ the status of W_x is not enough to perform the proper WP-action $P_x^{(x\to s)}$, as the darkness of W_x indicates both the failure of $e_s = (s \to s')$ and segment $s_x = (x \to s)$. Thus, the task in AG-NFL is to find a set of m-trails with cover length $\leq Z = |V| + 1$ which distinguishes e_s and segments $(x \to s)$ at each node of $\mathcal{X}$, i.e., $\forall x \in \mathcal{X}$ is a node of at least one m-trail which traverses e_s or s_x, but not both.

($\Rightarrow$) In a nutshell, we show that the AG-NFL -IO solution in the transformed graph with $\leq |V| + 1$ length must be a single $s' - t'$ simple path. It is important to observe that an m-trail traversing s_x (and disjoint from e_s) has useful information only for node x, while an m-trail traversing e_s disjoint from all s_x has useful information for all nodes along its path. First we show that, none of the two-hop s_x segments can be traversed in an m-trail solution with $\leq Z$ cover length. Let us assume indirectly, that the two-hop segment $s_y, y \neq t'$ is traversed by an m-trail T_y (distinguishes e_s and s_y only at $y \in \mathcal{X}$ with length 2), while all remaining $|V| - 2$ nodes in $\mathcal{X}$ can unambiguously localize s_x with $|V| - 1$ cover length. Even if we are able to allocate an m-trail $T_{s'}$ with $|V| - 1$ links traversing e_s, it spans at most $|V|$ different nodes, including s', s (because of e_s) and at least t (shortest way to get to t' is through link e_t)

which are not in $\mathcal{X}$. Thus, the status of e_s is known in at most $|V|-3$ nodes from $\mathcal{X}$, which contradicts the assumption that all remaining $|V|-2$ nodes in $\mathcal{X}$ can localize s_x. Note that a similar reasoning works if $y = t'$, except that in this case $T_{t'}$ has length 4.

Hence, each m-trail must traverse e_s, and each node in $\mathcal{X}$ must be on at least one m-trail. However, note that we need at least $|V|-2$ cover length (if we use original links E) to reach all nodes $\forall x \neq t' \in \mathcal{X}$ with m-trails traversing s. Thus, we have to reach t' with at most cover length 2, which is only possible through t using link e_t and one link from E. Thus, the AG-NFL-IO solution with $\leq Z$ cost must be a single spanning tree in $G = (V, E)$ extended with e_s and e_t. As only (simple or non-simple) paths are allowed in AG-NFL, the spanning tree in G must be an $s-t$ Hamiltonian path.

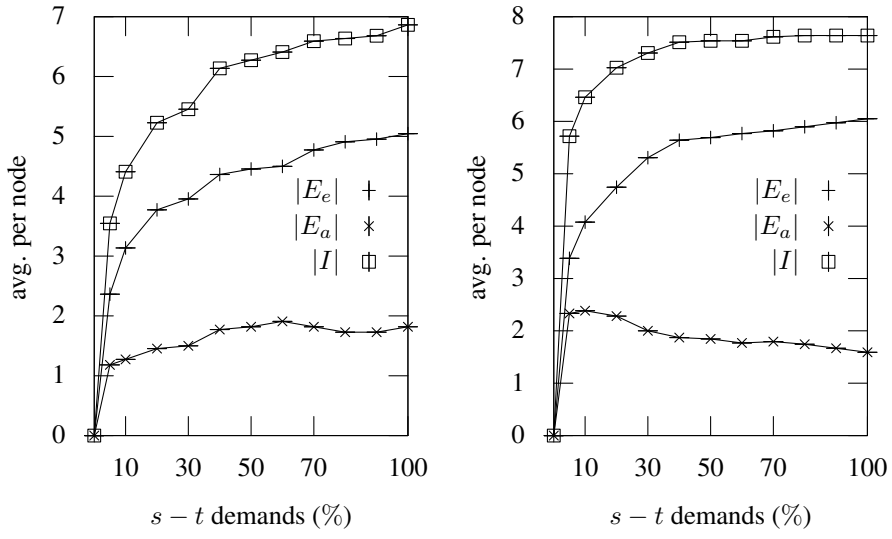
($\Leftarrow$) It is easy to convert the $s-t$ Hamiltonian path to an S-LP with $\leq Z = |V|+1$ length, simply extend it with e_s and e_t. Together with the W-LPs status information, all nodes in $\mathcal{X}$ can distinguish the failure of e_s from s_x. ■

5.7 Experimental Results

In this section we analyze the benefits of the link failure identification requirements we examined in this dissertation, and the resulting code collision types AG-NFL is built on. Furthermore, we compare the simulation results of different m-trail designs with different data plane dependencies. In the evaluation we use G-NFL as the baseline algorithm [89], which is the state-of-the-art solution for fast failure localization. We used real world topologies from the Internet Topology Zoo [49] in the comparisons. The W-LPs are bidirectional and are established using shortest-path routing. The routes of the P-LPs and corresponding WP- and PP-actions sets (and forbidden link-pairs $\mathcal{F}_v$) are derived form the result of SSP [37].

5.7.1 Code Collision Analysis

In Figure 5.7 we analyze the average number of exact identification ($|E_e|$), approximate identification ($|E_a|$) and no identification required ($|E_n| = |E| - |I|$) links per node on two real world topologies [49]. One can observe that the size of the neighborhood links do not vary significantly as the traffic increases, especially above 30% loaded unique $s-t$ pairs (with unique W-LPs and P-LPs). We see that the number of the links in the neighborhood ($|I|$) is relatively small, this means that the number of no identification required ($|E|-|I|$) links per node is much higher than the sum of exact and approximate links. For example, in the 39 node Bell Canada network the average number of exact links is around 5 and the number of approximate links is around 2 (from the 55 links). This means that the average number of no identification required links is around 48. This result suggests that the *A-N* collision type (Observation (ii)) should be more powerful, and more importantly always realizable (i.e., no conflicting switching actions) as it is independent from the concrete wavelength assignment. On the other hand, *E-E* and *A-A* collisions depend on the conflicting switching actions on the wavelengths, and generates forbidden link-pairs between



(a) AT&T (22 nodes, 38 links) (b) Bell Canada (39 nodes, 55 links)

Figure 5.7: The average number of exact ($|E_e|$) and approximate ($|E_a|$) links within the neighborhood ($|I| = |E_e| + |E_a|$) per node with varying traffic in Internet Topology Zoo topologies [49] [J4].

two exact and two approximate links. Hence, even if we assume that each exact and approximate link forms a forbidden link-pair (loosing the benefits of *E-E* and *A-A* code collisions, i.e., Observation (i)), owing to the refinement of the neighborhood to exact and approximate links the number of S-LPs for unambiguous switching link group failure localization can be lowered significantly compared to G-NFL, demonstrated in Section 5.7.2.

5.7.2 M-trail Allocation Analysis

In order to demonstrate that our AG-NFL approach is beneficial even with a "malicious" wavelength assignment, in these simulations we define each *E-E* and *A-A* link-pair to be forbidden. As a result, even with a naive wavelength assignment our results can be further improved. In Table 5.2 the results on real world topologies are presented. In each network W-LPs are deployed between 30% of random $s-t$ pairs (17 to 223 W-LPs depending on the network size). We analyze the different m-trail designs with different data plane dependencies in terms of average number of wavelengths per link required ($\|\mathcal{T}\|/|E|$) and total number of m-trails (transmitters) b needed [89].

Note that, even this worst case performance of AG-NFL SOD-IO significantly outperforms (with $30-55\%$) G-NFL SOD-IO in both average cover length per link ($\|\mathcal{T}\|/|E|$) and the total number of S-LP transmitters (b) required. If we prepare our network for all the future traffic (LOD-IO) the resource consumption of course increases. The difference between the AG-NFL and G-NFL slightly decreases but AG-NFL LOD-IO still significantly outperforms (with $25-50\%$) G-NFL LOD-IO. In case we assume that no in-band monitoring is allowed, both by SOD-O and by LOD-O the resource consumption increases significantly (by AG-NFL and G-NFL too), this it due the fact that we lose a lot of valuable

W-LP information, and we have to obtain this information by adding S-LPs instead. This way AG-NFL is only $10 - 30\%$ better than G-NFL; however, the difference in the number of transponders and cover length is the same between the two algorithms as in the -IO schemes. Although with the exclusion of the in-band monitoring we ensure that no reconfiguration is needed upon W-LP teardown (Table 5.1), the increase in resource consumption is significant.

In terms of data plane dependency and resource consumption trade-off the LOD-IO scenario seems to be a good choice, since the resource consumption is not significantly higher than by SOD-IO and the m-trail allocation is independent from W-LP setups. This is important since the recalculation of the m-trail solution and the physical deployment of new S-LPs upon each W-LP setup would make network management difficult. On the other hand, the substitution of an W-LP to S-LP is straightforward and seems to be manageable upon a W-LP teardown event with the price of increased number of transponders.

Table 5.2: Results on real network topologies from the Internet Topology Zoo [49]. In the SOD scenario 30% of $s-t$ pairs loaded. In the LOD scenario the same 30% of W-LPs are active, while forbidden link-pairs calculated for 100% traffic. AG-NFL columns contain worst case results with only *A-N* code collisions allowed (Obs. (ii)) in addition to G-NFL [J4].

Network topology			$\|\mathcal{T}\|/\|E\|$								b							
			SOD-IO		SOD-O		LOD-IO		LOD-O		SOD-IO		SOD-O		LOD-IO		LOD-O	
Name	$\|V\|$	$\|E\|$	G-NFL	AG-NFL	G-NFL	AG-NFL	G-NFL	AG-NFL	G-NFL	AG-NFL	G-NFL	AG-NFL	G-NFL	AG-NFL	G-NFL	AG-NFL	G-NFL	AG-NFL
Abilane	11	14	2.14	1.14	3.71	2.92	2.14	1.57	3.71	3.00	13	8	26	23	13	10	26	22
Germany	17	25	2.20	1.32	3.48	2.60	2.52	1.48	3.96	3.12	26	19	45	39	29	19	49	43
BtEurope	17	30	1.50	0.86	2.90	2.20	2.03	1.63	3.36	2.90	25	13	52	39	31	24	56	48
AS6461	17	37	2.16	0.97	3.56	2.37	2.59	1.75	3.97	3.02	44	19	75	50	53	34	83	63
InternetMCI	18	32	2.53	1.03	4.12	2.75	3.09	1.81	4.59	3.68	38	16	70	50	45	27	74	63
AS1755	18	33	1.60	0.81	2.72	2.15	2.54	1.90	3.78	3.24	30	17	54	41	42	32	68	59
ChinaTelc	20	44	1.56	0.77	3.68	3.20	5.15	2.56	6.93	4.75	37	18	86	73	105	52	145	101
AS3967	21	36	3.19	1.94	5.02	4.02	3.41	2.16	5.38	4.52	52	33	89	71	55	36	94	79
BellSouth	21	36	0.75	0.36	2.27	2.02	0.75	0.55	2.27	2.11	16	7	54	48	16	10	54	48
AT&T	22	38	1.76	0.86	4.05	3.34	2.50	1.73	4.71	4.23	33	17	76	60	48	35	90	80
NSF	26	43	3.00	2.00	5.60	4.76	3.65	2.97	6.32	5.79	55	38	106	91	65	54	116	107
BICS	27	42	2.88	1.07	5.42	4.40	3.42	2.02	5.90	4.85	50	22	105	83	58	37	110	89
AS3257	27	64	2.34	1.39	4.46	3.50	2.68	1.85	4.79	3.98	78	44	150	115	88	56	157	125
AS1239	30	69	2.39	1.01	5.49	3.98	3.94	2.14	7.18	5.37	80	35	181	130	121	65	226	164
Arnes	31	47	2.17	1.12	5.08	3.93	2.53	1.70	5.29	4.48	47	24	112	86	52	36	112	95
Geant	31	49	2.65	1.69	5.32	4.20	3.18	2.28	5.73	5.02	56	39	120	97	64	48	125	111
Italy	33	56	1.69	0.96	3.96	3.37	1.76	1.17	4.00	3.71	47	30	116	99	47	35	116	105
BtNAmerica	36	76	3.21	2.01	6.93	5.48	4.03	2.69	7.63	6.22	109	68	239	184	134	87	257	204
BellCanada	39	55	3.25	2.12	7.76	5.80	3.56	2.45	8.16	6.20	62	45	156	121	68	53	163	129

5.8 Summary of Results

In this chapter we investigated the AG-NFL m-trail allocation method which provides ultra-fast all-optical restoration for any shared protection scheme. Based on the switching action sets we divided the neighborhood links of a node into approximate and exact links. Based on these sets we identified *working segments* (Observation (i)) and allowed "non-disruptive" switching actions (Observation (ii)) to form switching link groups, whose failure can be identified as a whole instead of its individual links. In order to be able to identify the minimal set of conflicting switching actions the concept of forbidden link-pairs was introduced, which minimizes the number of switching link groups. As a result the AG-NFL enables a cheap implementation of any shared protection scheme for network operators. Note that our contribution is twofold. First, in order to reduce the dependency of AG-NFL on the data-plane dynamics, we introduced four different m-trail scenarios [86, 89] with decreasing data plane dependency for the price of increased monitoring resource usage (Section 5.5). Second, we improved the switching link group formation process by considering and excluding conflicting switching actions (forbidden link-pairs) in order to make all of these m-trail scenarios applicable for AG-NFL with an improved m-trail performance (Section 5.3). Through simulation we investigated the four m-trail designs with different data plane dependencies. We demonstrated that even in the worst case simulation scenarios (i.e. with a "malicious" wavelength assignment forcing us to define each *E-E* and *A-A* link-pair to be forbidden) the AG-NFL significantly outperforms the G-NFL.

5.8.1 Theses Summary

Thesis 3 *[C4, J4] I introduced a novel local failure localization framework called Advanced Global Neighborhood Failure Localization, which enables ultra fast recovery even for shared protection methods. I introduced a new concept called forbidden link-pairs which generalizes several m-trail problems (AG-NFL, G-NFL, NWL-UFL) and reduces their solutions complexity. I proved that both the AG-NFL SOD-IO and LOD-IO are NP-complete. These are the first complexity results for single link failure localization with m-trails. To solve this problem, I proposed a new heuristic.*

Thesis 3.1 (Generalized concept) *[J4] I introduced a new concept called forbidden link-pairs which generalizes several m-trail problems and reduces their solutions complexity. Based on this concept I proposed an efficient heuristic to solve these problems.*

Thesis 3.2 (AG-NFL complexity) *[J4] I proved that both the AG-NFL SOD-IO and LOD-IO are NP-complete.*

Chapter 6

Conclusion

In this dissertation two topics were investigated. Network coding based survivable routing methods with and without delay constraints in order to ensure *robust and instantaneous* failure recovery and *m-trail based local failure localization* in transport networks.

In particular in Section 3.1 a new practical (deployable), yet extremely resource efficient method called Survivable Routing with Network Coding (SRNC) was introduced for the arbitrary failure scenario. The method utilizes network coding to ensure resource efficiency. The user data is not split into arbitrary many parts as by the *Lower bound*, but into exactly two parts keeping the protection approach simple from equipment and management point of view. Hence the SRNC can be deployed in today's networks contrary to the *Lower bound*. Nonetheless the network coding capability has to be added in each node and the method is NP-complete. In Section 3.2 a very practical scenario is investigated i.e. the single link failure scenario. A new easily deployable and extremely resource efficient method called Survivable Routing with Diversity Coding (SRDC) is introduced. The method is easily deployable since the *coding itself can be done in the source and destination nodes, i.e. there is no need for a node capability update* (contrary to the SRNC). Several subproblems of SRDC were investigated depending on capacity and node capability constraints in the network. I proposed several ILPs and heuristics to solve the different subproblems. I proved that the $1 + 1$ dedicated protection is a 4/3 approximation of the subproblems without the capacity constraints (ICAN and ICCN) and does not approximate the problem with capacity constraints (CCAN and CCCN). For the ICAN subproblem I proposed an approximation algorithm. As shown in Section 6.1.1 the SRDC and SRNC are easily deployable in SDN networks, however, the SRDC does not consider any end-to-end delay characteristics, which are getting more and more into spotlight [23] with the proliferation of multi-media and streaming applications.

Several works are dealing with Quality-of-Service (QoS) routing [64] and differential delay (DD) aware [39] survivable routing in optical networks, however all of the methods are designed for disjoint paths only. In Chapter 4 we generalize these results for diversity coding-based survivable routing using DAGs. Hence, we define the before- and after-failure (single link) delay of an end-to-end DAG, and investigate the effect of QoS routing and DD-aware routing delay constraints on these optimal survivable

routing structures ensuring instantaneous recovery.

We show a graph transformation to an equivalent survivable routing problem, too, which traces back our problem to finding disjoint paths with delay constraints in a special graph. In Section 4.3 our complexity analysis, integer linear program and approximability result for the survivable QoS routing problem are presented, together with a heuristic solution for the differential delay aware routing problem in Section 4.4. The effectiveness of all routing problems (SRNC, SRDC, DARC) is confirmed through extensive simulations.

In the second part of the dissertation the focus is on the failure localization. I propose a new all-optical local failure localization framework, called Advanced Global Neighborhood Failure Localization (AG-NFL), that enables fast restoration even for the shared protection approaches, reducing the failure recovery time (t_R) significantly. Based on the switching action sets we divided the neighborhood links of a node into approximate and exact links. Based on these sets we identified *working segments* (Observation (i)) and allowed "non-disruptive" switching actions (Observation (ii)) to form switching link groups, whose failure can be identified as a whole instead of its individual links. In order to be able to identify the minimal set of conflicting switching actions the concept of forbidden link-pairs was introduced, which generalizes several m-trail problems (AG-NFL, G-NFL, NWL-UFL) and reduces their solutions complexity. As a result the AG-NFL enables a cheap implementation of any shared protection scheme for network operators. Through simulation we investigated the four m-trail designs with different data plane dependencies. We demonstrated that even in the worst case simulation scenarios (i.e. with a "malicious" wavelength assignment forcing us to define each *E-E* and *A-A* link-pair to be forbidden) the AG-NFL significantly outperforms the G-NFL.

6.1 Possible Application of the Results

The routing methods introduced in this dissertation i.e. the SRNC, SRDC and DARC methods could be easily deployed in SDN networks. The section describes our proof-of-concept SRNC implementation in a real-world transport network infrastructure, namely the OpenFlow Facility of the GÉANT European backbone network (GOFF). Since the SRDC and DARC does not require any complex node capability update they could be easily deployed in optical transport networks, too. The implementation of SRNC is a bigger challenge since the SRNC requires in-network coding. However, since the all-optical XOR already can be implemented [48] it is feasible.

6.1.1 Practical Applicability of SRNC and SRDC in Software Defined Networks

In order to investigate the possibility of the practical deployment of the new survivable routing based methods (SRNC, SRDC and DARC), we need to implement the modules which allow to form an arbitrary protection structure for the connection requests. In specific, when implementing the SRNC framework the network coding is the main challenge. Since the SRDC and DARC does not require network

coding inside the network (only on the edges i.e. in the source and destination node) it is much easier to implement. Thus, in this section we describe our proof-of-concept SRNC implementation in a real-world transport network infrastructure, namely the OpenFlow Facility of the GÉANT European backbone network (GOFF).

Design Considerations of Network Coding in Transport Networks

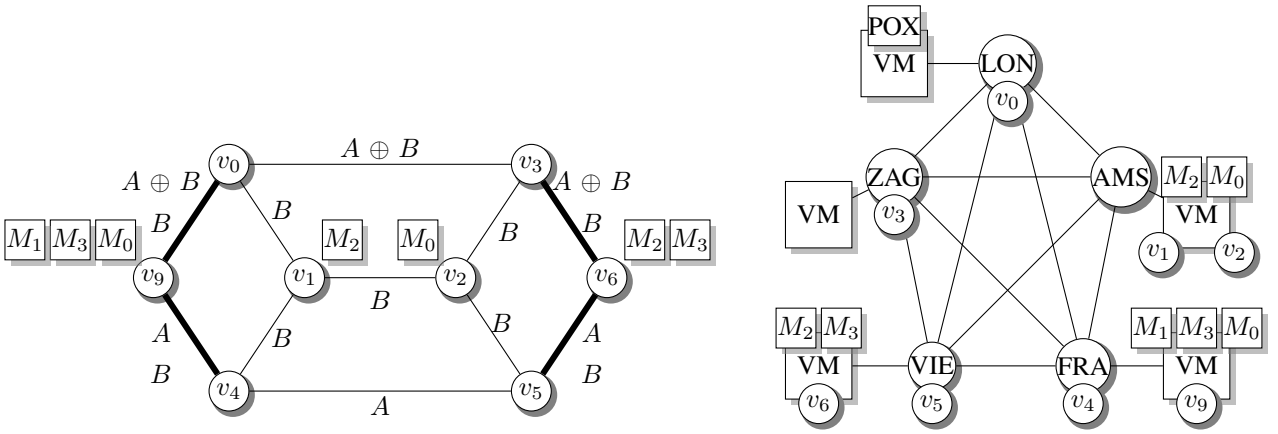
The existing implementations of network coding fall into two categories. They either put the coding/decoding functionality into the application layer of the end hosts [103], or they needed to substantially modify the software running in routers [47]. Relying on a modified end host requires end user involvement, which limits the possible deployment scenarios and reduces the number of coding possibilities. On the other hand, persuading equipment vendors to implement an experimental network coding function in their high-speed router is a slow process.

Luckily, the Software Defined Networking paradigm has been gaining acceptance in recent years [24]. SDN makes fast innovation possible by separating the control plane from the data plane. A control application (controller) can remotely "program" the packet forwarding behavior of an SDN capable switch. Unfortunately, though, the currently standardized protocol for controller-switch communication named OpenFlow [62] is not capable of instructing switches to modify packet payloads. However, the OpenFlow protocol conveniently allows steering traffic to middleboxes where complex packet manipulations can take place. Moreover, the advance in hardware technology now allows porting middlebox logic implemented on specialized hardware elements to normal software running on off-the-shelf hardware. In case of Network Function Virtualization (NFV), the network operator runs this packet processing software in a virtualized environment, for example, in a virtual machine. NFV allows dynamically placing so-called network functions next to different switches based on actual demands [25]. Thus, we decided to follow the NFV approach in our network coding implementation.

Identifying Network Functions for SRNC

Following the SDN/NFV principles, we implemented small building blocks as network functions (NF) allowing the creation of complex scenarios of SRNC. Our implementation facilitates practical deployment of SRNC, SRNC and DARC in high-speed networks, such as GOFF. Even if one NF cannot process packets at line speed, the SDN controller can divide traffic among multiple NFs of the same kind running on the same virtual machine connected to the OpenFlow switch. In the following, our NF-based prototype implementation of SRDC is demonstrated and possible implementation issues of the required network functions for SRDC (DARC) and SRNC are discussed.

Based on the flow values in the resilient subgraph of survivable routing the switching matrix configurations and necessary routing/network coding operations can be obtained. An example is presented on



(a) An example on a possible single link failure resilient SRDC solution and the corresponding placement of NFs. The bold edges carrying two units of data, while the thick edges forwarding one unit.

(b) The mapping of the butterfly topology to GOFF and the corresponding NF placement at the VMs

Figure 6.1: Experimental setup in the GÉANT OpenFlow Facility [J1]

the butterfly topology (as the resilient subgraph) in Figure 6.1, where the NF placement is shown assuming the network coding strategy in [16]. In order to deploy an arbitrary SRNC solution, the following types of NFs are required:

Sequencer (M_1):. Our SRNC implementation uses MPLS labels to mark different flows similarly as proposed in [18]. In this framework, the sequencer NF divides the user data (e.g., video stream) into parts A and B (odd and even packets in our implementation), and three MPLS labels are stacked to the packet header: the first identifies the payload type (i.e., flow A, B or $A \oplus B$), while the other two contain the sequence number, etc., of the A and B packets in the payload, respectively. The sequencer module is always placed to the source node[1], e.g., v_9 in Figure 6.1.

Splitter (M_0):. According to the definition in Section 2.4.1 the splitter duplicates the packets of its single incoming port and forwards them through two output ports. An OpenFlow switch can be programmed to follow this simple multicast rule, so no special NFs are needed (e.g., node v_2 duplicates the B flow in Figure 6.1a).

Merger (M_2):. According to the definition in Section 2.4.1 the merger node merges the two identical flows (packets) arriving on two different ports into one as demonstrated in Figure 6.1 too. In other words in practice the task of the merger NF is to forward one of the packets among the two identical copies arriving on the two incoming ports. Without loss of generality, the flow arriving on link (v_0, v_1) has lower delay, thus, in a failure-less state, the merger at v_1 forwards this flow and drops the duplicate packets arriving on link (v_4, v_1). If a failure occurs on the path traversing

[1] Note that the source node and destination node in this context refers to the ingress and egress node of the transport network, respectively. Thus, these modules are deployed as NFs, clearly differ from any coding implementation at the application layer of the end hosts.

(v_0, v_1), then v_1 should forward the traffic of the intact path arriving on link (v_4, v_1) instead. Note that, our goal is to maintain instantaneous recovery, while the detection of the failure of path on link (v_0, v_1) is never instantaneous, which results in traffic loss or increased recovery time. Thus, instead of detecting path failure, our merger NF implementation keeps track of the highest sequence number it forwarded. If a packet arrives to the merger either from link (v_0, v_1) or link (v_4, v_1), the merger checks whether its sequence is higher than current highest. If the answer is yes, then it forwards the packet and records its sequence number, otherwise it drops the packet. *This simple forwarding logic ensures instantaneous recovery from link failures.*

Coding/Decoding (M_3):. We implemented a simple XOR encoding that allows fast packet processing. Of course a more complex coding scheme can be implemented with the NFV approach. In our case from the two input streams A and B we create the $A \oplus B$ flow by XOR-ing the whole packet, and by restoring the checksum and other fields in the header for successful data transmission. Note that in a general SRNC scheme, where coding might be performed at intermediate nodes as well, or in the case of decoding at the destination (v_6) a buffer is also necessary in order to smooth the jitter caused by the end-to-end propagation delay difference between the different subflows. Otherwise, encoding and decoding network functions are very similar.

Note that, the Coding/Decoding type node is only necessary by SRNC, by SRDC the coding and decoding can and has to be done only in the source and destination node. However using the above network functions an arbitrary SRNC and or SRDC solution can be deployed in the network. Nonetheless, if the set of failure patterns $\mathcal{F}$ contains multiple link failures as well, coding schemes in [38, 41] might be necessary instead of simple XOR coding.

Implementation Results

In order to demonstrate the practical benefits of SRNC and SRDC for transport networks, we conducted our recovery time measurements on real equipments in a European size SDN network, the GÉANT OpenFlow Facility. The GOFF consists of five nodes connected as a full mesh. The switches are placed at Amsterdam, Frankfurt, Vienna, London and Zagreb connected with over 100 Gbps optical links. Each switch is connected to a server running a virtual machine (VM), on which the necessary network functions can be deployed. Our POX controller code is running at the London VM, and besides forwarding its main function is to map the butterfly topology in Figure 6.1a to the physical network in Figure 6.1b. Each of the A, B and $A \oplus B$ subflows are identified by different forwarding identifiers[2] and forwarded to the appropriate NFs obtained from the resilient SRNC or SRDC solution. We placed the source node with the corresponding NFs to the Frankfurt VM, while the destination node and decoder is running at the Vienna VM.

[2] We can use the custom network coding MPLS labels for this purpose, or the VLAN ID tag of the Ethernet header for forwarding the different subflows, depending on the OpenFlow version.

Measurement Results on Recovery Time

In this section, we present our measurement results which were conducted on the experimental setup in Fig 6.1b. We present our results as the average of multiple runs with different packet sizes and different packet arrival frequencies (from ms to sec) in order to avoid false data.

First, we have measured the performance of individual NFs in order to understand their performance characteristics. However, even in the most stressful scenarios, both the encoding and decoding delay was below the precision of the measurement ($\leq$ 1 ms). Thus, we can conclude that steering the traffic to the NFs at the virtual machines do not add any measurable delay to the path. Second, the end-to-end delay of the three end-to-end flows in the failure-less state was measured in the setup of Fig 6.1b. Note that the A flow in the mapped topology traverses physical link Frankfurt-Vienna, the shortest-delay path along the edges of the B flow traversing physical edges Frankfurt-Amsterdam and Amsterdam-Vienna, while the $A \oplus B$ flow is forwarded through links Frankfurt-London, London-Zagreb and Zagreb-Vienna. The end-to-end delay of the A, B and $A \oplus B$ flows are 41.6 ms, 48 ms and 69.4 ms, respectively.

As the main goal of our framework is to provide instantaneous recovery, we have investigated whether the $t_R < 50$ ms goal can be reached in a real transport network. From theory, the recovery time should be about the maximum of the pair-wise delay difference of the end-to-end paths of the original flows (A and B) and the encoded flow $A \oplus B$, as after a flow disrupts it can be reconstructed from the packets of the other two. This difference is $t_R = 27.8$ ms on average based on the previous results. This theoretical delay was completely backed by our measurement results. We have simulated the failure of the $v_1 - v_2$ virtual link forwarding the B flow at the Amsterdam VM, and measured the recovery time at the destination node (Vienna VM), i.e., the time difference between the last packet received from the original B flow, and the appearance of the first decoded B packet from the A and $A \oplus B$ flows. This delay was $t_R = 23$ ms on average.

Lessons Learned

In this section we have shown that in protection and restoration approaches without instantaneous recovery packet retransmission and maybe flow rerouting would be necessary, which causes severe packet loss and transport service outage (as both $t_l > 0$ and $t_n > 0$, and the recovery time t_R would also contain the end-to-end path delay when the flow is rerouted). We have measured the recovery of a single working path in the GOFF after a failure disrupts the connection, and even in the best scenario it was in the order of seconds. On the other hand, we demonstrated that even if a failure occurs, using our SRNC, DARC and SRDC approaches the source flow can be recovered instantaneously, without packet retransmission or flow rerouting.

Publications

Journal Publications

[J1] P. Babarczi, A. Pašić, J. Tapolcai, F. Németh, and B. Ladóczki. Instantaneous recovery of unicast connections in transport networks: Routing versus coding. *Elsevier Computer Networks*, 82(1):68–80, 2015.

[J2] A. Pašić, P. Babarczi, and A. Kőrösi. Diversity coding-based survivable routing with QoS and differential delay bounds. *Optical Switching and Networking*, 23(2):118–128, 2017.

[J3] P. Babarczi, J. Tapolcai, A. Pašić, L. Rónyai, E. R. Bérczi-Kovács, and M. Médard. Diversity coding in two-connected networks. *IEEE/ACM Transactions on Networking*, 25(4):2308–2319, 2017.

[J4] A. Pašić, P. Babarczi, and J. Tapolcai. Unambiguous switching link group failure localization in all-optical networks. *Networks*, 70(4):327–341, 2017.

Conference Proceedings Publications

[C1] P. Babarczi, J. Tapolcai, A. Pašić, S. R. Darehchi, and P.-H. Ho. New addressing scheme to increase reliability in mpls with network coding. In *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the*, pages 36–43. IEEE, 2013.

[C2] A. Pašić, J. Tapolcai, P. Babarczi, E. Bérczi-Kovács, Z. Király, and L. Rónyai. Survivable routing meets diversity coding. In *IFIP Networking*, pages 1–9, 2015.

[C3] A. Pašić and P. Babarczi. Delay aware survivable routing with network coding in software defined networks. In *Reliable Networks Design and Modeling (RNDM), 2015 7th International Workshop on*, pages 41–47. IEEE, 2015.

[C4] A. Pašić and P. Babarczi. Switching link group failure localization via monitoring trails in all-optical networks. In *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*, pages 92–99. IEEE, 2016.

[C5] Tornatore, Massimo and André, Joao and Babarczi, Péter and Braun, Torsten and Følstad, Eirik and Heegaard, Poul and Hmaity, Ali and Furdek, Marija and Jorge, Luisa and Kmiecik, Wojciech and

Mas Machucaand, Carmen and Martins, Lucia and Medeiros, Carmo and Musumeci, Francesco and Pašić, Alija and Rak, Jacek and Simpson, Steven and Travanca, Rui and Voyiatzis, Artemios. A survey on network resiliency methodologies against weather-based disruptions. In *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*, pages 23–34. IEEE, 2016. (3/18 = 0.16)

[C6] Gomes, Teresa and Tapolcai, János and Esposito, Christian and Hutchison, David and Kuipers, Fernando and Rak, Jacek and de Sousa, Amaro and Iossifides, Athanasios and Travanca, Rui and André, Joao and Jorge, Luísa and Martins, Lúcia and Ortiz Ugalde, Patricia and Pašić, Alija and Pezaros, Dimitrios and Jouet, Simon and Secci, Stefano and Tornatore, Massimo. A survey of strategies for communication networks to protect against large-scale natural disasters. In *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*, pages 11–22. IEEE, 2016. (3/15 = 0.2)

[C7] Pašic, Alija and Girao-Silva, Rita and Vass, Balázs and Gomes, Teresa and Babarczi, Péter FRADIR: A Novel Framework for Disaster Resilience. In *Resilient Networks Design and Modeling (RNDM), 2018 10th International Workshop on*, pages 1–7. IEEE, 2018. (3/4 = 0.75)

Independent Citations

[C6-1] K. Anazawa, T. Miyazaki, P. Li, and X. Wang. Big data synchronization among isolated data servers in disaster. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[C6-2] M. Aprile, N. Castro, F. Robledo, and P. Romero. Analysis and complexity of node-immunization under natural disasters. In *DRCN 2017-Design of Reliable Communication Networks; 13th International Conference; Proceedings of*, pages 1–8. VDE, 2017.

[C6-3] N.-H. Bao, G.-Q. Su, Y.-K. Wu, M. Kuang, and D.-Y. Luo. Reliability-sustainable network survivability scheme against disaster failures. In *Computer, Information and Telecommunication Systems (CITS), 2017 International Conference on*, pages 334–337. IEEE, 2017.

[C6-4] N.-H. Bao, Y.-K. Wu, G.-Q. Su, Y. Yuan, and D.-Y. Luo. Hierarchical fairness based re-provisioning in post-disaster telecom networks. In *Computer, Information and Telecommunication Systems (CITS), 2017 International Conference on*, pages 330–333. IEEE, 2017.

[C6-5] N. Castro, G. Ferreira, F. Robledo, and P. Romero. Graph fragmentation problem for natural disaster management. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 496–505. Springer, 2017.

[J2-1] P.-Y. Chen and M. Reisslein. Fiwi network throughput-delay modeling with traffic intensity control and local bandwidth allocation. *Optical Switching and Networking*, 28:8–22, 2018.

[C1-1] N. B. El Asghar, I. Jouili, and M. Frikha. Survivable inter-datacenter network design based on network coding. In *Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on*, pages 1192–1197. IEEE, 2017.

[C3-1] N. B. El Asghar, I. Jouili, and M. Frikha. Survivable inter-datacenter network design based on network coding. In *Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on*, pages 1192–1197. IEEE, 2017.

[C5-1] M. Gusev, S. Ristov, R. Prodan, M. Dzanko, and I. Bilic. Resilient iot ehealth solutions in case of disasters. In *Resilient Networks Design and Modeling (RNDM), 2017 9th International Workshop on*, pages 1–7. IEEE, 2017.

[J4-1] M. Jonsson, J. Rak, and A. Somani. Preface: Design of resilient communication networks. *Networks*, 70(4):281–282, 2017.

[C6-6] J. J. Kang and S. Adibi. Bushfire disaster monitoring system using low power wide area networks (lpwan). *Technologies*, 5(4):65, 2017.

[C6-7] J. J. Kang, I. Khodasevych, and S. Adibi. A disaster recovery system for location identification-based low power wide area networks (lpwan). In *Telecommunication Networks and Applications Conference (ITNAC), 2017 27th International*, pages 1–6. IEEE, 2017.

[J1-1] P. V. Pham, B. C. Chatterjee, A. H. Al Muktadir, and E. Oki. Instantaneous recovery route design scheme using multiple coding-aware protection scenarios. *Telecommunication Systems*, 64(1):75–85, 2017.

[C2-1] W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, and S. Sànchez-López. Managing resilience in carrier grade networks: Survey, open issues and trends. *Computer Communications*, 61:1–16, 2015.

[C2-2] W. Ramírez, X. Masip-Bruin, E. Marin-Tordera, M. Yannuzzi, M. Siddiqui, A. Martínez, and V. Lopez. Improving reliability in multi-layer networks with network coding protection. In *Optical Network Design and Modeling, 2014 International Conference on*, pages 240–245. IEEE, 2014.

[C2-3] W. Ramírez, X. Masip-Bruin, M. Yannuzzi, D. Montero, A. Martínez, and V. Lopez. Network coding-based protection scheme for elastic optical networks. In *Design of Reliable Communication Networks (DRCN), 2014 10th International Conference on the*, pages 1–8. IEEE, 2014.

List of Figures

Bibliography

[1] CPLEX. http://www.ilog.com/products/cplex/.

[2] LEMON: A C++ library for efficient modeling and optimization in networks. Technical report, http://lemon.cs.elte.hu.

[3] P. Agarwal, A. Efrat, S. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman. The resilience of wdm networks to probabilistic geographical failures. In *Proc. IEEE INFOCOM*, pages 1521–1529, 2011.

[4] R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network information flow. *IEEE Transactions on Inf. Theory*, 46(4):1204–1216, 2000.

[5] S. Ahuja, M. Krunz, and T. Korkmaz. Optimal path selection for minimizing the differential delay in ethernet-over-sonet. *Computer Networks*, 50(13):2349–2363, 2006.

[6] S. Ahuja, S. Ramasubramanian, and M. Krunz. Single link failure detection in all-optical networks using monitoring cycles and paths. 17(4):1080–1093, 2009.

[7] S. A. Aly, A. E. Kamal, and O. M. Al-Kofahi. Network protection codes: Providing self-healing in autonomic networks using network coding. *Computer Networks*, 56(1):99 – 111, 2012.

[8] S. Avci and E. Ayanoglu. Optimal algorithms for near-hitless network restoration via diversity coding. *IEEE Transactions on Communications*, 61(9):3878–3893, September 2013.

[9] E. Ayanoglu, I. Chih-Lin, R. Gitlin, and J. Mazo. Diversity coding for transparent self-healing and fault-tolerant communication networks. *IEEE Trans. on Communications*, 41(11):1677–1686, 1993.

[10] E. Ayanoglu, I. Chih-Lin, R. D. Gitlin, and J. E. Mazo. Diversity coding for transparent self-healing and fault-tolerant communication networks. *IEEE Transactions on Communications*, 41(11):1677–1686, 1993.

[11] P. Babarczi. *Survivable Optical Network Design with Unambiguous Shared Risk Link Group Failure Localization*. PhD thesis, Budapest University of Technology and Economics, 2011.

[12] P. Babarczi, G. Biczók, H. Overby, J. Tapolcai, and P. Soproni. Realization strategies of dedicated path protection: A bandwidth cost perspective. *Computer Networks*, 57(9):1974 – 1990, 2013.

[13] P. Babarczi, J. Tapolcai, and P.-H. Ho. Adjacent link failure localization with monitoring trails in all-optical mesh networks. *IEEE/ACM Transactions on Networking*, 19(3):907 – 920, June 2011.

[14] P. Babarczi, J. Tapolcai, and P.-H. Ho. SRLG failure localization with monitoring trails in all-optical mesh networks. In *Proc. Design Of Reliable Communication Networks (DRCN)*, pages 188–195, 2011.

[15] P. Babarczi, J. Tapolcai, P.-H. Ho, and M. Médard. Optimal Dedicated Protection Approach to Shared Risk Link Group Failures using Network Coding. In *Proc. IEEE ICC*, pages 3084–3088, 2012.

[16] P. Babarczi, J. Tapolcai, L. Rónyai, and M. Médard. Resilient flow decomposition of unicast connections with network coding. In *Proc. IEEE Intl. Symp. on Information Theory (ISIT)*, pages 116–120, 2014.

[17] I. Barla, F. Rambach, D. Schupke, and G. Carle. Efficient Protection in Single-Domain Networks using Network Coding. In *GLOBECOM 2010*, pages 1–6. IEEE, 2010.

[18] T. Biermann, A. Schwabe, and H. Karl. Creating butterflies in the core ï£¡ a network coding extension for mpls/rsvp-te. In *Networking*, pages 883–894, 2009.

[19] G. Brightwell, G. Oriolo, and F. B. Shepherd. Reserving resilient capacity in a network. *SIAM journal on discrete mathematics*, 14(4):524–539, 2001.

[20] G. Brightwell, G. Oriolo, and F. B. Shepherd. Reserving resilient capacity for a single commodity with upper-bound constraints. *Networks*, 41(2):87–96, 2003.

[21] M. R. Carter, M. P. Howard, N. Owens, D. Register, J. Kennedy, K. Pecheux, and A. Newton. Effects of catastrophic events on transportation system management and operations, howard street tunnel fire, baltimore city, maryland, july 18, 2001: Findings. 2002.

[22] C. Chekuri, A. Gupta, A. Kumar, J. S. Naor, and D. Raz. Building edge-failure resilient networks. In *Integer Programming and Combinatorial Optimization*, pages 439–456. Springer, 2006.

[23] P.-Y. Chen and M. Reisslein. Fiwi network throughput-delay modeling with traffic intensity control and local bandwidth allocation. *Optical Switching and Networking*, 28:8–22, 2018.

[24] A. Csaszar, W. John, M. Kind, C. Meirosu, G. Pongracz, D. Staessens, A. Takacs, and F.-J. Westphal. Unifying cloud and carrier network: Eu fp7 project unify. In *IEEE/ACM Utility and Cloud Computing*, pages 452–457, Dec 2013.

[25] C. Cui et al. Network functions virtualisation. Introductory White Paper, ETSI, Oct. 2012.

[26] G. Ellinas, E. Bouillet, R. Ramamurthy, J.-F. Labourdette, S. Chaudhuri, and K. Bala. Routing and restoration architectures in mesh optical networks. *Optical Networks Magazine*, pages 91–106, January/February 2003.

[27] C. Fragouli and E. Soljanin. Information flow decomposition for network coding. *IEEE Transactions on Information Theory*, 52(3):829–848, 2006.

[28] A. Fumagalli and L. Valcarenghi. IP restoration vs. WDM protection: is there an optimal choice? *Network, IEEE*, 14(6):34–41, 2000.

[29] M. Garey and D. Johnson. *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences*. WH Freeman and Company, San Francisco, Calif, 1979.

[30] K. N. Georgakilas, K. Katrinis, A. Tzanakaki, and O. B. Madsen. Impact of Dual-Link Failures on Impairment-Aware Routed Networks. In *Transparent Optical Networks, 2003. Proceedings of 2010 12th International Conference on*. IEEE, 2010.

[31] W. Grover, J. Doucette, M. Clouqueur, D. Leung, and D. Stamatelakis. New options and insights for survivable transport networks. 40(1):34–41, Jan. 2002.

[32] W. Grover and D. Stamatelakis. Cycle-oriented distributed preconfiguration: ring-like speed with mesh-like capacity for self-planning network restoration. In *Proc. IEEE ICC*, volume 1, pages 537–543 vol.1, 1998.

[33] L. Guo and H. Shen. Efficient approximation algorithms for computing k-disjoint minimum cost paths with delay constraint. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2012 13th International Conference on*, pages 627–631, 2012.

[34] L. Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[35] N. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V. Chan. Non-Adaptive Fault Diagnosis for All-Optical Networks via Combinatorial Group Testing on Graphs. In *IEEE INFOCOM*, pages 697–705, 2007.

[36] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations research*, 17(1):36–42, 1992.

[37] P.-H. Ho, J. Tapolcai, and T. Cinkler. Segment shared protection in mesh communication networks with bandwidth guaranteed tunnels. *IEEE/ACM Trans. on Networking*, 12(6):1105–1118, 2004.

[38] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Trans. on Inf. Theory*, 52(10):4413–4430, 2006.

[39] S. Huang, C. Martel, and B. Mukherjee. Survivable multipath provisioning with differential delay constraint in telecom mesh networks. *IEEE/ACM Transactions on Networking*, 19(3):644–656, 2011.

[40] L. Huawei Technologies Co. White paper on technological developments of optical networks. *white paper,Huawei Technologies Co., Ltd*, 2016.

[41] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Tran on InfTh*, 51(6):1973–1982, 2005.

[42] H. C. Joksch. The shortest route problem with constraints. *Journal of Math. analysis and applications*, 14(2):191–197, 1966.

[43] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó. Lagrange relaxation based method for the qos routing problem. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 859–868. IEEE, 2001.

[44] A. Kamal. 1+ N network protection for mesh networks: network coding-based protection using p-cycles. *IEEE/ACM Transactions on Networking*, 18(1):67–80, 2010.

[45] A. E. Kamal and M. Mohandespour. Network coding-based protection. *Optical Switching and Networking*, 11, Part B:189 – 201, 2014.

[46] D. Karger, R. Motwani, and G. D. S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997.

[47] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. *IEEE/ACM Transactions on Networking (TON)*, 16(3):497–510, 2008.

[48] J. H. Kim, Y. M. Jhon, Y. T. Byun, S. Lee, D. H. Woo, and S. H. Kim. All-optical xor gate using semiconductor optical amplifiers without additional input beam. *IEEE Photonics Technology Letters*, 14(10):1436–1438, 2002.

[49] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The Internet Topology Zoo. `http://www.topology-zoo.org`.

[50] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Trans. on Networking*, 11(5):782–795, 2003.

[51] J. Konemann, S. Leonardi, G. Schafer, and S. van Zwam. From primal-dual to cost shares and back: a stronger LP relaxation for the Steiner forest problem. *Automata, languages and programming*, pages 930–942, 2005.

[52] R. E. Korf, E. L. Schreiber, and M. D. Moffitt. Optimal sequential multi-way number partitioning. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM-2014)*, 2013.

[53] A. Kotb and K. E. Zoiros. Performance analysis of all-optical xor gate with photonic crystal semiconductor optical amplifier-assisted mach–zehnder interferometer at 160 gb/s. *Optics Communications*, 402:511–517, 2017.

[54] P. Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015.

[55] U. Kozat, G. Liang, K. Kokten, and J. Tapolcai. On optimal topology verification and failure localization for software defined networks. *IEEE/ACM Transactions on Networking*, 24(5):2899–2912, Oct 2016.

[56] B. Ladóczki, C. Fernandez, O. Moya, P. Babarczi, J. Tapolcai, and D. Guija. Robust network coding in transport networks. In *34th IEEE INFOCOM Demo session*, pages 27–28, 2015.

[57] S. LaPerrière. Taiwan earthquake fiber cuts: a service provider view. *NANOG39, Febraury*, 5, 2007.

[58] C.-L. Li, S. T. McCormick, and D. Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105–115, 1990.

[59] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, 2001.

[60] X. Ma and G.-S. Kuo. Optical switching technology comparison: optical mems vs. other technologies. *IEEE communications magazine*, 41(11):S16–S23, 2003.

[61] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of failures in an IP backbone. In *Proc. IEEE Infocom*, volume 4, pages 2307–2317. Citeseer, 2004.

[62] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

[63] F. Musumeci, M. Tornatore, and A. Pattavina. A power consumption analysis for IP-over-WDM core network architectures. *Journal of Optical Commun. and Networking*, 4(2):108–117, 2012.

[64] A. Orda and A. Sprintson. Efficient algorithms for computing disjoint QoS paths. In *23th IEEE INFOCOM*, volume 1, 2004.

[65] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. In *Proc. INOC*, 2007.

[66] H. Overby, G. Biczók, P. Babarczi, and J. Tapolcai. Cost Comparison of 1+1 Path Protection Schemes: A Case for Coding. In *Proc. IEEE International Conference on Communications (ICC)*, pages 3100–3105, 2012.

[67] F. Pakzad, M. Portmann, W. L. Tan, and J. Indulska. Efficient topology discovery in software defined networks. In *Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on*, pages 1–8. IEEE, 2014.

[68] D. Papadimitriou and E. Mannie. Analysis of generalized MPLS-based recovery mechanisms. Internet Draft, Sept. 2003.

[69] D. Papadimitriou and E. Mannie. Analysis of generalized multi-protocol label switching GMPLS-based recovery mechanisms (including protection and restoration). Technical report, RFC 4428, March, 2006.

[70] L. Quan, J. Heidemann, and Y. Pradkin. Trinocular: Understanding internet reliability through adaptive probing. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 255–266. ACM, 2013.

[71] J. Rak. *Resilient Routing in Communication Networks*. Springer, 2015.

[72] J. Rak, D. Hutchison, E. Calle, T. Gomes, M. Gunkel, P. Smith, J. Tapolcai, S. Verbrugge, and L. Wosinska. Recodis: Resilient communication services protecting end-user applications from disaster-based failures. In *Transparent Optical Networks (ICTON), 2016 18th International Conference on*, pages 1–4. IEEE, 2016.

[73] D. S. Reeves and H. F. Salama. A distributed algorithm for delay-constrained unicast routing. *IEEE/ACM Transactions on Networking (TON)*, 8(2):239–250, 2000.

[74] S. Rouayheb, A. Sprintson, and C. Georghiades. Robust network codes for unicast connections: A case study. *IEEE/ACM Transactions on Networking*, 19(3):644–656, 2011.

[75] L. Sahasrabuddhe, S. Ramamurthy, and B. Mukherjee. Fault management in IP-over-WDM networks: WDM protection versus IP restoration. *IEEE JSAC*, 20(1):21–33, 2002.

[76] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester. Fast failure recovery for in-band openflow networks. In *Design of reliable communication networks (drcn), 2013 9th international conference on the*, pages 52–59. IEEE, 2013.

[77] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester. Openflow: Meeting carrier-grade recovery requirements. *Computer Communications*, 36(6):656–665, 2013.

[78] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester. In-band control, queuing, and failure recovery functionalities for openflow. *IEEE Network*, 30(1):106–112, 2016.

[79] K.-S. Sohn, S. Y. Nam, and D. K. Sung. A distributed lsp scheme to reduce spare bandwidth demand in mpls networks. *IEEE Transactions on Communications*, 54(7):1277–1288, 2006.

[80] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145, 2002.

[81] A. Srivastava. Flow aware differential delay routing for next-generation Ethernet over SONET/SDH. In *IEEE ICC*, volume 1, pages 140–145, 2006.

[82] A. Srivastava, S. Acharya, M. Alicherry, B. Gupta, and P. Risbood. Differential delay aware routing for ethernet over SONET/SDH. In *24th IEEE INFOCOM*, volume 2, pages 1117–1127, 2005.

[83] D. Stokes. Optical transport networks and bandwidth demand. `https://insight.nokia.com/optical-transport-networks-and-bandwidth-demand`. Accessed: 2018-04-19.

[84] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[85] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[86] J. Tapolcai, P.-H. Ho, P. Babarczi, and L. Rónyai. On achieving all-optical failure restoration via monitoring trails. In *Proc. 32nd IEEE INFOCOM*, pages 380–384, April 2013.

[87] J. Tapolcai, P.-H. Ho, P. Babarczi, and L. Rónyai. *Internet Optical Infrastructure*. Springer, 2014.

[88] J. Tapolcai, P.-H. Ho, P. Babarczi, and L. Rónyai. On signaling-free failure dependent restoration in all-optical mesh networks. *IEEE/ACM Transactions on Networking*, 22(4):1067–1078, Aug 2014.

[89] J. Tapolcai, P. H. Ho, P. Babarczi, and L. Rónyai. Neighborhood failure localization in all-optical networks via monitoring trails. *IEEE/ACM Transactions on Networking*, 23(6):1719–1728, Dec 2015.

[90] J. Tapolcai, P.-H. Ho, L. Rónyai, P. Babarczi, and B. Wu. Failure localization for shared risk link groups in all-optical mesh networks using monitoring trails. *IEEE/OSA Journal of Lightwave Technology*, 29(10):1597–1606, 2011.

[91] J. Tapolcai, B. Wu, P.-H. Ho, and L. Rónyai. A novel approach for failure localization in all-optical mesh networks. *IEEE/ACM Transactions on Networking*, 19(1):275–285, 2011.

[92] J. Vasseur, M. Pickavet, and P. Demeester. *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers, 2004.

[93] P. Vizarreta, C. M. Machuca, and W. Kellerer. Controller placement strategies for a resilient sdn control plane. In *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*, pages 253–259. IEEE, 2016.

[94] H. Wang, E. Modiano, and M. Médard. Partial path protection for WDM networks: End-to-end recovery using local failure information. In *Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on*, pages 719–725. IEEE, 2002.

[95] J. Wang, C. Qiao, and H. Yu. On progressive network recovery after a major disruption. In *Proc. IEEE INFOCOM*, pages 1925–1933, 2011.

[96] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.

[97] Wikipedia. Internet outage Wikipedia, the free encyclopedia. [Online; accessed 22-05-2017].

[98] B. Wu, P.-H. Ho, J. Tapolcai, and P. Babarczi. Optimal allocation of monitoring trails for fast SRLG failure localization in all-optical networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.

[99] Y. Xiao, K. Thulasiraman, and G. Xue. Constrained shortest link-disjoint paths selection: A network programming based approach. *IEEE Trans. on Circuits and Systems I*, 53(5):1174–1187, 2006.

[100] G. Xue, W. Zhang, T. Wang, and K. Thulasiraman. On the partial path protection scheme for WDM optical networks and polynomial time computability of primary and secondary paths. *MANAGEMENT*, 3(4):625–643, 2007.

[101] R. Yadav and R. R. Aggarwal. Survey and comparison of optical switch fabrication techniques and architectures. *arXiv preprint arXiv:1004.4481*, 2010.

[102] Y. Yano, T. Ono, K. Fukuchi, T. Ito, H. Yamazaki, M. Yamaguchi, and K. Emura. 2.6 terabit/s wdm transmission experiment using optical duobinary coding. In *Optical Communication, 1996. ECOC'96. 22nd European Conference on*, volume 5, pages 3–6. IEEE, 1996.

[103] Y. Zhu, B. Li, and J. Guo. Multicast with network coding in application-layer overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1):107–120, 2004.

Index