Fault Localization in All-Optical Mesh Networks

by

Mohammed Liakat Ali

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Fault management is a challenging task in all-optical wavelength division multiplexing (WDM) networks. However, fast fault localization for shared risk link groups (SRLGs) with multiple links is essential for building a fully survival and functional transparent all-optical mesh network.

Monitoring trail (m-trail) technology is an effective approach to achieve the goal, whereby a set of m-trails are derived for unambiguous fault localization (UFL). However, an m-trail traverses through a link by utilizing a dedicated wavelength channel (WL), causing a significant amount of resource consumption. In addition, existing m-trail methods incur long and variable alarm dissemination delay.

We introduce a novel framework of real-time fault localization in all-optical WDM mesh networks, called the monitoring-burst (m-burst), which aims at initiating a balanced trade-off between consumed monitoring resources and fault localization latency. The m-burst framework has a single monitoring node (MN) and requires one WL in each unidirectional link if the link is traversed by any m-trail. The MN launches short duration *optical bursts* periodically along each m-trail to probe the links of the m-trail. Bursts along different m-trails are kept non-overlapping through each unidirectional link by scheduling burst launching times from the MN and multiplexing multiple bursts, if any, traversing the link. Thus, the MN can unambiguously localize the failed links by identifying the lost bursts without incurring any alarm dissemination delay. We have proposed several novel m-trail allocation, burst launching time scheduling, and node switch fabric configuration schemes. Numerical results show that the schemes, when deployed in the m-burst framework, are able to localize single-link and multi-link SRLG faults unambiguously, with reasonable fault localization latency, by using at most one WL in each unidirectional link.

To reduce the fault localization latency further, we also introduce a novel methodology called nested m-trails. At first, mesh networks are decomposed into cycles and trails. Each cycle (trail) is realized as an independent virtual ring (linear) network using a separate pair of WLs (one WL in each direction) in each undirected link traversed by the cycle (trail). Then, sets of m-trails, i.e., nested m-trails, derived in each virtual network are deployed independently in the m-burst framework for ring (linear) networks. As a result, the fault localization latency is reduced significantly. Moreover, the application of nested m-trails in adaptive probing also reduces the number of sequential probes significantly. Therefore, practical deployment of adaptive probing is now possible. However, the WL consumption of the nested m-trail technique is not limited by one WL per unidirectional link. Thus, further investigation is needed to reduce the WL consumption of the technique.

Acknowledgements

I would like to thank my thesis supervisor Dr. Pin-Han Ho for his advice, guidance, encouragement, and unwavering support. Without his help, it would have been impossible to complete this thesis.

I wish to thank the members of my PhD thesis committee: Dr. Johnny Wong and Dr. Bin Ma of the David R. Cheriton School of Computer Science, Dr. Sagar Naik of the Department of Electrical and Computer Engineering, and Dr. Martin Maier of the Institut National de la Recherche Scientifique (INRS), Montreal, Canada for their advice and guidance. I also wish to thank Dr. C. Perry Chou, chair of my thesis defense.

I would like to thank Mary McPherson of the Writing Center for her help to make the thesis more understandable. I wish to thank Dr. Ahmad Dhaini for his advice regarding thesis completion in time. My thanks go to Dr. Anna Lubiw for her discussion and valuable comments. I also thank Dr. Bernard Wong, Dr. Craig Kaplan, Dr. Martin Karsten, Dr. Srinivasan Keshav, and Dr. Douglas Stinson for their help.

I wish to thank my co-authors during my study at the University of Waterloo: Dr. János Tapolcai for his advice, guidance, and cooperation, Dr. Basam Shihada and Dr. Suresh Subramaniam for their helpful comments, and Dr. Bin Wu for his patience in introducing me to the design of effective ILPs.

I would like to thank Maureen Jones of the Student Awards & Financial Aid office for her advice about funding. I also thank Margaret Towell and Helen Jardine for their help and cooperation. Thanks to my fellow students, staff, and teachers at the David R. Cheriton School of Computer Science for their consultations, cooperation, and help.

I am grateful to my wife Salma for her sincere support and encouragement. Without her help, this thesis work could never have been completed. My appreciation goes to my sons, Rafi and Shwapneel, who gladly adjusted their schedules during my long period of study whenever it was needed.

Dedication

To my wife, Salma Kader Lovely

and

To my sons, Rafi and Shwapneel

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Background

In optical networks, each lightpath is set up between a source and a sink node before the start of data transmission from the source node through the lightpath. In all-optical networks, data remain in the optical domain during transportation through the network. Data conversions are needed only twice in each lightpath: E/O data conversion at the source node and O/E data conversion at the sink node. Slow and costly O/E/O data conversions are not needed at the intermediate nodes of the lightpath. As a result, all-optical technology has ushered in an era of phenomenal growth of data transmission rates. In addition, transparent all-optical networks can accommodate different data formats, protocols, or bit-rates at the same time.

Optical networks in general are built as fault tolerant systems. The survivability of optical networks is realized using failure detection and traffic protection/restoration techniques [41]. Therefore, high availability of an optical network is ensured. After detecting a failure event, protection/restoration techniques restore user traffic using spare capacity of the network. However, protection/restoration techniques may or may not localize link faults. Link protection/restoration schemes have to localize link faults, but path protection/restoration schemes may not localize link faults. By localizing link faults, however, performance of the path protection/restoration schemes can be enhanced significantly. Thus, fast fault localization schemes that identify faulty links unambiguously will help to reduce huge data loss during fault events in all-optical networks. Consequently, the fault localization schemes will improve QoS provided by the network.

Moreover, prompt recovery of the faulty links ensures high throughput of the network. Since fault localization is the very first step in bringing faulty links back into service again, fast and precise link fault localization in all-optical WDM mesh networks is needed for speedy recovery of faulty links. Thus, fast and unambiguous link fault localization is essential for building fully survival transparent all-optical mesh networks that operate with high throughput.

Network components are becoming more reliable with time, but component failures are inevitable in the long run. However, in this study we consider all network components except fiber links as invulnerable. Due to greater environmental exposure of optical cables, link faults such as fiber cuts happen much more frequently.

Link faults can be localized at various layers of the protocol stack of the network. Any link fault will trigger alarms in upper layers in due course of time and may cause an *alarm storm* if the fault is not localized and bypassed fast enough [15]. An alarm storm will make fault localization much more difficult. In fact, fault localization from the alarms that spread over various layers in a communication network is proven NP-complete in [24] by showing equivalence between the problem of finding the best explanation of the received alarms and the set cover problem. Thus, fast and unambiguous fault localization at the optical layer is desirable and crucial.

In opaque optical networks, link fault localization is comparatively easy because of electronic terminations of lightpaths at their intermediate nodes. Hence, data overhead bits are available at both end nodes of each link. Thus, link failures can be detected, correlated, and localized unambiguously at the end nodes of the faulty links by using performance metrics such as the bit error rate (BER) that can be derived from the overhead bits. As fault localization can be done on a single hop basis, single-link or multi-link shared risk link group (SRLG) fault localization becomes straightforward.

On the other hand, in all-optical networks, data overhead bits are not available at the intermediate nodes of a lightpath due to lack of electronic terminations. As a result, link failures can only be detected at intermediate nodes of a lightpath by using techniques such as loss of light (LOL) that need additional expensive monitoring equipment. Moreover, as failure indication propagates downstream through each lightpath traversing a faulty link, fault detection and fault localization schemes become separate processes [33], and fault localization turns into a multi-hop process. Therefore, link fault localization becomes a challenging task in all-optical networks.

Fault localization schemes in all-optical networks use traffic carrying user lightpaths or dedicated supervisory lightpaths (S-LPs) to detect link failures. Dedicated S-LPs are provisioned temporarily or permanently. To detect link failures at intermediate nodes of

lightpaths, dedicated monitors are used.

Link-based monitoring is widely used in the current commercial carrier backbones. An in-band link fault localization scheme is incorporated in the Link Management Protocol (LMP) [33], which is a new link management protocol introduced by Generalized Multi-Protocol Label Switching (GMPLS). The LMP detects LOL at each intermediate node of user lightpaths.

Link fault localization via in-band monitoring is based on the working status of user lightpaths. The schemes that utilize link failure detection only at the destination node of the lightpaths can use overhead bits of the data as in [29][44][74]; hence, the schemes need no additional expensive monitors for LOL detection or estimation of parameters like the signal-to-noise ratio (SNR). On the other hand, link failure detection at the intermediate nodes is based on techniques such as LOL detection or electronic processing of the signal tapped from the incoming/outgoing ports of the nodes by dedicated monitors; in this case, the schemes need $O(|E|)$ dedicated and expensive monitors. Fault localization with a reduced number of monitors is dealt with in [37][47].

In a probing scheme, temporary S-LPs instead of user lightpaths are set up periodically as per design requirements. A probe signal is sent through a temporary S-LP. If the probe signal arrives at the receiving node, there is no faulty link in the temporary S-LP route. Otherwise, there is at least one faulty link in the path. The temporary S-LP is torn down as soon as the on-off status of the traversed links is determined from the probing result [21][57]. In adaptive probing schemes [57], the next probing region is selected based on the result of the current probe. The probing process continues until all faulty links, if any, are localized. As probing is done sequentially, the adaptive probing schemes incur long and variable fault localization latency. Moreover, the number of monitors $|M|$ required by the adaptive probing schemes to detect link failures is in $O(|E|)$. In non-adaptive probing schemes [21], all the temporary S-LPs needed to localize link faults are set up simultaneously.

An out-of-band link-based fault localization scheme can be devised by using a single WL in each undirected link as a dedicated S-LP. Thus, the scheme needs $O(|E|)$ WLs, where $|E|$ is the number of links in the network. The required number of monitors $|M|$ of the link-based scheme is also in $O(|E|)$ [2][34][54][63][64][66].

Link fault localization via out-of-band monitoring is based on working status of dedicated permanent S-LPs. A set of permanent S-LPs are launched, and each S-LP is monitored at its receiver [1][2][51][54][61][63][64][66][70][71][72][73]. To reduce the required number of monitors $|M|$ in the order of the logarithm to the number of SRLGs, simple/non-simple m-cycles and m-trails are investigated in [54][60][61][63]. The S-LP allocation prob-

lem of these studies is a topology coding process: each SRLG is assigned a unique SRLG code. When an SRLG fails, each S-LP traversing any link of the SRLG is disrupted; as a result, the monitor at the receiver of the affected S-LP raises an alarm and the alarm is broadcast in the control plane. The network controller derives an alarm code based on the alarming and non-alarming monitors after collecting all the flooded alarms; then, the controller localizes the failed SRLG by matching the alarm code with an SRLG code [54].

The first concern during set up of a fault localization scheme is the time complexity of finding a set of lightpaths and/or locations of monitors such that link faults can be localized unambiguously as per the design requirement. There are two more major concerns regarding each fault localization scheme in all-optical networks: monitoring resource consumption and fault localization latency.

Usually, two types of monitoring resources are considered: the number of monitors $|M|$ and the number of dedicated wavelength channels (WLs) Λ. Monitors in all-optical networks are costly; in addition to their hardware and software costs, alarm management cost increases disproportionately as the number of monitors increases. On the other hand, dedicated supervisory WLs cannot be used for user data transportation. The trade-off between the two kinds of resources is defined in the literature as follows, where cost ratio γ determines the relative importance of the resources [54][63][65][66][73].

$$\begin{aligned} \text{“}Monitoring\ cost &= \gamma * monitor\ cost + bandwidth\ cost \\ &= \gamma * number\ of\ monitors + cover\ length\ in\ number\ of\ \text{WLs”} \\ &= \gamma * |M| + \Lambda \end{aligned} \tag{1.1}$$

In-band schemes that detect link failures only in destination nodes of the lightpaths incur insignificant monitoring cost. The link based and sequential probing schemes incur high monitor cost but minimum bandwidth cost. High bandwidth cost is a common concern of out-of-band schemes. Monitoring cost of the m-trail schemes deployed in the m-burst framework is reduced significantly.

Once a fault event occurs, a fault localization scheme incurs fault localization latency for failure detection, alarm dissemination, and alarm correlation. The fault localization latency can be expressed as follows.

$$\begin{aligned} Fault\ localization\ latency &= failure\ detection\ delay \\ &\quad + alarm\ dissemination\ delay \\ &\quad + alarm\ correlation\ delay \\ T &= T_{fd} + T_{ad} + T_{ac} \end{aligned} \tag{1.2}$$

All fault localization schemes incur failure detection delay, T_{fd}, that is more or less deterministic except in the adaptive probing schemes. The adaptive probing schemes are subject to long and variable failure detection delays. Reducing T_{fd} is the main concern of the schemes developed in this thesis.

In most fault localization schemes, alarms have to be flooded in the control domain since a monitor may be located at any node of the network, and each monitor may not be collocated with the network controller or the routing entities. The network controller or the routing entities can localize the fault only after collecting all the alarms. Thus, alarm dissemination delay, T_{ad}, is incurred. All fault localization schemes except a few incur alarm disseminate delays, but the problem is severe in some in-band schemes where each alarm has to traverse the affected lightpath that triggers the alarm, and in all adaptive probing schemes where alarms are generated sequentially. The alarm dissemination delay T_{ad} of the schemes developed in this thesis will be 0 because the schemes monitor the network from a single MN.

For successful deployment of fault localization schemes, the required computation to localize faults after collection of all the alarms should be done in real time. In other words, minimum alarm correlation delay T_{ac} is desired. This is a major concern of the fault localization schemes that need coordination among MNs. Alarm correlation delay T_{ac} of the schemes that utilize topology coding and alarm code table (ACT) look up is in O(1).

In this thesis, we deal with both single-link and multi-link SRLG fault localization. We consider a single link and/or multiple links up to d links as an SRLG. Thus, each failure event may involve a single link or up to d links. However, the context will make clear whether we are dealing with single-link or multi-link SRLG fault localization. We shall be explicit whenever needed.

1.2 Motivation

Fault localization is not studied as thoroughly as protection/restoration schemes in all-optical networks. Even optical channel monitoring (OCM) and optical performance monitoring (OPM) are more thoroughly investigated. It is probably due to the idea inherited from opaque optical networking that fault localization is easy, but it turns out that fault localization in all-optical network is really challenging.

At this point, single-link SRLG fault localization in all-optical networks is well studied in the literature. Many optimal and near optimal, in terms of monitoring cost $\gamma * |M| +$

Λ, single-link SRLG fault localization schemes are proposed. These schemes achieve the number of monitors $|M|$ in the order of the logarithm to the number of links $|E|$.

On the other hand, only a few efficient multi-link SRLG fault localization schemes have been proposed so far. The efficient schemes are available only for networks with special topology with high connectivity [21]. In general, the upper bound of the number of monitors $|M|$ for multi-link SRLG fault localization in an arbitrary all-optical network is in the order of $O(|\Psi|^2)$, where $|\Psi|$ is the number of SRLGs under consideration [1]. Monitoring resource requirements in terms of the number of monitors $|M|$ and the number of WLs Λ are high. Moreover, alarm dissemination delay T_{ad} remains an issue for almost all the multi-link SRLG fault localization schemes in all-optical networks.

To deal with the issues, Wu et al. [61] recently proposed an out-of-band method called the local unambiguous failure localization (L-UFL) scheme, which uses several monitoring nodes (MNs). Failure detection delay T_{fd} of the scheme is deterministic based on propagation delay along affected m-trails. The scheme avoids alarm dissemination delay T_{ad} altogether by enabling each MN to localize link faults. Moreover, alarm correlation delay T_{ac} of the scheme after collecting the alarms is in $O(1)$ because the scheme uses ACT look up. However, the scheme needs WLs per link $\Lambda_{avg} = 2$ to localize multi-link SRLG faults among 6 SRLGs in a 10-node topology. The required Λ_{avg} is considered high for localizing faults among such a small set of SRLGs.

We believe that there is scope to decrease monitoring resources further in terms of both the number of monitors $|M|$ and the number of WLs Λ while keeping fault localization latency T within acceptable limits at the same time for multi-link SRLG fault localization. Moreover, we want a balanced trade-off between consumed monitoring resources and fault localization latency.

1.3 Statement of Problem

To reduce the number of monitors $|M|$ and the required WLs Λ for fault localization, we propose a novel framework called the monitoring burst (m-burst) in this thesis. The m-burst framework has a single monitoring node (MN) and uses at most a single WL in each unidirectional link of the network. M-trail monitoring structure is utilized to detect link failures in the framework. As both the launched and received monitoring signals are available at the MN, significantly simplified hardware and software design of the monitors is possible [73]. The m-burst framework will be deployed to localize single-link and/or multi-link SRLG faults in all-optical mesh networks.

As a first step, a set of m-trails that provides a unique code for each SRLG under consideration is identified as a solution of the m-trail allocation problem. Then, the MN launches short duration optical bursts called monitoring bursts (m-bursts) repeatedly. Burst collision is avoided altogether by scheduling burst launching times from the MN and multiplexing multiple bursts, if any, traversing a unidirectional link of the network. In other words, the m-burst framework has to deal with three relevant problems: 1) to find a set of m-trails as a solution of the m-trail allocation problem, 2) to find the burst starting time from the MN of the m-burst along each m-trail in the solution such that no burst overlaps another in the network, and 3) to configure in real time the switch fabrics of the nodes to multiplex multiple bursts, if any, through the monitoring WL in each unidirectional link.

The MN launches one m-burst along each m-trail of the solution to probe the links traversed by the m-trail and wait for the complete return of all the m-bursts; once all the m-bursts return to the MN completely, it launches the next set of m-bursts. A *monitoring period* is defined as the duration of time from the launching of any m-burst from the MN to the complete return to the MN of that m-burst and of each m-burst launched once along each of the remaining m-trails in the solution. In other words, the monitoring period is the failure detection time T_{fd} needed by the MN.

During a fault event, the MN forms an alarm code based on which burst returns to the MN in due course of time and which one fails to return. The MN localizes the fault by matching the alarm code with an SRLG code.

The framework will completely avoid alarm dissemination. Thus, there will be no alarm dissemination delay, i.e., T_{ad} is 0. In addition, the alarm correlation delay T_{ac} will be in order of O(1) because ACT look up will be used to correlate alarms to faulty link(s). Thus, the fault localization latency T in the framework consists of mainly failure detection time T_{fd} needed by the MN.

This thesis mainly explores using at most a single dedicated WL per unidirectional link of an all-optical mesh network to localize single-link, or single-link and multi-link SRLG faults from a single MN with reasonable fault localization latency.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 provides related work on fault localization in all-optical networks. The existing methods are primarily classified on the number of simultaneous faulty links: single-link

and multi-link SRLGs. The single-link SRLG fault localization methods are again classified into five groups: in-band, probing, out-of-band, hybrid, and non-alarm disseminating. Similarly, the multi-link SRLG fault localization methods are classified into four groups: in-band, probing, out-of-band, and non-alarm disseminating. The review provides a comparative study of the complexity of setting up the fault localization schemes, monitoring cost, and fault localization latency of the existing and the proposed methods.

Chapter 3 describes the proposed m-burst framework in detail. As stated earlier, the m-burst framework has three relevant problems: m-trail allocation, burst launching time scheduling and node switch fabric configuration scheduling. The problems are formulated in the chapter. The methods proposed in the subsequent chapters provide solutions to the problems.

Chapter 4 provides an implementation of the m-burst framework for single-link SRLG fault localization in all-optical mesh networks. The m-trail allocation and burst launching time scheduling problems are formulated as joint and separate optimization problems via Integer Linear Programs (ILPs). Two corresponding heuristic algorithms are also devised to solve the problems in large networks. The burst launching time scheduling schemes ensure that the arrival times of two bursts to the sending node of a unidirectional link are separated by L ms if the bursts traverse the link, where L is the burst length in ms. The burst launching time scheduling methods developed in the chapter are used in other chapters directly or indirectly. Node switch fabric configuration scheduling related issues are also discussed.

Chapter 5 presents an implementation of the m-burst framework for multi-link SRLG fault localization in all-optical mesh networks, where an SRLG can have at most d links. A novel m-trail allocation scheme called the M-Trail Cover against the Faults (MCF) method is proposed. For each faulty SRLG, the MCF m-trail allocation method ensures that each remaining undirected link is traversed by at least one m-trail that is disjoint from the faulty SRLG. The MCF m-trail allocation scheme and the burst launching time scheduling problems are formulated as a joint optimization problem via an ILP, where the ILP takes a set of enumerated unique m-trails as the input. A heuristic algorithm is also devised to implement the MCF m-trail allocation scheme in large networks by finding an unaffected shortest path from the MN to each link. To avoid m-trail enumeration in ILP formulation and to reduce fault localization latency, an extension of the MCF m-trail allocation scheme called the Disjoint path-based M-Trail Cover against the Faults (DMCF) method is also proposed in the chapter. The DMCF m-trail allocation method ensures that each undirected link is traversed by $(d + 1)$ otherwise link-disjoint m-trails. The DMCF m-trail allocation scheme is formulated as an optimization problem via an ILP. Another heuristic algorithm is also devised to implement the DMCF m-trail allocation scheme in

large networks by finding link-disjoint shortest paths from the MN to each link.

Chapter 6, first provides an implementation of the m-burst framework for single-link SRLG fault localization in all-optical linear networks. Then the chapter provides an implementation of the m-burst framework for dual-link SRLG fault localization in all-optical ring networks. It is shown both theoretically and experimentally that the m-trail allocation, burst launching time scheduling, and node switch fabric configuration scheduling problems of the m-burst frameworks for linear and ring networks can be solved by inspection. Next, the techniques are used for an implementation of the m-burst framework for dual-link SRLG fault localization in all-optical mesh networks. Then, the techniques are used for another implementation of the m-burst framework for multi-link SRLG fault localization in all-optical mesh networks. A mesh network is decomposed into trails and cycles that serve as virtual linear and ring networks, respectively. Then, single-link and dual-link SRLG faults are localized in the virtual linear and ring networks using nested m-trails derived in the respective networks by inspection. As the sets of nested m-trails are deployed independently in the m-burst framework for linear or ring networks, the m-trail allocation, burst launching time scheduling, and node switch fabric configuration scheduling problems of the m-burst framework for mesh networks can be solved by inspection as well. An ILP is formulated to decompose mesh networks for dual-link and multi-link SRLG fault localization in mesh networks. A heuristic algorithm is also provided for network decomposition. Thorough theoretical analysis of the m-trail allocation, burst launching time scheduling, and node switch fabric configuration scheduling problems in the m-burst frameworks for SRLG fault localization in linear, ring, and mesh networks is conducted. Application of nested m-trails in adaptive probing is also investigated in the chapter.

Chapter 7 describes contributions of the thesis, provides the list of the publications based on the thesis, outlines future work, and draws conclusions.

Appendix A provides two simplified disjoint path algorithms. These algorithms avoid network transformation, link direction reversal, and vertex splitting.

Appendix B shows all the networks used in the numerical experiments in this thesis.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we discuss the related work on fault localization in all-optical networks to place our work in this thesis in an appropriate context. The underlying failure-indication propagation model, stated or implied, in all-optical networks is that failure indication will propagate downstream along the lightpaths established through each faulty link of the network. This deterministic propagation model is based on the optical signal flow model in all-optical networks. The propagation model helps to overcome the problems related with the inability to access data overhead bits at intermediate nodes of lightpaths, simplifies alarm correlation techniques, and facilitates devising numerous fault localization schemes in all-optical networks. We discuss single-link SRLG fault localization and multi-link SRLG fault localization in separate sections. The fault localization schemes are classified in each section. The main classification criterion is the kind of lightpaths used for failure detection. The other classification criteria are structures of S-LPs, ways of alarm dissemination, and use of monitors at intermediate nodes of user lightpaths.

Traffic protection/restoration techniques are deployed to ensure high availability of optical networks [41]. However, the performance of a traffic protection/restoration method depends on the relevant failure detection and fault localization schemes. An efficient and precise fault localization scheme in all-optical WDM mesh networks enables fast and automatic failure-dependent restoration in the event of any SRLG failure and is essential for high network throughput.

In an optical network where data go through O/E/O conversions in each node, failure detection and fault localization schemes can utilize data overhead bits. In SONET, the

end nodes of a faulty link localize the fault using the overhead bits and suppress the redundant alarms in both directions. On the other hand, in all-optical networks, failure detection and fault localization schemes cannot access the overhead bits at the intermediate nodes of a lightpath. As a result, the end nodes of a faulty link cannot localize the fault using bit-based schemes [30]. In fact, a failure detection scheme in all-optical networks needs additional expensive monitoring equipment at the intermediate nodes only for link failure detection. Moreover, as failure detection and fault localization schemes are separate processes in all-optical networks [33], fault localization schemes are to be built on top of failure detection schemes.

In 1997, Li and Ramaswami [35] proposed a Finite State Machine (FSM) based scheme for fault detection, isolation and recovery in broadcast-and-select and wavelength-routed all-optical networks. In wavelength-routed networks, one WL, called λ_0, is designated as a control channel that goes through O/E/O conversion in each amplifier. Each link is managed by two FSMs located at the end nodes. On the other hand, the FSM located at each amplifier controls two FSMs located at the two far ends of the link segments connected with the amplifier using λ_0. Moreover, router and converter faults are also managed by using FSMs. The method needs too many FSMs to be practically useful.

As we have described in [7], fault management is defined in the GMPLS as a set of real-time tasks performed sequentially right after the occurrence of any fault [39]. The tasks include fault localization and notification that are defined as a series of electronic signaling mechanisms. In all-optical networks, each downstream node of a failed user lightpath is subject to loss of light (LOL). By applying the Link Management Protocol (LMP) [32] coupled with a signaling protocol such as the resource reservation protocol with traffic engineering (RSVP-TE), the downstream node will send an alarm to its upstream node on the affected lightpath. After receiving the alarm, the upstream node checks the corresponding input port and forwards the alarm to farther upstream if the node itself is subject to LOL too. Otherwise, the downstream link is the faulty link, and the upstream node initiates protection/restoration procedures. Guo and Kuo [19] proposed an enhancement of the LMP scheme to expedite the fault localization process: whenever a node detects LOL in a lightpath, it sends alarms to both upstream and downstream nodes on the affected lightpath.

As we have identified in [7], the GMPLS-based approaches have three major weaknesses. First, multiple downstream nodes issue alarms simultaneously. The number of lightpaths traversing through the faulty link determines the number of alarms; in fact, the total length of the lightpaths measured in terms of the number of WLs determines the number of alarms. Such a large number of alarms could easily lead to an *alarm storm* in the control plane, and the alarm storm may crash the network. Second, the approaches cannot

handle multi-link SRLG failure events unambiguously because a node can only be aware of a faulty link if the link is a downstream link of any lightpath traversing the node, but the node has no way to know the status of a link if all lightpaths traversing the node are link-disjoint from the link. Therefore, when a multi-link SRLG fails, a node may only be able to identify a subset of the faulty links in the SRLG and may select a protection path for restoration that includes some of the faulty links. Third, due to extensive electronic signaling mechanism and nodal processing, the approaches may incur delay in hundreds of milliseconds just for the fault localization/notification process, and the delay is added to the overall restoration time. Note that a slow restoration not only causes data loss but also has negative impacts on the upper layer protocols such as OSPF and TCP.

To improve the GMPLS-based approaches, link-based monitoring [2][34][54][63][64][66] has been considered such that every link is exclusively monitored via a single-hop supervisory lightpath (S-LP). Once a failure occurs, each monitor subject to LOL will localize the upstream link as a faulty link and notify the network controller or the corresponding decision nodes (e.g., edge routers) for subsequent restoration processes.

The link-based schemes can localize faults that involve up to $|E|$ links, while the probability of faults in large number of links simultaneously is very low [16]. Although an effective solution to the problems of conventional GMPLS-based approaches, the link-based monitoring approach requires $|E|$ WLs along with $|E|$ transmitters/monitors that are considered precious resources in optical networks. Moreover, like the GMPLS-based approaches, link-based monitoring strongly relies on electronic signaling for failure notification, which leads to considerable control complexity and long restoration time. Thus, numerous schemes based on sophisticated designs and various assumptions were extensively reported in the past decades.

In this thesis, we are interested mainly in multi-link SRLG fault localization in all-optical networks. Thus, attack monitoring in general or fault localization only in WL granularity will not be discussed. Moreover, fault localization in discrete optical elements other than links or nodes is beyond the scope of the thesis. We will discuss exiting single-link and multi-link SRLG fault localization methods in Section 2.2 and Section 2.3, respectively. Fault localization using the m-burst framework is discussed in Section 2.4. Section 2.5 concludes the chapter.

2.2 Single-Link SRLG Fault Localization

The single-link fault localization schemes can be classified mainly based on the kind of lightpaths utilized for link failure detection: user lightpaths and supervisory lightpaths

(S-LPs). The schemes are classified broadly as in-band, probing, out-of-band, hybrid, and non-alarm dissemination methods. The non-alarm disseminating fault localization schemes are identified as a separate group because the methodology offers a very promising way to decrease fault localization latency significantly; in addition, all fault localization methods to be developed in this thesis will avoid alarm dissemination altogether.

2.2.1 In-band Methods

In-band methods utilize only user lightpaths for link failure detection. Each in-band method monitors lightpaths at their destination nodes. Whenever an in-band method detects link failure at intermediate nodes of lightpaths, the method needs additional expensive monitoring equipment. The in-band methods can be further classified based on whether a method needs failure detection at intermediate nodes or not.

Methods Detecting Failures at intermediate Nodes

In 2002, Stanic et al. [47] outlined a single network component fault localization method in transparent optical networks. The fault propagation model of the method is that fault indications propagate downstream of traffic lightpaths established through the faulty component, and each downstream active monitor traversed by at least one affected lightpath will raise an alarm. It is assumed that aggregate power monitors are pre-installed in each input port of network nodes. Thus, the number of monitors is in $O(|E|)$. The method activates the optimal number of monitors to localize a single faulty component.

At first, an alarm-matrix is created where each row represents a component of the network, and each column represents a monitor. An entry of the alarm-matrix is 1 if the monitor raises an alarm when the component becomes faulty and 0 otherwise. Each row is the alarm vector of the corresponding network component. In pre-processing, each row with zero alarm vector is deleted, and all rows having identical alarm vectors are combined together to form a single row. Then, redundant monitors are deleted from the alarm-matrix such that the resultant alarm-matrix has no zero alarm vectors, and all alarm vectors remain distinct. The authors prove in [45] that minimizing active monitors by deactivating redundant monitors is NP-complete using a polynomial time transformation of the Exact Three Cover (X3C) problem to the decision version of the problem addressed in the paper, called the Redundant Monitor Deactivation Problem (RMDP).

The authors formulate an ILP with constraints to ensure that each row vector is unique and non-zero, and propose a heuristic algorithm called the Greedy Min. The heuristic

method checks each monitor once if it can be deleted from the alarm-matrix. Monitors are selected in descending order of the total number of alarms raised for all faults by each monitor. The time complexity of the Greedy Min method is given in [48] as $O(n^2k)$, where n is the number of fault vectors, and k is the number of monitors in the pre-processed alarm-matrix. The method uses a fault localization tree to localize a single component fault after getting all alarms of a fault event and, consequently, needs $O(m)$ steps for alarm correlation, where m is the number of monitors in the optimized alarm-matrix; on the other hand, m-trail based methods need only $O(1)$ steps for alarm correlation.

In 2004, Sichani and Mouftah [43] proposed the Rolling Back Signaling Protocol (RBSP) to localize single-link faults in all-optical networks. Whenever the destination node of a lightpath does not receive an expected data stream, the node sends an alarm called Rolling Back Alarm (RBA) through the control channel OSC to its upstream node in the affected lightpath. An RBA recipient node checks whether it is receiving data or not through the affected lightpath. If the node is not receiving data, the node passes the RBA to its upstream node in the affected lightpath because the link between the RBA recipient node and its downstream node in the affected lightpath is not faulty. Otherwise, if the RBA recipient node is receiving data, it sends a fault ACK signal to its downstream node in the affected lightpath, indicating that the link between the RBA recipient node and the downstream node is faulty. After receiving an ACK signal, the downstream node starts a link restoration process and floods the network with fault location information. The minimum number of monitors required by RBSP is in $O(|E|)$.

In 2008, Khair, Zheng, and Mouftah [27] extended the limited-perimeter vector matching (LVM) [44] method for multi-domain all optical networks. Each domain edge node is equipped with additional monitors. When a link becomes faulty, the destination node of each lightpath traversing the faulty link and each edge node traversed by at least one affected lightpath detect the failure. The nodes exchange alarm messages to localize the faulty link.

If an ingress node of a domain detects the failure but its upstream egress node of the adjacent domain does not, the inter-domain link between the egress and the ingress nodes is faulty. Otherwise, if the destination node of each affected lightpath in a domain and each egress node of the domain traversed by at least one affected lightpath, and the corresponding upstream ingress nodes of the domain traversed by the affected lightpaths detect the failure, the fault is not within the domain. On the other hand, if the destination node of each affected lightpath in a domain and each egress node of the domain traversed by at least one affected lightpath detect the failure, but no ingress nodes of the domain detect the failure, the fault is within the domain. The faulty link is localized by using the LVM that utilizes only nodes and links of the faulty domain. The LVM considers each ingress

(egress) node traversed by an affected lightpath as a virtual source (sink) of the lightpath if it is not a real source (sink) of the lightpath. The minimum number of additional monitors required by the method is in $O(|E_{id}|)$, where E_{id} is the set of inter-domain links.

Methods Detecting Failures Only at Destination Nodes

In 2005, Zeng, Vukovic, and Huang [74] proposed an end-to-end fault localization scheme. The source node of each lightpath sends hello packets to the destination node periodically through the lightpath. If a fixed number of consecutive hello packets are lost during a pre-defined detection interval, the destination node will send an alarm to the network management system (NMS) and notify the affected nodes of the lightpath using the control plane. If path restoration is used, only the source node of the lightpath will be notified to initiate traffic recovery process. On the other hand, all upstream nodes will be notified if link restoration is used. After collecting all alarms, the NMS will localize the fault based on alarm distribution in multiple lightpaths and network topology. An optional traffic reinstatement process can start only after the faulty element is repaired. The method incurs long failure detection delay T_{fd} because the detection interval has to be large enough in order to avoid spurious alarm generation.

In 2007, Sichani and Mouftah [44] proposed the limited-perimeter vector matching (LVM) fault localization protocol. The LVM limits the fault localization process to within a small area of the network. Each destination node of the lightpaths passing through the faulty link detects link failure and floods the network with a message comprising the length of the affected lightpath and the ID of the node. If a node is the destination of more than one affected lightpath, the length of the shortest affected lingthpath is used in the message. The node with the shortest affected lightpath will be selected as the executive node; in case of a tie, the node with the lowest node ID will be selected. The executive node will form a link vector called the affected link vector (ALV) using its shortest affected lightpath and multicast the ALV to all the nodes in the ALV and their neighbors. These nodes define an area with a limited perimeter.

Each node of the area will form a link vector using nodes of the lightpath terminated at the node. Then, the node compares links of the link vector with that of the ALV to form a binary vector that consists of the same links as the ALV and a status bit. If the lightpath terminated at the node is affected by the fault, the status bit is assigned 0; each matched and each non-matched link is assigned 1 and 0 in the binary vector, respectively. Otherwise, the status bit is assigned 1; each matched and each non-matched link is assigned 0 and 1 in the binary vector, respectively. The binary vector is send to the executive node. The executive node localizes faults by performing logical `AND` operation on all the received

binary vectors. If a fault cannot be localized unambiguously, the executive node expands the perimeter of the area by including adjacent nodes of each node in the current area.

In 2008, Khair, Zheng, and Mouftah [25][28] extended the LVM again for single-link fault localization in multi-domains without using monitors at domain edge nodes. When the sink of an affected lightpath detects failure, the LVM is run within the domain of the sink; each ingress node of the domain traversed by an affected lightpath is considered as the virtual source of the lightpath. If the faulty link is within the domain, the LVM localizes the fault. Otherwise, each ingress node traversed by an affected lightpath sends relevant message to its upstream egress node of the adjacent domain. Each such egress node independently runs the LVM in its domain; again, each ingress node of the domain traversed by an affected lightpath is considered as the virtual source of the lightpath. The process will continue until the faulty intra-domain link is localized or the LVM is run in all the domains of the sources of the affected lightpaths. In the later case, the faulty link is an inter-domain link. During inter-domain link fault localization, the egress node of the source domain of each affected lightpath is considered as a virtual sink and each domain is considered as a node connected by inter-domain links. The LVM is run in the transformed network to localize the faulty inter-domain link.

In 2009, Khair et al. [26] proposed two ILP-based schemes to enhance the performance of the LVM method. It is observed that the LVM is able to localize a single-link fault unambiguously if at least two distinct lightpaths traverse the faulty link. In the first ILP, traffic distribution is optimized to satisfy static traffic demand such that the probability of fault localization is maximized. In the second ILP, the average hop length is minimized to reduce fault localization latency. The authors also propose a heuristic algorithm to optimize traffic distribution and to minimize fault localization latency by manipulating link cost. For each request, the cost of each link of the network is updated based on all previous WL assignments. Then, the shortest path between the s-d node pair of the request is found by using Dijkstra's shortest path algorithm. Link cost manipulation ensures traffic distribution optimization, and shortest path based lightpaths ensure minimum fault localization latency.

Discussion

The in-band methods that detect link failure only at the destination nodes of lightpaths can utilize data overhead bits, but most of the in-band methods that detect link failure at intermediate nodes of lightpaths need $O(|E|)$ additional monitors. In-band fault localization schemes except the method proposed in [74] do not incur any bandwidth cost; in fact, the method in [74] uses an insignificant amount of bandwidth resources for hello packets.

Failure detection delay T_{fd} of the schemes except the method in [74] is deterministic and short. The method in [74] incurs a predetermined but long failure detection delay T_{fd}. However, the in-band schemes incur long and variable alarm dissemination delay T_{ad} and alarm correlation delay T_{ac}.

2.2.2 Probing Methods

Probing schemes utilize dedicated S-LPs temporarily for link failure detection. As soon as the status of an S-LP is determined, the S-LP is torn down.

In 2007, Harvey et al. [21] provided outlines of *non-adaptive* probing schemes based on combinatorial group testing (CGT). A network is probed periodically, but the set of probes is launched simultaneously in a non-adaptive probing scheme. The sets of probes for single-link fault localization in all-optical linear, complete, 2-D and tree networks are identified. In the linear networks, $\lceil \frac{n}{2} \rceil$ probes are required where n is the number of nodes of the network. The authors also provide an upper bound on the number of probes required to localize single-link faults in complete, 2-D, tree, and any arbitrary networks as $\mathrm{O}(\log_2 n)$, $\mathrm{O}(\log_2 n)$, $\min\{\mathrm{O}(D \log_2 n), \mathrm{O}(D + \log_2 n)\}$, and $\mathrm{O}(D + \log_2 n)$, respectively, where D is the diameter of the network.

2.2.3 Out-of-band Methods

Out-of-band fault localization methods utilize S-LPs only for link failure detection. The methods are further classified based on the structure of the used S-LPs: m-cycle, m-trail, and m-tree methods. An m-cycle is a non-simple cycle and can traverse a node multiple times but a link only once. The transmitter, the receiver, and the monitor of an m-cycle are collocated at any on-cycle node. An m-cycle is considered as an m-trail. Typically, an m-trail is a non-simple trail and can traverse a node multiple times but a link only once. However, if an m-trail in the proposed m-burst framework traverses a link, it has to traverse the link from both directions. An m-tree can traverse a node multiple times but a link only twice, once in each direction. The receiver and the monitor of an m-trail or an m-tree are collocated at the sink of the S-LP.

M-cycle Methods

In 2004, Zeng et al., in a milestone paper [72], introduced the monitoring cycle, later renamed the m-cycle, for failure detection and path performance monitoring in all-optical

mesh networks. Two heuristic algorithms are proposed to find cycle covers of the network: Heuristic Depth First Search (HDFS) and Shortest Path Eulerian Matching (SPEM). However, the methods do not localize link faults unambiguously.

In the HDFS, all the nodes and links are initially marked uncovered, and m-cycles are formed by using Depth First Search (DFS) iteratively. Heuristic parts of the method are in the process of selection of the starting link of a DFS and next nodes/links during each DFS. Each starting link of a DFS has to be an uncovered one. If possible, an uncovered node/link is selected as the next node/link during a DFS. Otherwise, the largest and smallest node numbers are used alternatively during successive searches. Once a node/link is included in a cycle, the node/link is marked as covered. When a search reaches a previously visited part, an m-cycle is formed. Then a new DFS will be started if there is at least one uncovered link.

In a Eulerian graph, the SPEM can find a cycle cover easily because there exists a Eulerian cycle that cover all the links of the graph. Moreover, a set of m-cycles can be formed by traversing the Eulerian cycle end to end. Whenever a node is revisited, an m-cycle is formed and the links of the m-cycle are deleted from the Eulerian cycle. Link traversal of the Eulerian cycle continues until all the links are traversed and the starting node is reached. The set of m-cycles formed in this way will be a cycle cover of the network, and the m-cycles will have no common link. If the network is not a Eulerian graph, the SPEM transforms it into a Eulerian graph by adding links between odd degree node pairs such that the total number of added links is minimized. Then the SPEM identifies m-cycles in the modified network as if it is a real Eulerian graph but avoids those cycles that are formed with only two links. The m-cycles of a cycle cover in a non-Eulerian graph have some common links due to the added links. Performance of the SPEM is reported as better than that of the HDFS in terms of WL consumption.

In 2005, Zeng, Huang, and Vukovic [70] proposed another heuristic algorithm called Heuristic Spanning Tree (HST) to find a cycle cover of the network. The HST uses the breath first spanning tree (BFST) construction technique. Heuristic parts of the algorithm are in the process of selection of the root node and next nodes to be expanded in order to add links to the tree. At first, the node with maximum nodal degree is selected as the root of the tree T, and all links incident on the node are added to T. The method iterates until nodal degree labels of all the nodes in T become zero.

In the beginning of each iteration, nodal degree label of each node in T is updated by excluding all links incident on the node that has both end nodes in T. Next, node u with maximum nodal degree label among all nodes in T is selected as the next node to be expanded. Then each link incident on u that does not connect u with any node in T

is added to T. At the end of the iterative phase, T becomes a spanning tree. For each non-tree link called chord, an m-cycle is formed by using the cord and links in the tree called trunks. The HST constructs total $(|E| - |V| + 1)$ m-cycles for a cycle cover.

In 2006, Wu and Yeung [64] proposed M^2-CYCLE algorithm to localize single-link faults in all-optical networks based on enumeration of all minimum-length m-cycles (m^2-cycles) through each link of the network. The method has three phases. In the initialization phase, all the links are marked as uncovered, enumerated m^2-cycles are sorted on ascending order of their lengths, and the sorted m^2-cycles are assigned to the list θ. The expansion phase iterates as long as there is any uncovered link. Let F and T be two sets of links. In the beginning of each iteration, F and T are made empty. Now, θ is searched to find an m^2-cycle that traverses at least one uncovered link. If such an m^2-cycle is found, all the uncovered links traversed by the m^2-cycle is added to set F and an inner loop starts.

In the inner loop, each link in F is selected once: one link in each iteration. If any m^2-cycle derived from the selected link traverses any uncovered link, the m^2-cycle is added to tentative solution B, all uncovered links traversed by the m^2-cycle are marked as covered and newly covered links are added to set T. If T is not empty at the end of the inner loop, the links of F are replaced by the links of T, T is made empty, and the inner loop starts again with the new links in F. Otherwise, the method exits the inner loop and a new iteration of the expansion phase is started.

In the refinement phase, at first, an alarm code table (ACT) is constructed. Next, each column of the ACT is selected to check if the corresponding m^2-cycle can be deleted from B without violating two constraints: no link code should become zero, and two or more unique link codes should not become identical after deletion of the m^2-cycle. If the constraints are satisfied, the m^2-cycle is deleted from B. Finally, each link pair is checked whether their link codes are identical or not. If any link pair with identical link code is found, the shortest m^2-cycle among m^2-cycles derived from the links is added to B. At the end of the refinement phase, B has a set of m-cycles that can localize single-link faults. It is claimed that the M^2-CYCLE method outperforms the HST method "no matter how the spanning tree is constructed".

In 2007, Wu and Yeung [65] proposed a method called the heuristic ILP (HILP) [66] to solve the m-cycle design problem with minimum cost. The cost is defined as monitor and bandwidth costs, i.e., $Cost = Monitor\ cost + \beta * Bandwidth\ cost = number\ of\ Monitors + \beta * cover\ length\ in\ number\ of$ WLs, where β is a cost ratio. The method incorporates a trade-off between monitor and bandwidth costs. Moreover, the HILP allows the more versatile non-simple m-cycles in the solution.

In the HILP, cycles are defined with flow constraints only; in fact, only requirement of

the flow constraints for each cycle in the ILP solution and each node of the network is that the cycle is either node-disjoint from the node or only even number of on-cycle links should be connected with the node. Thus, the cycle definition is weak. As a result, the HILP provides cycle-sets instead of cycles in the solution; a cycle-set may contain more than one cycle. Thus, monitor counting in the objective function is not possible. Hence, the sum of link codes is used as heuristic measure of the monitor cost instead of monitor counting. The method provides a unique decimal code for each link or each segment of the network based on the cycle-sets traversing the link or the segment. Thus, the method is able to localize single-link faults unambiguously. It is claimed that the method outperforms the HST and M^2-CYCLE methods with a significant performance gain.

In 2007, Zeng and Vukovic [73] proposed a heuristic algorithm for unambiguous single-link fault localization called variant Constraint Cycle Cover Problem (vCCCP) that minimize monitoring cost while keeping cycle length within the given limit. The monitoring cost is defined as $\alpha M + \sum_{e_i \in E} \beta_i x_i$ where αM is the cost of monitors, x_i is the number of m-cycles traversing link e_i, and β_i is the cost coefficient to reserve a WL in link e_i. The vCCCP method has two phases: shortest-path candidate-cycle construction (SPCC), and branch and bound (B&B) search.

In the SPCC, candidate cycles are enumerated as follows. $\forall (u, v) \in E, \forall w \in V$: $u \neq w \neq v$, the method finds shortest paths between nodes u and w, and nodes v and w after removing link (u, v) temporarily. Then it forms a cycle using the link and two shortest paths. Next, non-simple cycles, two link cycles and cycles exceeding the given length limit are removed from the candidate cycle set, and the remaining cycles are sorted based on the cycle cost. If the candidate cycle set is not enough to localize single-link faults unambiguously, the SPCC adds more cycles to the set to enable UFL.

In the B&B search, a full binary tree is constructed. The root of the tree represents a virtual cycle with zero cost. One candidate cycle is assigned to all nodes of a level of the binary tree. Cycles are assigned from the root to the leaf in ascending order of cycle costs. Each node of a level is labeled either 0 or 1. Each non-leaf node is connected with two children nodes labeled 0 and 1. Each route/sub-route from the root to a leaf represents a combination of candidate cycles where a particular candidate cycle is included only if the route visits any label 1 node of the candidate cycle. Thus, the route/sub-route can be a tentative solution. The depth-first search is conducted in the tree to find solutions. At each level, link coverage and condition of UFL are checked. Once a route/sub-route satisfies link coverage and condition of UFL, the route/sub-route is taken as a new solution and sub-tree, if any, rooted at the node is pruned because adding more cycle from the sub-tree to the new solution will increase cost. Then monitoring cost of the new solution is calculated. If the cost is less than that of the current solution, the new solution is taken

as current solution. At the end of tree traversal, the current solution is taken as the near optimal solution.

In May 2009, Wu, Yeung, and Ho in another milestone paper [66] proposed an optimal ILP based method to localize single-link faults in all-optical networks. To solve the monitor counting problem in the formulated ILP, additional flow constraints among on-cycle nodes are introduced to keep each cycle-set a single structure. Though the additional flow constraints are complex, the technique is a significant development toward formulation of simplified ILPs such that each monitoring structure remains a single connected component.

In August 2009, Ahuja, Ramasubramanian, and Krunz proposed a heuristic algorithm called Monitoring Cycle Problem with Multiple Monitoring Locations (MC-M) to localize single-link faults in three-edge-connected all-optical networks from a given set of monitoring locations, which are also known as monitoring nodes (MNs). The MC-M is the second algorithm proposed in [2]. To apply the MC-M to an arbitrary network, the minimum number of MNs and their locations can be derived by using the monitor placement algorithm proposed in the paper.

The MC-M finds a set of monitoring cycles (MCs), which are simple m-cycles, as a solution of the fault localization problem. The MC-M has two modes of operation. In the MC-M with information exchange mode, each link of the network is traversed by a unique subset of m-cycles in the solution. Monitors of m-cycles traversing a faulty links raise alarm in the control plane. A routing entity can localize a single faulty link after collecting all the alarms. On the other hand, in the MC-M without information exchange mode, each MN can localize independently a single faulty link in its cloud without alarm dissemination.

The MC-M has two phases. In first phase, one cloud is formed for each MN. Each link of the network is included to the cloud of its nearest MN. In second phase, each MN is considered in turn. All links not in the cloud of the current MN are assigned large weights to avoid them in cycle formation as much as possible. Next, m-cycles are added to the solution by running the MC-1 method from the current MN until each link of the cloud of the current MN is traversed by a unique set of m-cycles: each m-cycle derived for the current MN traverses the MN. The MC-1 is described in Section 2.2.5. In the MC-M with information exchange mode, each link of a cloud is traversed by a unique set of m-cycles considering all m-cycles added to the solution so far; but in the MC-M without information exchange mode, each link of a cloud is traversed by a unique set of m-cycles that does not include any m-cycles derived for other clouds. To enhance performance of the method, a technique called cloud transfer is introduced. If a non-cloud link is traversed by a unique set of m-cycles when a new m-cycle is added to the solution, the link is transferred to the

cloud of the current MN.

M-trail Methods

In 2008, Wu, Ho, and Yeung in another milestone paper [62] proposed the monitoring trail (m-trail) to localize single-link faults unambiguously in the presence of two-edge cuts without using any additional link-based monitor. Like an m-cycle, an m-trail needs a transmitter, a receiver, and a dedicated S-LP but has no cycle constraint. The authors formulated an ILP to find a set of m-trails with minimum monitoring cost to localize single-link faults unambiguously. *Monitoring cost* is defined as *monitor cost* + *bandwidth cost* = γ * *number of monitors* + *cover length*, where γ is a cost ratio. The method ensures a unique decimal code for each undirected link of the network. As a unique code represents a unique set of m-trails, each link of the network will be traversed by a unique set of m-trails.

M-trails are defined with flow and voltage constraints. With only flow constraints, an m-trail can have at most two odd degree nodes and may consist of multiple connected components. Voltage constraints are introduced in the paper to supplement the weak cycle/trail definition provided by flow constraints and to ensure that each m-trail in the ILP solution is a single connected component that traverses the sink of the m-trail. The voltage constraint for the jth m-trail in an ILP solution at each node u is defined as $r_u^j + \sum_{(u,v)\in E}(q_{uv}^j - q_{vu}^j) \geq \lambda\, z_u^j$, where λ is a predefined small constant, binary variable r_u^j is 1 if u is the sink of the m-trail and 0 otherwise, voltage variable q_{uv}^j can take any non-negative fractional value if $u \rightarrow v$ is an on-trail vector of the m-trail, and binary variable z_u^j is 1 if the m-trail visits u and 0 otherwise. Voltage constraints stipulate that voltages of outgoing on-trail links should be equal to or greater than that of incoming on-trail links of each node except the sink of each m-trail in the solution. It is proved that if any connected component of an m-trail does not traverse the sink of the m-trail, the component will encounter a voltage conflict. The concept of voltage constraints is a significant development to simplify ILP formulation of out-of-band monitoring schemes.

In April 2009, Tapolcai, Wu, and Ho [54] proposed an efficient heuristic algorithm for finding an m-trail solution set in logarithmic order of the number of links $|E|$ in the network to localize single-link faults unambiguously. The method solves the m-trail design problem in large networks and has three modules: 1) Random Code Assignment (RCA), 2) Cost evaluation, and 3) Random Code Swapping (RCS). The method maintains a unique alarm code for each link all the time.

In the RCA, each link is assigned a unique alarm code. Thus $\lceil \log_2(|E|+1) \rceil$ linksets are formed but each linkset may not be an m-trail. If the jth binary bit of the code of a

link is 1, the link is in linkset L_j. The cost evaluation is done once after the RCA. Next, the RCS and cost evaluation are done one after the other as long as there is any possibility of monitoring cost reduction. Before every cost evaluation, however, valid m-trails are formed by splitting or merging linksets if necessary. The RCS is performed independently on each linkset L_j in each round to reduce number of odd-degree nodes in the linkset by swapping link codes of a link pair randomly. If alarm codes of two links are identical in all bit positions except in the jth bit, they can be swapped. Thus, swapping will change links of linkset L_j and their connection pattern only.

It is proved in [54] that a ring network with more than four links needs $\lceil \frac{|E|}{2} \rceil$ m-trails, any two-connected network needs at most $\lceil \frac{|E|}{2} \rceil$ m-trails and a fully meshed network needs $6 + \lceil \log_2(|E| + 1) \rceil$ m-trails for UFL. In [55], the upper bound for fully meshed network is improved to $4 + \lceil \log_2(|E| + 1) \rceil$ m-trails for UFL. It is found that the number of m-trails is upper bounded by $\lceil \log_2(|E| + 1) \rceil + \frac{\#\text{degree-2 nodes}}{2}$ when cost of a monitor is very high compare to that of a WL.

In August 2009, Ahuja, Ramasubramanian, and Krunz proposed a heuristic algorithm called the Monitoring with Paths and Cycles (MPC) to localize single-link faults in three-edge-connected all-optical networks from a given set of monitoring locations, which are also known as monitoring nodes (MNs). The MPC is the third algorithm proposed in [2]. To apply the MPC to an arbitrary network, minimum number of MNs and their locations can be derived using the monitor placement algorithm provided in the paper.

The MPC uses network transformation and finds a set of monitoring paths (MPs) and monitoring cycles (MCs) in the transformed network. MPs and MCs are simple trails and cycles, respectively. Thus, MPs and MCs are simple m-trails.

In the MPC, each MN is split into two virtual MNs, and each link connected to the MN is also split to connect the virtual MNs with the link's other end node. All virtual MNs are connected to a virtual super MN J using auxiliary connectors. Now a modified MC-1 is applied to the transformed graph. The MC-1 is described in section 2.2.5. The modified MC-1 finds a unique set of cycles for each link of the network considering the transformed graph as a flow network to avoid self-loops. If a cycle traverses both the virtual MNs of an MN, it will be a cycle. Otherwise, it will be a trail.

In 2010, Haddad, Doumith, and Gagnaire [20] proposed a heuristic algorithm called the meta-heuristics for monitoring trail assignment (MeMoTA) to solve the m-trail design problem. Initially, the number of m-trails T is equal to $\lceil \log_2(|E| + 1) \rceil$, and the MeMoTA uses Tabu search technique iteratively to find a solution that has T m-trails as described below.

In the beginning of a Tabu search, the MeMoTA assigns a unique link code that is chosen randomly from 1 to $2^T - 1$ to each link of the network, modifies the corresponding linksets into initial m-trails and adds the initial m-trails to an empty ACT. Whenever a linkset becomes an m-trail, the MeMoTA maintains the structure of the linkset as an m-trail all the time. Then, it selects an m-trail t_j from the ACT randomly and reshape the m-trail by adding and removing links starting from the highest ambiguity reducing group without violating the full link coverage constraint. The full link coverage constraint ensures that each link of the network is traversed by at least one m-trail. If all reshaped m-trails lead to high monitoring cost, t_j is added to Tabu list ξ. The m-trail remains in the Tabu list until monitoring cost improves due to reshaping of another m-trial. When all the m-trails are added to ξ, the search is terminated and the new solution in the ACT is evaluated.

If the new solution is an optimal solution, the MeMoTA terminates. Otherwise, if the new solution is better than the reference solution, the new solution is taken as the reference solution. Now, If the search does not return the reference solution n_r times or does not terminate n_i times without improvement, a new Tabu search is conducted without increasing T. On the other hand, if the search returns the reference solution n_r times or terminates n_i times without improvement, the MeMoTA checks the reference solution. If the reference solution achieves UFL, the MeMoTA terminates. Otherwise, next Tabu search is conducted with incremented T.

M-tree Methods

In 2010, Tapolcai, Rónyai, and Ho [53] investigated application of the monitoring tree (m-tree) structure. The upper bound on the number of m-trees to achieve unambiguous single-link fault localization in the chocolate bar graphs is derived as $\lceil 0.42 + \log_2(|E| + 2) \rceil$. The upper bound on the number of m-trees to achieve unambiguous single-link fault localization in two dimensional lattice networks is derived as $3 + \lceil \log_2(|E| + 1) \rceil$ by using chocolate bar graphs.

In 2010, Doumith, Zahr, and Gagnaire [17] proposed a new construction of a monitoring tree (m-tree) in ILP to localize single-link faults unambiguously. The m-tree uses a single WL per undirected link of the network. Thus, bandwidth cost Λ is equal to $|E|$. The m-tree has a single transmitter at the root node that sends a single supervisory signal along a head link. At each node, the received supervisory signal is terminated, forwarded along another link, or sent over two or more outgoing links. If the signal is terminated or forwarded along a single link, one monitor is needed at the node. If a link becomes faulty,

all the downstream monitors will raise alarms. The minimum number of monitors required is $\lceil \frac{|E|+1}{2} \rceil$.

Discussion

An out-of-band fault localization scheme is a very convenient way to handle link fault localization in the optical layer. However, when m-cycle [72] monitoring structure was proposed in 2004, it was intended only for the failure detection and the performance monitoring in all-optical networks. It turns out that m-cycles can be deployed for link fault localization as well. However, an m-cycle based method has a major weakness: a fault in a two-edge cut cannot be localized unambiguously by using only m-cycles. To deal with the problem, m-trail [62] monitoring structure is proposed. As stated earlier, an m-trail has no cycle constraints. In m-tree monitoring structure, routing constraints are further relaxed.

It becomes apparent soon after the introduction of m-cycles that real number of m-cycles $|\mathfrak{M}|$ required for unambiguous single-link fault localization is in $\mathrm{O}(|\log_2(E+1)|)$ [65]. For each m-cycle/m-trail/m-tree, the out-of-band methods require one WL in each unidirectional link traversed by the monitoring structure. Hence, bandwidth cost Λ for single-link SRLG fault localization is reasonable, but the cost is too high for multi-link SRLG fault localization.

The failure detection delay T_{fd} of the out-of-band methods is deterministic and short, but the methods incur long and variable alarm dissemination delay T_{ad}. As most of the methods use ACT look up, the alarm correlation delay T_{ac} is in $\mathrm{O}(1)$.

2.2.4 Hybrid Methods

In 2011, Stanic and Subramaniam [46] proposed a fault localization method using both user lightpaths and dedicated S-LPs. At first an alarm matrix is constructed with the user lightpaths. Each row of the matrix represents an element of the network and each column of the matrix represents a monitor. Then k-shortest paths/cycles between all node-pairs are enumerated excluding all the established user lightpaths. The authors formulate an ILP and propose a heuristic to choose a minimum set of S-LPs from the set of enumerated shortest paths/cycles for UFL.

Both the ILP and the heuristic utilize the alarm matrix and the set of enumerated shortest paths/cycles as input. The ILP chooses a minimum set of S-LPs from the set of enumerated shortest paths/cycles such that the final alarm matrix constructed with

the user lightpaths and the chosen S-LPs will have no zero row or identical rows. In the heuristic, a minimum cost S-LP from the set of enumerated shortest paths/cycles is selected one at a time until UFL is achieved. The cost of choosing an S-LP is the number of zero rows and the number of identical rows of the enhanced alarm matrix if the S-LP is added to the existing alarm matrix.

2.2.5 Non-Alarm Disseminating Methods

Non-alarm disseminating fault localization schemes also utilize dedicated S-LPs only for link failure detection. In fact, a non-alarm disseminating fault localization scheme is a special kind of out-of-band fault localization scheme. In a typical out-of-band fault localization scheme, whenever a monitor detects any link failure, the monitor broadcasts an alarm in the control plane with the highest priority in order to enable the controller for localizing link faults. However, any alarm dissemination process makes the design of the control plane complicated and is a source of variable fault localization latency. Obviously, a non-alarm disseminating fault localization scheme avoids both the problems.

In August 2009, Ahuja, Ramasubramanian, and Krunz [2] proposed a single-link fault localization method in three-edge-connected all-optical networks using monitoring cycles (MC), which are simple m-cycles. The method can localize single-link faults without alarm dissemination from a single monitoring location, which is also known as monitoring node (MN). The upper bound of the number of m-cycles is given as $\frac{|E|(|E|-1)}{2} + 1$, i.e. $O(|E|^2)$. An ILP that takes enumerated cycles as input is formulated with two constraints. The first constraint is that for each link pair, at least one m-cycle traverses one of them but not both. The second constraint is that each link of the network must be traversed by at least one m-cycle.

A heuristic algorithm called the Monitoring Cycle Problem with One Monitoring Location (MC-1) is also proposed. The first phase of the MC-1 method is to get a cycle cover. For each uncovered link, an m-cycle consisting of the link and two link-disjoint shortest paths from the MN to the end nodes of the link is added to the solution. In second phase, each group of links traversed by the same set of m-cycles is considered iteratively until each link of the network is traversed by a unique set of m-cycles. A link from the group is selected randomly. The weights of the remaining links of the group are assigned very large values temporarily. Next, an m-cycle consisting of the selected link and two link-disjoint shortest paths from the MN to the end nodes of the selected link is added to the solution. The worst-case complexity of the MC-1 method is $O(|E|^2(|V| \log_2 |V| + |E|))$.

In 2010, Wu et al. [61] proposed a single-link fault localization method in all-optical

networks such that multiple given MNs are able to localize faults without any alarm dissemination altogether. Each MN detects status of each m-trail passing through the MN by signal tapping. To enable status checking by multiple MN, m-trails in the method are closed structures. An ILP is formulated with constraints to derive a unique code for each link of the network in each MN. The number of constraints is in $O(m|E|^2)$ where m is the number of the given MNs. Each MN localizes single-link faults using its local alarm code table (LACT).

In 2011, He et al. [22] proposed a greedy algorithm to localize faults from a given set of MNs. Each MN detects LOL by signal tapping from the m-trails traversing the MN and can localize single-link faults independently without any alarm dissemination. The method selects m-trails based on *Min Wavelength Max Information* principle. Information gain of an m-trail is defined as $\eta = \sum_i \delta_i/\omega$, where δ_i is the number of unique codes added to the LACT of the ith MN by the m-trail and ω is the length of the m-trail in the number of hops. The method considers each MN once. The method selects an MN randomly and iterates as described below until the LACT of the current MN has unique codes for all links of the network. Once the current LACT achieves unique link codes, it is ensured that no link code of the current LACT is 0 and there are no redundant m-trails in the current LACT.

A unique link code for each link adjacent to the current MN is derived indirectly by randomly generating trail codes, which identify link traversal by m-trails. Then, link codes for the remaining links are assigned randomly and reshuffled among them. Next, all connected components (CCs) of each linkset are considered in sequence. Each CC that does not pass though the current MN is discarded. If a CC is a valid m-trail, it is saved as a trail candidate (TC). Otherwise, valid m-trails that pass through the current MN are searched from arbitrarily selected odd-degree node pairs of the CC. If any valid m-trail is found, it is saved as a TC. Finally, each TC is refined by splitting and looping to ensure that if the TC visits multiple MN, it must be a closed structure. Otherwise, it must be terminated at the current MN.

Now, if a refined TC achieves a preset information gain, it is processed further to enhance its information gain by flipping each bit of its trail code. Then, the TC is added as a new m-trail to the LACT of each MN that is traversed by the TC.

The failure detection delay T_{fd} of the non-alarm disseminating fault localization methods is deterministic and short. The methods incur no alarm dissemination delay. i.e., T_{ad} is 0. As the methods in [22][61] use ACT look up, the alarm correlation time T_{ac} of the methods for alarm correlation is in $O(1)$.

2.3 Multi-Link SRLG Fault Localization

The multi-link SRLG fault localization schemes can also be classified mainly based on the kind of lightpaths utilized for link failure detection. The schemes are classified broadly as in-band, probing, out-of-band, and non-alarm dissemination methods. The non-alarm disseminating SRLG fault localization schemes are identified as a separate group again.

2.3.1 In-band Methods

In 1993, Deng, Lazar, and Wang [16] proposed an in-band Bayesian network based probabilistic method to localize multiple node faults in an early transparent all-optical network, called the *Linear Lightwave Network* (LLN). The method is based on power measurement of user lightpaths, inference model trimming and belief function updates to localize multiple faults via a multi-stage process. The worst case complexity is $O(Z|V|^Z)$ where $|V|$ is the number of nodes and Z is the numbers of incorrect output power values at egress of the network. The algorithm is modified to achieve complexity in $O(Z|V|^d)$ and $O(Z|V|^{T+1})$ for node faults involving up to given d nodes and unknown T nodes, respectively. As it is needed to measure input and output powers of the node with the largest belief whenever the belief value is less than 1, the method needs $2|E|$ monitors and incurs long and variable fault localization latency.

In 2005, Mas, Tomkos, and Tonguz [37] introduced one of the earliest fault localization methods in all-optical networks to localize dual optical component faults. The method has to deal with the problem of optimal alarm set placement, which is shown to be NP-complete [42]. Alarm domains of single and dual components traversed by the user lightpaths are determined. A binary tree is created using all alarms in the system where leafs are either an alarm domain or empty. This part is pre-calculated every time there is a change in user lightpaths; tree construction is avoided in [38]. After a failure event, the binary tree is traversed by using the set of ringing alarms to reach a leaf, which may points to multiple components. Hence, the method may not localize faulty component(s) unambiguously. Moreover, it needs $O(|A|)$ steps for alarm correlation, where $|A|$ is the number of alarms, but m-trail based schemes need $O(1)$ steps for the purpose.

In 2009, Khair, Zheng, and Mouftah [29] proposed a method, called the Parallel Limited Perimeter Vector Matching (P-LVM), for multi-link fault localization in all-optical networks. Once a link failure event occurs, which interrupts some lightpaths, the sink nodes of the disrupted lightpaths will flood the network with messages that include link set of the lightpaths. After receiving the messages, the nodes build lightpath queues, called

Executive Route Queues (ERQs). Each ERQ represents a single-link fault event. The sink of the shortest lightpath of each ERQ becomes the Executive Sink (ES) of the ERQ. Each ES conducts remaining fault localization activities independently using the LVM [44], a single-link fault localization method. The P-LVM has many advantages of a typical distributed method, but has to deal with loss and delay of messages; consequently, the P-LVM incurs long and variable fault localization latency. The P-LVM is applied to multi-domain scenario in [68].

2.3.2 Probing Methods

In 2005, Wen, Chan, and Zheng [57] proposed an *adaptive* run-length probing scheme in all-optical networks based on probabilistic link failure model. All network links are probed sequentially in a probing session to localize all faulty links. Maximum probing length K is derived based on approximately equal probabilities of a single-link fault and no fault in a fiber segment of a Eulerian trail. An optical signal is sent through maximum K links each time. If there is no fault in a fiber segment, the probe arrives at the remote end node. Otherwise, the leftmost faulty link of the segment is localized by using 2^m-splitting and probing techniques repeatedly. The method needs $d \log_2 K + \frac{|E|-d+1}{K} + (d-2)$ probes to identify d faulty links and can localize up to $|E|$ faulty links. The method needs $|E|$ monitors and typically incurs very long fault localization latency due to sequentially probing.

To localize node and link failures, the authors in [58] utilize a network transformation that maps both undirected links and nodes of the network into directed arcs of a directed graph. Then, the directed graph is probed to localize link and node faults by using the run-length scheme. The authors in [59] extend the run-length scheme for non-Eulerian networks by converting non-Eulerian networks into a Eulerian network using the Disjoint-Trail Decomposition and the Path-Augmentation approaches.

In 2007, Harvey et al. [21] outlined two *non-adaptive* probing schemes based on Combinatorial Group Testing (CGT). The methods find fault-free link sets in a network G and identify probes to localize faulty links using each link set as a hub. The first method is for the network topologies that can accommodate at least $(d+1)$ edge-disjoint spanning trees. The upper bound of the minimum number of probes $L^*(G,d)$ to localize up to d link failures is $\mathrm{O}\big(d * T^*(|E|,d)\big)$, where $T^*(|E|,d)$ is the minimum number of non-adaptive group tests needed to identify up to d failed elements among $|E|$ elements. The upper bound of $T^*(|E|,d)$ is $\mathrm{O}(d^2 \log_2 |E|)$. Thus $L^*(G,d)$ is in $\mathrm{O}(d^3 \log_2 |E|)$. The second method is for the network topologies that have at least d edge-disjoint spanning trees $\{T_1, \ldots, T_d\}$. Now, $L^*(G,d)$ is in $\mathrm{O}\big(d * T^*(|E|,d) + \sum_{i=1}^{d} L^*(T_i, d=1)\big)$. Obviously, the second method

needs more probes. The schemes are for highly connected networks and cannot be applied in networks with limited connectivity to achieve the bounds on $L^*(G, d)$.

2.3.3 Out-of-band Methods

In 2008, Ahuja, Ramasubramanian, and Krunz [1] outlined a multi-link SRLG failure localization method in arbitrarily connected networks to localize faults involving up to d links. The authors define the feasibility of the S-LP routing problem in terms of the network topologies and deal with the problem of MN selection. The method requires that each component found after removal of any $(d + 1)$ links have at least one MN. Once all MNs are identified, all MNs are merged to a central MN J. The transformed graph becomes $(d + 2)$-edge connected. The method finds a set of m-trails in the transformed network running the modified heuristic MC-1 from MN J. The modified MC-1 is described in section 2.3.4.

In 2010, Wu et al. [60] provided an approach for optimally allocating m-trails to achieve unambiguous SRLG failure localization using ILP where decimal alarm codes of each pair of SRLGs are kept dissimilar. Thus the number of constraints is in $O(|\Psi|^2)$ which is reduced to $O(|E||\Psi|)$ in Section 5.2. Similar ideas of [60] are further explored on bi-directional monitoring trails (bm-trails) in [51]. A bm-trail is a non-simple cycle/path that can traverse any node multiple times but a link at most twice: one time in each direction. Initially, each SRLG with up to d arbitrary links is assigned a unique code from generated non-adaptive $\bar{d}$-separable CGT codes. Then code pairs are swapped keeping code uniqueness of the SRLGs using the Greedy Code Swapping (GCS) method to form bm-trails in each bit position of the SRLG codes in such a way that the overall cost of failure localization is minimized. The worst-case complexity of the method is $O\big(d^4|V||E|^2 \log^2 |E|(1+h_\Delta)\big)$, where h_Δ is the difference between maximum and minimum Hamming weight of the generated CGT codes. The method is applied in a *dense*-SRLG scenario.

In 2011, Babarczi, Tapolcai, and Ho [10] introduced an algorithm, called the Adjacent-Link Failure Localization (AFL), that deals with *sparse*-SRLGs. The set of SRLGs includes all single-link and some adjacent-link SRLGs. Since the code of each dual-link SRLG is bit-wise OR of its constituent single-link codes, there are code dependencies. The code dependencies are removed by graph partitioning. Then the Random Code Assignment (RCA) and the Random Code Swapping (RCS) methods [54] are applied to each partition to find codes for single-link faults. A unique code for each SRLG is derived by direct summing of the single-link codes of all the partitions. Complexity of AFL is $O(\Delta^2|V||E|^3 \log_2 |E|)$, where Δ is maximal nodal degree of the graph. The worst-case complexity is improved to

$O\big((|E|+1)^d(|V|\log_2|V|+|E|)\big)$ in Section 5.2 where d is 2 for all single-link and adjacent-link SRLGs.

2.3.4 Non-Alarm Disseminating Methods

In 2008, Ahuja, Ramasubramanian, and Krunz [1] proposed a multi-link SRLG failure localization method using monitoring cycles (MCs), which are simple m-cycles. The method can localize faults involving up to d links from a single monitoring location, which is also known as MN. The authors prove that to localize up to d simultaneous link faults in a network from a single MN using m-cycles, the network must be $(d+2)$-edge connected.

An ILP is formulated based on enumerated cycles. The first constraint of the ILP is that for each SRLG pair, at least one m-cycle traverses one SRLG but not both. The second constraint of the ILP is that each SRLG under consideration must be traversed by at least one m-cycle.

Another version of the heuristic algorithm MC-1 is also proposed in [1] for multi-link SRLG fault localization from a single MN. Initially, the tag value for each SRLG is set to 0. Then, a pair of SRLGs having the same tag value is considered iteratively until the tag value of each SRLG becomes unique. In each iteration, the MC-1 randomly selects a link that is in only one SRLG of the pair, and all links of the other SRLG are removed temporarily. Next, an m-cycle consisting of the selected link and two link-disjoint shortest paths from the MN to the end nodes of the selected link is added to the solution. The weight of each link traversed by the m-cycle is increased by a large value, and tag value of each SRLG traversed by the m-cycle is updated. At the end of the iterative phase, if an SRLG is not traversed by any m-cycle, a new m-cycle consisting of one link of the SRLG and two link-disjoint shortest paths from the MN to the end nodes of the link is added to the solution. The complexity of the method is $O(|\Psi|^2|V|\log_2|V|)$ where $|\Psi|$ is the number of SRLGs under consideration. The required number of m-cycles $|\mathfrak{M}|$ is upper bounded by $O(|\Psi|^2)$, which is the same as $O\big((|E|+1)^{2d}\big)$; the upper bound is reduced to $O(|\Psi|)$ in Section 5.2 and to $(d+1)|E|$ in Section 5.3.

In 2011, He et al. [23] proposed a heuristic method, called the MNUFL, to localize multi-link SRLG faults from a given set of MNs. Each MN detects LOL using signal tapping from the m-trails traversing the MN and can localize link faults involving up to d links independently without any alarm dissemination. The MNUFL selects m-trails based on *Enhanced Min Wavelength Max Information* principle. Let m be the number of distinct SRLG codes in the set of SRLG codes C and the number of SRLGs having C_i as their SRLG code be $|C_i|$. Thus $\sum_{i=1}^{m}|C_i| = |\Psi|$, where $|\Psi|$ is the total number of SRLGs under

consideration. The entropy of C is defined as $H(C) = \sum_{i=1}^{m} -p_i \log p_i$, where $p_i = \frac{|C_i|}{|\Psi|}$. Information gain is defined as the difference between entropies of the set of SRLG codes before and after addition of a new m-trail to a solution.

The MNUFL derives MN-valid m-trails and adds them to the solution iteratively until each SRLG code becomes unique. An m-trail becomes MN-valid if it traverses all the designated MNs. As described below, the MNUFL maximizes information gain during the search for each new MN-valid m-trail.

The MNUFL derives a new trail by adding links as long as the entropy can be increased. If the new trail is not an MN-valid m-trail, an info-gaining event group with respect to the new trail is chosen. (An info-gaining event group with respect to an m-trail is a set of SRLGs having the same code, but the code of a subset of the SRLGs changes to a different value when the m-trail is added to the solution.) Next, an SRLG that is link-disjoint from the new trail is selected from the chosen info-gaining event group. Then for each pair of connected components (CCs) in the new trail, a set of links that is link-disjoint from the selected SRLG but connects the pair of CCs is stored as the pair's hyperedge. Finally, the MNUFL adds new links to the new trail iteratively until the new trail becomes an MN-valid m-trail; in each iteration, a link that provides maximum info-gain among the links of an arbitrarily selected hyperedge is added to the new trail.

2.4 Fault Localization via M-Burst Framework

As we have stated in [7], the proposed m-burst framework aims at solving a number of legacy problems inherent in the context of fault localization in all-optical mesh networks. First, the m-burst framework overcomes the limitations of link-based monitoring, which serves as an alternative to the state-of-the-art industry solution. Second, the conventional m-trail based approaches may consume a huge amount of monitoring resources that make them impractical for deployment. For example, the authors in [61] show that on average 2 WLs along each link are needed in order to localize 6 SRLGs in a 10-node topology. Moreover, the authors in [51] show that more than 10 WLs along each link are needed to localize any faulty SRLG with up to 3 links in randomly generated large topologies. The m-burst framework makes those conventional m-trail based approaches more feasible due to significantly reduced resource consumption achieved by multiplexing in the time domain the traversing multiple m-trails, if any, in each unidirectional link. Third, since the switch fabric of each network node is configured periodically as per the switching schedule determined in advance, there will be no real-time control complexity and signaling

overhead. The resultant failure monitoring system is truly all-optical, signaling-free, and deterministic.

In m-burst framework, the number of monitors $|M|$ and the number of m-trails $|\mathfrak{M}|$ are no longer the same. As each m-burst is launched from the single MN along a single WL in an outgoing link and the burst returns to the MN via a single WL in an incoming link, transmitters, receivers, and monitors of m-trails can be shared. In fact, the number of required monitors $|M|$ in m-burst framework is less than or equal to the nodal degree of the MN. Hence, the number of m-trails $|\mathfrak{M}|$ has no significance from the monitoring cost perspective, and the monitor cost $\gamma * |M|$ is determined by the nodal degree of the MN. Moreover, the bandwidth cost Λ is in $O(|E|)$ because at most one WL in each unidirectional link is used.

When an m-burst fails to return, the MN has to wait for a monitoring period before it correlates alarms. Thus, unlike any other fault localization method, the failure detection delay T_{fd} in the m-burst framework is significantly large; in fact, T_{fd} is equivalent to the fault localization latency T as the alarm dissemination delay T_{ad} is 0 and the alarm correlation delay T_{ac} is in $O(1)$.

As the number of monitors $|M|$ is determined by the nodal degree of the MN and the bandwidth cost Λ is in $O(|E|)$, it turns out that reducing monitoring period or equivalently T_{fd} is the primary objective in the m-burst framework.

2.5 Conclusions

In-band fault localization schemes incur minimum monitor cost if monitors are not used at the intermediate nodes of lightpaths to detect LOL. Almost all in-band schemes incur no bandwidth cost. On the other hand, the performance of an in-band scheme in a network heavily depends on the current traffic matrix of the network, a source of great uncertainty in fault localization. Moreover, the in-band schemes are subject to long and variable alarm dissemination delays. Adaptive probing based fault localization schemes also incur long and variable alarm dissemination delays, but the schemes are able to localize all faulty links. The performances of out-of-band fault localization schemes are stable and predictable, but the schemes incur high monitoring cost. Although not as severe as in in-band or adaptive probing based schemes, alarm dissemination delays remain an issue with out-of-band schemes except in non-alarm dissemination schemes.

At this point, many efficient methods, in terms of monitoring resource utilization, are proposed for single-link SRLG fault localization in all-optical networks. However, the

scenario is quite different with multi-link SRLG fault localization in all-optical networks. The existing multi-link SRLG fault localization methods need considerable amount of monitoring resources; therefore, multi-link SRLG fault localization in all-optical networks is still an open problem for investigation.

Chapter 3

The M-Burst Framework

3.1 Introduction

In this thesis, we develop single-link and multi-link SRLG fault localization schemes in all-optical mesh networks. The schemes are to be deployed within a novel fault localization framework called the monitoring burst (m-burst). The framework has one given MN and uses one dedicated WL in each unidirectional link of the network if the link is traversed by an m-trail. The MN is the source and sink of the S-LP of each m-trail and sends a short duration optical burst through the S-LP periodically. The MN detects link failure in an m-trail if the burst along the m-trail fails to return to the MN in due time.

As mentioned earlier, the m-burst framework has three components: 1) an m-trail allocation scheme for single-link or multi-link SRLG fault localization, 2) an m-burst launching time scheduler, and 3) a node switch fabric configuration scheduler. The fault localization scheme provides a set of m-trails as a solution such that each SRLG under consideration will be traversed by a unique subset of the m-trails. The m-burst launching time scheduler will determine the burst starting time from the MN along each m-trail in each monitoring period such that bursts remain non-overlapping in any link of the network. The node switch fabric configuration scheduler will specify switching times for each node in order to multiplex multiple m-bursts, if any, in the time domain, through each unidirectional link in each monitoring period. The switching times for each node are derived from the m-trail routes and the burst launching times. Burst collision is avoided altogether with the help of two schedulers.

The MN sends the schedule for node switch fabric configuration to every node of the network once in the beginning of fault monitoring using a signaling protocol such as the

Resource Reservation Protocol with Traffic Engineering (RSVP-TE). Each node periodically configures its switch fabric at the reference time instant, according to the assigned schedule, without relying on any additional signaling from the MN prior to every switching activity, since the m-trail deployment depends only on the network topology, which is considered rather static. The schedule is usually updated after pre-defined long intervals.

In this chapter, we describe the construction and working principles of the m-burst framework. We also formulate three relevant problems in the m-burst framework: m-trail allocation, m-burst launching time scheduling, and node switch fabric configuration scheduling problems.

The rest of the chapter is organized as follows. Section 3.2 provides a detailed description of the m-burst framework. In Section 3.3, three problems of the m-burst framework are formulated individually. Section 3.4 concludes the chapter.

3.2 Description of the Framework

Figure 3.1 shows a network with six nodes and nine links to explain how the proposed m-burst framework works. Node 0 is the MN. We deal with single-link fault localization. Each undirected link of the network is a single-link SRLG; thus, there are nine SRLGs.

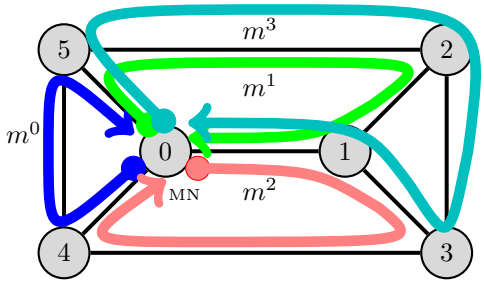


Figure 3.1: Single-link SRLG fault localization.

A set of four m-cycles is a solution for the problem of identifying the single-link SRLG faults. Each m-cycle starts from and returns to the MN. Each SRLG is traversed by a unique set of m-cycles. Each SRLG code is derived from the corresponding set of m-cycles traversing the SRLG. Each binary bit of the SRLG code represents an m-cycle. If the jth m-cycle traverses the SRLG, the corresponding bit of the SRLG code is assigned 1; otherwise, it is 0 [54]. SRLG codes are the decimal values of the binary codes. Formation

of the SRLG codes is shown in Table 3.1. The MN maintains an alarm code table (ACT) that is the mapping between SRLG codes and the SRLGs.

Table 3.1: Alarm Code Table (ACT)

SRLG	m^3	m^2	m^1	m^0	SRLG Codes
(0, 1)	1	1	1	0	14
(0, 4)	0	1	0	1	5
(0, 5)	1	0	1	1	11
(1, 2)	0	0	1	0	2
(1, 3)	1	1	0	0	12
(2, 3)	1	0	0	0	8
(2, 5)	1	0	1	0	10
(3, 4)	0	1	0	0	4
(4, 5)	0	0	0	1	1

To derive burst launching times from the MN, let us assume that propagation delay δ is the same along each link and burst length L is several times larger than δ. We try to find the launching time of each burst as early as possible, avoiding collision with other bursts. The m-bursts along m-cycles m^0 and m^2 can be launched in the beginning of a monitoring period because they do not use any unidirectional link of the network that is being used by another m-cycle. Thus, the burst launching times s^0 and s^1 are 0.

As both m-cycles m^1 and m^3 use unidirectional links (0, 5), (5, 2) and (1, 0), only one burst along the m-cycles can be launched in the beginning of a monitoring period. However, if we launch a burst along m-cycle m^3 in the beginning of a monitoring period, we will get the least monitoring period. Thus, the burst launching time s^3 is 0. In that case, the burst along m-cycle m^1 must be launched at least after L ms from the beginning of each monitoring period to avoid collision in links (0, 5) and (5, 2). To avoid collision in link (1, 0), the burst along m-cycle m^1 should not reach node 1 before $(4\delta + L)$ ms from the beginning of a monitoring period. Thus, the earliest burst launching time along m-cycle m^1, i.e., s^1 should be $(4\delta + L - 3\delta) = (\delta + L)$ ms in order to avoid all burst collisions. As a result, the m-burst along m-cycle m^1 will reach nodes 5, 2 and 1 after the m-burst along m-cycle m^3 completely passes through each of the three nodes.

The time required by the jth m-burst to return to the MN avoiding collisions is $T^j = s^j + tp^j + L$, where s^j, tp^j and L are the burst launching time along the jth m-cycle from the MN in ms, the total propagation delay of the burst in ms along the jth m-cycle, and the burst length in ms, respectively. Thus, $T^0 = 0 + 3\delta + L = 3\delta + L$, $T^1 = (\delta + L) + 4\delta + L = 5\delta + 2L$, $T^2 = 0 + 4\delta + L = 4\delta + L$ and $T^3 = 0 + 5\delta + L = 5\delta + L$. Within $(5\delta + 2L)$ ms, every burst will return to the MN completely. Thus, the monitoring period is $(5\delta + 2L)$

ms.

As each m-cycle is probed once every $(5\delta + 2L)$ ms, burst launching times s^0, s^2, and s^3 along m-cycles m^0, m^2, and m^3, respectively, will be $t^0 = k * (5\delta + 2L)$, where k is a non-negative integer. Similarly, burst launching time s^1 along m-cycle m^1 will be $t^1 = (\delta + L) + k * (5\delta + 2L)$. In the example, t^1 always lags t^0 by $(\delta + L)$ ms. Burst launching times are shown in Figure 3.2.

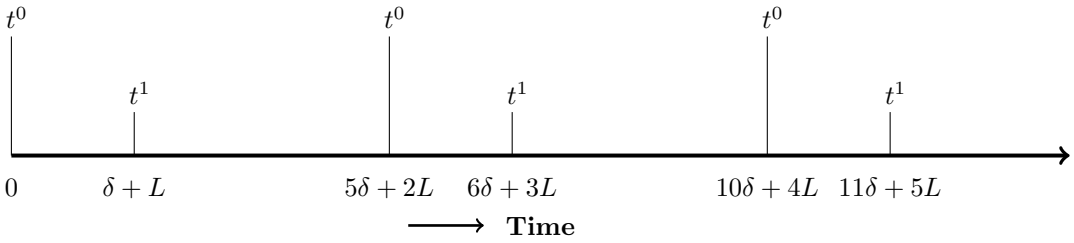
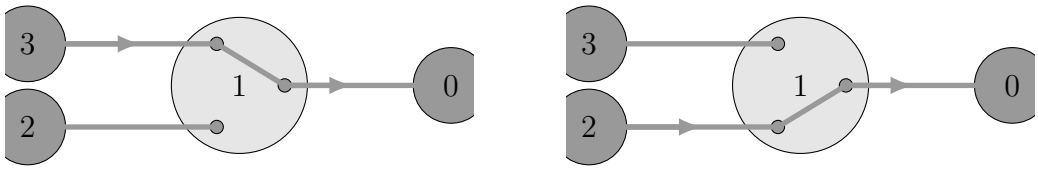


Figure 3.2: Burst launching schedule from the MN.

Now the only remaining issue is the real time configuration of the switch fabric of each network node in each monitoring period. Note that at most one WL is used along any unidirectional link in the m-burst framework. If two or more bursts are arriving through one incoming link to a node and leaving through another outgoing link from the node, the connection between the incoming and outgoing links need not be changed at all. On the other hand, if two or more bursts are arriving through the same incoming link to a node and leaving through different outgoing links from the node, or two or more bursts are arriving through different incoming links to a node and leaving through the same outgoing link from the node, the corresponding connections of the switch fabric of the node has to be changed in each monitoring period as per the burst arrival schedule. In our example, the connections of the switch fabrics of only nodes 1 and 2 have to be changed every monitoring period to allow the bursts along m-cycles m^1 and m^3 to pass the nodes without collision.

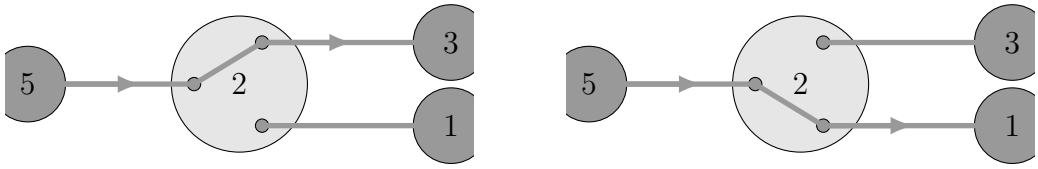
Figure 3.3 shows the node switch fabric configuration of node 1. The node has to enable a connection for L ms from node 3 to node 0 at every time instant $4\delta + k(5\delta + 2L)$ and from node 2 to node 0 at every time instant $(4\delta + L) + k(5\delta + 2L)$ for m-bursts along m-cycles m^3 and m^1, respectively. Similarly, Figure 3.4 shows the node switch fabric configuration of node 2. The node has to enable a connection for L ms from node 5 to node 3 at every time instant $2\delta + k(5\delta + 2L)$ and from node 5 to node 1 at every time instant $(3\delta + L) + k(5\delta + 2L)$ for m-bursts along m-cycles m^3 and m^1, respectively.

There are three switching strategies. First, each node can start changing its configuration at the instant each m-burst reaches the node to allow the m-burst to traverse



(a) at $t = 4\delta + k(5\delta + 2L)$ (b) at $t = (4\delta + L) + k(5\delta + 2L)$

Figure 3.3: Configuration of the switch fabric of node 1.



(a) at $t = 2\delta + k(5\delta + 2L)$ (b) at $t = (3\delta + L) + k(5\delta + 2L)$

Figure 3.4: Configuration of the switch fabric of node 2.

the node; for instance, node 1 in Figure 3.3 can start switching at $4\delta + k(5\delta + 2L)$ and $(4\delta + L) + k(5\delta + 2L)$. In a fault-free network state, the leading edge up to t_s ms of each burst will be affected, where t_s is the switching time of each node, and the last $(L - t_s)$ ms of the burst will remain intact.

Second, each node can start changing its configuration t_s ms before the instant an m-burst completely passes through the node, to allow the next m-burst to traverse the node; e.g., node 1 can start switching at $(4\delta + L) + k(5\delta + 2L) - t_s$ and $(4\delta + 2L) + k(5\delta + 2L) - t_s$. In a fault-free network state, $(L - t_s)$ ms of the leading edge of each burst will remain intact, and the last t_s ms of each burst will be distorted.

Third, each node can start changing its configuration at the instant an m-burst completely passes through the node, to allow the next m-burst to traverse the node; e.g., node 1 can start switching at $(4\delta + L) + k(5\delta + 2L)$ and $(4\delta + 2L) + k(5\delta + 2L)$. In a fault-free network state, the leading edge of the next burst will be affected if the burst reaches the node within t_s ms of the start of switching. In a fault-free network state, at most the first t_s ms of the next burst will be distorted, and at least the last $(L - t_s)$ ms of the burst will remain intact.

Let us assume that each node will start switching at the instant an m-burst completely passes through the node. Consequently, intact burst sizes will be increased. Note that each burst carries minimal data such as the identity of the m-burst. It is to be ensured only that the last $(L - t_s)$ ms of each burst is sufficient to identify the m-burst. Let t_s be 10 ms [14][41][67]. If burst length L is 20 ms, at least the last 10 ms of each m-burst will reach the MN intact. During normal operation of the network as shown in Figure 3.1, all bursts except the m-burst along m-cycle m^1 will reach the MN intact. Only the last $(20 - 10) = 10$ ms of the m-burst along m-cycle m^1 will return to the MN intact. On the other hand, if link (1, 2) becomes faulty, the m-burst along m-cycle m^1 will be lost completely.

Now if bursts along m-cycles m^0, m^1 and m^3 fail to return to the MN within a monitoring period due to a failure event, the generated decimal alarm code will be $2^0+2^1+2^3 = 11$. The MN can identify link (0, 5) as the faulty single-link SRLG from the ACT given in Table 3.1, using 11 as an index.

3.3 Problem Formulation

3.3.1 M-Trail Allocation

Like any out-of-band monitoring structure allocation problem, an m-trail allocation problem in the m-burst framework is to find a sequence of connected unidirectional links in the network $G = (V, E)$. As stated or implied in every monitoring structure allocation problem in the literature, an m-trail in the m-burst framework can be described as a sequence of unidirectional links. In other words, "$(v_0, v_1), (v_1, v_2), \ldots, (v_{n-2}, v_{n-1}), (v_{n-1}, v_n)$" is an m-trail where $v_j \in V$: v_j is the jth node of the m-trail for $j = 0, \ldots, n$ and "$(v_i, v_{i+1}) \in E$ for $i = 0, \ldots, n - 1$" provided that "$(v_i, v_{i+1}) \neq (v_j, v_{j+1})$ if $i \neq j$" [54].

Every monitoring structure has to satisfy certain constraints based on specific design requirements. The m-burst framework enforces the following constraints. First, each m-trail has to traverse a designated node, which is the MN. Second, as m-bursts are launched from the MN and returned to the MN, each m-trail has to be a closed loop; consequently, $v_0 = v_n$. Third, if an m-trail is a trail and traverses a unidirectional link (u, v), it has to traverse the unidirectional link (v, u); but if the m-trail is a cycle and traverses a unidirectional link (u, v), it cannot traverse the unidirectional link (v, u). Moreover, an m-trail is a non-simple path that can visit a node multiple times. Note that every m-trail in the m-burst framework is in fact a virtual entity because no WL is exclusively reserved for it nor is any persistent optical flow going through it.

In the m-burst framework, an SRLG failure is defined as the failure of all the links contained in the SRLG, but an SRLG is considered traversed by an m-trail if one or more links contained in the SRLG is traversed by the m-trail.

The solution of the m-trail allocation problem in the m-burst framework consists of a set of m-trails $\mathfrak{M} = \{m^0, m^1, \ldots, m^{(J-1)}\}$ where $J = |\mathfrak{M}|$. Let $\forall(u,v),(v,u) \in E$, the set of m-trails traversing through the undirected link (u,v) be denoted as $\varphi_{(u,v)}$. If m^0, m^1 and m^2 traverse through an undirected link (w,x), and m^2, m^3 and m^4 traverse through an undirected link (y,z), we have $\varphi_{(w,x)} = \{m^0, m^1, m^2\}$ and $\varphi_{(y,z)} = \{m^2, m^3, m^4\}$, respectively. Again, let the set of m-trails traversing SRLG ψ_i be denoted as φ_{ψ_i}. The set of m-trails traversing a single-link SRLG is obviously the set of m-trails traversing through the only link of the SRLG. Now, if a SRLG $\psi_j = \{(w,x)\}$, we have $\varphi_{\psi_j} = \varphi_{(w,x)} = \{m^0, m^1, m^2\}$. On the other hand, the set of m-trails traversing a multi-link SRLG is the union of the sets of m-trails traversing through the links of the SRLG. Here, $\forall\psi_i \in \Psi$: ψ_i is a multi-link SRLG, $\varphi_{\psi_i} = \bigcup_{\forall(u,v)\in\psi_i} \varphi_{(u,v)}$. Now, if a multi-link SRLG $\psi_k = \{(w,x),(y,z)\}$, we have $\varphi_{\psi_k} = \varphi_{(w,x)} \cup \varphi_{(y,z)} = \{m^0, m^1, m^2, m^3, m^4\}$.

To localize single-link or multi-link SRLG failures unambiguously, the set of m-trails traversing each SRLG should be unique. To fulfill the condition, the m-trail allocation problem will be considered as a topology coding process, where each SRLG is assigned a unique SRLG code.

A binary link code is defined as "an ordered sequence of binary bits $[b^{(J-1)} \cdots b^1\ b^0]$ where each m-trail in $\mathfrak{M}$ has a fixed bit position. Specifically, m-trail m^j occupies the jth bit position of link codes. If m-trail m^j traverses link (u,v), b^j will be 1 in the link code of link (u,v); otherwise, b^j will be 0" [54]. An SRLG code is defined similarly. The SRLG code of an SRLG is derived by bitwise logical `OR` operation on the link codes of the links in the SRLG. An alarm code table (ACT) maintains the mapping between SRLG codes and the set of SRLGs.

By properly designing a set of m-trails with respect to the SRLGs under consideration, it is ensured that the on-off status of the monitors of m-trails during a failure event will define a valid alarm code. The MN maps the failure event to a particular SRLG by matching the alarm code with an SRLG code in the ACT. Therefore, if each single-link or multi-link SRLG is assigned a unique SRLG code equivalently each SRLG is traversed by a unique set of m-trails, any single-link or multi-link SRLG failure can be localize unambiguously by identifying the set of disrupted m-trails.

3.3.2 M-Burst Launching Time Scheduling

The m-burst scheduling problem is to find m-burst launching time s^j along each m-trail $m^j \in \mathfrak{M}$ during a monitoring period so as to minimize the fault localization latency T while avoiding any burst collision altogether in the network. Let p^j_{uv} be the burst propagation delay from the MN to node u incurred by the burst along m-trail m^j until it is about to traverse unidirectional link (u, v). Thus, the burst along m-trail m^j reaches node u at $(s^j + p^j_{uv})$.

Now, $\forall(u, v) \in E$ and $\forall m^k, m^l \in \mathfrak{M}$: m^k and m^l traverse directional link (u, v), the burst scheduler at the MN has to ensure that $|(s^k + p^k_{uv}) - (s^l + p^l_{uv})| \geq L$, where L is the burst length, to avoid burst collision in the directional link (u, v) including its end nodes u and v. In other words, the absolute difference between the arrival times of each burst pair at any node while traversing the same outgoing link from the node must be at least equal to the burst length L.

As the propagation delay along an m-trail is determined by the route of an m-trail from the MN to node u, the burst arrival time p^j_{uv} to node u is fixed for m-trail m^j that is traversing link (u, v). Therefore, the burst scheduler can adjust only s^j to ensure separation of the bursts in time domain if m-trail allocation and burst scheduling are done separately.

Moreover, the burst scheduler has to minimize fault localization latency T. Now, $T \geq T^j = s^j + tp^j + L$, $\forall m^j \in \mathfrak{M}$, where tp^j is the end-to-end propagation delay along m-trail m^j. Clearly, with a set of m-trails determined a priori, the burst scheduler can adjust only s^j to minimize T as well.

3.3.3 Node Switch Fabric Configuration Scheduling

Once the set of m-trails as the solution of the fault localization problem is identified and the burst starting time along each m-trail is derived, the MN calculates burst arrival times to each node of the network. Now, $\forall u \in V$, $\forall(u, v) \in E$, and $\forall m^j \in \mathfrak{M}$, the burst arrival time to node u is $s^j + p^j_{uv}$ if m-trail m^j visits the node and next hop node of the m-trail is v. Then, the MN informs each node the monitoring period, the route of each m-trail traversing the node, and the burst arrival time along the m-trail using the RSVP-TE.

As stated earlier, if each burst that arrives to node u through the incoming link (t, u) is leaving through the same outgoing link (u, v), the node has to provide the connection from node t to node v but need not change the connection during fault monitoring. On the other hand, if more than one bursts arrive to node u through the incoming link (t, u) but leave through different outgoing links or more than one bursts arrive to the node through

different incoming links but leave through the same outgoing link (u, v), the node has to configure its switch fabric during every monitoring period.

Let us assume that switch fabric configuration starts after each burst passes through a node completely. In other words, when the burst along m-trail m^j passes through the node completely, i.e., at $s^j + p^j_{uv} + L$, the node starts changing its switch fabric to allow traversal of the next burst. Thus, at least $(L - t_s)$ ms of each burst will remain intact, where t_s is the switching time of the node.

3.4 Conclusions

In m-burst framework, S-LPs are multiplex in the time domain to reduce WL usage for monitoring. The S-LP for an m-trail is utilized only during its burst traversal along the m-trail. Thus, S-LP multiplexing is equivalent to fast lightpath set up and tear-down.

The schedule for node switch fabric configuration depends only on the routes of m-trails in the solution and burst starting times along the m-trails. Thus m-trail allocation and m-burst scheduling problems completely determine the timing to change the configuration. Actual calculation of the timing to change the configuration is routine, simple, and easy. Thus, we will not discuss the node switch fabric configuration scheduling problem at length. For all practical purposes, the problems we are dealing with in this thesis are the m-trail allocation and m-burst launching time scheduling problems.

Chapter 4

M-Burst on Single-Link SRLGs

4.1 Introduction

A network operator is committed up to 99.999% (five 9s) availability of connections as a part of service level agreements with its customers [41]. A high availability of network backbone is ensured by using fault detection, fault localization and traffic protection/restoration techniques. Instantaneous and precise failure localization in all-optical WDM mesh networks is essential in enabling fast and automatic failure-dependent restoration in the event of the failure of any shared risk link group (SRLG), which in turn serves as a key functional block in achieving an efficient and effective fault management system.

In the absence of the electronic data processing at the intermediate nodes, one of the most common methods in the literature for achieving instantaneous failure localization in all-optical WDM networks is via out-of-band monitoring, whereby a set of supervisory lightpaths (S-LPs) are launched and closely monitored at the receivers. A monitor broadcasts an alarm in the control plane with the highest priority as soon as any irregularity such as loss-of-light (LoL) is detected. By collecting all the alarms issued, the network controller or routing entities are expected to come up with a valid alarm code that uniquely indicates an SRLG failure event. Note that the alarm dissemination should take place in the network control plane in real-time right after the occurrence of each failure, which not only takes a significant amount of network resources of the control plane during fault events, but also introduces additional fault localization latency and control complexity.

Some studies suggest monitoring a set of m-trails terminated at a single monitoring node (MN), which can completely remove the alarm dissemination complexity [1][2][61].

In particular, the studies in [61] further suggest that a node obtain the on-off status of a lightpath by tapping the optical signal, by which the fault localization decision can be made locally at the node. However, with the signaling complexity resolved, every link needs to consume a single wavelength channel (WL) to support the traversal of an m-trail, which leads to significant increase of the resource consumption. According to an early assessment [51], a feasible unambiguous failure localization (UFL) solution takes 10 or more wavelength channels on average along each link when all the SRLGs with up to 3 links are considered.

To reduce monitoring resource consumption, time division multiplexing (TDM) of S-LPs at intermediate nodes could be used. But that would require expansive O/E/O conversion of S-LPs at the intermediate nodes. Moreover, it would go against transparent characteristics of the all-optical network.

This thesis aims to solve the abovementioned problem of high monitoring resource requirement by introducing a novel framework of all-optical failure localization, called the *monitoring burst* (m-burst). With the m-burst framework, a single MN is allocated with a set of close-loop S-LPs, also known as m-cycles. The MN launches optical bursts along the m-cycles to detect their on-off status. Due to the close-loop shape of an m-cycle, the launched optical burst will be received by the MN if all the links along the m-cycle are working properly, and the burst will be lost if any link along the m-cycle has failed. Thus, different from the scenario of local unambiguous failure localization (L-UFL) defined in [61] where an MN relies on the on-off status of each traversing S-LP to determine each bit in the alarm code, the MN of the m-burst framework determines each bit of the alarm code based on whether the corresponding burst is received at the expected time instant or not.

Since each m-burst is a short-duration optical flow, we no longer need to reserve statically a whole wavelength channel (WL) in each link traversed by an m-cycle; on the other hand, multiple optical bursts can be multiplexed in the time domain on the same WL, which is expected to reduce significantly the consumed monitoring resources. Such saving of monitoring resources, nonetheless, cannot be achieved without additional complications and overhead. Due to lack of optical buffers, an optical burst has a deterministic arrival and departure time instants at each intermediate node once it is launched from the MN. Thus, it is possible that two m-bursts will *collide* at any unidirectional link traversed by two corresponding m-cycles. Note that the proposed m-burst framework has an essential prerequisite that any burst loss should be due only to link failure instead of any other reason such as burst collision, otherwise the MN would be subject to a false alarm or an illegal alarm code. Provided with k wavelengths (full wavelength convertibility assumed) along each unidirectional link for supporting the failure localization system, burst collision can be avoided only if no more than k bursts have their traversal time periods overlapped with

those of others along any unidirectional link. Therefore, an immediate problem stemming from the proposed m-burst framework is how to completely avoid any burst collision.

In this chapter[1], we set our goal so as to explore the feasibility of the proposed m-burst framework by formulating the burst collision avoidance task as a burst scheduling problem. In order to avoid completely any possible burst collision, our strategy is to manipulate the timing of launching each optical burst at the MN. The chapter focuses on the scenario of single-link SRLGs with a single MN, where each unidirectional link requires one single supervisory WL if the link is traversed by any m-cycle for the failure localization system. Note that three relevant problems of the m-burst framework—the m-cycle allocation, m-burst launching time scheduling, and node switch configuration scheduling problems—are formulated in Section 3.3 of Chapter 3.

The rest of this chapter is organized as follows. Two problems, the burst launching time scheduling for getting non-overlapping bursts in each unidirectional links of the network and the signaling for switching to multiplex bursts in the time domain, are discussed in Section 4.2.

In Section 4.3, an integer linear program (ILP) is formulated to solve jointly the m-cycle allocation problem and to obtain collision-free burst scheduling under the proposed framework. Numerical experiments on small networks are conducted, and the results are given in the section.

In Section 4.4, two ILPs are formulated to solve separately the m-cycle allocation and burst scheduling problems. Numerical results are compared in the section.

Section 4.5 provides heuristic algorithms to solve separately m-cycle allocation and burst scheduling problems in large networks. Numerical results are also compared in the section. The heuristic burst scheduling algorithm will be used in Chapter 5 and Chapter 6 as well.

Section 4.6 concludes the chapter.

4.2 Problem Analysis

4.2.1 M-Burst Launching Time Scheduling

An immediate problem raised under the m-burst framework is the timing for launching the burst for each m-cycle, such that there is not any collision (or *burst contention* in the

[1]The content of this chapter has been published as a workshop paper [9]

context of OBS) in the process of resource reservation. A burst collision simply yields an incorrect monitoring and failure location result. Note that the MN is the only sender and receiver of the bursts for all the m-cycles. Therefore, a well-designed scheduling scheme for the time instant of launching the bursts from the MN is expected to nicely resolve any collision at any link. Without the scheduling effort, collisions may occur among the bursts of any $k+1$ m-cycles that traverse through a common link having k reserved wavelengths.

There are two extreme cases in the scheduling problem. One is that each link contains sufficient wavelength channels for monitoring, such that the MN can launch all the optical bursts at the same time without any collision. In this case, the monitoring delay is simply the delay along the longest m-cycle. The other extreme is that the MN conservatively keeps no more than one burst on-the-fly in the network, where an optical burst is launched after confirming the probing result of the previously launched burst. This obviously can avoid any collision under any circumstance, at the expense of taking the longest possible monitoring delay, which is the sum of the delays of all the m-bursts.

In the m-burst framework, at most a single WL channel is used for monitoring in each unidirectional link. Given the m-cycle solution set $\mathfrak{M}$ of the single-link SRLG failure localization problem, monitoring delay can be minimized by finding the earliest burst launching time from the MN along each m-cycle in $\mathfrak{M}$ and at the same time avoiding burst collisions throughout the network. Each link is considered unidirectional for scheduling m-bursts.

Lemma 4.2.1. *The necessary and sufficient condition for avoiding burst collision throughout a network is that* $\forall (u,v) \in E$ *and* $\forall m^i, m^j \in \mathfrak{M}$: m^i *and* m^j *traverse unidirectional link* (u,v)*, the interval between the burst arrival times to node* u *along m-cycles* m^i *and* m^j *is greater than or equal to the burst length* L*; i.e.,* $|(s^i + p^i_{uv}) - (s^j + p^j_{uv})| \geq L$*, where* s^i *and* s^j *are the burst launching times from the MN along m-cycles* m^i *and* m^j*,* p^i_{uv} *and* p^j_{uv} *are the burst propagation delays from the MN to node* u *of the m-bursts along m-cycles* m^i *and* m^j*, respectively.*

Proof. If each burst pair arrives at the sending node of each unidirectional link at least L ms apart, two bursts cannot collide with each other in the unidirectional link including its end nodes because the second burst arrives to the sending node only after the first burst passes through the node completely. Therefore, there will be no burst collision in the network. This proves the sufficient condition. On the other hand, if there is no burst collision in the network, it means that there is no burst collision in any unidirectional link, including its end nodes. It implies that each burst pair arrives at the sending node of each unidirectional link at least L ms apart. This proves the necessary condition. □

In ILP formulation, as joint optimization for single-link SRLG fault localization, we deal with burst separation in each unidirectional link using two constraints and an additional binary variable g_{uv}^{ij} for each pair of bursts along m-cycles m^i and m^j that traverse a unidirectional link (u, v). The first constraint is $(s^i + p_{uv}^i) - (s^j + p_{uv}^j) \geq (1 - (N + 1) * g_{uv}^{ij}) * L + (e_{uv}^i + e_{uv}^j - 2) * N$ and the second constraint is $(s^j + p_{uv}^j) - (s^i + p_{uv}^i) \geq (g_{uv}^{ij} - N * (1 - g_{uv}^{ij})) * L + (e_{uv}^i + e_{uv}^j - 2) * N$. The binary variable g_{uv}^{ij} is 1 if $(s^i + p_{uv}^i) < (s^j + p_{uv}^j)$ and 0 otherwise.

In heuristic burst scheduling, we deal with burst separation in each unidirectional link by finding the earliest possible collision-free burst launching time s^j from the MN for the burst along each m-cycle m^j that provide minimum fault localization latency.

Obviously, in joint optimization, both the burst launching times and propagation delays to the sending node of each unidirectional link can be manipulated to achieve non-overlapping bursts in each unidirectional link. However, in separate optimization, only the burst launching times can be manipulated for the purpose.

4.2.2 Node Switch Fabric Configuration Scheduling

With the proposed m-burst framework, the MN is given and a set of m-cycles are identified, where an optical burst with a fixed length is launched along each m-cycle in order to determine if there is any failed link along the m-cycle. With the short-duration bursts, the monitoring resources can be reduced by multiplexing bursts of multiple m-cycles in the time domain along some wavelength links. Therefore, the proposed m-burst is expected to significantly improve the performance reported by the previous studies where static S-LPs are employed.

With such saving of monitoring resources, m-burst is subject to longer monitoring delay and larger signaling overhead. The longer monitoring delay is due to the discrete time instants on burst return to the MN and any possible mechanism for ensuring that a burst loss is merely due to link failure instead of any other reason. The larger signalling overhead, on the other hand, is caused by the signalling effort for real-time configuration of switch fabrics for the m-cycles. Therefore, a careful design is required to mitigate the above two issues.

Here, the monitoring delay is defined as the shortest duration starting when the MN launches the first burst until it can come up with a valid alarm code (including the code $00\cdots0$ showing no failure). With m-bursts, monitoring delay is an index for how fast the MN can respond to any failure event considered in the m-cycle deployment, while the

number of m-cycles and their total cover length concern the amount of signaling overhead in the node switch fabric configuration.

Different from the case with static S-LPs, the proposed m-burst framework should be supported by a specific resource reservation scheme. We suggest in [9] to use the Just-Enough-Time (JET) [40] scheme that takes advantage of tell-and-go signaling and delayed reservation scheme. With delayed reservation, the source node sends the control packet first in order to reserve the switching time in the switch fabric at each intermediate node of the route, and the optical burst is launched after an offset time. With tell-and-go, the offset time is minimized by sending the optical burst without waiting for the acknowledgement; thus, the offset time can be as short as simply the sum of nodal processing time at each intermediate node. Thus, JET has each node specified with the exact time instants when the optical burst will arrive and depart, respectively, such that the switch fabric of the node will be reserved only when the optical burst traverses through the node, and will be released after the departure of the burst without any further signaling notice.

In the m-burst framework, MN can be considered as only edge node and all other nodes of the network as core nodes in the context of OBS [36]. Most of the activities of traditional OBS are done as preprocessing in the framework. Burst assembling become trivial because bursts are fixed length optical signals. At most one monitoring wavelength channel at each direction of each link is reserved. The routes of the m-cycles are found as the solution of the m-cycle allocation problem. Contention for resources across the network is resolved by scheduling m-burst launching times from MN. As the routes of the bursts are known and stable, calculation of burst offsets from their respective control packets also becomes trivial. The launching time of a control packet is calculated by deducting the corresponding offset from the burst launching time.

The remaining main functions for MN are sending control packets and bursts, which can be considered as routine work. The only remaining work for the core nodes is resource scheduling. The estimated set-up and estimated release scheme for resource reservation can be used because burst size is fixed and control packets carry offset information.

Although we initially wanted to use the same resource reservation mechanism as JET, a non-trivial difference exists between the two in the aspect of design premises and implementations. The bursts in the proposed m-burst framework are taken as a means of out-of-band monitoring that inspects the on-off status of the m-cycles. Instead of bearing any significant modulated information bit for data transfer as in OBS, m-bursts are fixed length optical signals carrying minimal information such as identification of m-cycles. In this regards, a much more relaxed requirement on time precision and optical signal quality can be expected, which makes the proposed framework easily implementable.

In the m-burst framework, one m-burst is launched from the MN along each m-cycle every T ms. Each m-burst will traverse on-cycle unidirectional links along a particular m-cycle. Within T ms, every burst returns to the MN if there is no faulty SRLG in the network. Each m-burst will arrive to on-cycle nodes periodically, exactly after each T ms from its last arrival time. Thus, after much deliberation, we have come to the conclusion that we need not send a control packet ahead of each m-burst. Instead of that, if each burst arrival time, the corresponding m-cycle route, and the value of T are communicated to each node once at the beginning of the monitoring and after long intervals, the node will be able to configure its switch fabric correctly for burst multiplexing.

4.3 Joint Optimization

This section[2] formulates an ILP to solve jointly the m-cycle allocation and m-burst scheduling problems of the m-burst framework. The ILP is given in subsection 4.3.1. Numerical experiments on small network are conducted to investigate feasibility of the application of the m-burst framework for single-link SRLG fault localization. The results are provided in subsection 4.3.2.

4.3.1 ILP Formulation for Joint Optimization

The ILP will take a network and a single MN in the network as inputs. The ILP will find a set of m-cycles and derive burst launching times from the MN along the m-cycles at the same time. The set of m-cycles provided by the ILP as a solution will be used to localize single-link SRLG faults in the network. A single monitoring wavelength channel (WL) in each unidirectional link will be assigned if the link is traversed by any m-cycle in the solution.

Notation List

G The network, $G = (V, E)$, where V is the set of nodes and E is the set of unidirectional links of the network.

MN The monitoring node in the network, $MN \in V$.

[2]The content of this section has been published as a workshop paper [9]

r_1, r_2 Predefined cost ratios.

J Predefined maximum number of allowed m-cycles in an ILP solution.

j, k Indices of m-cycles $j, k \in \{0, 1, 2, \ldots, J-1\}$.

m^j Binary variable, it is equal to 1 if the jth m-cycle is in an ILP solution and 0 otherwise.

d_u^j Binary variable, it is equal to 1 if node u is the sink of the jth m-cycle and 0 otherwise. Note that only the MN can be the sink of an m-cycle in an ILP solution.

e_{uv}^j Binary variable, it is equal to 1 if the jth m-cycle traverses unidirectional link (u, v) and 0 otherwise.

δ Predefined small positive constant. It is the minimum voltage increase along an m-cycle, $|E|^{-1} \geq \delta > 0$.

q_{uv}^j Fractional variable, it is defined as the *voltage* of the vector $u \rightarrow v$ for unidirectional link (u, v). It assumes an arbitrary positive value if the jth m-cycle traverses the link and 0 otherwise.

z_u^j Binary variable, it is equal to 1 if the jth m-cycle visits node u and 0 otherwise. The predefined constant δ and the variables z_u^j and q_{uv}^j help to keep each m-cycle in an ILP solution a single connected component.

α_{uv} Integer variable, the unique decimal alarm code of the undirected link (u, v).

β Predefined small positive constant, $2^{-J} \geq \beta > 0$.

f_{uv}^{xy} Binary variable, it is equal to 1 if $\alpha_{uv} > \alpha_{xy}$ and 0 if $\alpha_{uv} < \alpha_{xy}$. The predefined constant β and the variable f_{uv}^{xy} help to find unique decimal alarm code for each undirected link of the network.

n Integer variable, it is the maximum number of m-cycles in an ILP solution traversing through any unidirectional link of the network.

N A large predefined number representing ∞.

L Predefined constant burst length in ms.

c_{uv} Predefined burst propagation delay in ms through unidirectional link (u, v).

s^j Real variable, the burst launching time in ms from the MN along the jth m-cycle.

T Real variable, total monitoring delay or the fault localization latency in ms.

h^j_{MNv} Binary variable, it is equal to 1 if unidirectional link (MN, v) is the head link of the jth m-cycle and 0 otherwise.

r^j_{tMN} Binary variable, it is equal to 1 if unidirectional link (t, MN) is the tail link of the jth m-cycle and 0 otherwise.

w^j_{tuv} Binary variable, it is equal to 1 if the jth m-cycle traverses unidirectional link (t, u) immediately before traversing unidirectional link (u, v) and 0 otherwise. This is the link continuity variable. It helps to calculate properly burst propagation delays from the MN to the on-cycle nodes along non-simple m-cycles.

tp^j Real variable, the total round trip burst propagation delay in ms along the jth m-cycle.

p^j_{uv} Real variable, the burst propagation delay in ms along the jth m-cycle from the MN to on-cycle node u when the next hop node from node u along the m-cycle is node v and N otherwise.

g^{jk}_{uv} Binary variable, it is equal to 1 if the burst along the jth m-cycle arrives to node u before the burst along the kth m-cycle, i.e., $(s^j + p^j_{uv}) < (s^k + p^k_{uv})$ and 0 if the burst along the jth m-cycle arrives to the node after the burst along the kth m-cycle, i.e., $(s^j + p^j_{uv}) > (s^k + p^k_{uv})$. It helps collision-free burst propagation through network links.

ILP for M-Cycle Allocation and Burst Scheduling

The specific ILP formulation is provided as follows.

Objective:

$$Minimize \ \ \{T + r_1 * \sum_j m^j + r_2 * n\} \tag{4.1}$$

Subject to the following constraints:

The constraints (4.2) to (4.15) and their variables are related with m-cycle formation and mostly taken from the method in [63]; please refer to the paper for a thorough explanation of the constraints and the variables. The constraint (4.16) is for finding the

maximum number of m-cycles traversing through any unidirectional link of the network. The constraints (4.17) to (4.35) are related with collision-free m-burst scheduling.

A single sink is allowed for an m-cycle.

$$\sum_{u \in V} d_u^j \leq 1, \quad \forall j \tag{4.2}$$

Only the MN can be the sink of an m-cycle.

$$d_{MN}^j = \sum_{u \in V} d_u^j, \quad \forall j \tag{4.3}$$

A sink can be assigned only to an m-cycle in an ILP solution.

$$m^j \geq d_{MN}^j, \quad \forall j \tag{4.4}$$

The indices of the m-cycles in an ILP solution are lower than the indices of the m-cycles not in the solution.

$$m^j \geq m^{j+1}, \quad \forall j : j \leq J - 2 \tag{4.5}$$

A lower bound on the number of m-cycles in a solution is given to help the ILP solver to find the optimal solution faster.

$$\sum_j m^j \geq \lfloor \log_2 |E| \rfloor + 1 \tag{4.6}$$

An m-cycle can traverse an undirected link in one direction only.

$$e_{uv}^j + e_{vu}^j \leq 1, \quad \forall (u, v) \in E : u < v, \forall j \tag{4.7}$$

This is the flow conservation defined for each node of the network. It enforces that each monitoring structure consists of cycle(s).

$$\sum_{(u,v) \in E} (e_{uv}^j - e_{vu}^j) = 0, \quad \forall u \in V, \forall j \tag{4.8}$$

A node is visited by an m-cycle if any adjacent link of the node is traversed by the m-cycle.

$$z_u^j \geq e_{uv}^j + e_{vu}^j, \quad \forall u \in V : (u, v) \in E, \forall j \tag{4.9}$$

Voltages of the vectors corresponding to on-cycle links can be assigned non-zero values.

$$q_{uv}^{j} \leq e_{uv}^{j}, \quad \forall (u,v) \in E, \forall j \tag{4.10}$$

This is node voltage constraint. It ensures that each m-cycle in an ILP solution is a single connected component. If a simple or non-simple m-cycle traverses a node except the sink, the sum of the vector voltage values for the node's outgoing links traverses by the m-cycle will be greater than the sum of the vector voltage values for the node's incoming links traverses by the m-cycle. However, as stated in [63], "the specific vector voltage values for the links are not important."

$$d_{u}^{j} + \sum_{(u,v)\in E} (q_{uv}^{j} - q_{vu}^{j}) \geq \delta * z_{u}^{j}, \quad \forall u \in V, \forall j \tag{4.11}$$

All decimal link alarm codes must be positive integers.

$$\alpha_{uv} \geq 1, \quad \forall (u,v) \in E : u < v \tag{4.12}$$

The decimal alarm code of an undirected link is based on the m-cycles traversing the link.

$$\alpha_{uv} = \sum_{j} 2^{j} * (e_{uv}^{j} + e_{vu}^{j}), \quad \forall (u,v) \in E : u < v \tag{4.13}$$

The constraints (4.14) and (4.15) ensure a unique decimal alarm code for each undirected link of the network. $\forall (u,v),(x,y) \in E : u < v$, $x < y$, and $(u,v) \neq (x,y)$,

$$\beta + \beta * (\alpha_{uv} - \alpha_{xy}) \leq f_{uv}^{xy} \tag{4.14}$$

$$\beta + \beta * (\alpha_{xy} - \alpha_{uv}) \leq 1 - f_{uv}^{xy} \tag{4.15}$$

The calculation of the maximum number of m-cycles traversing through any unidirectional link of the network.

$$n \geq \sum_{j} e_{uv}^{j}, \quad \forall (u,v) \in E \tag{4.16}$$

The constraint assigns lower and upper bounds of burst launching times. Burst starting times of only m-cycles in an ILP solution can be assigned positive values.

$$0 \leq s^j \leq m^j * N, \quad \forall j \tag{4.17}$$

The calculation of total round trip burst propagation delay along the jth m-cycle.

$$tp^j = \sum_{(u,v)\in E} c_{uv} * e^j_{uv}, \quad \forall j \tag{4.18}$$

The calculation of collision-free total monitoring delay or the fault localization latency that is equal to the largest elapse time for any m-burst from the start of the monitoring period to the return of the burst back to the MN.

$$T \geq s^j + tp^j + L, \quad \forall j \tag{4.19}$$

An on-cycle link from the MN can be the head link of the jth m-cycle.

$$h^j_{MNv} \leq e^j_{MNv}, \quad \forall j \tag{4.20}$$

Only an m-cycle in an ILP solution has a head link from the MN.

$$\sum_{(MN,v)\in E} h^j_{MNv} = m^j, \quad \forall j \tag{4.21}$$

An on-cycle link to the MN can be the tail link of the jth m-cycle.

$$r^j_{tMN} \leq e^j_{tMN}, \quad \forall j \tag{4.22}$$

Only an m-cycle in an ILP solution has a tail link to the MN.

$$\sum_{(t,MN)\in E} r^j_{tMN} = m^j, \quad \forall j \tag{4.23}$$

The constraints (4.24), (4.25), (4.26), and (4.27) identify a unique on-cycle incoming link to a node for each on-cycle out-going link from the node and allow calculation of burst propagation delays to the node properly along non-simple m-cycles in an ILP solution.

Each link continuity variable of the jth m-cycle through node u can be assigned 1 if the corresponding incoming link to the node is an on-cycle link of the m-cycle.

$$w^j_{tuv} \leq e^j_{tu}, \quad \forall u \in V : (t,u) \in E, \forall j \tag{4.24}$$

Only one link continuity variable is allowed to be 1 for an on-cycle incoming link for the jth m-cycle to node u.

$$\sum_{(u,v)\in E} w^j_{tuv} = e^j_{tu}, \quad \forall u \in V : (t,u) \in E, \forall j \tag{4.25}$$

Only one link continuity variable is allowed to be 1 for an on-cycle outgoing link for the jth m-cycle from node u.

$$\sum_{(t,u)\in E} w^j_{tuv} = e^j_{uv}, \quad \forall u \in V : (u,v) \in E, \forall j \tag{4.26}$$

The link continuity variable for the incoming tail link to the MN and outgoing head link from the MN of the jth m-cycle is ensured to be 1.

$$w^j_{tMNv} + (1 - r^j_{tMN}) \geq h^j_{MNv}, \forall (MN,v) \in E, \forall j \tag{4.27}$$

The lower and upper bounds of the burst propagation delay to node u through the jth m-cycle.

$$0 \leq p^j_{uv} \leq N, \quad \forall (u,v) \in E, \forall j \tag{4.28}$$

The burst propagation delay to node u along the jth m-cycle is infinity if the unidirectional link (u,v) is not traversed by the m-cycle.

$$p^j_{uv} \geq (1 - e^j_{uv}) * N, \quad \forall (u,v) \in E, \forall j \tag{4.29}$$

The burst propagation delay at the MN toward the first link of the jth m-cycle from the MN is assigned as 0. The constraint acts as a base case for the recursive burst propagation delay calculations as per constraints (4.31) – (4.33).

$$p^j_{MNv} \leq (1 - h^j_{MNv}) * N, \quad \forall (MN,v) \in E, \forall j \tag{4.30}$$

When node u is not the MN and the next hop node along the jth m-cycle is v, the constraint is for the calculation of the burst propagation delay along the m-cycle from the MN through the head link of the m-cycle to node u.

$$p^j_{uv} \geq p^j_{tu} + c_{tu} - (1 - w^j_{tuv}) * N, \forall (u,v),(t,u) \in E : u \neq MN, \forall j \tag{4.31}$$

When node u is the MN and the next hop node along the jth m-cycle is v, the constraint is for the calculation of the burst propagation delay along the m-cycle from the MN to itself. The burst propagation delay increases by the propagation delay of the incoming link if the outgoing link is not the head link of the m-cycle. Thus, $\forall (MN,v),(t,MN) \in E, \forall j$,

$$h^j_{MNv} * N + p^j_{MNv} \geq p^j_{tMN} + c_{tMN} - (1 - w^j_{tMNv}) * N \tag{4.32}$$

The total burst propagation delay of the jth m-cycle is the burst propagation delay up to the MN through the tail link of the m-cycle.

$$tp^j \geq p^j_{tMN} + c_{tMN} - (1 - r^j_{tMN}) * N, \forall (t,MN) \in E, \forall j \tag{4.33}$$

The constraint (4.34) and (4.35) enforce difference between arrival times of any two m-bursts along the jth and kth m-cycles to node u that are traversing same outgoing link (u,v) from the node to the burst length L. Thus, they ensure collision-free burst propagation through any link of the network. They are the key constraints for collision-free burst scheduling. Thus, $\forall u \in V : (u,v) \in E, \forall j,k : j \neq k$,

$$(s^j + p^j_{uv}) - (s^k + p^k_{uv}) \geq (1 - (N+1) * g^{jk}_{uv}) * L + (e^j_{uv} + e^k_{uv} - 2) * N \tag{4.34}$$

$$(s^k + p^k_{uv}) - (s^j + p^j_{uv}) \geq (g^{jk}_{uv} - N * (1 - g^{jk}_{uv})) * L + (e^j_{uv} + e^k_{uv} - 2) * N \tag{4.35}$$

The objective (4.1) aims at minimizing total monitoring delay or the fault localization latency T. It also minimizes the number of m-cycles $\sum_j m^j$ and the maximum number of m-cycles traversing through any unidirectional link n. Relative importance of the objectives is manipulated with the help of cost ratios r_1 and r_2.

4.3.2 Numerical Results

We have developed a model of the ILP for joint optimization in the modeling language GNU MathProg and translated the MathProg model using the GLPK solver *glpsol* into an MPS file for each network to run the program in ILOG CPLEX 11.1.

We assume that the burst length L is 20 ms and the propagation delay c_{uv} through any unidirectional link (u, v) is 2 ms in the numerical experiment. We have also assumed that one supervisory wavelength channel (WL) will be assigned in each unidirectional link for monitoring purposes if the link is traversed by any m-cycle. We conducted the experiment on a network with 9 nodes and 14 links. Node 1 is the given MN.

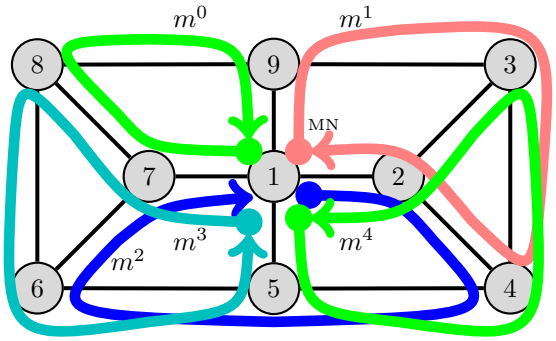


Figure 4.1: M-cycle solution set provided by the joint ILP for a network with 9 nodes and 14 links.

The ILP solution has $\sum_j m^j = 5$ m-cycles for single-link SRLG fault localization. M-cycles m^0, m^1, m^2, m^3, and m^4 traverse 4, 5, 6, 5, and 5 links, respectively. The solution allows at most $n = 2$ m-cycles in any unidirectional link of the network. The network with the m-cycles is shown in Figure 4.1.

The solution provides a unique decimal alarm code for each undirected link of the network. The decimal alarm code of a link is derived based on the m-cycles that traverse the link, without considering the direction of traversal through the link. Thus, unidirectional links (u, v) and (v, u) have the same decimal alarm code. Table 4.1 shows the alarm code table (ACT), which is the mapping between link failure events and link decimal alarm codes. The decimal alarm codes are given in column *Dec.* of the table.

Any single-link failure will disrupt a unique set of m-bursts. As a result, a unique decimal alarm code will be generated. The generated decimal alarm code identifies the failed link. For example, if m-cycles m^1, m^2, and m^4 are disrupted due to a single-link failure, the generated decimal alarm code will be 22. The MN can identify link $(1, 2)$ as the failed link from the ACT, using 22 as index.

The burst starting times s^0, s^1, s^2, s^3, and s^4 along the corresponding m-cycles given by the solution are 0 ms, 20 ms, 0 ms, 20 ms, and 0 ms, respectively. The total burst propagation delays tp^0, tp^1, tp^2, tp^3, and tp^4 through the m-cycles are 8 ms, 10 ms, 12 ms,

Table 4.1: ACT using the Joint ILP

Link	m^4	m^3	m^2	m^1	m^0	Dec.
(1, 2)	1	0	1	1	0	22
(1, 5)	1	1	0	0	0	24
(1, 7)	0	1	1	0	1	13
(1, 9)	0	0	0	1	1	3
(2, 3)	1	0	0	0	0	16
(2, 4)	0	0	1	1	0	6
(3, 4)	1	0	0	1	0	18
(3, 9)	0	0	0	1	0	2
(4, 5)	1	0	1	0	0	20
(5, 6)	0	1	1	0	0	12
(6, 7)	0	0	1	0	0	4
(6, 8)	0	1	0	0	0	8
(7, 8)	0	1	0	0	1	9
(8, 9)	0	0	0	0	1	1

10 ms, and 10 ms, respectively. The fault localization latency or total monitoring delay T is 50 ms. Thus, T is derived in the solution either from m-cycle m^1 as $s^1 + tp^1 + L = 20 + 10 + 20 = 50$ ms or from m-cycle m^3 as $s^3 + tp^3 + L = 20 + 10 + 20 = 50$ ms.

Figure 4.2 shows m-burst traversal timing through each unidirectional link traversed by more than one m-bursts. Each rectangle in the figure represents the period that the m-burst along the m-cycle traverses through the link. The period is $c_{uv} + L = 2 + 20 = 22$ ms. All links in the figure are traversed by 2 m-bursts back to back that means 2 bursts traverse through a link, one immediately after the other. Back-to-back periods are indicated by black rectangles. As shown in the figure, bursts along m-cycles m^0 and m^3 traverse unidirectional links (1, 7) and (7, 8) back to back, and bursts along m-cycles m^4 and m^1 traverse unidirectional link (2, 1) back to back. Thus, the m-bursts are non-overlapping through any link of the network. Hence, the m-bursts are collision-free throughout the network.

To determine whether the m-burst framework can be applied to any arbitrary mesh network, we have run the program with five more networks, with a single arbitrarily chosen MN for each one and keeping all other assumptions the same as the above experiment. Numbers of m-cycles $\sum_j m^j$, maximum numbers of m-cycles n through any unidirectional link, and monitoring delays T of the solutions are given in Table 4.2. Networks in the table are identified by the numbers of nodes and links.

For each of the five networks, the m-burst framework is able to identify a set of m-cycles to achieve unambiguous failure localization (UFL) for the single-link SRLG failures. The

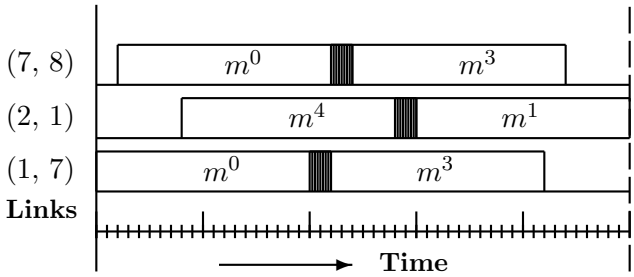


Figure 4.2: Schedule of link traversal by multiple m-bursts along the m-cycles provided by the joint ILP.

maximum number of m-cycles through any unidirectional link of each network is low. The fault localization latency or total monitoring delays from 26 ms to 52 ms of the solutions for the networks are optimal. For each network, the m-burst framework is able to achieve collision-free burst scheduling by keeping the m-bursts along the m-cycles non-overlapping throughout the network.

Table 4.2: The Performance of the Joint ILP in Additional Networks

Performance metrics	Networks				
	4 nodes 6 links	5 nodes 8 links	6 nodes 9 links	7 nodes 12 links	8 nodes 12 links
$\sum_j m^j$	3	4	4	6	4
n	1	1	2	2	2
T	26	26	48	48	52

4.4 Separate Optimization

The proposed m-burst framework has two distinct objectives: (1) minimization of the number of m-cycles and (2) minimization of the fault localization latency or total monitoring delay through m-burst scheduling along the m-cycles. We have formulated an ILP for the m-burst framework as a joint optimization problem of m-cycle design and m-burst scheduling in subsection 4.3.1 and numerically tested the ILP in subsection 4.3.2. To investigate the effectiveness of the m-burst framework, we investigate m-cycle and m-burst problems as separate optimization problems and compare the numerical results in this section.

4.4.1 ILP Formulation for M-Cycle Allocation

The main objective of the m-cycle design problem is to find the minimum number of m-cycles that provide unique alarm codes for all undirected links of a network. Another objective is to limit the maximum number of m-cycles traversing through any unidirectional link, which will limit the fault localization latency indirectly. These objectives are the same as those in the m-cycle allocation part of the objective of the ILP for joint optimization. Thus, we have taken relevant portions of the objective and constraints as well as used constants and variables with the same meaning from the ILP for joint optimization.

ILP for M-Cycle Allocation

The specific ILP formulation is provided as follows.

Objective:

$$Minimize \quad \left\{ \sum_{j} m^j + r_1 * n \right\} \tag{4.36}$$

Subject to the constraints (4.2) to (4.16) of the ILP for the joint optimization problem given in Section 4.3.1. Note that the constraints (4.2) to (4.15) are related to m-trail formation and mostly taken from the method in [63]. The constraint (4.16) is for finding the maximum number of m-cycles traversing through any unidirectional link of the network.

The objective (4.36) aims at minimizing the number of m-cycles $\sum_j m^j$ and the maximum number of m-cycles n traversing through any unidirectional link. The relative importance of the objectives is manipulated with the help of cost ratio r_1.

4.4.2 ILP Formulation for M-Burst Launching Time Scheduling

This subsection provides an ILP formulation for the m-burst launching time scheduling that takes a set of m-cycles as an input. The set of m-cycles is derived by the ILP for m-cycle allocation given in subsection 4.4.1 as a solution for single-link SRLG fault localization. Note that each m-cycle in the solution set traverses the given MN. Assume that one WL is assigned in each unidirectional link traversed by at least one m-cycle in the solution set.

As the paths of m-cycles are given, the calculation of the burst propagation delay along the m-cycles is done as preprocessing; doing so decreases the complexity of the problem

significantly. However, we have to consider burst traversal directions along the m-cycles as new variables because the burst traversal along the m-cycles in the given direction may not provide the minimum fault localization latency. As burst propagation delays from the MN to on-cycle nodes depend on the burst flow direction along an m-cycle, two spin variables per m-cycle are used to keep track of propagation delays properly.

Notation List

G The network, $G = (V, E)$, where V is the set of nodes and E is the set of unidirectional links of the network.

N A large number representing ∞.

L Predefined constant burst length in ms.

M Number of m-cycles in the solution.

i, j Indices of m-cycles $i, j \in \{0, 1, 2, \ldots, M-1\}$.

sp^{jr} Binary variables for spins of the jth m-cycle. Given spin sp^{j0} is equal to 1 if the m-burst along the jth m-cycle traverses each on-cycle link in the given direction and 0 otherwise. Reverse spin sp^{j1} is equal to 1 if the m-burst along the jth m-cycle traverses each on-cycle link in the opposite direction and 0 otherwise.

p_k^{jr} Pre-calculated propagation delay in ms of the m-burst from the MN to the kth node of the jth m-cycle along given or reverse spin sp^{jr}.

tp^j Pre-calculated round trip burst propagation delay in ms of the m-burst along the jth m-cycle starting from and returning to the MN.

s^{jr} Real variable, the m-burst launching time in ms from the MN along the jth m-cycle. It can be assigned non-zero value only if the jth m-cycle with sp^{jr} spin is 1.

g_{uv}^{irjt} Binary variable, it is equal to 1 if the m-burst along the ith m-cycle with sp^{ir} spin arrives to node u before the m-burst along the jth m-cycle with sp^{jt} spin and both the bursts traverse (u, v), and 0 otherwise. It helps collision-free burst propagation through network links.

T Real variable, the maximum fault localization latency in ms.

ILP for Burst Launching Time Scheduling

The specific ILP formulation is provided as follows.

Objective:

$$Minimize \ \ \{T\} \tag{4.37}$$

Subject to the following constraints:

Each m-cycle has only one non-zero spin.

$$sp^{j0} + sp^{j1} = 1, \qquad \forall j \tag{4.38}$$

The constraint ensures that positive burst starting times can be assigned only to the bursts along m-cycles with valid spins.

$$s^{jr} \leq sp^{jr} * N, \qquad \forall j, r \tag{4.39}$$

Constraints (4.40) and (4.40) enforce minimum difference between the arrival times of two m-bursts to node u when the two m-bursts along the ith and jth m-cycles traverse the same outgoing link (u, v) from the node. The minimum difference is the burst length L. Thus, they ensure collision free burst propagation through any link of the network. Here, the kth node of ith m-cycle considering sp^{ir} spin or the lth node of jth m-cycle considering sp^{jt} spin is the same node u. Moreover, the next hop node along the ith or jth cycle considering the respective spins is also the same node v. Now $\forall (u, v), i, j, r, t, l, k$,

$$(s^{ir} + p_k^{ir}) - (s^{jt} + p_l^{jt}) \geq (1 - (N + 1) * g_{uv}^{irjt}) * L + (sp^{ir} + sp^{jt} - 2) * N \tag{4.40}$$

$$(s^{jt} + p_l^{jt}) - (s^{ir} + p_k^{ir}) \geq (g_{uv}^{irjt} - N * (1 - g_{uv}^{irjt})) * L + (sp^{ir} + sp^{jt} - 2) * N \tag{4.41}$$

The fault localization latency is the largest elapse time for any m-burst with valid spin from the start of the monitoring period and return of the last bit of the m-burst to the MN.

$$T \geq s^{jr} + tp^{j} + L + (sp^{jr} - 1) * N, \quad \forall j, r \tag{4.42}$$

The objective (4.37) aims to minimize the fault localization T by manipulating spins of the m-cycles and the starting time of the m-bursts.

4.4.3 Numerical Results

We have solved two ILPs in modeling language GNU MathProg separately. The MathProg model for the m-cycle allocation ILP is translated using the GLPK solver *glpsol* into an MPS file for each network to run the program in ILOG CPLEX 11.1. The solution of the m-cycle allocation ILP is a set of m-cycles. The MathProg model for the m-burst scheduling ILP is also translated using the GLPK solver *glpsol* into another MPS file for the network to run the program in ILOG CPLEX 11.1 using the set of m-cycles derived by the m-cycle allocation ILP as an input.

At first, we run the program for m-cycle allocation in a network with 9 nodes and 14 links; it is the same network used in the joint optimization problem. Node 1 is the given MN.

Table 4.3: M-Cycle Solution Set Provided by the M-Cycle Allocation ILP

M-cycles	Paths
m^0	$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 1$
m^1	$1 \rightarrow 9 \rightarrow 3 \rightarrow 2 \rightarrow 1$
m^2	$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 9 \rightarrow 1 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 1$
m^3	$1 \rightarrow 7 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 3 \rightarrow 2 \rightarrow 1$
m^4	$1 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 1$

A set of 5 cycles form an m-cycle solution for any single-link failure localization. The paths of the m-cycles are shown in Table 4.3. M-cycles m^0, m^1, m^2, m^3, and m^4 traverse 5, 4, 9, 7, and 6 links, respectively. The m-cycles define a set of unique decimal alarm codes for the network links as shown in Table 4.4. The solution allows at most 2 m-cycles through any unidirectional link of the network.

The m-cycle solution set provided by the m-cycle allocation method is used as an input to the m-burst scheduling problem. To run the program for m-burst scheduling, we assume that the burst length L is 20 ms, and the propagation delay δ through each unidirectional link is 2 ms in the experiment.

As shown in Figure 4.3, orientations of the bursts along the m-cycles m^1, m^2 and m^4 are reversed to get the minimum fault localization latency or total monitoring delay T. The burst starting times s^0, s^1, s^2, s^3, and s^4 along the corresponding m-cycles given by the solution are 0 ms, 40 ms, 30 ms, 2 ms, and 18 ms, respectively. The total burst propagation delays tp^0, tp^1, tp^2, tp^3, and tp^4 through the m-cycles are 10 ms, 8 ms, 18 ms, 14 ms, and 12 ms, respectively. Thus, the fault localization latency or total monitoring delay T is derived from either m-cycle m^1 as $s^1 + tp^1 + L = 40{+}8{+}20 = 68$ ms or from m-cycle m^2 as $s^2 + tp^2 + L = 30{+}18{+}20 = 68$ ms. But this delay is sub-optimal.

Table 4.4: ACT using the Separate ILPs

Link	m^4	m^3	m^2	m^1	m^0	Dec.
(1, 2)	0	1	1	1	1	15
(1, 5)	1	0	1	0	1	21
(1, 7)	0	1	1	0	0	12
(1, 9)	1	0	1	1	0	22
(2, 3)	0	1	0	1	1	11
(2, 4)	0	0	1	0	0	4
(3, 4)	0	0	1	0	1	5
(3, 9)	0	1	1	1	0	14
(4, 5)	0	0	0	0	1	1
(5, 6)	1	0	1	0	0	20
(6, 7)	1	1	1	0	0	28
(6, 8)	0	1	0	0	0	8
(7, 8)	1	0	0	0	0	16
(8, 9)	1	1	0	0	0	24

Figure 4.4 shows m-burst traversal timing through each unidirectional link traversed by more than one m-bursts. Back-to-back m-bursts are indicated in the figure by black rectangles, which mean the two bursts traverse through a unidirectional link one immediately after the other. The bursts along m-cycles m^4 and m^2, m^0 and m^4, and m^3 and m^4 traverse unidirectional links (1, 9), (5, 1), and (7, 6) back to back, respectively. With the scheduling effort, burst collision was completely avoided by manipulating the burst launching time from the MN along each m-cycle.

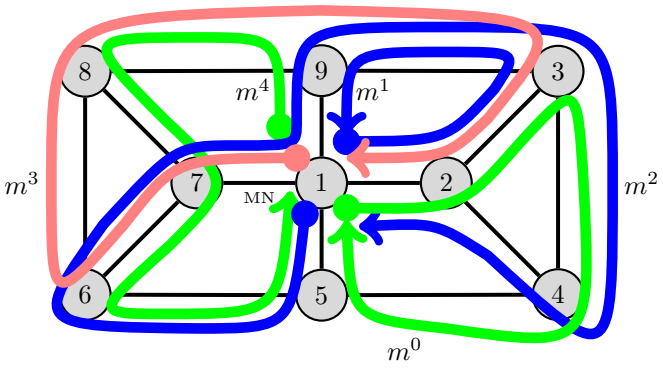


Figure 4.3: M-cycle solution set provided by the separate ILPs for a network with 9 nodes and 14 links.

Both the joint and the separate optimization methods solve the related problems of the

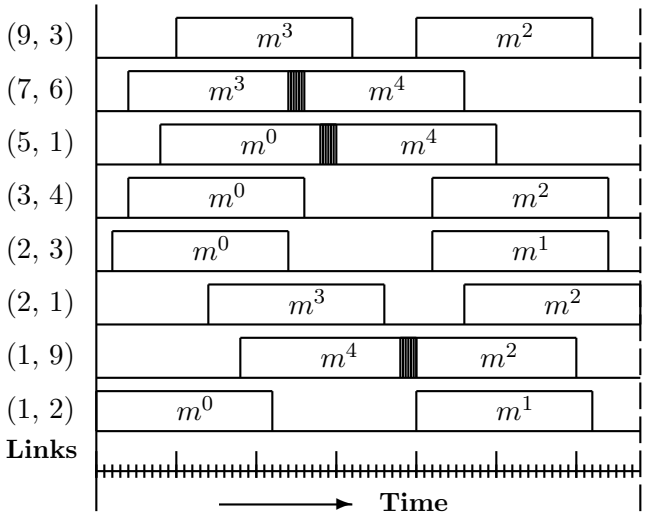


Figure 4.4: Schedule of link traversal by multiple m-bursts along the m-cycles provided by the separate ILPs.

m-burst framework for the same network with 9 nodes and 14 links. The joint optimization method provides smaller fault localization latency T. Now to compare the performance of the methods further, we run the separate optimization method for 5 more networks as well.

The performances of the methods are compared in Table 4.5. Networks are identified in the table with the numbers of nodes and links. The number of m-cycles $\sum_j m^j$ in the solution set by both the methods are the same for all the networks except the network with 7 nodes and 12 links where the joint optimization method required a higher number of m-cycles. The higher number of m-cycles in the solution set to reduce the fault localization latency is not unexpected of the joint optimization method because of the comparative importance of the fault localization latency and the number of m-cycles is represented by the cost ratio in the objective function of the ILP. The maximum number of m-cycles n traversing through any unidirectional link of the network derived by both the methods is the same for all the networks. In 50% of the networks, the joint optimization method outperforms the separate optimization method in terms of the minimum fault localization latency T.

Table 4.5: The Comparative Performance of the Joint and Separate ILP Methods

Network	Joint			Separate		
	$\sum_j m^j$	n	T	$\sum_j m^j$	n	T
4 nodes, 6 links	3	1	26	3	1	26
5 nodes, 8 links	4	1	26	4	1	26
6 nodes, 9 links	4	2	48	4	2	50
7 nodes, 12 links	6	2	48	4	2	54
8 nodes, 14 links	4	2	52	4	2	52
9 nodes, 14 links	5	2	50	5	2	68

4.5 Heuristic Algorithms

As the ILP methods cannot be applied in a large network to get a solution in reasonable time, we propose the heuristic algorithms for m-cycle allocation and m-burst scheduling in subsections 4.5.1 and 4.5.2, respectively.

4.5.1 M-Cycle Allocation Method

The pseudo code of the m-cycle allocation method to localize single-link SRLG failures is given in Algorithm 4.1. Two main inputs of the algorithm are a network and a single monitoring node (MN). Each m-cycle starts from and returns to the MN. The primary objective is to find the minimum number of m-cycles that is able to localize single-link failures unambiguously. All links are considered unidirectional. Initially, the m-cycle solution set $\mathfrak{M}$ and the alarm code table (ACT) $\mathfrak{A}$ are empty.

The outer *while* loop is to find the minimum cost m-cycle solution set $\mathfrak{M}$. The cost of a solution set is defined as $c_{new} = r_1 * \sum_j m^j + r_2 * n + \sum_j \sum_{(u,v)} e^j_{uv}$ where $\sum_j m^j$ is the total number of m-cycles in the solution, n is the maximum number of m-cycles in the solution traversing through any unidirectional link of the network, and $\sum_j \sum_{(u,v)} e^j_{uv}$ represents the total number of links traversed by the m-cycles in the solution. r_1 and r_2 are constants defining relative importance of the cost components. The loop iterates until the current minimum cost c_{min} does not improve for n_{max} times.

The inner *while* loop finds a sufficient number of unique m-cycles in a solution set to localize single-link SRLG faults unambiguously. Thus, each link has to be traversed by a unique subset of m-cycles in the solution set. The unique subset of m-cycles defines a unique alarm code for each undirected link of the network. As a result, the faulty link can

be identified unambiguously from the disrupted m-cycles. Each m-cycle is found by using the helper function FindMCycle.

In line 7, the m-cycles solution set $\mathfrak{M}$ and the alarm code table $\mathfrak{A}$ are returned.

Algorithm 4.1: The M-Cycle Allocation Method for Single-Link SRLG Fault Localization

Input: $G(V, E)$, MN, and n_{max}
Output: $\mathfrak{M}$ and $\mathfrak{A}$
1 Initialize $\mathfrak{A}$, $\mathfrak{M} \leftarrow \phi$, $c_{min} \leftarrow \infty$, $n_{min} \leftarrow 0$
while $n_{min} < n_{max}$ **do**
2 $\mathfrak{A}_{new}$, $\mathfrak{M}_{new} \leftarrow \phi$
  while *# distinct alarm codes in* $\mathfrak{A}_{new}$ $<$ *# undirected link in* G **do**
3 $m^j \leftarrow$ FindMCycle($\mathfrak{A}_{new}$)
    if $m^j \notin \mathfrak{A}_{new}$ **then**
4 $\mathfrak{A}_{new}$, $\mathfrak{M}_{new} \leftarrow m^j$
5 Update c_{new} using m-cycles in $\mathfrak{M}_{new}$, $n_{min} \leftarrow n_{min} + 1$
  if $c_{new} < c_{min}$ **then**
6 $\mathfrak{A} \leftarrow \mathfrak{A}_{new}$, $\mathfrak{M} \leftarrow \mathfrak{M}_{new}$, $c_{min} \leftarrow c_{new}$, $n_{min} \leftarrow 0$
7 Return $\mathfrak{A}, \mathfrak{M}$

The helper function FindMCycle is a randomized local search method that finds a valid m-cycle. Note that for each undirected link (u, v) in the network, n_{uv} is initially 1, and each time the link is traversed by an m-cycle in $\mathfrak{A}_j$, n_{uv} is incremented.

The outer *while* loop of the function iterates until a valid m-cycle is not found. Each search for m-cycle m^j starts from the MN and terminates at the MN. In other words, each m-cycle takes the MN as its first node, visits other nodes possibly multiple times, and returns to the MN. Initially, the MN is the current node u of m^j.

The inner *while* loop of the function finds the next current nodes of m-cycle m^j randomly in sequence. In each iteration of the inner loop, node v is chosen from the u adjacent nodes as a tentative next current node. If there is no outgoing unidirectional link from the current node u that can be included in m-cycle m^j, the current search is aborted and a new search is started as the next iteration of the outer *while* loop. Otherwise, if the undirected link (u, v) is not already in m-cycle m^j, the link will be included in m-cycle m^j with probability p. The link will certainly be included in m-cycle m^j if an undirected link (u, v) is not traversed by any m-cycle in $\mathfrak{A}_j$. Moreover, if $p \leq \frac{1}{n_{uv}}$, the link will be included in m-cycle m^j. If a unidirectional link (u, v) is included in m-cycle m^j, node v becomes the current node u, and the next iteration of the inner loop starts.

Once the MN is chosen as the current node u of m-cycle m^j a second time, the m-cycle becomes a valid one and the function returns m-cycle m^j in line 8.

Function FindMCycle($\mathfrak{A}_j$)

Input: $G(V, E)$, MN, and $\mathfrak{A}_j$
Output: m^j

1 Initialize $m^j \leftarrow \phi$
while $m^j == \phi$ **do**
2 $u \leftarrow$ MN
  while $u \neq MN \vee |m^j| == \phi$ **do**
3 Choose a node $v \in V : (u, v) \in E$ randomly with probability p
  if *there is no outgoing link from node u that can be chosen* **then**
4 $m^j \leftarrow \phi$
5 Exit the inner *While* loop.
  if $(u, v) \notin m^j \wedge (v, u) \notin m^j \wedge p \leq \frac{1}{n_{uv}}$ **then**
6 $m^j \leftarrow (u, v)$
7 $u \leftarrow v$
8 Return m^j

4.5.2 M-Burst Launching Time Scheduling

Once a set of m-cycles $\mathfrak{M}$ that provides a unique code for each single-link SRLG is derived by the heuristic given in subsection 4.5.1, the burst launching times from the MN are calculated. The set of starting times S of the m-bursts along the m-cycles are derived by the burst scheduling algorithm given in Algorithm. 4.2. The algorithm is based on Tabu search [31].

Tabu search is a metaheuristic that starts with a current solution (found randomly or based on the specific problem instance), and then iterates as long as the current solution seems to be improving. A Tabu list keeps track of the recent current solutions in order to avoid the same solution as the current solution repeatedly. Each solution in the Tabu list is a Tabu solution and remains so for a predefined number of current solution updates. In each iteration of Tabu search, the best solution in the neighbourhood of the current solution is found. Usually in permutation problems, each neighbourhood is identified by pairwise swapping of the positions in the current solution. Each pairwise swapping of the positions provides another potential solution and is called a move. A move is associated with a move value. If the best move is an improvement on the current solution, it is

accepted as the current solution. Otherwise, the best move that is not a Tabu solution is selected as the current solution [31].

In Algorithm 4.2, the m-cycles in $\mathfrak{M}$ are stored in an ordered list *seq*. A sequence of m-cycles is defined by the positions of m-cycles in *seq* at a point in time. The primary objective of the burst scheduling heuristic is to find the minimum fault localization latency T, avoiding burst collisions altogether. The near optimal sequence of m-cycles that provides minimum T is found by repeatedly swapping positions of m-cycles pair-wise in *seq* as per Tabu search rules [31].

In line 1, T is set to ∞. The outer *while* loop finds the minimum T. Each iteration of the outer *while* loop has two sections: initialization in lines 2-7 and Tabu search in lines 8-16.

In line 2, the Tabu list, the ordered list *seq*, new spin set sp_{new}, and new propagation delay table P are initialized. Then, a position in *seq* for each m-cycle is chosen randomly and spin sp^j_{new} of the m-cycle is also chosen randomly in lines 3 and 4. The spin will determine if the m-burst will traverse on-cycle links in a given or opposite direction. Then, the burst propagation delay p^j_{uv} from the MN to node u of each unidirectional link (u, v) traversed by m-cycle m^j is calculated in line 5, considering the spin of the burst. Next, a new fault localization latency T_{new} and new set of burst launching times S_{new} are initialized using the helper function FindMaxDelay in line 6. Finally, the spin set *sp*, the set of burst launching times S, and the minimum fault localization latency T are updated if T_{new} is less than T in line 7.

The inner *while* loop is the Tabu search section. It has two sub-sections: Tabu move and update. The inner loop will find minimum T_{new} for a particular spin set sp_{new} of the m-cycles.

In Tabu move sub-section, lines 8-12, all the move values corresponding to the pair-wise swapping of the current positions of the m-cycles in *seq* are calculated by using the helper function FindMaxDelay. A move is either a Tabu or non-Tabu move. The best Tabu and non-Tabu moves are found in the sub-section that involves the most expensive calculations.

In the update sub-section, lines 13-16, the relevant parameters *seq*, S_{new}, and T_{new} are updated based on the move values. If the maximum improvement of T_{new} is provided by the best Tabu move, it is used to update the relevant parameters. Otherwise, the best non-Tabu move is used to update the relevant parameters. Then, the Tabu list is updated for the next iteration of the inner *while* loop. Next, *sp*, S, and T are updated if T_{new} is less than T in line 16.

The spin set *sp*, the set of starting times S, and fault localization latency T are returned in line 17.

Algorithm 4.2: M-Burst Launching Time Scheduler

Input: $G(V, E)$, MN, n, and $\mathfrak{M}$
Output: sp, S, and T
begin
Initialize $T \leftarrow \infty$
while *# iteration of non-improving* $T < n$ **do**
Initialize Tabu list, seq, sp_{new}, and P
for *each* $m^j \in \mathfrak{M}$ **do**
Randomly choose a free position k in seq and the spin sp^j_{new} of the m-trail
$seq[k] \leftarrow m^j$, $sp_{new} \leftarrow sp^j_{new}$
$P \leftarrow$ Find burst propagation delay p^j_{uv}, $\forall (u, v) \in E$
S_{new}, $T_{new} \leftarrow$ FindMaxDelay(P, seq)
if $T_{new} < T$ **then**
$sp \leftarrow sp_{new}$, $S \leftarrow S_{new}$, $T \leftarrow T_{new}$
while *# iteration of non-improving* $T_{new} < n$ **do**
Initialize T_{nt}, $T_t \leftarrow \infty$
for *each* i*th and* j*th positions in* seq **do**
$seq_{ij} \leftarrow$ swap positions of the m-trails
S_{ij}, $T_{ij} \leftarrow$ FindMaxDelay(P, seq_{ij})
if *it is a non-Tabu move* $\wedge$ $T_{ij} < T_{nt}$ **then**
$seq_{nt} \leftarrow seq_{ij}$, $S_{nt} \leftarrow S_{ij}$, $T_{nt} \leftarrow T_{ij}$
if *it is a Tabu move* $\wedge$ $T_{ij} < T_t$ **then**
$seq_t \leftarrow seq_{ij}$, $S_t \leftarrow S_{ij}$, $T_t \leftarrow T_{ij}$
if $T_t \leq T_{new} \wedge T_t < T_{nt}$ **then**
$seq \leftarrow seq_t$, $S_{new} \leftarrow S_t$, $T_{new} \leftarrow T_t$
else
$seq \leftarrow seq_{nt}$, $S_{new} \leftarrow S_{nt}$, $T_{new} \leftarrow T_{nt}$
Update Tabu list
if $T_{new} < T$ **then**
$sp \leftarrow sp_{new}$, $S \leftarrow S_{new}$, $T \leftarrow T_{new}$
Return sp, S, T

In Algorithm 4.2, seq can have $O(|\mathfrak{M}|!)$ number of sequences of m-cycles. The Tabu search technique helps to find a sequence of m-cycles in seq, which provides near-optimal T without searching all the sequences of m-cycles.

The helper function FindMaxDelay calculates the starting times S_{sq} of m-bursts from the MN along the m-cycles from the first position to the last position of the given sq. Each burst is kept non-overlapping in each unidirectional link with all other bursts along the m-cycles in the preceding positions of sq. Thus, $\forall m^i$ in the preceding position of each m-cycle m^j in sq, if the burst along m^j arrives to the sending node u of a link (u, v) before the burst along m^i completely passes through u, the starting time of the burst s^j has to be increased to satisfy the condition $(s^j + p^j_{uv}) \geq (s^i + p^i_{uv} + L)$ in order to avoid collision in

(u, v). In this case, checking with the m-cycles in the preceding positions of m^j has to be restarted from the beginning. In other words, the function finds the earliest possible burst launching time s^j from the MN for the m-burst along each m-cycle m^j. Here, L is the burst length, and p^i_{uv} and p^j_{uv} are the burst propagation delays along m-cycles m^i and m^j from the MN to node u, respectively. Note that p^i_{uv}, p^j_{uv}, and L are constants in function FindMaxDelay.

The fault localization latency T_{sq} for the given sequence sq is the maximum value of $(s^j + p^j_{uv} + L)$ provided by any m-cycle m^j. S_{sq} and T_{sq} are returned in line 11 of function FindMaxDelay.

Function FindMaxDelay(P, sq)

```
   Input: G(V, E), MN, and 𝔐
   Output: S_sq and T_sq
   begin
1      S_sq ← φ, m^j ← sq[0], s^j ← 0
2      S_sq ← s^j, T_sq ← s^j + tp^j + L
       foreach k ∈ {1 ... |sq|} do
3          m^j ← sq[k], s^j ← 0
4          foreach l ∈ {0 ... (k − 1)} do
5              m^i ← sq[l], get s^i from S_sq
               foreach unidirectional link (u, v) ∈ E do
                   if both m^i and m^j visit (u, v) then
6                      Get p^i_uv and p^j_uv from P
                       if (s^j + p^j_uv + L) > (s^i + p^i_uv) ∧ (s^j + p^j_uv) < (s^i + p^i_uv + L) then
7                          s^j ← (s^i + p^i_uv + L − p^j_uv)
8                          GoTo line 4
9          S_sq ← s^j
           if (s^j + tp^j + L) > T_sq then
10             T_sq ← s^j + tp^j + L
11     Return S_sq, T_sq
```

For the sequence of m-trails in sq, the helper function FindMaxDelay finds T_{sq} taking $\Omega(|E||\mathfrak{M}|^2)$ steps.

4.5.3 Numerical Results

We have implemented the heuristic algorithms for m-cycle allocation and m-burst scheduling described in subsections 4.5.1 and 4.5.2, respectively, in Java. We run the heuristic

methods in an Intel Centrino Duo, 1.00GB RAM, 2.00GHz, Windows XP Home Edition Version 2002 Service Pack 2 Toshiba Satellite T2500 laptop, keeping all the assumptions the same as those in the ILP experiments. In addition to all the networks used in the ILP experiments, we have tested the heuristic methods in networks CERNet, SmallNet, NSFNet and Bellcore. To deal with the degree-2 nodes, we have added links to CERNet, NSFNet and Bellcore networks: one link each to CERNet and Bellcore, and two links to NSFNet. The comparative performance of the heuristic and the ILP methods for single-link SRLG failure localization is shown in Table 4.6. The results from the joint ILP, the separate ILPs, and the heuristics are given in columns J, S, and H, respectively. The numbers of m-cycles derived by the heuristic and the ILP methods are almost the same. The fault localization latencies derived by the heuristic methods are reasonable.

Table 4.6: The Comparative Performance of the Heuristic and the ILP Methods

Networks	No. of SRLGs	Performance metrics					
		$\sum_j m^j$			T ms		
		J	S	H	J	S	H
4 nodes, 6 links	6	3	3	3	26	26	26
5 nodes, 8 links	8	4	4	4	26	26	26
6 nodes, 9 links	9	4	4	4	48	50	50
7 nodes, 12 links	12	6	4	5	48	54	50
8 nodes, 12 links	12	4	4	5	52	52	52
9 nodes, 14 links	14	5	5	5	50	68	50
CERNet + 1 link	17	–	–	6	–	–	72
SmallNet	22	–	–	7	–	–	106
NSFNet + 2 links	23	–	–	7	–	–	108
Bellcore + 1 link	29	–	–	10	–	–	98

4.6 Conclusions

This chapter deals with single-link SRLG fault localization and presents implementations of a novel framework of failure localization via out-of-band monitoring in all-optical WDM networks, called the monitoring burst (m-burst) framework. With the proposed m-burst framework, one supervisory WL channel in each unidirectional link traversed by any m-cycle is used for failure localization, and the monitoring node (MN) launches short-duration optical bursts along a set of cycles called m-cycles to probe their on-off status. The MN thus can achieve unambiguous failure localization (UFL) for the SRLGs under consideration, using significantly less monitoring resources than are necessary in the previously reported studies.

An ILP is formulated as a joint design of m-cycle allocation and derivation of launching times of the bursts along the m-cycles to minimize the fault localization latency and signaling overhead. As a preliminary study on the proposed framework, we have conducted numerical experiments on small networks. The results demonstrate the feasibility and easy implementation of the framework. We have also formulated two ILPs as separate optimization problems for m-cycle allocation and m-burst scheduling. The joint optimization method outperforms the separate optimization method to find the fault localization latency for 50% of the tested networks. As the ILP methods cannot be applied in large networks to get a solution in a reasonable time, we have also devised heuristic algorithms for m-cycle allocation and burst scheduling to find the near-optimal solution of the problems. The m-cycle allocation method is a randomized local search method. The burst scheduling method finds the earliest burst launching time from the MN for the burst along each m-cycle. The heuristic methods incur reasonable fault localization latency for single-link SRLG fault localization.

In the numerical experiments for the m-burst framework, bursts are kept non-overlapping throughout the network by manipulating the launching time of each m-burst from the MN in the ILP and heuristic methods. The burst launching time scheduling techniques developed in this chapter will be used directly or indirectly in the rest of this thesis. In particular, Tabu search based Algorithm 4.2 will be used for burst scheduling, with most of the m-trail allocation methods developed later on.

Once the m-trail allocation and burst scheduling problems are solved, we need to multiplex multiple bursts in each unidirectional link traversed by more than one m-cycle to avoid burst collisions completely. In this chapter, we have shown that the scheduling for node switch fabric configuration can be done from the burst launching times and m-cycle routes as simple and systematic calculations. Moreover, the switching schedule follows a fixed pattern in each monitoring period. Thus, instead of sending control packets ahead of each burst as in OBS, the switching schedule can be sent to the nodes separately from the bursts themselves. Moreover, the updates of the switching schedule can be sent after long intervals.

The methods developed in this chapter provide a proof of concept of the m-burst framework. We have shown that single-link SRLG faults in all-optical networks can be localized from a single MN with reasonable fault localization latency by using at most a single WL in each unidirectional link of the network. Hence, this work proves our thesis on single-link SRLG fault localization in all-optical mesh networks. We shall deal with single-link and multi-link SRLG fault localization in all-optical networks in the next chapter.

Chapter 5

M-Burst on Multi-Link SRLGs

5.1 Introduction

Fault management is a challenging task in all-optical networks. The Link Management Protocol (LMP) [33] proposed under the GMPLS framework employs sequential co-ordination between adjacent nodes along each lightpath. As a result, it is subject to long fault management delay and high protocol complexity. To mitigate the problems, link-based monitoring [2][34][64] has been considered in the literature, where every link is exclusively monitored via a single-hop supervisory lightpath (S-LP) launched with a constant optical signal.

More efficient fault localization methods are proposed that use a set of multi-hop S-LPs such as a monitoring cycle (m-cycle or MC) [1][64][70][72], monitoring trail (m-trail) [54][63], or monitoring tree (m-tree) [17][21][53]. A monitoring node (MN) that terminates an S-LP can obtain the failure status of the group of links traversed by the S-LP. A set of S-LPs is allocated such that each undirected link of a network is traversed by a unique subset of the S-LPs. Therefore, the network controller will be able to localize any shared risk link group (SRLG) failure unambiguously according to the collected alarms issued by the MNs.

The failure localization schemes based on multi-hop S-LPs generally take a large amount of wavelength links (WLs), especially when the number of considered SRLGs is large. For example, the method in [51] has indicated that it takes more than 10 WLs per link on average for the S-LPs to localize all the SLRG failures with up to 3 links. Such high monitoring resource consumption is not acceptable in some circumstances.

On the other hand, some studies suggest monitoring a set of S-LPs terminated at a common MN in order to completely remove the alarm dissemination/collection complexity

[1][2][52][61]. The method in [1] ensures that for each pair of SRLGs, there exists at least two S-LPs, with either each S-LP traversing only one SRLG, or one of them traversing one SRLG and another S-LP traversing both SRLGs. Moreover, each SRLG must be traversed by at least one S-LP. Thus, the number of required S-LPs is upper-bounded by $O(|\Psi|^2)$, where $|\Psi|$ is the number of SRLGs under consideration. The studies in [52][61] suggest that a node can obtain the on-off status of an S-LP by tapping the optical signal, by which the failure localization decision can be made locally at each node without taking any alarm dissemination overhead.

In Chapter 3, the monitoring bursts (m-bursts) framework is introduced for fault localization in all-optical mesh networks as a remedy to the observed problems. With m-burst framework, a set of S-LPs called m-trails is derived such that all of them originate from and terminate at a single MN, and each link is traversed by a unique subset of m-trails. Next, instead of launching persistent optical signals along each m-trail, short-duration optical bursts are sent from the MN along the m-trail to inspect the failure status of the m-trail. By synchronizing the switching fabric configuration of intermediate nodes of the m-trails in advance, each network link can reserve as few as a single WL along which multiple optical bursts can be multiplexed in the time domain. Accordingly, the MN follows a specific schedule of launching the burst for each m-trail in order to achieve collision-free probing for all the m-bursts during a monitoring period. The relevant problem formulation in detail for m-trail allocation and m-burst scheduling under the proposed m-burst framework is provided in Section 3.3 of Chapter 3.

In Chapter 4, the m-burst framework is applied for single-link SRLG fault localization using m-cycles. A single MN is traversed by a set of m-trails in the form of cycles called m-cycles. The MN launches optical bursts along the m-cycles to detect the on-off status of the m-cycles. Due to close-loop shape of an m-cycle, the launched optical bursts will be received by the MN if all the links along the m-cycle are working properly, but the burst will be lost if any link along the m-cycle fails. Like a conventional m-trail fault localization approach, the MN designates a set of m-cycles where each m-cycle provides a single bit in the alarm code based on whether the corresponding burst is received at the expected time instant or not. Since each m-burst is a short-duration optical flow, we no longer need to reserve statically a whole WL along each link of an m-cycle. Multiple optical bursts can be multiplexed in the time domain on the same WL, which significantly reduces the consumed monitoring resources. As a result, each link may consume as few as a single WL for fault localization to support multiple m-cycles traversing through the link. This is possible if provided with a proper burst scheduling mechanism at the MN, which manipulates the launching time of each optical burst such that burst collision at any link can be completely avoided. However, the study in Chapter 4 covers only single-link fault localization.

In this chapter[1], we extend the study in Chapter 4 by providing a complete study on the m-burst framework under a multi-link SRLG failure scenario. We develop comprehensive analysis for gaining deeper understanding of the m-trail allocation problem with a single MN. Given the maximum number of simultaneous faulty links d that can possibly occur in the network, the methods proposed in this chapter find a set of m-trails to identify any SRLG failure and determines the instants of launching m-bursts in order to avoid burst collision along any link of the network.

To minimize the failure localization latency, a new problem for m-trail allocation and burst scheduling is formulated in Section 5.2, in which a set of m-trails and the timing of launching the burst for each m-trail are jointly determined, in order to achieve the maximum parallelism of burst traversals. The basic idea of the proposed novel M-Trial Cover against the Faults (MCF) method is to ensure that in case of any SRLG failure, each remaining link is traversed by at least one m-trail that is disjoint from the faulty SRLG. The MCF method needs the number of m-trails in $O(|\Psi|)$. i.e., $O((|E|+1)^d)$, where d is the maximum number of links in any SRLG.

Obviously, with more m-trails, less parallelism can be achieved, causing longer failure localization latency. In Section 5.3, we improve the MCF m-trail allocation method of Section 5.2 by having no more than $(d+1)|E|$ m-trails in the solution. The basic idea of the proposed novel Disjoint path-based M-Trial Cover against the Faults (DMCF) method is to ensure that each link is traversed by $(d+1)$ otherwise link-disjoint m-trails, such that when any SRLG with up to d links fails, each remaining link is traversed by at least one m-trail that is disjoint from the faulty SRLG. This approach means a reduction of the asymptotic bound on the number of consumed m-trails from $O((|E|+1)^d)$ to $O((d+1)|E|)$, which is a significant improvement of the state-of-the-art in the considered scenario. Note that the reduction of m-trails not only reduces the monitoring delay under the proposed m-burst framework, but also greatly simplifies the network control and management.

The rest of the chapter is organized as follows. In Section 5.2, a novel multi-link SRLG fault localization method called MCF is proposed. Theoretical analysis of the MCF m-trail allocation method for multi-link SRLG fault localization is provided in subsection 5.2.1. In subsection 5.2.2, an ILP is formulated to solve the m-trail allocation and burst scheduling problems jointly. In subsection 5.2.3, a heuristic m-trail allocation algorithm to localize multi-link SRLG failures is provided. Numerical experiments are conducted and the results are given in subsection 5.2.4.

In Section 5.3, a novel disjoint path-based extension of MCF method called DMCF

[1]The content of this chapter has been published as a journal paper [7] and is accepted as a conference paper [6]

is proposed. Theoretical analysis of the DMCF m-trail allocation method for multi-link SRLG fault localization is given in subsection 5.3.1. In subsection 5.3.2, an ILP is formulated to solve the m-trail allocation problem. A heuristic m-trail allocation algorithm to localize multi-link SRLG failures is provided in subsection 5.3.3. Numerical experiments are conducted, and the results are given in subsection 5.3.4.

Section 5.4 concludes the Chapter.

5.2 MCF: A Multi-Link SRLG Fault Localization Method

This section[2] deals with multi-link SRLG fault localization in all-optical mesh networks. In fact, the m-burst framework is extended for multi-link SRLG fault localization. We propose a novel m-trail allocation scheme called the m-trail cover against the faults (MCF) for multi-link SRLG fault localization and provide theoretical justifications of the scheme. Specifically, the proposed m-trail allocation method ensures that any healthy link is traversed by at least one uninterrupted m-trail during an SRLG failure. As a proof of concept, we formulate the m-trail allocation and the burst scheduling problems as a joint optimization problem via an Integer Linear Program (ILP) and implement the method for all possible SRLGs with up to $d = 3$ links in any network. A heuristic method for the m-trail allocation scheme is proposed and implemented for multi-link SRLG fault localization. To derive burst launching times along the m-trails provided by the heuristic method, Algorithm 4.2, given in Chapter 4 for burst scheduling, is used. Numerical results for small networks show that the scheme is able to localize single-link and multi-link SRLG faults unambiguously with a very small amount of fault localization latency.

5.2.1 Theoretical Analysis

Let the sets of the SRLGs, the m-trails, and the m-trails traversing through an undirected link (u, v) be denoted as Ψ, $\mathfrak{M}$, and $\varphi_{(u,v)}$, respectively.

Lemma 5.2.1. *$\forall (u, v), (w, x) \in E$, if $\varphi_{(u,v)} \subset \varphi_{(w,x)}$ or $\varphi_{(w,x)} \subset \varphi_{(u,v)}$, multi-link SRLG faults involving (u, v) and (w, x) cannot be unambiguously localized from the MN by detecting only disrupted m-trails.*

[2]The content of this section has been published as a journal paper [7]

Proof. We use proof by contradiction. Assume that the lemma is false, i.e., in case of $\varphi_{(u,v)} \subset \varphi_{(w,x)}$ or $\varphi_{(w,x)} \subset \varphi_{(u,v)}$, multi-link SRLG faults involving (u, v) and (w, x) can be unambiguously localized from the MN by detecting only disrupted m-trails.

Let two SRLGs be $\psi_i = \{(u, v)\}$ and $\psi_j = \{(u, v), (w, x)\}$, and the sets of m-trails traversing links (u, v) and (w, x) be $\varphi_{(u,v)} = \{m^0, m^1, m^2\}$ and $\varphi_{(w,x)} = \{m^0, m^2\}$, respectively. Here, $\varphi_{(w,x)} \subset \varphi_{(u,v)}$. If the SRLG ψ_j becomes faulty, m-trails m^0, m^1 and m^2 will be disrupted. Detecting disrupted m-trails, the MN cannot determine whether ψ_i or ψ_j is faulty because both the SRLGs are traversed by m-trails m^0, m^1 and m^2. It is a contradiction. □

Corollary 5.2.2. *$\forall(u, v), (w, x) \in E$, if $\varphi_{(u,v)} \nsubseteq \varphi_{(w,x)}$ and $\varphi_{(w,x)} \nsubseteq \varphi_{(u,v)}$, single-link faults can be unambiguously localized from the MN by detecting only disrupted m-trails.*

Corollary 5.2.3. *$\forall(u, v) \in E$: (u, v) is not adjacent to the MN, at least two m-trails must traverse (u, v) to enable unambiguous localization of multi-link SRLG faults from the MN by detecting only disrupted m-trails.*

However, it is not sufficient that every link of a network is traversed by a unique set of m-trails to localize multi-link SRLG faults unambiguously. To show the fact, let two multi-link SRLGs be $\psi_i = \{(u, v), (w, x), (y, z)\}$ and $\psi_j = \{(u, v), (y, z)\}$, and the sets of m-trails traversing links (u, v), (w, x) and (y, z) be $\varphi_{(u,v)} = \{m^0, m^1\}$, $\varphi_{(w,x)} = \{m^1, m^2\}$ and $\varphi_{(y,z)} = \{m^2, m^3\}$, respectively. Here, $\varphi_{(u,v)}$, $\varphi_{(w,x)}$ and $\varphi_{(y,z)}$ are unique sets. Hence, (u, v), (w, x) or (y, z) is traversed by a unique set of m-trails. If the SRLG ψ_i becomes faulty, m-trails m^0, m^1, m^2 and m^3 will be disrupted. Detecting disrupted m-trails, the MN cannot determine whether multi-link SRLG ψ_i or multi-link SRLG ψ_j is faulty because both multi-link SRLGs are traversed by the same set of m-trails.

Lemma 5.2.4. *$\forall\psi_i, \psi_j \in \Psi$: $\psi_i \neq \psi_j$ and $|\psi_i| = |\psi_j| = l$, if the set of m-trails traversing ψ_i is not the same set of m-trails traversing ψ_j, i.e., $\varphi_{\psi_i} \neq \varphi_{\psi_j}$, any multi-link SRLG fault involving l links can be unambiguously localized from the MN by detecting only disrupted m-trails.*

Proof. From inspection, a multi-link SRLG fault involving l links will disrupt a unique set of m-trails because $\varphi_{\psi_i} \neq \varphi_{\psi_j}$. □

Lemma 5.2.5. *$\forall\psi_i, \psi_j \in \Psi$: $\psi_i \subset \psi_j$, if the set of m-trails traversing ψ_i is a proper subset of those traversing ψ_j, i.e., $\varphi_{\psi_i} \subset \varphi_{\psi_j}$, any multi-link SRLG faults involving either ψ_i or ψ_j can be unambiguously localized from the MN by detecting only disrupted m-trails.*

Proof. From inspection, at least one more m-trail traverses the superset SRLG. Thus the multi-link SRLG fault involving either ψ_i or ψ_j will disrupt different numbers of m-trails. □

To identify multi-link faults unambiguously, it is evident that a) m-trails traversing a link cannot be a subset of those traversing another link of the network, b) each SRLG with the same number of links should be traversed by a unique set of m-trails, and c) a set of links must be traversed by strictly less numbers of m-trails than those traversing its superset.

The proposed MCF m-trail allocation method ensures that during a failure event each healthy link of the network is traversed by at least one uninterrupted m-trail. Faulty links are identified from the MN by detecting disrupted m-trails. Now we prove that the set of m-trails identified as a solution by the proposed method has the above mentioned characteristics; as a consequence, the method has the capability to localize multi-link SRLG faults unambiguously.

Theorem 5.2.6. *$\forall\psi_i \in \Psi$, if each link in $E\backslash\psi_i$ is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_i, then*

i) m-trails traversing each link is not a subset of those traversing another link of the network;

ii) each SRLG having the same number of links are traversed by a unique set of m-trails; and

iii) m-trails traversing each SRLG is a proper subset of those traversing any of its superset SRLGs.

Proof. i) $\forall(u,v),(w,x) \in E$, let single-link SRLGs be $\psi_j = \{(u,v)\}$ and $\psi_k = \{(w,x)\}$. As each link in $E\backslash\psi_j$ including link (w,x) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_j, at least one m-trail that does not traverse ψ_j will traverse ψ_k because $(w,x) \in \psi_k$. Hence, $\varphi_{(w,x)} \nsubseteq \varphi_{(u,v)}$.

Similarly, as each link in $E\backslash\psi_k$ including link (u,v) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_k, at least one m-trail that does not traverse ψ_k will traverse ψ_j because $(u,v) \in \psi_j$. Hence, $\varphi_{(u,v)} \nsubseteq \varphi_{(w,x)}$.

ii) $\forall\psi_j,\psi_k \in \Psi$ where $\psi_j \neq \psi_k$ and $|\psi_j| = |\psi_k|$ indicates that there exists at least one link that is in ψ_j but not in ψ_k and vice versa. Assume that (u,v) and (w,x) are such links where $(u,v) \in \psi_j \wedge (u,v) \notin \psi_k$ and $(w,x) \in \psi_k \wedge (w,x) \notin \psi_j$.

As each link in $E \backslash \psi_j$ including link (w, x) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_j, at least one m-trail that does not traverse ψ_j will traverse ψ_k because $(w, x) \in \psi_k$. Hence, $\varphi_{\psi_k} \nsubseteq \varphi_{\psi_j}$.

Similarly, as each link in $E \backslash \psi_k$ including link (u, v) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_k, at least one m-trail that does not traverse ψ_k will traverse ψ_j because $(u, v) \in \psi_j$. Thus, $\varphi_{\psi_j} \nsubseteq \varphi_{\psi_k}$. Hence, ψ_j and ψ_k are traversed by unique sets of m-trails.

iii) $\forall \psi_j, \psi_k \in \Psi$: $\psi_j \subset \psi_k$. As $\psi_j \subset \psi_k$, all the m-trails that traverse ψ_j will also traverse ψ_k. Hence, $\varphi_{\psi_j} \subseteq \varphi_{\psi_k}$.

Again, $\psi_j \subset \psi_k$ indicates that there exists at least one link that is in ψ_k but not in ψ_j. Assume that (w, x) is such a link where $(w, x) \notin \psi_j \wedge (w, x) \in \psi_k$. As each link in $E \backslash \psi_j$ including link (w, x) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_j, at least one m-trail that does not traverse ψ_j will traverse ψ_k because $(w, x) \in \psi_k$. Thus, $|\varphi_{\psi_j}| < |\varphi_{\psi_k}|$. □

Theorem 5.2.7. *$\forall \psi_i \in \Psi$, if each link in $E \backslash \psi_i$ is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_i, single-link and multi-link SRLG faults can be unambiguously localized from the MN by detecting only disrupted m-trails.*

Proof. $\forall \psi_j, \psi_k \in \Psi$: $|\psi_j| \leq |\psi_k|$. Thus, ψ_j may or may not be a subset of ψ_k. But ψ_k can not be a subset of ψ_j. Thus, there arise two cases.

Case 1, $\psi_j \subseteq \psi_k$: In this case $|\psi_j|$ must be less than $|\psi_k|$, i.e., $\psi_j \subset \psi_k$. It implies that all the m-trails that traverse ψ_j will also traverse ψ_k. i.e., $\varphi_{\psi_j} \subseteq \varphi_{\psi_k}$. Moreover, $\exists (w, x) \in E : (w, x) \notin \psi_j \wedge (w, x) \in \psi_k$. As each link in $E \backslash \psi_j$ including link (w, x) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_j, we have $|\varphi_{\psi_j}| < |\varphi_{\psi_k}|$.

Case 2, $\psi_j \not\subset \psi_k$: In this case $|\psi_j|$ is less than or equal to $|\psi_k|$. In either situation $\exists (u, v), (w, x) \in E : (u, v) \in \psi_j \wedge (u, v) \notin \psi_k$ and $(w, x) \notin \psi_j \wedge (w, x) \in \psi_k$. As each link in $E \backslash \psi_k$ including link (u, v) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_k, we have $\varphi_{\psi_j} \nsubseteq \varphi_{\psi_k}$. Again, as each link in $E \backslash \psi_j$ including link (w, x) is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_j, we have $\varphi_{\psi_k} \nsubseteq \varphi_{\psi_j}$.

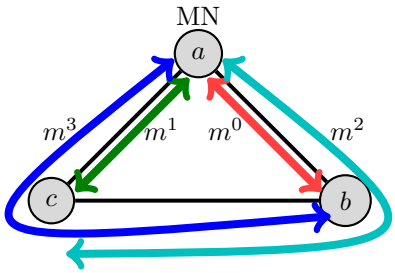
Thus, each SRLG is traversed by a unique set of m-trails. Hence, disrupted m-trails will identify the faulty SRLG. □

We observe that Theorem 5.2.7 is a sufficient but not a necessary condition for unambiguous localization of single-link and multi-link SRLG faults from the MN by detecting only disrupted m-trails. To show that the theorem is not a necessary condition, let two

SRLGs be ψ_i and ψ_j where $\psi_j \not\subset \psi_i$ and they are traversed by two sets of m-trails namely $\varphi_{\psi_i} = \{m^1, m^2, m^3\}$ and $\varphi_{\psi_j} = \{m^1, m^2\}$, respectively. Thus, ψ_i and ψ_j are traversed by unique sets of m-trails. Consequently, any fault involving either ψ_i or ψ_j can be unambiguously localized even though ψ_j is not traversed by at least one m-trail which is link-disjoint from ψ_i.

Example: Figure 5.1 shows a three link network to explain how the proposed multi-link SRLG fault localization method works. Node a is the MN. Five SRLGs are given as $\psi_1 = \{(a, b)\}$, $\psi_2 = \{(b, c)\}$, $\psi_3 = \{(c, a)\}$, $\psi_4 = \{(a, b), (b, c)\}$, $\psi_5 = \{(b, c), (c, a)\}$. As shown in Figure 5.1(a), four m-trails are needed to identify five SRLG failures. For each of the single-link SRLG ψ_1, ψ_2 and ψ_3 failure, the remaining two links are traversed by m^1 and m^3, m^0 and m^1, and m^0 and m^2 which are link-disjoint from ψ_1, ψ_2 and ψ_3, respectively. For each of the double link SRLG ψ_4 and ψ_5 failure, the remaining link is traversed by m^1 and m^0 which are link-disjoint from ψ_4 and ψ_5, respectively. Hence, $\forall \psi_i \in \Psi$, each link in $E \backslash \psi_i$ is traversed by at least one m-trail that is link-disjoint from the SRLG ψ_i.

Figure 5.1(b) shows the ACT derived from the m-trails. The decimal alarm codes are given in column *Dec.* of the ACT. The decimal codes of SRLGs ψ_1, ψ_2, ψ_3, ψ_4 and ψ_5 are 5, 12, 10, 13 and 14, respectively. Since each SRLG has a unique code, it indicates that each SRLG is traversed by a unique set of m-trails. Therefore, the MN will be able to localize SRLG faults unambiguously by detecting only disrupted m-trails.



(a) M-cycles in the solution set

SRLGs	m^3	m^2	m^1	m^0	Dec.
ψ_1	0	1	0	1	5
ψ_2	1	1	0	0	12
ψ_3	1	0	1	0	10
ψ_4	1	1	0	1	13
ψ_5	1	1	1	0	14

(b) Alarm code table (ACT)

Figure 5.1: M-trail solution set and the ACT for a network with 3 node and 3 link.

5.2.2 ILP Formulation for the MCF Method

With a single MN and a set of SRLGs Ψ where each $\psi \in \Psi$ consists of either single or multiple links of the network, the proposed ILP is a manipulation of Theorem 5.2.7 in such

a way that $\forall\psi \in \Psi$ each link in $E\backslash\psi$ is traversed by at least one m-trail that is link-disjoint from ψ while visiting the MN. Different from any previous research, the ILP determines the burst schedule and minimizes the monitoring period by manipulating the routes of m-trails where each SRLG is traversed by a unique set of m-trails.

To reduce the problem size, we use an enumerated set of m-trails as an input to the ILP. For each node of the network, k unique shortest cycles/trails (m-trails) are derived by using Suurballe's algorithm for shortest pairs of disjoint paths [50] and Yen's algorithm for k-shortest loop-less paths [69]; each enumerated m-trail traverses the node and the MN.

List of Notation

G The network, $G = (V, E)$, where V is the set of nodes and E is the set of unidirectional links of the network.

MN The monitoring node in the network, $MN \in V$.

r_1, r_2 Predefined cost ratios.

N A large predefined number that represents ∞.

J Total number of enumerated m-trails.

ee^j_{uv} It is equal to 1 if the jth enumerated m-trail traverses unidirectional link (u, v) and 0 otherwise.

es^j_ψ It is equal to 1 if the jth enumerated m-trail traverses any link of SRLG ψ, and 0 otherwise.

des^{jab}_ψ It is equal to 1 if $es^j_\psi == 0 \wedge (ee^j_{ab} == 1 \vee ee^j_{ba} == 1)$ where undirected link $(a, b) \in E\backslash\psi$, and 0 otherwise.

p^{jr}_k Pre-calculated propagation delay in ms of the m-burst from the MN to the kth node of the jth enumerated m-trail following the given or reverse spin sp^{jr}.

tp^j Pre-calculated total propagation delay in ms of the m-burst along the jth enumerated m-trail starting from and returning to the MN.

m^j Binary variable, it is equal to 1 if the jth enumerated m-trail is in an ILP solution and 0 otherwise.

sp^{jr} Binary variables called spins of the jth enumerated m-trail. As propagation delays from the *MN* to on-trail nodes depend on the burst flow direction along an m-trail, two spin variables per enumerated m-trail are used to keep track of propagation delays properly. It can be assigned a non-zero value only if the enumerated m-trail is in an ILP solution. Given spin sp^{j0} is equal to 1 if the jth enumerated m-trail that is in an ILP solution traverses its on-trail links in the enumerated direction and 0 otherwise. Similarly, reverse spin sp^{j1} is equal to 1 if the jth enumerated m-trail that is in an ILP solution traverses its on-trail links against the enumerated direction and 0 otherwise.

e^{j}_{uv} Binary variable, it is equal to 1 if the jth m-trail in an ILP solution traverses unidirectional link (u, v) with spin sp^{j0} or unidirectional link (v, u) with spin sp^{j1}, and 0 otherwise.

g^{irjt}_{uv} Binary variable, it is equal to 1 if the m-burst along the ith m-trail in an ILP solution with spin sp^{ir} arrives to node u before the m-burst along the jth m-trail in the ILP solution with spin sp^{jt} and both the bursts traverse unidirectional link (u, v), and 0 otherwise. It helps collision-free burst propagation through network links.

s^{jr} Variable m-burst launching time in ms from the *MN* along the jth enumerated m-trail. It can be assigned a non-zero value only if the jth enumerated m-trail with spin sp^{jr} is in an ILP solution.

T Variable maximum fault localization latency in ms.

n Integer variable, it is the maximum number of m-trails in an ILP solution traversed through any unidirectional link of the network.

α_{ψ} Integer variable, decimal SRLG code of ψ.

ILP for M-Trail Allocation and M-Burst Launching Time Scheduling

The specific ILP formulation is provided as follows.

Objective:

$$Minimize \ \{T + r_1 * n + r_2 * \sum_{j} m^j\} \tag{5.1}$$

Subject to the following constraints:

At least one enumerated m-trail that is link-disjoint from SRLG ψ but traverses undirected link (a, b) must be in an ILP solution. This is the key constraint that implements the proposed MCF m-trail allocation method for multi-link SRLG fault localization.

$$\sum_{j} m^j * des_{\psi}^{jab} \geq 1, \quad \forall \psi \in \Psi, \forall (a,b) \in E \backslash \psi \tag{5.2}$$

Each enumerated m-trail in an ILP solution has only one non-zero spin.

$$sp^{j0} + sp^{j1} = m^j, \qquad \forall j \tag{5.3}$$

The constraint ensures that positive burst starting times can be assigned only to the bursts along enumerated m-trails that are in an ILP solution with valid spins.

$$s^{jr} \leq sp^{jr} * N, \qquad \forall j, r \tag{5.4}$$

Constraints 5.5 and 5.6 enforce minimum difference between arrival times of two m-bursts to node u to the burst length L when the m-bursts along the ith and jth enumerated m-trails traverse the same outgoing link (u, v) from the node. Thus, the constraints ensure collision free burst propagation through any link of the network. Here, node u is the kth node of the ith enumerated m-trail considering spin sp^{ir} and the lth node of the jth enumerated m-trail considering spin sp^{jt}. Moreover, node v is the next hop node along the ith and jth enumerated m-trails considering their spins. Now $\forall i, r, k, j, t, l, (u, v)$,

$$(s^{ir} + p_k^{ir}) - (s^{jt} + p_l^{jt}) \geq (1 - (N+1) * g_{uv}^{irjt}) * L + (sp^{ir} + sp^{jt} - 2) * N \tag{5.5}$$

$$(s^{jt} + p_l^{jt}) - (s^{ir} + p_k^{ir}) \geq (g_{uv}^{irjt} + N * (g_{uv}^{irjt} - 1)) * L + (sp^{ir} + sp^{jt} - 2) * N \tag{5.6}$$

The maximum fault localization latency is the largest elapse time for any m-burst with valid spin from the start of the monitoring period to the return time of the last bit of the m-burst to the MN.

$$T \geq s^{jr} + m^j * tp^j + L + (sp^{jr} - 1) * N, \quad \forall j, r \tag{5.7}$$

Link traversal by m-trails in an ILP solution considering their spins

$$e_{uv}^j \geq m^j * ee_{uv}^j + sp^{j0} - 1, \quad \forall (u,v) \in E \quad \forall j \tag{5.8}$$

$$e^j_{vu} \geq m^j * ee^j_{uv} + sp^{j1} - 1, \quad \forall (u,v) \in E \quad \forall j \tag{5.9}$$

Calculation of the maximum number of m-trails traversing through any unidirectional link of the network

$$n \geq \sum_j e^j_{uv}, \quad \forall (u,v) \in E \tag{5.10}$$

The following constraint is for the calculation of decimal codes, but it is not a constraint of the ILP per se. However, using the codes, we can verify if the method derives a unique code for each SRLG or not.

$$\alpha_\psi = \sum_j 2^j * es^j_\psi, \quad \forall \psi \in \Psi \tag{5.11}$$

Objective 5.1 aims to minimize mainly the fault localization latency. In order to find the fault localization latency, the number of m-trails traversing through any unidirectional link of the network and the total number of m-trails $|\mathfrak{M}|$ are also minimized.

From Eq. 5.2, we have the number of constraints in the order of $O(|E||\Psi|)$ where $|E|$ is the number of network links and $|\Psi|$ is the number of SRLGs under consideration. As $|\Psi| \leq (|E|+1)^d$, $O(|E||\Psi|)$ is equivalent to $O(|E|(|E|+1)^d)$ where d is the maximum number of links in any SRLG. From Eq. 5.11, we have the number of variables in the order of $O(|\Psi|)$ based on the variable α_ψ.

5.2.3 Heuristic Algorithm for M-Trail Allocation

The ILP is not scalable to the problem size; thus, a heuristic m-trail allocation algorithm for multi-link SRLG fault localization is proposed in Algorithm 5.1. The set of m-trails provided by the MCF m-trail allocation method is then used as the input of the burst scheduler Algorithm 4.2, given in Chapter 4, in order to find burst launching times from the MN and fault localization latency.

All links are considered undirected in Algorithm 5.1. In line 1, the m-trail solution set $\mathfrak{M}$ and ACT $\mathfrak{A}$ are initialized as empty. In lines 2-3, an m-trail is found for each node of the network G based on the node's shortest path to the MN. In lines 4-5, the shorter m-trail based on shortest distances $d(u)$ and $d(v)$ is selected from the two m-trails that

terminate at the end nodes u and v of each link (u, v). Then the link is concatenated to the m-trail as its last link to derive a new m-trail. The new m-trail is added to $\mathfrak{M}$ and $\mathfrak{A}$ in line 6. In the outer *for* loop, each SRLG $\psi \in \Psi$ is considered in turn. In line 7, each m-trail that traverses any link of $\psi \in \Psi$ is deleted from $\mathfrak{A}$ to form a temporary ACT $\mathfrak{A}_\psi$. Then for each link not in the SRLG ψ and if the alarm code of the link becomes 0 in $\mathfrak{A}_\psi$ (i.e., $\mathfrak{A}_\psi^{ab} == 0$), the link is added to an array Q_ψ in line 8. Next, for each link in Q_ψ if the alarm code of the link remains 0 in $\mathfrak{A}_\psi$, a new m-trail is found in line 9 that traverses the link but is link-disjoint from the SRLG ψ using the helper function FindMTrail. Finally, the new m-trail is added to $\mathfrak{M}$, $\mathfrak{A}$ and $\mathfrak{A}_\psi$ in line 10. In line 11, the set of m-trails $\mathfrak{M}$ as the solution of the fault localization problem and ACT $\mathfrak{A}$ are returned.

Algorithm 5.1: The MCF M-Trail Allocation Method

```
   Input: G(V, E), MN, and d
   Output: 𝔐 and 𝔄
   begin
1      Initialize 𝔐, 𝔄 ← ϕ
2      Find shortest distances from the MN in G
       foreach u ∈ V do
3          Build an m-trail m_u^i based on the shortest path p(u) from the MN
       foreach (u, v) ∈ E do
           if d(u) ≤ d(v) then
4              Build m-trail m^j with (u, v) and m_u^i
           else
5              Build m-trail m^j with (u, v) and m_v^i
6          𝔐, 𝔄 ← m^j
       foreach ψ ∈ Ψ do
           foreach m^j ∈ 𝔐: m^j ∈ φ_ψ do
7              Delete m^j from 𝔄 to form 𝔄_ψ
           foreach link (a, b) ∈ E\ψ: 𝔄_ψ^ab == 0 do
8              Q_ψ ← (u, v)
           foreach link (a, b) ∈ Q_ψ: 𝔄_ψ^ab == 0 do
9              m^j ← FindMTrail( 𝔄_ψ, ψ, a, b)
10             𝔐, 𝔄, 𝔄_ψ ← m^j
11     Return 𝔐, 𝔄
```

Function FindMTrail finds a new m-trail that traverses link (a, b) but is link-disjoint from SRLG ψ. In lines 1-2, the ordered list $\mathfrak{L}$ stores nodes a and b in ascending order of the lengths of their shortest paths $p(a)$ and $p(b)$ from the MN, respectively. The first node

in $\mathfrak{L}$ will be used to find the new m-trail before the second one. In the inner *for* loop, neighborhoods of both nodes are checked in turn. If any neighboring node v of node u is visited by an m-trail that is link-disjoint from SRLG ψ, the new m-trail is derived by using the portion of the m-trail from the MN to node v and links (u, v) and (a, b). If such an m-trail is not found in the outer *for* loop, link (a, b) and all links of SRLG ψ are deleted from G to derive a truncated network G_ψ in line 7. Then, two shortest paths $p(a)$ and $p(b)$ from the MN to the nodes a and b, respectively are found in G_ψ. The new m-trail m^j is derived by using link (a, b) and the shorter of the two shortest paths $p(a)$ and $p(b)$. In line 11, m-trail m^j is returned.

Function FindMTrail($\mathfrak{A}_\psi$, ψ, a, b)

Input: $G(V, E)$, MN, and $\mathfrak{M}$
Output: m^j
begin
1 Build an ordered list $\mathfrak{L} = [a, b]$
if $d(a) > d(b)$ **then**
2 Swap positions of a and b in $\mathfrak{L}$
foreach $u \in \mathfrak{L}$ **do**
foreach $v \in N(u)$ *and* $w \in N(v)$ **do**
if $v \neq a \wedge v \neq b \wedge w \neq a \wedge w \neq b \wedge \mathfrak{A}_\psi^{vw} \geq 1$ **then**
3 Find any m-trail m^i that visits v
4 Get the portion m_v^i of the m-trail from the MN to v
5 Add (a, b) and (u, v) to m_v^i to form an m-trail m^j
6 Return m^j
7 $G_\psi \leftarrow$ delete (a, b) and each $(u, v) \in \psi$ from G
8 Find shortest distances from the MN in G_ψ
if $d(a) \leq d(b)$ **then**
9 Build m-trail m^j based on the shortest path $p(a)$ from the MN and (a, b)
else
10 Build m-trail m^j based on the shortest path $p(b)$ from the MN and (a, b)
11 Return m^j

To find an ordered list of 2 nodes based on the shortest path, the helper function FindMTrail requires at most $O(|V| \log_2 |V| + |E|)$ steps via Dijkstra's algorithm. The outer *for* loop iterates twice only: one for each end node of link (a, b). For each node, it requires $O(1)$ steps to check the neighborhood of the node for uninterrupted m-trails. It takes at most $O(\log_2 |\mathfrak{M}|)$ steps to search an uninterrupted m-trail. Thus, the outer *for* loop needs $O\big(d \log_2(|E| + 1)\big)$ steps as $|\mathfrak{M}| \in O(|\Psi|)$ shown below and $|\Psi| \leq (|E| + 1)^d$, where d is

the maximum number of simultaneous faulty links. Line 7 needs O(1) steps and line 8 requires at most $O(|V|\log_2|V|+|E|)$ steps via Dijkstra's algorithm. To build an m-trail $O(E)$ steps are required. Thus, complexity of the helper function is $O(|V|\log_2|V|+|E|)$.

The overall worst-case complexity of the method is derived as follows. Line 2 of Algorithm 5.1 requires $O(|V|\log_2|V|+|E|)$ steps. Line 3 needs $O(E)$ steps. Lines 4-6 needs requires $O(|E|)$ steps. The outer *for* loop iterates $O(|\Psi|)$ times. Deleting m-trails from $\mathfrak{A}$ needs $O(|E|)$ steps. To populate Q_ψ requires $|E|$ steps. As each SRLG can disrupt a small number of m-trails, the number of links that are not being traversed by at least one uninterrupted m-trail will also be small. Thus, $|Q_\psi|$ can be considered as constant. To find a new m-trail for a link in Q_ψ requires complexity of $O(|V|\log_2|V|+|E|)$. Thus, the outer *for* loop takes the worst-case complexity of $O\big(|\Psi|(|V|\log_2|V|+|E|)\big)$.

Next, we derive asymptotic upper bound of the number of m-trails denoted as $|\mathfrak{M}|$. In line 6 of Algorithm 5.1, $|E|$ m-trails are added to the solution. In line 10, upper bound of m-trails to cover links in Q_ψ while disjoint from any link of ψ is $O(1)$ because a small number of m-trails can be devised to cover all such links. Thus, we have $|\mathfrak{M}| \in O(|\Psi|)$, compared with $|\mathfrak{M}| \in O(|\Psi|^2)$ in [1].

5.2.4 Numerical Results

We have developed a model of the ILP described in Section 5.2.2 in the modeling language GNU MathProg and translate the MathProg model using the GLPK solver *glpsol* into an MPS file to run the program in ILOG CPLEX 11.1.

We assume that the burst length L is 20 ms, burst propagation delay through any link (u, v) is 2 ms, and there is at most one supervisory WL along each unidirectional link in the numerical experiment. We also assume that r_1 is 0.1 and r_2 is 0.01. As minimization of n helps to minimize the fault localization latency indirectly, r_1 is assigned a low value. Again, as WL consumption is almost fixed for a network and bursts are launched from and returned to the MN, minimization of the number of m-trails turns into a secondary issue. Thus, r_2 is assigned even a lower value than that of r_1.

We conducted the experiment on a network with 7 nodes and 12 links. Node 0 is assigned as the MN. An SRLG could consist of up to $d = 3$ links, where each SRLG with two or three links is node-disjoint from the MN. In our case, there are in total 96 SRLGs consisting of 12 single-link, 28 dual-link and 56 triple-link SRLGs as shown in Table 5.1.

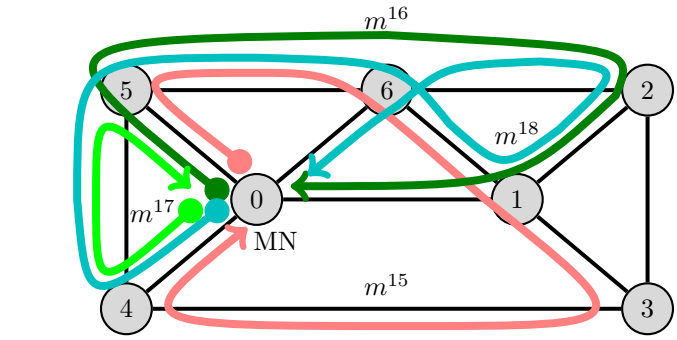
As shown in Figure 5.2, the ILP solution has $|\mathfrak{M}| = \sum_j m^j = 19$ m-trails. The set of m-trails populates the network ACT, which is the mapping between each possible SRLG code

Table 5.1: List of SRLGs in a Network with 7 Nodes and 12 Links

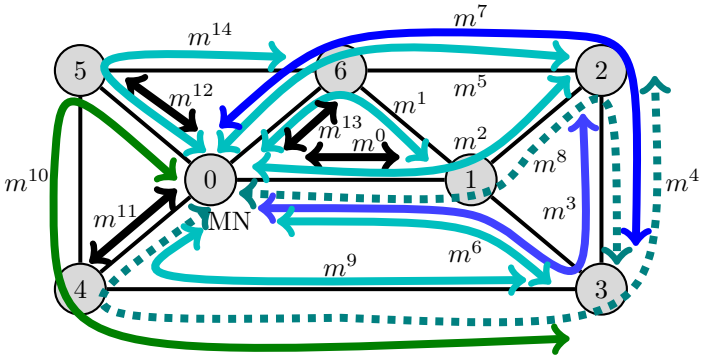
(0 1)	(1 3)(5 6)	(1 2)(1 6)(3 4)	(1 3)(2 6)(5 6)
(0 4)	(1 6)(2 3)	(1 2)(1 6)(4 5)	(1 3)(3 4)(4 5)
(0 5)	(1 6)(2 6)	(1 2)(1 6)(5 6)	(1 3)(3 4)(5 6)
(0 6)	(1 6)(3 4)	(1 2)(2 3)(2 6)	(1 3)(4 5)(5 6)
(1 2)	(1 6)(4 5)	(1 2)(2 3)(3 4)	(1 6)(2 3)(2 6)
(1 3)	(1 6)(5 6)	(1 2)(2 3)(4 5)	(1 6)(2 3)(3 4)
(1 6)	(2 3)(2 6)	(1 2)(2 3)(5 6)	(1 6)(2 3)(4 5)
(2 3)	(2 3)(3 4)	(1 2)(2 6)(3 4)	(1 6)(2 3)(5 6)
(2 6)	(2 3)(4 5)	(1 2)(2 6)(4 5)	(1 6)(2 6)(3 4)
(3 4)	(2 3)(5 6)	(1 2)(2 6)(5 6)	(1 6)(2 6)(4 5)
(4 5)	(2 6)(3 4)	(1 2)(3 4)(4 5)	(1 6)(2 6)(5 6)
(5 6)	(2 6)(4 5)	(1 2)(3 4)(5 6)	(1 6)(3 4)(4 5)
(1 2)(1 3)	(2 6)(5 6)	(1 2)(4 5)(5 6)	(1 6)(3 4)(5 6)
(1 2)(1 6)	(3 4)(4 5)	(1 3)(1 6)(2 3)	(1 6)(4 5)(5 6)
(1 2)(2 3)	(3 4)(5 6)	(1 3)(1 6)(2 6)	(2 3)(2 6)(3 4)
(1 2)(2 6)	(4 5)(5 6)	(1 3)(1 6)(3 4)	(2 3)(2 6)(4 5)
(1 2)(3 4)	(1 2)(1 3)(1 6)	(1 3)(1 6)(4 5)	(2 3)(2 6)(5 6)
(1 2)(4 5)	(1 2)(1 3)(2 3)	(1 3)(1 6)(5 6)	(2 3)(3 4)(4 5)
(1 2)(5 6)	(1 2)(1 3)(2 6)	(1 3)(2 3)(2 6)	(2 3)(3 4)(5 6)
(1 3)(1 6)	(1 2)(1 3)(3 4)	(1 3)(2 3)(3 4)	(2 3)(4 5)(5 6)
(1 3)(2 3)	(1 2)(1 3)(4 5)	(1 3)(2 3)(4 5)	(2 6)(3 4)(4 5)
(1 3)(2 6)	(1 2)(1 3)(5 6)	(1 3)(2 3)(5 6)	(2 6)(3 4)(5 6)
(1 3)(3 4)	(1 2)(1 6)(2 3)	(1 3)(2 6)(3 4)	(2 6)(4 5)(5 6)
(1 3)(4 5)	(1 2)(1 6)(2 6)	(1 3)(2 6)(4 5)	(3 4)(4 5)(5 6)

and the corresponding SRLG. The ACT is shown in Table 5.2. Each SRLG has an entry in the ACT of the network. An SRLG code is determined by the m-trails traversing through the SRLG. For example, link $(1, 2)$ is traversed by m-trails m^2, m^8, m^{16} and m^{18}, thus its code is 101 0000 0001 0000 0100. The corresponding decimal code is 327940. As link $(3, 4)$ is traversed by m-trails m^4, m^9, m^{10} and m^{15}, its code is 000 1000 0110 0001 0000. The corresponding decimal code is 34320. Similarly, link $(5, 6)$ is traversed by m-trails m^{14}, m^{15}, m^{16} and m^{18}, hence its code is 101 1100 0000 0000 0000. The corresponding decimal code is 376832. The code of SRLG $\psi = \{(1, 2), (3, 4), (4, 5)\}$ is the bit-wise `OR` of its link codes, i.e., 101 1100 0111 0001 0100. The corresponding decimal code is 378644.

When an SRLG fails, a unique set of m-trails will be disrupted, which results in a unique alarm code. For example, if m^2, m^5, m^7, m^8, m^{10}, m^{16}, m^{17} and m^{18} are disrupted, the generated alarm code will be 111 0000 0101 1010 0100. The corresponding decimal alarm code is $2^{18} + 2^{17} + 2^{16} + 2^{10} + 2^8 + 2^7 + 2^5 + 2^2 = 460196$. The MN can identify the SRLG consists of links $(1, 2)$, $(2, 6)$ and $(4, 5)$ as the failed SRLG from the ACT in Table



(a) M-cycles in the solution set



(b) M-trails in the solution set

Figure 5.2: M-trail solution set for a network with 7 nodes and 12 links.

5.2 using 460196 as an index.

The solution allows at most $n = 6$ m-trails through any unidirectional link of the network.

The starting time of the bursts s^j, total propagation delay through the m-trail tp^j, burst length L, and fault localization latency T^j for the burst along the jth m-trail is shown in Table 5.3. Maximum fault localization latency T is 126 ms. Thus, T is derived in the solution from m-trail m^2, m^7, m^9 or m^{14}.

Due to space constraint, Figure 5.3 shows collision-free m-burst traversal timing through only links from the MN to node 3 via nodes 1 and 2, and back to the MN using the same links during a monitoring period. Each rectangle represents an m-burst and the m-burst is labeled with the corresponding m-trail. The traversal period of an m-burst through a link is equal to the burst propagation delay δ through the link and the burst length L. Thus,

Table 5.2: Alarm Code Table (ACT)

Dec.	SRLG	Dec.	SRLG	Dec.	SRLG	Dec.	SRLG
408	(2 3)	360680	(1 3)(2 6)	376992	(2 6)(5 6)	427480	(1 3)(2 3)(4 5)
32840	(1 3)	360682	(1 3)(1 6)(2 6)	376994	(1 6)(2 6)(5 6)	427536	(3 4)(4 5)
33240	(1 3)(2 3)	360710	(1 2)(1 6)	377064	(1 3)(2 6)(5 6)	427538	(1 6)(3 4)(4 5)
34320	(3 4)	360780	(1 2)(1 3)	377092	(1 2)(5 6)	427608	(1 3)(3 4)(4 5)
34392	(1 3)(3 4)	360782	(1 2)(1 3)(1 6)	377094	(1 2)(1 6)(5 6)	427928	(2 3)(3 4)(4 5)
34712	(2 3)(3 4)	360862	(1 2)(1 6)(2 3)	377164	(1 2)(1 3)(5 6)	428560	(0 4)
34776	(1 3)(2 3)(3 4)	360870	(1 2)(1 6)(2 6)	377240	(2 3)(5 6)	459936	(2 6)(4 5)
65869	(0 1)	360890	(1 6)(2 3)(2 6)	377242	(1 6)(2 3)(5 6)	460036	(1 2)(4 5)
250880	(0 5)	360924	(1 2)(1 3)(2 3)	377244	(1 2)(2 3)(5 6)	460188	(1 2)(2 3)(4 5)
270498	(0 6)	360940	(1 2)(1 3)(2 6)	377252	(1 2)(2 6)(5 6)	460196	(1 2)(2 6)(4 5)
294914	(1 6)	360952	(1 3)(2 3)(2 6)	377272	(2 3)(2 6)(5 6)	460216	(2 3)(2 6)(4 5)
294986	(1 3)(1 6)	362160	(2 6)(3 4)	377304	(1 3)(2 3)(5 6)	492706	(1 6)(2 6)(4 5)
295322	(1 6)(2 3)	362162	(1 6)(2 6)(3 4)	378384	(3 4)(5 6)	492776	(1 3)(2 6)(4 5)
295386	(1 3)(1 6)(2 3)	362232	(1 3)(2 6)(3 4)	378386	(1 6)(3 4)(5 6)	492806	(1 2)(1 6)(4 5)
296466	(1 6)(3 4)	362260	(1 2)(3 4)	378456	(1 3)(3 4)(5 6)	492876	(1 2)(1 3)(4 5)
296538	(1 3)(1 6)(3 4)	362262	(1 2)(1 6)(3 4)	378544	(2 6)(3 4)(5 6)	493232	(2 6)(3 4)(4 5)
296858	(1 6)(2 3)(3 4)	362332	(1 2)(1 3)(3 4)	378644	(1 2)(3 4)(5 6)	493332	(1 2)(3 4)(4 5)
327840	(2 6)	362396	(1 2)(2 3)(3 4)	378776	(2 3)(3 4)(5 6)	508928	(4 5)(5 6)
327940	(1 2)	362420	(1 2)(2 6)(3 4)	394240	(4 5)	508930	(1 6)(4 5)(5 6)
328092	(1 2)(2 3)	362424	(2 3)(2 6)(3 4)	394648	(2 3)(4 5)	509000	(1 3)(4 5)(5 6)
328100	(1 2)(2 6)	376832	(5 6)	427010	(1 6)(4 5)	509088	(2 6)(4 5)(5 6)
328120	(2 3)(2 6)	376834	(1 6)(5 6)	427080	(1 3)(4 5)	509188	(1 2)(4 5)(5 6)
328124	(1 2)(2 3)(2 6)	376904	(1 3)(5 6)	427082	(1 3)(1 6)(4 5)	509336	(2 3)(4 5)(5 6)
360610	(1 6)(2 6)	376906	(1 3)(1 6)(5 6)	427418	(1 6)(2 3)(4 5)	509456	(3 4)(4 5)(5 6)

the period is $\delta + L = 2 + 20 = 22$ ms.

Table 5.3: Monitoring Delays along the M-Trails

m^j	$s^j + tp^j + L = T^j$	m^j	$s^j + tp^j + L = T^j$
0	$0 + 4 + 20 = 24$	10	$34 + 12 + 20 = 64$
1	$0 + 8 + 20 = 28$	11	$40 + 4 + 20 = 64$
2	$98 + 8 + 20 = 126$	12	$74 + 4 + 20 = 98$
3	$32 + 12 + 20 = 64$	13	$24 + 4 + 20 = 48$
4	$0 + 12 + 20 = 32$	14	$98 + 8 + 20 = 126$
5	$58 + 8 + 20 = 86$	15	$54 + 12 + 20 = 86$
6	$78 + 8 + 20 = 106$	16	$14 + 10 + 20 = 44$
7	$94 + 12 + 20 = 126$	17	$20 + 6 + 20 = 46$
8	$54 + 12 + 20 = 86$	18	$72 + 14 + 20 = 106$
9	$98 + 8 + 20 = 126$		

The periods when two m-bursts traverse through a link back-to-back are indicated by black rectangles in Figure 5.3. The m-bursts along m-trails m^6 and m^2, m^{18} and m^2, and

m^3 and m^8 traverse links (0, 1), (1, 2) and (2, 3), respectively, back-to-back. Similarly, the m-bursts along m-trails m^0 and m^{16}, m^{16} and m^3, m^8 and m^6, and m^6 and m^2 traverse link (1, 0) back-to-back. Each black rectangle in the figure spans 2 ms, which is the same as the burst propagation delay through a link. It is obvious that whenever two m-bursts traverse a link back-to-back, they remain collision-free in the link. On the other hand, the m-burst along m-trail m^8 starts traversing link (0, 1) or (1, 0) once the m-burst along m-trail m^3 completed traversal of link (0, 1) or (1, 0), respectively. Thus, the m-bursts along m-trails m^3 and m^8 remain 2 ms apart in links (0, 1) and (1, 0).

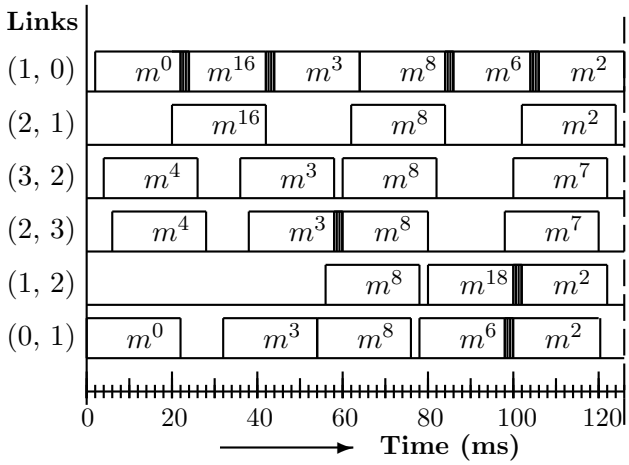


Figure 5.3: Collision-free link traversal by multiple m-bursts.

To verify the proposed algorithm, we run the ILP on three more networks with an arbitrarily chosen MN in each network while keeping all other assumptions the same as those in the above experiment.

The simulation results in terms of the number of m-trails $\sum_j m^j$, the maximum number of m-trails n through any unidirectional link, and the maximum fault localization latency T of the solutions are given in Table 5.4. The results for single-link SRLG fault localization are taken from Section 4.3. The results for single-link and multi-link SRLG fault localization are shown in columns S and M, respectively.

For each of the four networks, the method is able to identify a set of m-trails to achieve unambiguous failure localization (UFL) for the multi-link SRLGs failures. Maximum fault localization latency T for each network is optimal. Maximum number of m-trails n through any unidirectional link for each network is low. For each network, the method is able to achieve collision-free burst scheduling by keeping the m-bursts along the m-trails non-overlapping through any link of the network.

By taking the single-link SRLG result as reference, when multi-link SRLGs are considered and increased by 1.67, 2.25, 4.89, and 8 times, the number of m-trails is increased by 3, 3, 3.75, and 3.17 times, the maximum number of m-trails traversing through any unidirectional link is increased by 3, 3, 2.5, and 3 times, and the maximum fault localization latency is increased by 2.62, 2.54, 2.42, and 2.63 times, respectively. It shows that the proposed approach is scalable to the number of SRLGs.

Table 5.4: The Comparative Performance in Additional Networks

Performance metrics	Networks							
	4 nodes 6 links		5 nodes 8 links		6 nodes 9 links		7 nodes 12 links	
	S	M	S	M	S	M	S	M
$\sum_j m^j$	3	9	4	12	4	15	6	19
n	1	3	1	3	2	5	2	6
$T \ \ ms$	26	68	26	66	48	116	48	126

The proposed heuristic m-trail allocation method is implemented on an Intel Centrino Duo, 1.00GB RAM, 2.00GHz, Windows XP Home Edition Version 2002 Service Pack 2 Toshiba Satellite T2500 laptop. In addition to all the networks used in the ILP experiments, we have tested the heuristic method on 4 larger topologies: CERNet, SmallNet, NSFNet and Bellcore. We have modified the topologies of NSFNet by adding two links and Bellcore by adding one link. For all the networks, all the SRLGs with up to arbitrary three links are considered, where each SRLG with two or three links is node-disjoint from the MN. Once a set of m-trails is derived by the m-trail allocation method for a network, the burst scheduling is done in the network using Algorithm 4.2 of subsection 4.5.2 to derive burst launching times and fault localization latency T with $\delta = 2$ ms and $L = 20$ ms. The result is shown in column H in Table 5.5. The number of m-trails derived by the heuristic is approximately similar to that found with the ILP for multi-link SRLG fault localization. The method minimizes the maximum failure localization latencies.

5.3 DMCF: A Disjoint Path-Based Extension of the MCF Method

This section[3] also deals with multi-link SRLG fault localization in all-optical mesh networks. We introduce a novel m-trail allocation scheme called the Disjoint path-based

[3]The content of this section is accepted as a conference paper [6]

Table 5.5: The Comparative Performance of the Heuristic and the ILP MCF Methods

Networks	No. of SRLGs			Performance metrics					
				$\sum_j m^j$			T ms		
	S	M	H	S	M	H	S	M	H
4 nodes, 6 links	6	10	10	3	9	8	26	68	94
5 nodes, 8 links	8	18	18	4	12	11	26	66	94
6 nodes, 9 links	9	44	44	4	15	15	48	116	138
7 nodes, 12 links	12	96	96	6	19	20	48	126	152
8 nodes, 12 links	12	132	132	4	–	27	52	–	228
9 nodes, 14 links	14	179	179	5	–	24	50	–	164
CERNet	16	236	236	–	–	32	–	–	258
SmallNet	22	1162	1162	–	–	64	–	–	502
NSFNet + 2 links	23	1353	1353	–	–	98	–	–	876
Bellcore + 1 link	29	2053	2053	–	–	86	–	–	742

M-trail Cover against the Faults (DMCF) under the m-burst framework for achieving local unambiguous failure localization (L-UFL), in which a single monitoring node (MN) can localize any failure on a shared risk link group (SRLG) with up to d links by inspecting the optical bursts traversing through it. The DMCF method is an extension of the MCF method given in Section 5.2.

Specifically, the proposed MCF m-trail allocation method requires that any healthy link is traversed by at least one uninterrupted m-trail during an SRLG failure, which can be achieved by launching no more than $(d+1)$ link-disjoint m-trails originating from the MN for each link. Based on such a sufficient condition, we develop a solution approach based on integer linear program (ILP) for m-trail allocation, and implement the method for SRLGs with $d = 3$. To avoid the high computation complexity in solving the ILP, a heuristic algorithm is developed for deriving $(d+1)$ link-disjoint m-trails between the MN and each link of the network.

Again, to derive burst launching times along the m-trails provided by the heuristic method, Algorithm 4.2, given in subsection 4.5.2 for burst scheduling, is used. Numerical results show that the proposed method, by solving the ILP and heuristics, yields significantly better performance than that of the methods in Section 5.2 and in [1] in large network topologies.

5.3.1 Theoretical Analysis

This section analyzes the m-trail allocation problem for local unambiguous failure localization (L-UFL) with a single MN and provides two theorems that serve as a basis for the subsequent ILP formulation and heuristic design.

Let the set of SRLGs and the set of m-trails traversing through undirected link (u, v) be denoted as Ψ and $\varphi_{(u,v)}$, respectively. A necessary condition for the eligibility of the solution is that each SRLG is traversed by a unique subset of the m-trails in the solution. This directly results in the fact that the m-trails that are not disrupted by a faulty SRLG should traverse all healthy links of the network. Such a fact is formalized in Theorem 5.2.7 given in Section 5.2 as a sufficient condition, in which $\forall\psi \in \Psi$, if each link in $E\backslash\psi$ is traversed by at least one m-trail that is disjoint from the SRLG ψ, any SRLG fault can be unambiguously localized by the MN according to the disrupted m-trails.

Let the maximum number of faulty links be d, and each undirected link be traversed by $(d+1)$ m-trails that are link-disjoint in other links. Moreover, each m-trail in the solution traverses the MN. We will demonstrate that the requirements of Theorem 5.2.7 can be satisfied with dramatically reduced number of m-trails, specifically less than or equal to $(d+1)|E|$.

A solution of the m-trail allocation problem for multi-link SRLG fault localization is a set of m-trails $\mathfrak{M}$, where each m-trail is passing through the MN. The solution must have at least three characteristics as described in Theorem 5.3.1.

Theorem 5.3.1. *$\forall(u, v) \in E$: (u, v) is an undirected link, if (u, v) is traversed by at least $(d+1)$ m-trails that are link-disjoint in other links, then*

i) the m-trails traversing each link is not a subset of those traversing another link of the network;

ii) each SRLG having the same number of links are traversed by a unique set of m-trails; and

iii) the m-trails traversing each SRLG is a proper subset of that traversing any of its superset SRLGs.

Proof. i) $\forall(u, v), (w, x) \in E$, if $\varphi_{(u,v)} \cap \varphi_{(w,x)} = \phi$, $\varphi_{(u,v)}$ can not be a subset of $\varphi_{(w,x)}$ and vice versa. In case $\varphi_{(u,v)} \cap \varphi_{(w,x)} \neq \phi$, let m^j traverse both the links, i.e., $m^j \in \varphi_{(u,v)} \wedge m^j \in \varphi_{(w,x)}$.

As the remaining d m-trails traversing link (u, v) are link-disjoint from m^j in other links, they cannot traverse link (w, x). Thus $\varphi_{(u,v)} \nsubseteq \varphi_{(w,x)}$. Similarly, $\varphi_{(w,x)} \nsubseteq \varphi_{(u,v)}$.

ii) $\forall \psi_i, \psi_j \in \Psi$: $\psi_i \neq \psi_j$ and $|\psi_i| = |\psi_j| = l$. This indicates that there are x links that are in ψ_i but not in ψ_j and vice versa. Here $1 \leq x \leq l \leq d$. For each one of the x links in ψ_i, at most l m-trails can traverse both the link and ψ_j, and at least one m-trail must be disjoint from ψ_j. Thus $\varphi_{\psi_i} \nsubseteq \varphi_{\psi_j}$. Similarly, $\varphi_{\psi_j} \nsubseteq \varphi_{\psi_i}$.

iii) $\forall \psi_i, \psi_j \in \Psi$: $\psi_i \subset \psi_j$. As $\psi_i \subset \psi_j$, all the m-trails that are traversing ψ_i will also traverse ψ_j. Hence, $\varphi_{\psi_i} \subseteq \varphi_{\psi_j}$. Moreover, as $\psi_i \subset \psi_j$, there exists at least one link that is in ψ_j but not in ψ_i. Assume that (w, x) is such a link, where $(w, x) \notin \psi_i \wedge (w, x) \in \psi_j$. Since an SRLG has at most d links, $|\psi_i| < d$. Thus at most $(d-1)$ m-trails that are traversing (w, x) can traverse ψ_i and at least two m-trails that do not traverse ψ_i will traverse ψ_j because $(w, x) \in \psi_j$. Thus, $|\varphi_{\psi_i}| < |\varphi_{\psi_j}|$. □

Theorem 5.3.2. $\forall (u, v) \in E$: *(u, v) is an undirected link, if (u, v) is traversed by at least $(d+1)$ m-trails that are link-disjoint in other links, where each m-trail is passing through the MN, single-link and multi-link SRLG faults involving up to d links can be unambiguously localized from the MN by detecting disrupted m-trails only.*

Proof. $\forall \psi \in \Psi$, single-link and multi-link SRLG faults involving up to d links in ψ can disrupt up to d m-trails that are traversing any healthy undirected link $(u, v) \in E \backslash \psi$. As each link of the network is traversed by $(d + 1)$ m-trails that are link-disjoint in other links, at least one m-trail that is link-disjoint from the faulty SRLG ψ will traverse (u, v). Hence, the condition of Theorem 5.2.7 is satisfied. Therefore, single-link and multi-link SRLG faults can be unambiguously localized from the MN by detecting disrupted m-trails only. □

5.3.2 ILP Formulation for the DMCF Method

This section provides an ILP for implementing the condition in Theorem 5.3.2. In contrast to the MCF method, given in Section 5.2, the developed ILP for the DMCF method ensures that each undirected link is visited by at least $(d+1)$ m-trails that are link-disjoint in other links. This guarantees that each undirected link in $E \backslash \psi$, $\forall \psi \in \Psi$, is traversed by at least

one m-trail that is link-disjoint from the SRLG ψ as $|\psi| \leq d$. The derived solution is sufficient for achieving L-UFL. Each m-trail in an ILP solution visits the MN as well.

List of Notation

G The network, $G = (V, E)$, where V is the set of nodes and E is the set of unidirectional links of the network.

MN The monitoring node in the network, $MN \in V$.

J Predefined maximum number of allowed m-trails in an ILP solution.

i, j Indices of m-trails $i, j \in \{0, 1, 2, \ldots, J-1\}$.

r_1, r_2 Predefined cost ratios.

δ Predefined small positive constant. It is the minimum voltage increase along an m-trail, $|E|^{-1} \geq \delta > 0$.

m^j Binary variable, it is equal to 1 if the jth m-trail is in an ILP solution and 0 otherwise.

mt^j Binary variable, it is equal to 1 if m^j is a trail and 0 otherwise.

so_u^j Binary variable, It is equal to 1 if node u is the source of the jth m-trail and 0 otherwise. Note that only MN can be the source of an m-trail.

si_u^j Binary variable, It is equal to 1 if node u is the destination of the jth m-trail and 0 otherwise. If the MN is also the destination of an m-trail, it will be a cycle.

ef_{uv}^j Binary variable, it is equal to 1 if the jth m-trail, which is either a trail or a cycle, traverses unidirectional link (u, v) en route to its destination node and 0 otherwise.

q_{uv}^j Fractional variable, It is defined as *voltage* of the vector $u \rightarrow v$ for unidirectional link (u, v). It assumes an arbitrary positive value if the jth m-trail traverses link (u, v) and 0 otherwise.

z_u^j Binary variable, it is equal to 1 if the jth m-trail visits node u and 0 otherwise. The variables z_u^j and q_{uv}^j along with predefined constant δ help to keep each m-trail in an ILP solution a single connected component.

er_{uv}^{j} Binary variable, it is equal to 1 if the jth m-trail, which is a trail not a cycle, traverses unidirectional link (u, v) while returning from its destination node to the MN and 0 otherwise.

e_{uv}^{j} Binary variable, it is equal to 1 if the jth m-trail traverses unidirectional link (u, v) and 0 otherwise.

eu_{uv}^{j} Binary variable, it is equal to 1 if the jth m-trail traverses undirected link (u, v) and 0 otherwise.

c^{ij} Binary variable, it is equal to 1 if both the ith and jth m-trails traverse any common undirected link and 0 otherwise.

ld_{uv}^{ij} Binary variable, it is equal to 1 if both the ith and jth m-trails traverse undirected link (u, v) but the ith m-trail is link-disjoint from the jth m-trail in all other undirected links, and 0 otherwise.

df_{uv}^{j} Binary variable, it is equal to 1 if the jth m-trail is counted as one of the otherwise link-disjoint m-trails traversing undirected link (u, v), and 0 otherwise.

l Integer variable, it is the maximum number of potential collision for any m-trail in an ILP solution.

n Integer variable, it is the maximum number of m-trails in an ILP solution traversing through any unidirectional link of the network.

es_{ψ}^{j} Binary variable, it is equal to 1 if the jth m-trail traverses any undirected link of SRLG ψ, and 0 otherwise.

α_{ψ} It is the decimal alarm code of SRLG ψ. Each decimal alarm code must be a positive number.

ILP for M-Trail Allocation

The specific ILP formulation is provided as follows.

Objective:

$$Minimize\left\{ n + l + r_1 * \sum_{j} m^{j} + r_2 * \sum_{j} \sum_{\forall (u,v) \in E} e_{uv}^{j} \right\} \tag{5.12}$$

Subject to the following constraints:

The constraints (5.13) to (5.21) and their variables are related with m-trail formation and mostly taken from the method in [63]; please refer to the paper for a thorough explanation of the constraints and the variables. A single source node and a single destination node are allowed for each m-trail in an ILP solution. Only the MN can be the source of an m-trail.

$$\sum_{u \in V} so_u^j = m^j, \;\; so_{MN}^j = \sum_{u \in V} so_u^j, \;\; \sum_{u \in V} si_u^j = m^j, \;\; \forall j \tag{5.13}$$

The indices of the m-trails in an ILP solution are the lowest ones. A lower bound of the number of m-trails in an ILP solution helps to find the solution faster.

$$m^j \geq m^{j+1}, \quad \forall j : j \leq J - 2 \tag{5.14}$$

$$\sum_j m^j \geq d * \lfloor \log_2(E+1) \rfloor \tag{5.15}$$

The jth m-trail can traverse an undirected link at most once en route to its destination node. A valid m-trail must traverse at least one unidirectional link.

$$ef_{uv}^j + ef_{vu}^j \leq m^j, \quad \forall (u,v) \in E : u < v, \forall j \tag{5.16}$$

$$\sum_{\forall (u,v) \in E} (ef_{uv}^j + ef_{vu}^j) \geq m^j, \quad \forall j \tag{5.17}$$

The flow conservation defined for each node of the network enforces that each m-trail consists of connected components.

$$\sum_{(u,v) \in E} (ef_{uv}^j - ef_{vu}^j) = (so_u^j - si_u^j), \quad \forall u \in V, \forall j \tag{5.18}$$

If any link incident on node u is traversed by the jth m-trail, the node is considered visited by the m-trail. Voltages of the vectors corresponding to on-trail links can be assigned non-zero values. The node voltage constraint (5.21) ensures that each m-trail in an ILP solution is a single connected component.

$$z_u^j \geq ef_{uv}^j + ef_{vu}^j, \quad \forall u \in V : (u,v) \in E, \forall j \tag{5.19}$$

$$q^j_{uv} \leq ef^j_{uv}, \quad \forall (u,v) \in E, \forall j \tag{5.20}$$

$$si^j_u + \sum_{(u,v)\in E} (q^j_{uv} - q^j_{vu}) \geq \delta * z^j_u, \quad \forall u \in V, \forall j \tag{5.21}$$

If the MN is the destination node of the jth m-trail, it is a cycle and a trail otherwise.

$$mt^j = m^j - si^j_{MN}, \quad \forall j \tag{5.22}$$

The constraints (5.23) – (5.25) ensure that if the jth m-trail is a cycle, it can traverse an undirected link at most once. On the other hand, if the m-trail is a trail and traverses an undirected link, it has to traverse the link from both directions.

$$er^j_{uv} \leq ef^j_{vu}, \quad \forall (u,v) \in E, \forall j \tag{5.23}$$

$$ef^j_{uv} \; + \; er^j_{vu} \leq m^j \; + \; mt^j, \quad \forall (u,v) \in E, \forall j \tag{5.24}$$

$$m^j \; + \; er^j_{vu} \geq mt^j \; + \; ef^j_{uv}, \quad \forall (u,v) \in E, \forall j \tag{5.25}$$

The jth m-trail can traverse unidirectional link (u,v) either en route to its destination node or while it is returning from its destination node to the MN.

$$e^j_{uv} = ef^j_{uv} + er^j_{uv}, \quad \forall (u,v) \in E, \forall j \tag{5.26}$$

The constraint derives the maximum number of m-trails traversing through any unidirectional link of the network.

$$n \geq \sum_j e^j_{uv}, \quad \forall (u,v) \in E \tag{5.27}$$

If both the ith and jth m-trails traverse any common link, c^{ij} will be 1. Moreover, c^{ij} and c^{ji} must assume the same value.

$$c^{ij} + 1 \geq e^i_{uv} + e^j_{uv}, \quad \forall (u,v) \in E, \forall i,j : i <> j \tag{5.28}$$

$$c^{ij} = c^{ji}, \quad \forall i, j \tag{5.29}$$

The constraint finds the maximum number of m-trails that traverse common link(s) with any m-trail.

$$l \geq \sum_{\forall i: i<>j} c^{ij}, \quad \forall j \tag{5.30}$$

If the jth m-trail traverse undirected link (u, v) from either direction en route to its destination node, the link is considered traversed by the m-trail.

$$eu^{j}_{uv} = ef^{j}_{uv} + ef^{j}_{vu}, \quad \forall (u, v) \in E : u < v, \forall j \tag{5.31}$$

Constraints (5.32) and (5.33) identify otherwise link-disjoint m-trail pairs that are traversing each undirected link (u, v). The ith and jth m-trails can be considered link-disjoint except in link (u, v) if both the m-trails traverse the link. ld^{ij}_{uv} and ld^{ji}_{uv} must assume the same value.

$$ld^{ij}_{uv} \leq eu^{i}_{uv}, \quad ld^{ij}_{uv} \leq eu^{j}_{uv}, \quad ld^{ij}_{uv} = ld^{ji}_{uv}, \forall (u, v) \in E : u < v, \forall i, j : i < j \tag{5.32}$$

Moreover, the ith m-trail has to be link-disjoint from the jth m-trail in all other undirected links. Now, $\forall (u, v), (w, x) \in E : u < v \wedge \ w < x \ \wedge \ (u, v) \neq (w, x), \ \forall i, j : i < j$,

$$ld^{ij}_{uv} \ + \ eu^{i}_{wx} \ + \ eu^{j}_{wx} \leq 2 \tag{5.33}$$

An m-trail traversing the link (u, v) can be considered otherwise link-disjoint from at most d m-trails that are also traversing the link. At least $(d+1)$ mutually otherwise link-disjoint m-trails should traverse each link of the network.

$$\sum_{i: i<>j} ld^{ij}_{uv} = d * df^{j}_{uv}, \quad \forall (u, v) \in E : u < v, \forall j \tag{5.34}$$

$$\sum_{j} df^{j}_{uv} = (d + 1), \quad \forall (u, v) \in E : u < v \tag{5.35}$$

If any undirected link (u, v) in SRLG ψ is traversed by the jth m-trail, the SRLG is considered traversed by the m-trail. If no link in SRLG ψ is traversed by the jth m-trail,

es^j_ψ is 0. The decimal alarm code α_ψ of SRLG ψ is determined by the set of traversing m-trails.

$$es^j_\psi \geq eu^j_{uv},\ \forall\psi \in \Psi, \forall(u,v) \in \psi : u < v, \forall j \tag{5.36}$$

$$es^j_\psi \leq \sum_{\forall(u,v)\in\psi:u<v} eu^j_{uv},\ \forall\psi \in \Psi, \forall j \tag{5.37}$$

$$\alpha_\psi = \sum_j 2^j * es^j_\psi,\quad \forall\psi \in \Psi \tag{5.38}$$

The objective function (5.12) aims at minimizing the number of potential collisions between the m-bursts indirectly by minimizing both n and l. The number of m-trails and usage of WLs in the ILP solution are also minimized where each undirected link is traverse by at least $(d+1)$ m-trails that are link-disjoint in other links.

From Eq. (5.36), we have the number of constraints in the order of $\mathrm{O}(d^2|E||\Psi|)$ because $|\psi| \leq d$ and $J \leq (d+1)|E|$, where $|E|$ is the number of undirected links, $|\Psi|$ is the number of SRLGs under consideration, and d is the maximum number of links in any SRLG. From Eq. (5.38), we have the number of variables in the order of $\mathrm{O}(|\Psi|)$, based on the variable α_ψ.

The scheduling algorithm will take the ILP solution, a set of m-trails, for a network as an input. It turns out that there is little performance differences between using the ILP given in sub-section 4.4.2 and the heuristic Algorithm 4.2 presented in sub-section 4.5.2 in solving the scheduling problem. We will therefore use the heuristic algorithm for burst launching time scheduling in this section.

5.3.3 Heuristic Algorithms for the DMCF Method

In this sub-section, two heuristic algorithms are proposed. The first one is a heuristic m-trail allocation algorithm for multi-link SRLG fault localization given in Algorithm 5.2. Like the ILP in sub-section 5.3.2 for m-trail allocation, the heuristic is based on Theorem 5.3.2, where each m-trail in the solution visits the MN. Next, the redundant m-trails in the solution are removed by using the second heuristic described in function RemoveRedundantMTrails. The resultant set of m-trails is then used as the input of the burst scheduler described in Algorithm 4.2 in sub-section 4.5.2 to find fault localization latency.

M-Trail Allocation for SRLG Fault Localization

The pseudo code of the proposed heuristic algorithm for the m-trail allocation problem is given in the Algorithm 5.2. The basic idea is to reuse each m-trail as much as possible such that every link is traversed by at least $(d+1)$ m-trails that are link-disjoint in other links, and there exists a set of m-trails that is disjoint from each SRLG while covering the rest of the network.

In line 1, the solution set of m-trails $\mathfrak{M}$, the alarm code table (ACT) $\mathfrak{A}$, and the set of the m-trail sets L are initialized as empty. Each element of L is a set of m-trails traversing a particular link. In lines 2 and 3, each undirected link of the network is assigned a distance that is the smaller one of the two shortest distances of the end nodes of the link from the MN. In line 4, links are en-queued to Q based on the descending order of their distances. Thus, m-trails traversing links that are far away from the MN will be derived before the m-trails traversing links near the MN.

The *while* loop will consider each undirected link in turn. Each iteration of the loop covers lines 5-18. In line 5, an undirected link (u, v) is de-queued from Q and a unique network G_{uv} is derived from G for (u, v). In line 6, the set of m-trails L_{uv} currently traversing the link (u, v) is retrieved from L. The intention is to reduce the total number of m-trails in the solution by reusing the largest possible set of m-trails already traversing link (u, v) but link-disjoint in all other links.

In line 7, each set of disjoint m-trails in L_{uv} are found using a helper function, not shown, $(d+1)$ times. Each time n from $(d+1)$ to 1 is used as another parameter. In the helper function, all the combinations of n m-trails from L_{uv} are verified whether they are link-disjoint or not in all links except link (u, v) using bitwise `AND` operation on the pairs of *m-trail codes*. The m-trail code of an m-trail is derived from link traversal of the m-trail. All disjoint set of m-trails are assigned to L^{ds}_{uv}. The cardinality of the largest set of disjoint m-trails in L^{ds}_{uv} is assigned to max in line 8.

For example, if no m-trail is currently traversing link (u, v), L_{uv} and L^{ds}_{uv} will be empty, and $max = 0$. If m^j is currently traversing link (u, v), $L_{uv} = \{m^j\}$, $L^{ds}_{uv} = \{\{m^j\}\}$, and $max = 1$. If m^i and m^j are currently traversing link (u, v) and m^i and m^j are not disjoint in all other links, $L_{uv} = \{m^i, m^j\}$, $L^{ds}_{uv} = \{\{m^i\}, \{m^j\}\}$, and $max = 1$. On the other hand, if m^i and m^j are currently traversing link (u, v) and m^i and m^j are disjoint in all other links, $L_{uv} = \{m^i, m^j\}$, $L^{ds}_{uv} = \{\{m^i, m^j\}, \{m^i\}, \{m^j\}\}$, and $max = 2$.

If max is less than $(d+1)$, this indicates that (u, v) is not yet traversed by at least $(d+1)$ m-trails that are link-disjoint in other links. In this case, each m-trail set L^j_{uv} in L^{ds}_{uv} will be considered in turn. Now, the m-trails in L^j_{uv} traverse (u, v) but disjoint in all other

links of G_{uv}. We tentatively assume that the m-trails in L^j_{uv} are already found for (u, v) and remaining k_j disjoint paths for the link will be derived in line 11 using Algorithm A.1, which is based on Suurballe's [50] and Bhandari's [12] algorithms for disjoint paths. If no set in L^{ds}_{uv} can be used to find reduced number of new otherwise link-disjoint paths, in line 15 we derive $(d+1)$ otherwise disjoint paths without using any m-trail in L_{uv}. We assume that the MN and each undirected link are separated by the min-cut with cardinality at least $(d+1)$. If the network G is $(d+1)$ connected, there will be at least $(d+1)$ link-disjoint paths between the MN and each end node of each link of the network as per Menger Theorem [3]. An m-trail is formed using each new path and unidirectional links (u, v) and (v, u) and the m-trail is added to $\mathfrak{M}$, $\mathfrak{A}$, and L in lines 17 and 18, respectively.

The m-trails solution set $\mathfrak{M}$ for multi-link SRLG fault localization and ACT $\mathfrak{A}$ are returned in line 19.

Let us derive the upper bound of the number of m-trails in a solution, $|\mathfrak{M}|$. The *while* loop of Algorithm 5.2 iterates $|E|$ times, where $|E|$ is the number of undirected links. At most $(d+1)$ m-trails are added to $\mathfrak{M}$ in each iteration of the loop. Thus, we have $|\mathfrak{M}| \leq (d+1)|E|$.

Now, let us derive complexity of the algorithm. Line 2 needs $\mathrm{O}(|V| \log_2 |V| + |E|)$ steps to find node distances from the MN using Dijkstra's algorithm. Line 3 needs $|E|$ steps to assign distances to the links. Line 4 needs $|E| \log_2 |E|$ steps to en-queue the links to Q in descending order of their distances.

The *while* loop iterates $|E|$ times. As m-trails derived for each link are link-disjoint in other links of the network, the number of m-trails traversing any link will be at most $|E|$. Thus, for each undirected link (u, v), $|L_{uv}| \leq |E|$. Now, to find all combination of up to $(d+1)$ m-trails in L_{uv}, $\mathrm{O}\big((|E|+1)^{(d+1)}\big)$ steps are required. To perform pair wise `AND` operations in a set of m-trails with cardinality $(d+1)$, $\mathrm{O}\big((d+1)^2\big)$ steps are needed. Thus, line 7 requires at most $\mathrm{O}\big((d+1)^2(|E|+1)^{(d+1)}\big)$ steps.

The first *for* loop inside the *while* loop will iterate $|L^{ds}_{uv}|$ times, which is equivalent to $\mathrm{O}\big((|E|+1)^d\big)$ because the cardinality of the largest m-trail set in L^{ds}_{uv} will be less than or equal to d inside the loop. Consequently, at most d disjoint paths for a link will be searched $\mathrm{O}\big((|E|+1)^d\big)$ times in line 11. Finding d disjoint path from the MN to each link needs $\mathrm{O}(d|V||E|)$ steps using Algorithm A.1 after minor modifications. Line 14 needs $\mathrm{O}(|E|)$ steps. Thus, the first *for* loop needs $\mathrm{O}\big(d|V||E|(|E|+1)^d\big)$ steps.

Line 15 needs $\mathrm{O}((d+1)|V||E|)$ steps. Finding the paths from D and forming m-trails based on the paths requires $\mathrm{O}((d+1)|E|)$ steps in line 17. Thus, overall complexity of the algorithm will be $\mathrm{O}\big(d(d+|V|)(|E|+1)^{(d+2)}\big)$.

Algorithm 5.2: The DMCF M-Trail Allocation Method

Input: $G(V, E)$, MN, and d
Output: $\mathfrak{M}$ and $\mathfrak{A}$
begin
Initialize $\mathfrak{M}, \mathfrak{A}, L \leftarrow \phi$, $k \leftarrow (d+1)$
$\forall u \in V$, find shortest distance $d(u)$ from the MN
foreach *undirected link* $(u, v) \in E$ **do**
  Assign $min\big(d(u), d(v)\big)$ as the distance of (u, v).
Assign the links to a queue Q in descending order of their distances.
while $Q \neq \phi$ **do**
  $(u, v) \leftarrow$ Extract from Q, $G_{uv} \leftarrow G$
  Retrieve the set of m-trails L_{uv} traversing (u, v) from L.
  Find all disjoint m-trails sets in L_{uv} and assign these sets in descending order of their cardinalities to L^{ds}_{uv}.
  $max \leftarrow$ the cardinality of the largest set of m-trail in L^{ds}_{uv}
  if $max < k$ **then**
    foreach $L^{j}_{uv} \in L^{ds}_{uv}$ **do**
      $k_j \leftarrow (k - |L^{j}_{uv}|)$
      Delete each unidirectional link in G_{uv} traversed by any m-trail in L^{j}_{uv}.
      Find k_j link-disjoint paths in G_{uv} from the MN to (u, v).
      Assign the link-disjoint paths to D.
      if $|D| == k_j$ **then**
        k disjoint paths are found for (u, v). Now, exit the for loop.
      else
        Restore each unidirectional link in G_{uv} traversed by any m-trail in L^{j}_{uv}.
    if *k disjoint paths are not found for (u, v) using any m-trail in L_{uv}* **then**
      Find k link-disjoint paths in G_{uv} from the MN to (u, v) without deleting any link of G_{uv} traversed by any m-trail in L_{uv}.
      Assign the link-disjoint paths to D.
    foreach *path* $p(u) \in D$ **do**
      Form m-trail m^j with $p(u)$, (u, v) and (v, u)
      Add m^j to $\mathfrak{M}$, $\mathfrak{A}$, and L
Return $\mathfrak{M}, \mathfrak{A}$

Post Processing Module

Note that Theorem 5.3.2 serves as a sufficient condition for L-UFL, which yields some redundant m-trails against the optimal case. Therefore, both results by the ILP and Algorithm 5.2 should go through a post process in order to remove any redundant m-trail. We shall describe the proposed post processing module now.

Once the set of m-trails $\mathfrak{M}$ and ACT $\mathfrak{A}$ are derived by solving the ILP or implementation of Algorithm 5.2, the function RemoveRedundantMTrails is invoked to remove the redundant m-trails from $\mathfrak{M}$ and $\mathfrak{A}$. We have applied two established rules to remove any redundant m-trail from a solution set: two or more SRLG codes should not become equal, and no SRLG code should become 0 after the removal of the m-trail [64].

In line 1, a conflict set is built for each m-trail. The m-trails in $\mathfrak{M}$ is assigned to Q in descending order of the cardinality of their conflict set in line 2. The *while* loop will consider each m-trail in turn. In line 4, an m-trail m^j is de-queued from Q. If m^j is removed from $\mathfrak{M}$, each link has to be traversed by at least one m-trail, and two or more SRLG codes should not become equal. These conditions are verified in lines 5 and 6, respectively. Here, Ψ is the set of SRLGs, α_i and α_j are SRLG codes of the SRLGs ψ_i and ψ_j, respectively. If either condition is not fulfilled, m-trail m^j cannot be removed and next iteration of the *while* loop starts. Otherwise, m-trail m^j is removed from $\mathfrak{M}$ and $\mathfrak{A}$ in line 7. In line 8, $\mathfrak{M}$ and $\mathfrak{A}$ are returned.

Function RemoveRedundantMTrails($\mathfrak{M}$, $\mathfrak{A}$)

Input: $G(V, E)$, Ψ, $\mathfrak{M}$, and $\mathfrak{A}$
Output: $\mathfrak{M}, \mathfrak{A}$
begin
  foreach $m^j \in \mathfrak{M}$ **do**
    foreach $(u, v) \in E$: *(u, v) is an undirected link and m^j traverses (u, v)* **do**
      foreach $m^i \in \mathfrak{M} \backslash m^j$ **do**
        if *m^i traverses (u, v)* **then**
1   Add m^i to the conflict set of m^j.
2 Assign the m-trails to a queue Q in descending order of the cardinalities of their conflict sets.
3 **while** $Q \neq \phi$ **do**
4    $m^j \leftarrow$ Extract from Q
     foreach $(u, v) \in E$: *(u, v) is an undirected link* **do**
       if *(u, v) is not traversed by at least one m-trail in $\mathfrak{M} \backslash m^j$* **then**
5   GoTo line 3.
     foreach $\psi_i \in \Psi$ **do**
       foreach $\psi_j \in \Psi \backslash \psi_i$ **do**
         if *$\alpha_i == \alpha_j$ considering all m-trails in $\mathfrak{M} \backslash m^j$* **then**
6   GoTo line 3.
7   Remove m^j from $\mathfrak{M}$ and $\mathfrak{A}$.
8 Return $\mathfrak{M}, \mathfrak{A}$.

We have, $|\mathfrak{M}| \leq (d+1)|E|$. To find conflict set, line 1 needs $\mathrm{O}\big((d+1)^2|E|^3\big)$ steps. Line

2 needs $O\big((d+1)|E|\log_2((d+1)|E|)\big)$ steps to en-queue the m-trails to Q in descending order of the cardinalities of their conflict sets. The *while* loop iterates $O\big((d+1)|E|\big)$ times. Line 5 needs $O(|E|)$ steps. Line 6 needs $O\big(|\Psi|^2\big)$ steps, where $|\Psi|$ is the number of SRLGs under consideration. Thus, overall complexity of the function RemoveRedundantMTrails is $O\big((d+1)|E||\Psi|^2\big)$.

5.3.4 Numerical Results

We have first solved the m-trail allocation problem using the ILP-based approach, given in Section 5.3.2, and ILOG CPLEX 11.1. The set of m-trails derived by solving the ILP is used as an input to the post process module, followed by the invocation of the burst scheduler described in Algorithm 4.2, given in sub-section 4.5.2, in order to find fault localization latency.

The experiment considers all the SRLGs with up to arbitrary three links, where each SRLG with two or three links is node-disjoint from the MN. Consequently, if an undirected link that is incident on the MN is traversed by a single-hop m-trail, the condition of Theorem 5.2.7 for the link will be satisfied. Similarly, if an undirected link that is incident on an MN neighboring node but node-disjoint from the MN is traversed by two otherwise link-disjoint m-trails and one of them is a two-hop m-trail, the condition of Theorem 5.2.7 for the link will also be satisfied. We assume that r_1 is 0.1 and r_2 is 0.01.

We conducted the experiment on a network with 7 nodes and 12 links. Node 0 is assigned as the MN. There are in total 96 SRLGs consisting of 12 single-link, 28 dual-link, and 56 triple-link SRLGs.

The solution of the ILP-based approach has $\sum m^j = 10$ m-trails, as shown in Figure 5.4. The ACT, not shown, of the network keeps the mapping between the alarm code and the corresponding SRLG. The alarm code of an SRLG is determined by the m-trails traversing through the SRLG. For example, link (1, 2) is traversed by m-trail m^0, its alarm code is 0 000 000 001; link (3, 4) is traversed by m-trails m^4 and m^9, its alarm code is 1 000 010 000; and link (5, 6) is traversed by m-trails m^6 and m^8, its alarm code is 0 101 000 000. On the other hand, since the SRLG $\psi = \{(1,2),(3,4),(5,6)\}$ is traversed by all the above mentioned m-trails, its alarm code 1 101 010 001 is the bitwise OR of its link alarm codes.

To run the burst scheduling heuristic using the set of m-trails, which is the solution of the ILP-based approach, we assume that the burst length L is 20 ms, burst propagation delay δ through any link (u, v) is 2 ms, and there is a single supervisory WL along each

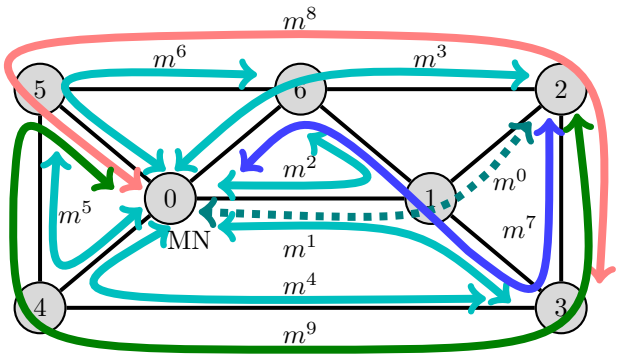


Figure 5.4: M-trail solution set for a network with 7 nodes and 12 links.

unidirectional link traversed by any m-trail. The starting time of the bursts s^j, total propagation delay through the m-trail tp^j, burst length L, and monitoring delay for the jth m-burst T^j in the network with 7 nodes and 12 links are shown in Table 5.6. Maximum monitoring delay, i.e., fault localization latency T is 80 ms. Thus, T is derived in the solution from the burst along m-trail m^7.

Table 5.6: Monitoring Delays along the M-Trails

m^j	$s^j + tp^j + L = T^j$	m^j	$s^j + tp^j + L = T^j$
0	$40 + 8 + 20 = 68$	5	$30 + 8 + 20 = 58$
1	$0 + 8 + 20 = 28$	6	$50 + 8 + 20 = 78$
2	$20 + 8 + 20 = 48$	7	$44 + 16 + 20 = 80$
3	$0 + 8 + 20 = 28$	8	$22 + 16 + 20 = 58$
4	$50 + 8 + 20 = 78$	9	$0 + 16 + 20 = 36$

Figure 5.5 illustrates the period when the m-bursts traverse through links from the MN to node 2 via nodes 6, 1, and 3, and back to the MN. Each rectangle represents the time period that the corresponding m-burst traverses through a link. The time period is equal to propagation delay δ through the link, which is 2 ms, and burst length L, which is 20 ms, i.e., $\delta + L = 2 + 20 = 22$ ms. When one burst arrives at node u to traverse unidirectional link (u, v) at the same moment another burst has left node v after traversing the link, they are shown adjacent to each other. Back-to-back periods are indicated in Figure 5.5 by black rectangles when two bursts traverse through an unidirectional link one immediately after the other. The m-bursts in Figure 5.5 as well as all other bursts remain non-overlapping

through any link of the network. Hence, the m-bursts are collision-free throughout the network.

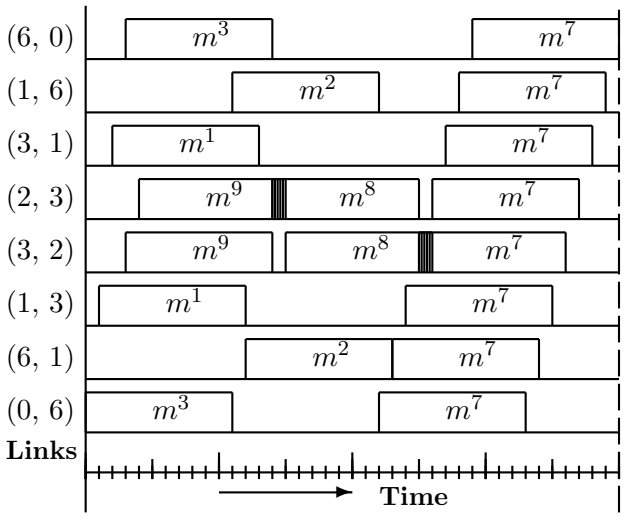


Figure 5.5: Schedule of m-bursts through selected unidirectional links.

The proposed heuristic DMCF method for m-trail allocation is implemented to compare a couple of methods in eight network topologies, namely the MCF method given in Section 5.2 and `MC-1` method proposed in [1]. For all the networks, all the SRLGs with up to arbitrary three links are considered, where each SRLG with two or three links is node-disjoint from the MN. Once a set of m-trails is derived by the m-trail allocation method for a network, the burst scheduling is done in the network using Algorithm 4.2 given in subsection 4.5.2 to derive burst launching times and fault localization latency T with $\delta = 2$ ms and $L = 20$ ms. The results are shown in Table 5.7.

In addition to the results by the ILP-based approach shown in column M_2, the `MC-1` method [1] is shown in columns H_0, which uses both m-cycles and m-paths based on the well-known necessary and sufficient conditions of UFL by comparing pair-wise SRLG codes, also called tag values, derived from the set of m-trails traversing each SRLG. The results by using the MCF method and the DMCF method are shown in column H_1 and H_2, respectively.

From the table, we see that the proposed DMCF method yields the best performance in terms of the least m-trails and the shortest fault localization latency among all the other method. However, the ILP-based approach is subject to huge computation complexity and

can only be implemented in generally small networks. In the experiment the ILP could only be used for networks consisting of up to 7 nodes and 12 links, in which we stopped running the program if it did not return a valid solution in 24 hours. Further, the proposed heuristic for m-trail allocation requires significantly fewer m-trails than required by all the other schemes, which translates to the shortest fault localization latency.

Table 5.7: The Comparative Performance of MC-1, MCF, and DMCF Methods

Networks	No. of SRLGs	$\sum m^j$				T			
		H_0	H_1	M_2	H_2	H_0	H_1	M_2	H_2
6 nodes, 12 links	96	17	19	12	12	108	152	88	88
7 nodes, 12 links	96	17	20	10	14	132	152	80	112
8 nodes, 12 links	132	22	27		15	172	228		120
9 nodes, 14 links	179	27	24		18	154	164		114
CERNet	236	29	32		20	168	258		152
SmallNet	1162	43	64		27	336	502		216
NSFNet + 2 links	1353	68	98		37	556	876		294
Bellcore + 1 link	2053	59	86		37	346	742		220

5.4 Conclusions

This chapter studies the m-burst framework for multi-link SRLG fault localization in all-optical mesh networks. In Section 5.2, the MCF method is presented. The method extends the m-burst framework for multi-link SRLG fault monitoring and localization in all-optical networks, which aims at achieving a truly all-optical, signaling-free, and deterministic fault localization plane. Detailed analysis of the proposed multi-link SRLG fault localization scheme is conducted. An ILP is formulated based on the developed theories, which are also the basis of the developed heuristic approach for large topologies. The ILP and the heuristic algorithm deal with the m-trail allocation and the burst scheduling problems as joint and separate optimization problems, respectively. By conducting extensive experiments, it is found that the proposed MCF method can effectively minimize the maximum fault localization latency.

In Section 5.3, with the minimization of the fault localization latency as the target, an optimization problem is formulated and solved in two parts: one is on m-trail allocation

and the other is on burst scheduling. Detailed analysis is provided. The theoretical proof shows that the proposed m-trail allocation method is able to achieve better performance. Numerical experiments are conducted to verify the proposed ILP-based approach and the heuristic algorithm, and compare them with two counterparts given in Section 5.2 and [1]. We conclude that the proposed DMCF method can significantly improve on the previously reported schemes in terms of both the number of m-trails and the total monitoring delay.

The methods developed in this chapter demonstrate that the m-burst framework can be applied for single-link and multi-link SRLG fault localization in all-optical networks as well. In other words, single-link and multi-link SRLG faults in all-optical networks can be localized from a single MN with reasonable fault localization latency by using at most a single WL in each undirectional link of the network. Hence, this research proves our thesis on single-link and multi-link SRLG fault localization in all-optical mesh networks.

Once the main objective of the thesis is achieved, we shall investigate techniques to minimize fault localization latencies further, using the m-burst framework for single-link, dual-link, and multi-link SRLG fault localization in all-optical networks in the next chapter. As always, there is a trade-off. Thus, we shall investigate methods that may need more than a single WL in some unidirectional links.

Chapter 6

M-Burst using Nested M-Trails

6.1 Introduction

The m-burst framework for single-link and multi-link SRLG fault localization in all-optical networks was introduced in Chapter 3. In the framework, fault localization is done from a single MN, and a single WL is assigned in each unidirectional link for fault localization if the link is traversed by any m-trail. The relevant problems of the framework are m-trail/m-cycle allocation, burst launching time scheduling, and node switch fabric configuration scheduling. The problems are formulated in Section 3.3.

In Chapter 4, the m-burst framework, using m-cycles, for single-link SRLG fault localization in all-optical mesh networks is presented. To deal with the m-cycle allocation and the burst launching time scheduling problems, the ILPs for joint and separate optimizations are formulated. Two heuristic algorithms are also developed to solve the problems. Moreover, the control mechanism for node switch fabric configuration is discussed in detail.

The m-burst framework, using m-trails, for multi-link SRLG fault localization in all-optical mesh networks is provided in Chapter 5. We have proposed two m-trail allocation methods. The MCF method ensures that during any fault event each healthy link of the network is traversed by at least one m-trail that is disjoint from the faulty SRLG. We have implemented the MCF method in ILP and heuristic for SRLGs with $d = 3$ links. Then the MCF method is extended using link-disjoint paths. The DMCF method requires that each undirected link of the network be traversed by at least $(d+1)$ otherwise link-disjoint m-trails. As a result, the number of m-trails required for fault localization and the fault localization latency are decreased significantly.

In this chapter, we first apply the m-burst framework in all-optical linear and ring networks. As in mesh networks, the framework has a single MN. A single WL in each unidirectional link of the networks is assigned because each unidirectional link will be traversed by at least one m-trail. The implementation of the m-burst framework in all-optical linear and ring networks is essential because not only are they important networks in their own right but also the insight gained from the practice will be applied in all-optical mesh networks.

We demonstrate in this chapter that dual-link SRLG faults can be localized from a single MN in the ring network using the m-burst framework. Based on this fact, we propose a novel technique called the *nested m-trail* method for mainly multi-link SRLG fault localization in all-optical mesh networks. Initially, a mesh network is decomposed into virtual ring and linear networks. Then, sets of m-trails are derived in the virtual networks. The nested m-trails are deployed in m-burst frameworks for the ring and linear networks to localize dual-link and single-link faults in the virtual ring and linear networks, respectively.

The rest of the chapter is organized as follows. In Section 6.2, single-link SRLG fault localization in all-optical linear networks is discussed. Section 6.3 provides dual-link SRLG fault localization in all-optical ring networks. In these sections, m-trail allocation, burst launching time scheduling, and node switch fabric configuration scheduling are discussed.

In Section 6.4 dual-link and in Section 6.5 multi-link SRLG fault localization in all-optical mesh network using the proposed novel *nested m-trail* technique are discussed. In Section 6.6, an ILP for decomposition of mesh networks into ring and linear networks is formulated. The corresponding heuristic is provided in Section 6.7. The numerical results for fault localization in mesh networks are given in Section 6.8.

The application of the *nest m-trail* technique for adaptive probing is given in Section 6.9. Section 6.10 concludes the chapter.

6.2 Single-Link SRLG Fault Localization in Linear Networks

Let $G = (V, E)$ be a linear network with $|V| = n$ nodes and $|E| = n - 1$ links, the nodes of the network be numbered from v_0 to v_{n-1} consecutively, the set of probes or m-trails be $\mathfrak{M}$, and the alarm code table (ACT) be $\mathfrak{A}$.

It is proved in [21] that to localize single-link faults unambiguously in linear network G, $|\mathfrak{M}| = \lceil \frac{n}{2} \rceil = \lceil \frac{|E|+1}{2} \rceil$ non-adaptive probes are required as shown in Figure 6.1; the same number of m-trails will be required for the purpose. However, each m-trail starts from its monitoring node (MN) through an outgoing monitoring wavelength channel (WL), visits its destination node, and returns to its MN through an incoming monitoring WL. The m-trail visits all the intermediate nodes between its MN and its destination node twice: once while going to its destination node and the second time while returning to its MN.

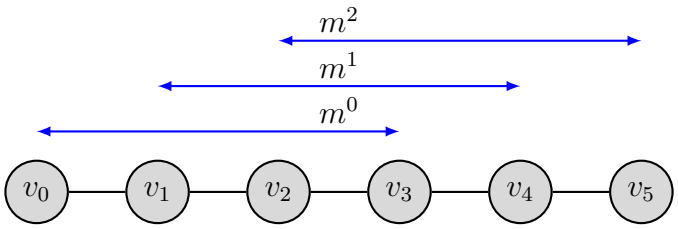


Figure 6.1: Single-link SRLG fault localization in a linear network. (adapted from [57]).

To localize single-link faults from a single MN, much more than $|\mathfrak{M}| = \lceil \frac{n}{2} \rceil = \lceil \frac{|E|+1}{2} \rceil$ m-trails are required because we have to ensure that each m-trail traverses the MN too, as shown in Figure 6.2.

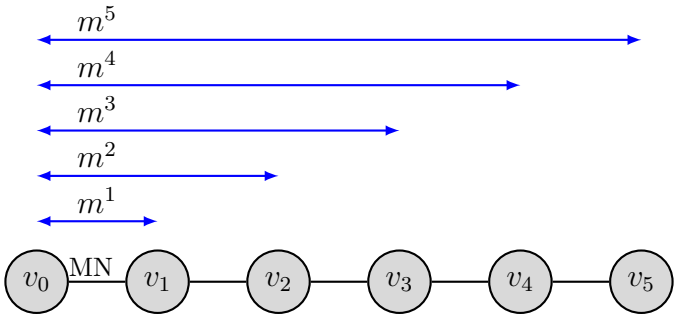


Figure 6.2: Single-link SRLG fault localization in a linear network from a single MN. End node v_0 is the MN.

Theorem 6.2.1. *If a linear network has a single MN and either end node of the network is the MN, it requires $|E|$ m-trails to localize single-link SRLG faults unambiguously from the MN.*

Proof. Let node v_0 be the MN. Each m-trail starts from the MN, reaches its destination node v_j, and returns back to the MN traversing all the intermediate nodes v_i twice, where $0 < i < j$. When node v_{n-1} is the MN, the proof will be similar. In this proof, we shall follow the same logical arguments of [21][63].

Assume that the number of required m-trails is less than $|E|$, i.e., $|\mathfrak{M}| < |E|$. In this case at least one node will not be a destination node of any m-trail as there are $n-1 = |E|$ nodes in addition to node v_0, which is the MN and the source node of every m-trail, in linear networks with $|E|$ links. In other words, $\exists v_k \in V$: v_k is not a destination node of any m-trail.

If $k = (n-1)$, the end link (v_{n-2}, v_{n-1}) will not be traversed by any m-trail. Hence, a single-link fault in the end link cannot be localized.

For $1 \leq k < (n-1)$, all the m-trails traversing the node v_k will traverse both of its adjacent links, i.e., (v_{k-1}, v_k) and (v_k, v_{k+1}). Hence, a single-link fault in either link (v_{k-1}, v_k) or (v_k, v_{k+1}) cannot be localized unambiguously. Thus, the number of required m-trails should be greater than or equal to $|E|$, i.e., $|\mathfrak{M}| \geq |E|$.

Let $|\mathfrak{M}|$ be equal to $|E| = n-1$ where each node is a destination node of only one m-trail, i.e., node v_k is the destination node of m-trail m^k. Next, assume that $\forall v_i, v_j$: $i \geq 1$, $1 < j \leq (n-1)$, and $i < j$. Let the sets of m-trails traversing links (v_{i-1}, v_i) and (v_{j-1}, v_j) be $\varphi_{(v_{i-1},v_i)}$ $\varphi_{(v_{j-1},v_j)}$, respectively.

Now, all the m-trails traversing the links beyond the node v_j, if any, will traverse both links (v_{i-1}, v_i) and (v_{j-1}, v_j). Let the set of these common m-trails be φ, which may be empty. Therefore, $\varphi_{(v_{i-1},v_i)} = \varphi \cup \{m^i, m^{i+1}, \ldots, m^{j-1}, m^j\}$ and $\varphi_{(v_{j-1},v_j)} = \varphi \cup \{m^j\}$, where m-trail subset $\{m^{i+1}, \ldots, m^{j-1}\}$ may be empty. Hence, both links will be traversed by unique sets of m-trails. Consequently, a single-link fault in the links can be localized unambiguously. Therefore, we do not need more than $|E|$ m-trails. □

Corollary 6.2.2. *If a linear network has a single MN and any node v_i where $i \neq 0 \wedge i \neq (n-1)$ is the MN, it requires $|E|$ m-trails to localize two faulty links unambiguously from the MN: one faulty link from the segment v_0 to v_i and another one from v_i to v_{n-1}.*

Figure 6.3 shows a linear network to explain Corollary 6.2.2, where an intermediate node v_2 is the MN. Now, a fault in segment v_0 to v_2 and another fault in segment v_2 to v_5

can be localized independently. From an intermediate node of a linear network, it is now possible to localize dual-link SRLG faults if the faulty links are located at different sides of the MN.

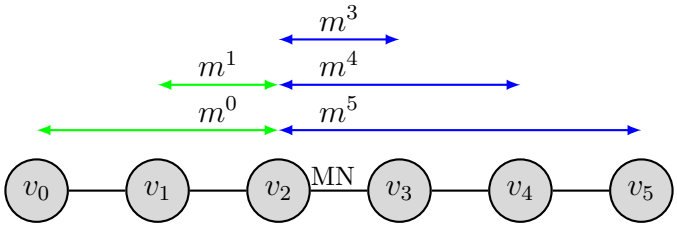


Figure 6.3: Single-link SRLG fault localization from a single MN in each segment at either side of the MN in a linear network. Intermediate node v_2 is the MN.

Theorem 6.2.3. *In linear networks, only a single faulty link can be localized unambiguously from a single MN if either end node is the MN.*

Proof. Let $|\mathfrak{M}| = |E|$ and each node except the MN be the destination node of an m-trail. Now, $\forall i, j\colon\ i < j$, and v_i and v_j are two adjacent intermediate nodes, link (v_i, v_j) will be traversed by a set of m-trails $\varphi_{(v_i,v_j)} = \{\varphi \cup \{m^j\}\}$, where v_j is the destination node of m-trail m^j, and φ is the set of m-trails traversing links beyond link (v_i, v_j) from the MN.

Let a link (v_k, v_l) beyond link (v_i, v_j) from the MN be traverse by a set of m-trails $\varphi_{(v_k,v_l)}$. Thus, $\varphi_{(v_k,v_l)} \subseteq \varphi \subset \varphi_{(v_i,v_j)}$. Now, if link (v_i, v_j) becomes faulty, all the m-trails in $\varphi_{(v_i,v_j)}$ will be disrupted and the MN will localize link (v_i, v_j) as faulty link irrespective of whether link (v_k, v_l) is faulty or not. Thus, if both links (v_i, v_j) and (v_k, v_l) become faulty simultaneously, the MN cannot localize the faulty links unambiguously. □

Corollary 6.2.4. *In linear networks, if any intermediate node is a single MN, only a single faulty link can be localized unambiguously from the MN in each segment from the MN to either end node.*

Corollary 6.2.5. *If a failure event involves a cut, minimum or not, and any number of links beyond the cut from the MN, the MN cannot localize the faulty links unambiguously. The MN will identify the links in the cut as the only faulty links.*

An m-trail is a non-simple path: it is allowed to traverse a node multiple times but each unidirectional link only once. As m-trails cannot traverse any unidirectional link multiple times, m-trails in linear networks have deterministic paths. Each m-trail in the m-burst framework for linear networks starts from its source node, reaches its destination node, and returns to the source node visiting each undirected link between the nodes twice: once from each direction.

Theorem 6.2.6. *If an end node is the single MN of a linear network, there will be* $|E|$ *unique m-trails.*

Proof. As the path of an m-trail is fully determined by the MN and the destination node of the m-trail, in this scenario each node except the MN can be the destination node of only one m-trail. Thus, there will be $|V| - 1 = |E|$ unique m-trails. □

As in all-optical mesh networks, the relevant problems of the m-burst framework in all-optical linear networks are m-trail allocation, burst launching time scheduling, and node switch fabric configuration scheduling problems. The problems will be discussed in subsections 6.2.1, 6.2.2, and 6.2.3 in detail, respectively.

6.2.1 M-Trail Allocation in Linear Networks

As $|\mathfrak{M}| = |E|$ m-trails are required to localize single-link faults unambiguously and there are $|E|$ unique m-trails in linear networks when an end node is the MN, we can enumerate m-trails in $\mathfrak{M}$ by inspection as described in Algorithm 6.1.

Algorithm 6.1: M-Trail Allocation in Linear Networks

Input: $G(V, E)$ and MN
Output: $\mathfrak{M}$ and $\mathfrak{A}$
begin
1 Initialize $\mathfrak{M}$, $\mathfrak{A} \leftarrow \phi$
foreach *node* $v_j \in V \backslash$MN **do**
2 Form m-trail m^j using all unidirectional links on the route between node v_j and the MN.
3 $\mathfrak{M}$, $\mathfrak{A} \leftarrow m^j$
4 Return $\mathfrak{M}$ and $\mathfrak{A}$.

Example: Let us derive m-trails by inspection for single-link SRLG fault localization in a linear network with 8 nodes and 7 links, where the nodes of the network are

numbered from v_0 to v_7 consecutively, and node v_0 is the MN. There are 7 m-trails. $\mathfrak{M} = \{m^1, m^2, m^3, m^4, m^5, m^6, m^7\}$. The destination nodes of the m-trails m^1, m^2, ..., m^7 are nodes v_1, v_2, ..., v_7, respectively. As each m-trail is fully determined by its destination node and the MN, we can easily find m-trails. For instance, m^1 will traverse unidirectional links (v_0, v_1) and (v_1, v_0), and m^3 will traverse unidirectional links (v_0, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_2), (v_2, v_1), and (v_1, v_0). Table 6.1 shows the ACT $\mathfrak{A}$ constructed from the link traversal of the m-trails in $\mathfrak{M}$. The decimal link alarm codes (Dec.) derived from the m-trails are unique.

Table 6.1: ACT for Single Link Fault Localization in a Linear Network with 7 Links

Links	m^7	m^6	m^5	m^4	m^3	m^2	m^1	Dec.
(v_0, v_1)	1	1	1	1	1	1	1	127
(v_1, v_2)	1	1	1	1	1	1	0	126
(v_2, v_3)	1	1	1	1	1	0	0	124
(v_3, v_4)	1	1	1	1	0	0	0	120
(v_4, v_5)	1	1	1	0	0	0	0	112
(v_5, v_6)	1	1	0	0	0	0	0	96
(v_6, v_7)	1	0	0	0	0	0	0	64

In the m-burst framework for linear networks, two dedicated WLs per undirected link are assigned for monitoring: one WL for each direction.

Now, can we remove the constraint of localizing only a single-link SRLG fault in linear networks from each side of the MN? The answer is a resounding yes. If we connect two ends of a linear network, it becomes a ring network. From a single MN, links of ring networks can be traversed from both clockwise and anti-clockwise directions. As a result, two single-link faults can be localized unambiguously from two directions irrespective of locations of two faulty links. However, before delving into dual-link SRLG fault localization in ring networks, we discuss scheduling of burst launching from the MN, fault localization latency, and configuration of node switch fabrics in linear networks.

In subsections 6.2.1 and 6.2.2, we assume that an end node of each linear network is the MN. If an intermediate node is the MN, we can solve the problem by considering the MN and each segment from the MN to either end node as an independent problem. We also assume that node v_0 is the MN. Let the source and the destination of each m-trail m^j be the MN and node v_j, respectively.

6.2.2 M-Burst Launching Time Scheduling in Linear Networks

Let δ and L be the burst propagation delay in ms through any unidirectional link of a network and the burst length in ms, respectively. The fault localization latency T is equal to the largest elapsed time for any m-burst from the start of the monitoring period to the complete return of the burst back to the MN; this is the last returning burst along m-trail m^l in linear networks. Thus, T is $s^l + tp^l + L$, where s^l and tp^l are burst launching time in ms from MN along the m-trail and total round trip burst propagation delay in ms along the m-trail, respectively.

Theorem 6.2.7. *The lower bound of the fault localization latency T to localize single-link SRLG fault from a single MN, which is an end node of a linear network, is $2\delta|E| + L$.*

Proof. Assume that all bursts are launched back-to-back from the MN and $\forall m^i, m^j$: $i = j - 1$, the burst along m-trail m^j is launched immediately before the burst along m-trail m^i. In other words, the bursts are launched back-to-back in descending order of the lengths of the corresponding m-trails. The burst along m-trail m^j has to traverse two more unidirectional links than the burst along m-trail m^i, i.e., links (v_i, v_j) and (v_j, v_i).

If $L \leq \delta$, the burst along m-trail m^i will complete traversal of its destination node v_i before the burst along m-trail m^j returns back to node v_i because the burst along m-trail m^j takes 2δ ms to return. On the other hand, the burst along m-trail m^i takes L ms, which is less than equal to δ ms, to completely pass through the node. Thus, the burst pair will traverse the links without any collision.

Therefore, the fault localization latency T will be equal to the burst traversal time along the longest m-trail. In other words, $T = s^l + tp^l + L = 0 + \delta \cdot 2|E| + L = 2\delta|E| + L$.

Note that T cannot be less than $2\delta|E| + L$ because T has to be greater than or equal to the burst traversal time along the longest m-trail. □

The lower bound can be achieved only when $L \leq \delta$ because in these cases the bursts can be launched back-to-back in descending order of the lengths of the corresponding m-trails and the bursts traverse links in parallel.

If $\delta < L \leq \delta(|E| - 1)$, we can launch only some bursts back-to-back in descending order of the lengths of the corresponding m-trails from the MN that traverse links in parallel. Before deriving upper bound of T, let us have a look on a few instances of burst length.

If $L = 2\delta$ and $|E|$ is even, bursts along all even numbered m-trails can be launched back-to-back in descending order of the lengths of the corresponding m-trails. After $2\delta + L$ delay to avoid collision between the longest and the shortest bursts, the bursts along all odd numbered m-trails can also be launched back-to-back in descending order of the lengths of the corresponding m-trails. Figure 6.4 shows snapshots of burst traversal after $t_{6\delta}$, $t_{9\delta}$, $t_{13\delta}$, and $t_{15\delta}$ seconds from the beginning of a monitoring period in a linear network with six links. As the last burst returns to the MN is the burst along the second longest m-trail, which is odd numbered, the fault localization latency is $T = s^l + tp^l + L = [0 + L(|E|/2 - 1) + (2\delta + L)] + 2\delta(|E| - 1) + L = L|E|/2 + 2\delta|E| + L = (2\delta + L/2)|E| + L = 3\delta|E| + L$. The fault localization latency T for the network is 20δ.

If $L = \delta(|E| - 1)$, we can launch all the bursts back-to-back in ascending order of the lengths of the corresonding m-trails. The last burst will traverse all links of the network. Thus, the fault localization latency T would be $s^l + tp^l + L = [0 + L(|E| - 1)] + 2\delta|E| + L = (2\delta + L)|E|$. However, T can be reduced further if the bursts are launched in descending order of the lengths of the corresponding m-trails except the burst along the shortest m-trail. As the bursts along the longest and the shortest m-trails can traverse the links in parallel, the burst along the shortest m-trail can be luanched immediately after the burst along the longest m-trail. In other words, bursts along the longest, the shortest and the second longest m-trails can be launched back-to-back. Then, each burst can be launched in descending order of the lengths of the corresponding m-trail. However, launching of each burst has to be delayed by $2\delta + L$ from that of the previous one. The last burst will traverse four unidirectional links. Thus, the fault localization latency T will be $s^l + tp^l + L = [0 + L + L + (L + 2\delta)(|E| - 3)] + 4\delta + L = (2\delta + L)|E| - 2\delta$.

If $L > \delta(|E| - 1)$, we cannot launch any burst back-to-back to traverse links in parallel completely with another burst. However, we can launch the bursts back-to-back from the MN or ensure that the bursts return to the MN back-to-back. Thus, partial parallelism can be achieved. Although, we have to ensure that each previously launched burst completes traversal of the destination nodes, if any one in its path, of the bursts launched afterward before they reach their respective destination nodes. The first burst will be launched at the beginning of a monitoring period as usual.

Theorem 6.2.8. *The upper bound of the fault localization latency T to localize single-link SRLG faults from a single MN, which is an end node of a linear network, is $(2\delta + L)|E|$.*

Proof. If bursts are launched in ascending order of the lengths of the corresponding m-trails, all burst can be launched back-to-back. As each burst has to traverse two more

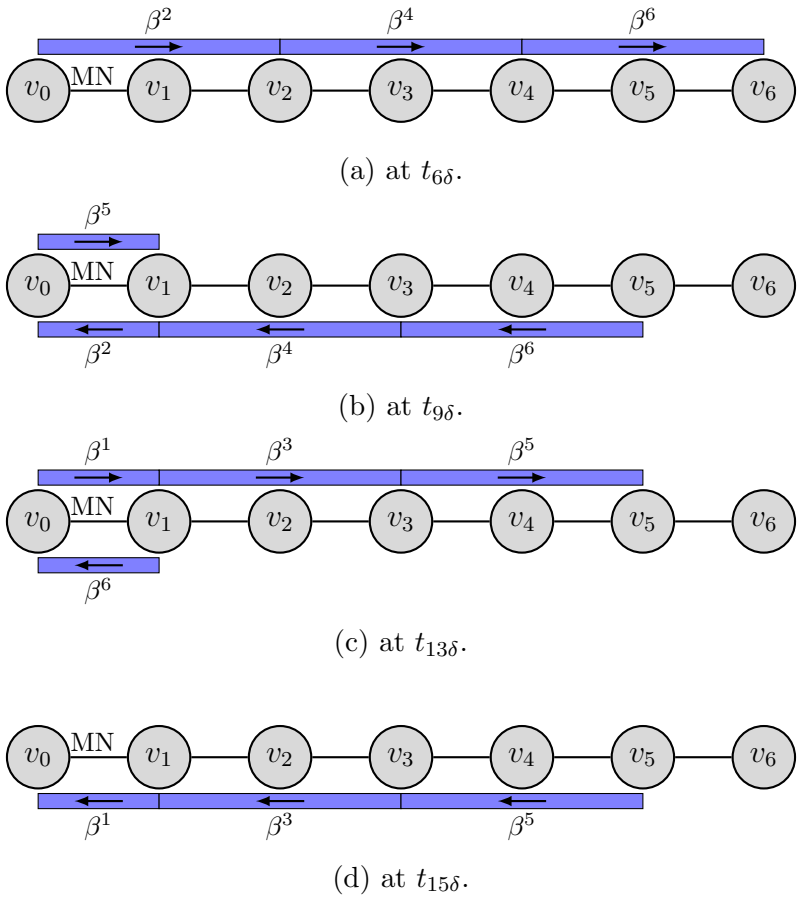


Figure 6.4: Burst traversal in a linear network to localize single-link SRLG faults from a single MN. End node v_0 is the MN. Burst β^j traverses the links along m-trail m^j. $L = 2\delta$.

unidirectional links than the burst launched immediately before it, bursts cannot collide because each predecessor burst will remain ahead of its successor bursts on the return route to the MN too. The last burst that returns to the MN will be the burst along the longest m-trail that traverses each unidirectional links of the network. Thus, the fault localization latency T will be $s^l + tp^l + L = [0 + L(|E| - 1)] + 2\delta|E| + L = (2\delta + L)|E|$.

Alternatively, if bursts are launched in descending order of the lengths of the corresponding m-trails, launching of each subsequent burst has to be delayed by $2\delta + L$ to

avoid collision. When a burst reaches its destination node, all the bursts along longer m-trails, if any, will have passed through the node completely while they are returning to the MN. The last burst that returns to the MN will be the burst along the shortest m-trail that traverses two unidirectional links. Thus, the fault localization latency T will be $s^l + tp^l + L = [0 + (2\delta + L)(|E| - 1)] + 2\delta + L = (2\delta + L)|E|$. □

Therefore, the fault localization latency in a linear network is dependent on burst length L and propagation delay δ through links. They determine how many bursts can traverse the links in parallel and burst launching delays to avoid collision.

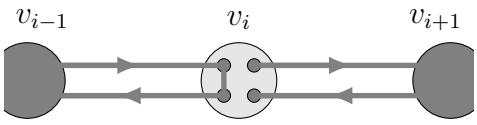
6.2.3 Node Switch Fabric Configuration Scheduling in Linear Networks

The MN connects the outgoing monitoring WL to the transmitter and the incoming monitoring WL to the receiver of the monitor. The other end node v_{n-1} connects the two WLs together. These connections will not be changed but the interconnection between WLs in intermediate nodes will be changed periodically.

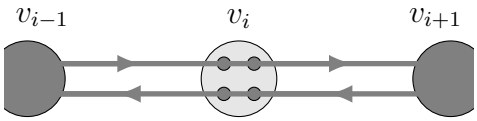
Although some values of L in the range $\delta < L \leq \delta(|E| - 1)$ provide lower values of the fault localization latency T with different burst launching sequences, we will discuss next only burst launching in ascending and descending order of the lengths of the corresponding m-trails for the reason that will become clear when we discuss burst scheduling in ring networks in sub-section 6.3.2.

Assume that bursts are launched back-to-back in ascending order of the lengths of the corresponding m-trails. Now, $\forall i$: $0 < i < (n - 1)$, node v_i connects two WLs from and to node v_{i-1} in the beginning of each monitoring period as shown in Figure 6.5a. At $t = (\delta + L)i + kT$, where $k = 0, 1, \ldots$, node v_i connects the WL from node v_{i-1} with the WL to node v_{i+1} and vice versa as shown in Figure 6.5b.

Similarly, assume that bursts are launched in descending order of the lengths of the corresponding m-trails keeping consecutive bursts $(2\delta + L)$ ms apart at launching time. Now, $\forall i$: $0 < i < (n - 1)$, each node v_i connects the WL from node v_{i-1} with the WL to node v_{i+1} and vice versa in the beginning of each monitoring period as shown in Figure 6.6a. At $t = (2\delta + L)(|E| - i) + i\delta + kT$, where $k = 0, 1, \ldots$, node v_i connects two WLs from and to node v_{i-1} as shown in Figure 6.6b.

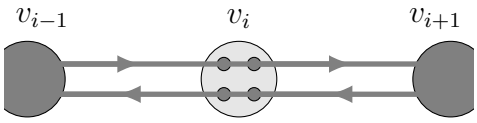


(a) at $t = kT$.

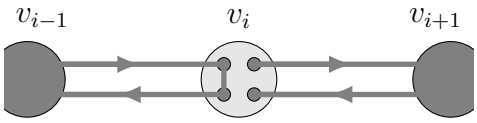


(b) at $t = (\delta + L)i + kT$.

Figure 6.5: Configuration of node switch fabric of intermediate node v_i for single-link SRLG fault localization from a single MN in linear networks when bursts are launched back-to-back in ascending order of the lengths of corresponding m-trails. End node v_0 is the MN.



(a) at $t = kT$.



(b) at $t = (2\delta + L)(|E| - i) + i\delta + kT$.

Figure 6.6: Configuration of node switch fabric of intermediate node v_i for single-link SRLG fault localization from a single MN in linear networks when bursts are launched in descending order of the lengths of corresponding m-trails. End node v_0 is the MN.

6.2.4 Numerical Results

Numerical experiments are conducted in linear networks with 3 to 12 links. A set of m-trails is derived for single-link SRLG fault localization in each network by inspection. Then, the burst starting times from the MN along the m-trails are scheduled using the heuristic Algorithm 4.2, given in Chapter 4. The burst propagation delay δ through a unidirectional

link is 2 ms. The burst length L is varied from 1 ms to 32 ms in 7 steps. The expected and actual fault localization latencies according to the burst lengths in each network are shown in columns E and A of Table 6.2, respectively. The actual fault localization latencies are consistent with the theoretically expected values.

Table 6.2: Fault Localization Latency T (ms) in Linear Networks

# of	$L = 1$ ms		$L = 2$ ms		$L = 4$ ms		$L = 8$ ms		$L = 16$ ms		$L = 20$ ms		$L = 32$ ms	
Links	E	A	E	A	E	A	E	A	E	A	E	A	E	A
3	13	13	14	14	≤ 20	20	36	36	60	60	72	72	108	108
4	17	17	18	18	≤ 28	28	48	48	80	80	96	96	144	144
5	21	21	22	22	≤ 34	32	≤ 56	56	100	100	120	120	180	180
6	25	25	26	26	≤ 40	40	≤ 68	64	120	120	144	144	216	216
7	29	29	30	30	≤ 46	44	≤ 80	76	140	140	168	168	252	252
8	33	33	34	34	≤ 52	52	≤ 88	88	160	160	192	192	288	288
9	37	37	38	38	≤ 60	56	≤ 104	96	≤ 176	176	216	216	324	324
10	41	41	42	42	≤ 64	64	≤ 116	104	≤ 196	192	240	240	360	360
11	45	45	46	46	≤ 70	68	≤ 128	116	≤ 216	208	≤ 260	260	396	396
12	49	49	50	50	≤ 76	76	≤ 140	128	≤ 236	224	≤ 284	280	432	432

6.3 Dual-Link SRLG Fault Localization in Ring Networks

Let $G = (V, E)$ be a ring network with $|V| = n$ nodes and $|E| = n$ links, the nodes of the network be numbered clockwise from v_0 to v_{n-1} consecutively, the set of m-trails be $\mathfrak{M}$, and the alarm code table (ACT) be $\mathfrak{A}$.

It is proved in [54] that $|\mathfrak{M}| = \lceil \frac{n}{2} \rceil = \lceil \frac{|E|}{2} \rceil$ m-trails are required to localize single-link SRLG faults unambiguously in a ring network with more than four nodes as shown in Figure 6.7. The well-known necessary and sufficient condition for unambiguous single-link SRLG fault localization is that each undirected link of the network has to be traversed by a unique set of m-trails.

As in linear networks, to localize single-link SRLG faults from a single MN in ring networks, much more than $|\mathfrak{M}| = \lceil \frac{n}{2} \rceil = \lceil \frac{|E|}{2} \rceil$ m-trails are required. In this case, the MN has to be either the source or the destination node of each m-trail as shown in Figure 6.8.

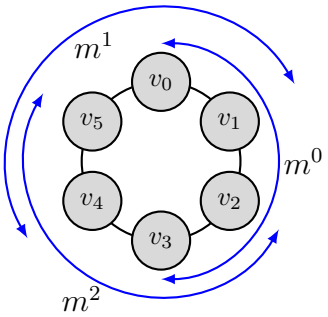


Figure 6.7: Single-link SRLG fault localization in a ring network.

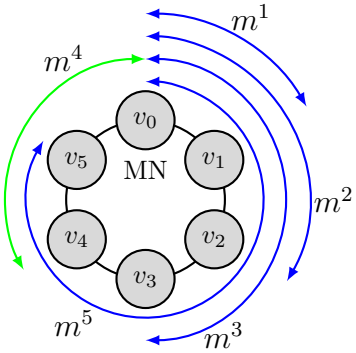


Figure 6.8: Single-link SRLG fault localization in a ring network from a single MN. Node v_0 is the MN.

Theorem 6.3.1. *If a ring network has a single MN, it requires* $(|E|-1)$ *m-trails to localize single-link SRLG faults unambiguously from the MN.*

Proof. It is also proved in [54] that a single-link SRLG fault can only be unambiguously localized in two adjacent links of a degree-2 node if the node is either source or destination of an m-trail. As each node of a ring network is a degree-2 node and the MN is the source of every m-trail, each node except the MN has to be the destination node of an m-trail. Thus, we need at least $(n-1) = (|E|-1)$ m-trails as there are $(n-1)$ nodes except the MN in a ring network.

Assume $|\mathfrak{M}| = (n-1) = (|E|-1)$. If each m-trail traverses links in clockwise (or anti-clockwise) direction from the MN, one link adjacent to the MN will not be traversed by any m-trail. Thus, fault in the link cannot be localized. As a result, m-trails has to traverse links in both clockwise and anti-clockwise directions.

Let node v_0 be the MN. Assume that m-trails $m^1, \ldots, m^{n-2}$ are traversing links in clockwise direction from the MN and a two-hop m-trail m^{n-1} is traversing links in anti-clockwise direction as shown in Figure 6.8 in a ring network with six nodes. In case of a solution set consisting of a multi-hop anti-clockwise m-trail, multiple anti-clockwise m-trails, or any combination of them, the proof will be similar.

Let destination nodes of m-trails $m^1, \ldots, m^{n-3}$ be $v_1, \ldots, v_{n-3}$, respectively. As each node from v_1 to v_{n-3} is a destination node of a clockwise m-trail, the links located clockwise from the MN to node v_{n-3} will be traverse by a unique set of m-trails irrespective of any common m-trails, if any, traversing the links [see proof of Theorem 6.2.1].

Now, we have to show that remaining three links (v_{n-3}, v_{n-2}), (v_{n-2}, v_{n-1}) and (v_{n-1}, v_0) are also traversed by unique sets of m-trails.

As the destination node of two-hop m-trail m^{n-1} is v_{n-2}, the destination node of m-trail m^{n-2} has to be v_{n-1}. Otherwise node v_{n-1} will not be a destination of any m-trail. Hence, link (v_{n-2}, v_{n-1}) will be traversed by both the m-trails. Thus, the sets of m-trails traversing links (v_{n-3}, v_{n-2}), (v_{n-2}, v_{n-1}) and (v_{n-1}, v_0) are $\varphi_{(v_{n-3},v_{n-2})} = \{m^{n-2}\}$, $\varphi_{(v_{n-2},v_{n-1})} = \{m^{n-2}, m^{n-1}\}$ and $\varphi_{(v_{n-1},v_0)} = \{m^{n-1}\}$, respectively. Thus, the links will be traversed by unique sets of m-trails.

Therefore, we need no more than $(n-1) = (|E|-1)$ m-trails to localize single-link SRLG faults in ring networks. □

In this section, we are mainly interested in dual-link SRLG fault localization in all-optical ring networks. If we deploy one set of m-trail in clockwise (anti-clockwise) direction from the MN, the ring network will look like a linear network from the MN in the direction for fault monitoring purposes. As per Corollary 6.2.4, the MN will localize the nearer faulty link after any fault event considering clockwise (anti-clockwise) direction irrespective of whether another link has failed or not. Thus, as per Corollary 6.2.2, we need two sets of m-trails to localize dual-link SRLG faults in ring networks unambiguously: one in clockwise

and another in anti-clockwise directions from the MN. Each m-trail set will be responsible for localizing only a single faulty link.

Let the nearer faulty link from the MN in clockwise direction be called the first faulty link and the nearer faulty link from the MN in anti-clockwise direction be called the second faulty link. The clockwise m-trail set will localize the first faulty link and the anti-clockwise m-trail set will localize the second faulty link. Thus, fault in clockwise (anti-clockwise) MN adjacent link will be localized unambiguously by the clockwise (anti-clockwise) m-trail set. Hence, each set has to localize faults in $(|E| - 1)$ links only.

Now, we propose a novel but simple method for dual-link SRLG fault localization in all-optical ring networks in Algorithm 6.2 that is an application of the m-burst framework in all-optical ring networks. As in all-optical mesh networks, the relevant problems of the m-burst framework in all-optical ring networks are m-trail allocation, burst launching time scheduling, and node switch fabric configuration scheduling problems. The problems will be discussed in subsections 6.3.1, 6.3.2, and 6.3.3 in detail, respectively.

Algorithm 6.2: Dual-Link SRLG Fault Localization in Ring Networks

Input: $G(V, E)$ and MN
Output: $\mathfrak{M}$, $\mathfrak{A}$, S, and T
begin
1. Derive by inspection the clockwise and anti-clockwise sets of m-trails from the MN in G. For clockwise (anti-clockwise) set of m-trails, consider $(|E| - 1)$ links that includes all links in G except the anti-clockwise (clockwise) MN adjacent link as a linear network.
2. Assign all the m-trails in the sets to $\mathfrak{M}$ and $\mathfrak{A}$.
3. Determine the burst launching times S from the MN along the m-trails.
4. Determine the fault localization latency T.
   foreach *node* $u \in V$ **do**
5. Determine node switch fabric configuration schedule based on the burst arrival and departure times along the m-trails visiting node u.
6. Return $\mathfrak{M}$, $\mathfrak{A}$, S, and T.

The m-trail deployment scheme in a ring network with 6 nodes and 6 links is shown in Figure 6.9. The clockwise set of m-trails is $\{m^1, m^2, m^3, m^4, m^5\}$ and the anti-clockwise m-trails is $\{m^{1'}, m^{2'}, m^{3'}, m^{4'}, m^{5'}\}$. Each set has 5 m-trails. The MN is the source node of each m-trail.

Theorem 6.3.2. *In a ring network, it requires* $|\mathfrak{M}| = 2(|E| - 1)$ *m-trails to localize dual-link SRLG faults unambiguously from a single MN.*

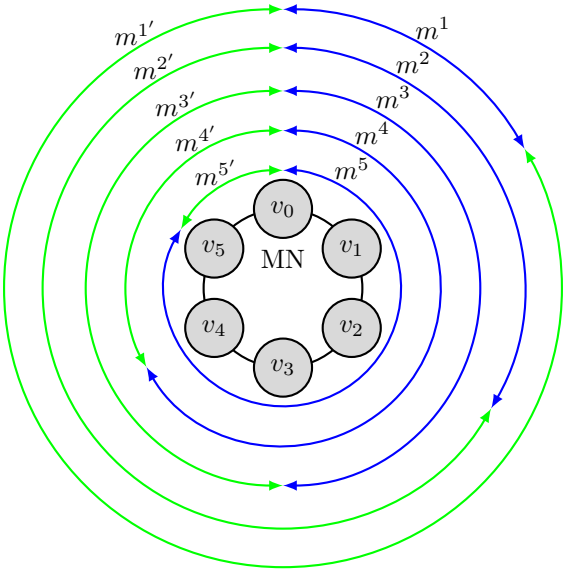


Figure 6.9: Dual-link SRLG fault localization in a ring network from a single MN. Node v_0 is the MN.

Proof. Without losing generality, assume that node v_0 is the MN. Let there be $(|E|-1)$ m-trails in each set, clockwise or anti-clockwise, to localize a single fault in $(|E|-1)$ links, and each node of the network except the MN be a destination node of an m-trail from each m-trail set. Now, we have to consider three cases.

Case 1: let the first faulty link be any link except (v_{n-1}, v_0) and the second faulty link be (v_{n-1}, v_0), which is the last link of the network considering links in clockwise direction from the MN. From the fault localization perspective, the ring network has two segments now. The clockwise segment includes all link except link (v_{n-1}, v_0) and the anti-clockwise segment has only one link, i.e., (v_{n-1}, v_0). Thus, according to Corollary 6.2.2, the first faulty link can be localized unambiguously by using the full set of clockwise m-trails and the second faulty link can be localized by using the single hop anti-clockwise m-trail from the MN.

Case 2: let the first faulty link be (v_0, v_1) and the second faulty link be any link except link (v_0, v_1), which is the first link of the network considering links in clockwise direction

from the MN. We can prove that both the first and second faulty links can be localized unambiguously using similar argument as in Case 1.

Case 3: the MN adjacent links, i.e., (v_0, v_1) and (v_{n-1}, v_0), are not faulty ones. In this case, the set of clockwise m-trails traversing the first faulty link and beyond will be disrupted. Thus, the MN will be able to localize only the first faulty link by using clockwise m-trails as per Theorem 6.2.3. Similarly, the MN will be able to localize only the second faulty link by using anti-clockwise m-trails.

Thus, we need no more than $|\mathfrak{M}| = 2(|E| - 1)$ m-trails to localize dual-link faults unambiguously from the MN.

Now, if either m-trail set has less than $|E| - 1$ m-trails, one or more nodes will not be a desitnation node of any m-trail from that set. As a result, faults in the adjacent links of those nodes cannot be localized unambigiously from the MN by using the m-trail set. Thus, each m-trail set must have at least $|E| - 1$ m-trails. □

6.3.1 M-Trail Allocation in Ring Networks

From the perspectives of clockwise or anti-clockwise set of m-trails, a ring network with $|E|$ links is a linear network with $|E|$ links spread out clockwise or anti-clockwise from the MN, respectively. As stated earlier, any fault in the last link considering the links clockwise from the MN will be localized only by the anti-clockwise m-trails and vice versa. Thus, each m-trail set will be responsible for fault localization in first $|E| - 1$ links of the ring network. As in linear networks, there will be $|E| - 1$ unique m-trails in each set of m-trails in ring networks because each node except the MN, i.e., $n - 1 = |E| - 1$ nodes, can be a destination node of only a single m-trail in the set. Refer to Theorem 6.2.6.

As $|E| - 1$ m-trails are required to localize single-link faults unambiguously by the clockwise (anti-clockwise) m-trails and there are $|E| - 1$ unique clockwise (anti-clockwise) m-trails in ring networks, we can enumerate m-trails in $\mathfrak{M}$ by inspection as described in Algorithm 6.3.

Example: Now, let us derive the sets of clockwise and anti-clockwise m-trails by inspection for a ring network with 4 nodes and 4 links, where the nodes of the network are numbered from v_0 to v_3 consecutively, and node v_0 is the MN. There are 6 m-trails: 3 in each set. $\mathfrak{M} = \{m^1, m^2, m^3, m^{1'}, m^{2'}, m^{3'}\}$. The destination nodes of the clockwise m-trails

Algorithm 6.3: M-Trail Allocation in Ring Networks

Input: $G(V, E)$ and MN
Output: $\mathfrak{M}$ and $\mathfrak{A}$
begin
1 Initialize $\mathfrak{M}$, $\mathfrak{A} \leftarrow \phi$
foreach *node* $v_j \in V \backslash$MN **do**
2 Form m-trail m^j using two unidirectional links, one in each direction, from each undirected link on the clockwise route from the MN to node v_j.
3 $\mathfrak{M}$, $\mathfrak{A} \leftarrow m^j$
4 Form m-trail $m^{j'}$ using two unidirectional links, one in each direction, from each undirected link on the anti-clockwise route from the MN to node v_j.
5 $\mathfrak{M}$, $\mathfrak{A} \leftarrow m^{j'}$
6 Return $\mathfrak{M}$ and $\mathfrak{A}$.

m^1, m^2, and m^3 are nodes v_1, v_2, and v_3, respectively. The destination nodes of the anti-clockwise m-trails $m^{1'}$, $m^{2'}$, and $m^{3'}$ are nodes v_1, v_2, and v_3, respectively. For instance, m^1 will traverse unidirectional links (v_0, v_1) and (v_1, v_0), and $m^{1'}$ will traverse unidirectional links (v_0, v_3), (v_3, v_2), (v_2, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_0). Table 6.3 shows ACT constructed from 6 m-trails found by inspection in the ring network. The decimal alarm codes (Dec.) of the single-link and dual-link SRLGs derived from the m-trails are unique. Thus, the sets of m-trails will be able to localize single-link and dual-link SRLG faults in the ring network.

Table 6.3: ACT for Dual-Link SRLG Fault Localization in a Ring Network with 4 Links

SRLG	$m^{1'}$	$m^{2'}$	$m^{3'}$	m^3	m^2	m^1	Dec.
(v_0, v_1)	0	0	0	1	1	1	7
(v_1, v_2)	1	0	0	1	1	0	38
(v_2, v_3)	1	1	0	1	0	0	52
(v_3, v_0)	1	1	1	0	0	0	56
$(v_0, v_1)(v_1, v_2)$	1	0	0	1	1	1	39
$(v_0, v_1)(v_2, v_3)$	1	1	0	1	1	1	55
$(v_0, v_1)(v_3, v_0)$	1	1	1	1	1	1	63
$(v_1, v_2)(v_2, v_3)$	1	1	0	1	1	0	54
$(v_1, v_2)(v_3, v_0)$	1	1	1	1	1	0	62
$(v_2, v_3)(v_3, v_0)$	1	1	1	1	0	0	60

6.3.2 M-Burst Launching Time Scheduling in Ring Networks

Theorem 6.3.3. *The lower bound of the fault localization latency T to localize dual-link SRLG faults from a single MN in a ring network is $2\delta(|E|-1)+L$.*

Proof. The lower bound can be achieved only when $L \leq \delta$. Bursts are launched from the MN along the clockwise and anti-clockwise m-trails independently. The bursts along each set of m-trails are launched in descending order of the lengths of the m-trails in the set; moreover, each burst is delayed exactly by δ ms from the launching time of its predecessor burst. In the beginning of a monitoring period, two bursts along the longest clockwise and anti-clockwise m-trails are launched.

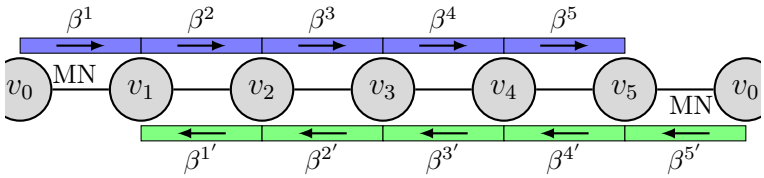
Consequently, each burst reaches its destination node, starts reversing traversal direction and reaches its next node in the reverse direction at the same time. Hence, there will be no collision. Figure 6.10a depicts the moment when all clockwise and anti-clockwise bursts reach their destination nodes and Figure 6.10b depicts the moment when all clockwise and anti-clockwise bursts reach next nodes from their destination nodes toward the MN.

As the longest m-trail of each set has the same length, the last bursts along the clockwise and anti-clockwise m-trails will return to the MN at the same moment. Therefore, the fault localization latency T will be equal to the burst traversal time along the longest m-trails. In other words, $T = s^l + tp^l + L = s^{l'} + tp^{l'} + L = 0 + \delta \cdot 2(|E|-1) + L = 2\delta(|E|-1) + L$.

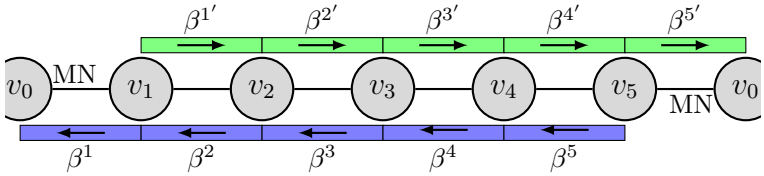
As the fault localization latency has to be greater than or equal to the burst traversal time along the longest m-trail, T cannot be less than $2\delta(|E|-1)+L$. . □

Theorem 6.3.4. *The upper bound of the fault localization latency T to localize dual-link SRLG faults from a single MN in a ring network is $(2\delta+L)(|E|-1)+\delta(|E|-2)$.*

Proof. The upper bound of the fault localization latency occurs when $L > \delta(|E|-2)$. There will be no interleaving of bursts along clockwise and anti-clockwise m-trails in any undirected link.



(a) at $t_{5\delta}$.



(b) at $t_{6\delta}$.

Figure 6.10: Burst traversal in a ring network to localize dual-link SRLG faults from a single MN. Node v_0 is the MN. Burst β^j (blue) traverses the links along clockwise m-trail m^j and burst $\beta^{j'}$ (green) traverses the links along anti-clockwise m-trail $m^{j'}$. $L = \delta$.

Bursts along each set of m-trails can be launched either ascending or descending order of the lengths of m-trails in the set with the same delay $(2\delta+L)(|E|-1)$. Refer to Theorem 6.2.8. Now, we have three options of burst selection: both sets of bursts in ascending order of the lengths of the corresponding m-trails, both sets of bursts in descending order of the lengths of the corresponding m-trails, or one set of bursts in ascending order and other set of bursts in descending order of the lengths of the corresponding m-trails.

If bursts along both the m-trail sets are launched in ascending order of the lengths of the corresponding m-trails, the bursts from two sets can be launched in parallel until the destination node of two bursts (one from each set) is the middle node of the network. Then bursts along one m-trail set cannot be launched until bursts along other m-trail set almost finished link traversal. Thus, the fault localization latency will be almost one and a half times of the delay $(2\delta + L)(|E| - 1)$ incurred by either set of bursts alone.

If bursts along both the m-trail sets are launched in descending order of the lengths of the corresponding m-trails, the burst along the shortest m-trail in the first m-trail set and the bursts along the longest m-trail of the second m-trail set can reach their common node

at the same time. As a result, the first burst along the second m-trail set can be launched only after the bursts along the first m-trail set almost finished link traversal. Thus, the fault localization latency will be almost double of the delay $(2\delta + L)(|E| - 1)$ incurred by either set of bursts alone.

If bursts along one m-trail set are launched in ascending order and bursts along other m-trail set are launched in descending order of the lengths of the corresponding m-trails, each pair of bursts (one from each set) can reach their common destination node at the same time. Without losing generality, assume that clockwise bursts are launched in descending order of the lengths of the corresponding m-trails. Launching of each subsequent clockwise burst has to be delayed by $2\delta + L$ from launching of its predecessor clockwise burst to avoid collision among the clockwise bursts while they are returning to the MN. When a burst reaches its destination node, all the bursts along longer m-trails, if any, will have passed through the node completely while they are returning to the MN.

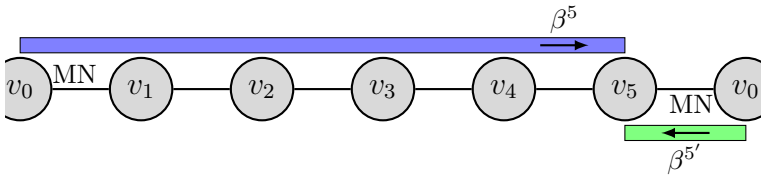
As an anti-clockwise burst can reach up to the destination node of the clockwise burst along the longest m-trail currently traversing links, the anti-clockwise bursts are launched back-to-back on ascending order of the lengths of the corresponding m-trails. In addition, the anti-clockwise burst along the shortest anti-clockwise m-trail is launched after appropriate delay in such way that the shortest anti-clockwise burst and the clockwise burst along the longest clockwise m-trail reach their common destination node at the same time. Consequently, each pair of clockwise and anti-clockwise bursts reaches their common destination node at the same time as shown in Figure 6.11. The clockwise and anti-clockwise bursts remain separated in time and space.

Again, without losing generality, let the burst along the longest clockwise m-trail be launched in the beginning of each monitoring period. Thus, the maximum fault localization latency of the bursts along the clockwise m-trails will be $s^l + tp^l + L = [0 + (2\delta + L)(|E| - 2)] + 2\delta + L = (2\delta + L)(|E| - 1)$.

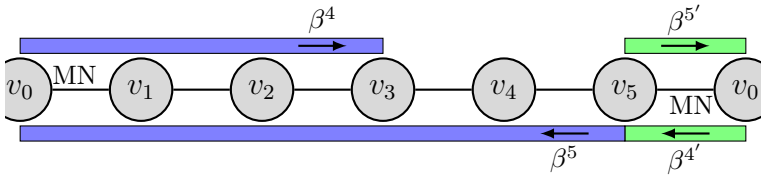
The delay between launching bursts along the longest clockwise and the shortest anti-clockwise m-trails should be $\delta(|E|-1)-\delta = \delta(|E|-2)$ such that both the bursts reach their common destination node v_{n-1} at the same time. Hence, the maximum fault localization latency of the bursts along the anti-clockwise m-trails will be $s^{l'} + tp^{l'} + L = [\delta(|E| - 2) +$

$L(|E|-2)] + 2\delta(|E|-1) + L = (2\delta + L)(|E|-1) + \delta(|E|-2)$.

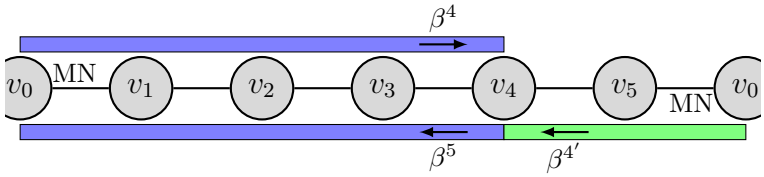
Thus, the upper bound of the fault localization latency T is $(2\delta + L)(|E|-1) + \delta(|E|-2)$. □



(a) at $t_{5\delta}$.



(b) at $t_{5\delta} + L$.



(c) at $t_{6\delta} + L$.

Figure 6.11: Burst traversal in a ring network to localize dual-link SRLG faults from a single MN. Node v_0 is the MN. Burst β^j (blue) traverses the links along clockwise m-trail m^j and burst $\beta^{j'}$ (green) traverses the links along anti-clockwise m-trail $m^{j'}$. $L \geq \delta(|E|-1)$.

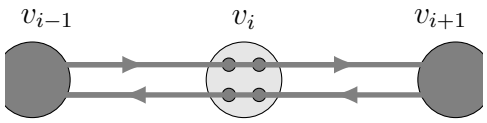
6.3.3 Node Switch Fabric Configuration Scheduling in Ring Networks

In the m-burst framework for ring networks, two dedicated wavelength channels (WLs) per undirected link are assigned for monitoring. One WL is for each direction. The

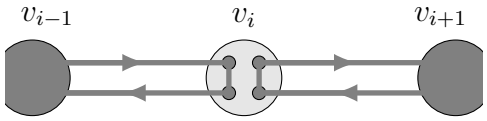
MN connects WLs to and from its clockwise adjacent node to the monitor for clockwise m-trails. The MN also connects WLs to and from its anti-clockwise adjacent node to the monitor for anti-clockwise m-trails. These connections will not be changed. We will consider $L > \delta(|E| - 2)$. Hence, $T = (2\delta + L)(|E| - 1) + \delta(|E| - 2)$.

Assume that at $t = 0$ the first burst launched is the burst along the longest clockwise m-trail. The burst along the shortest anti-clockwise m-trail is launched in such a way that these two bursts reach their common destination node at the same moment. Subsequent bursts along clockwise m-trails are launched in the descending order of the lengths of the corresponding m-trails. On the other hand, subsequent bursts along anti-clockwise m-trails are launched in the ascending order of the lengths of the corresponding m-trails.

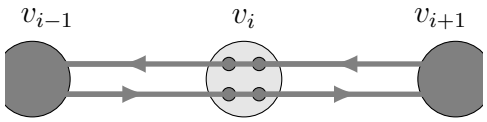
$\forall i$: $0 < i < n$, in the beginning of monitoring, i.e. at $t = 0$, intermediate node v_i connects the WL from node v_{i-1} with the WL to node v_{i+1} and vice versa as shown in Figure 6.12a. Note that node v_n and node v_0 is the same node.



(a) at $t = 0$.



(b) at $t = (2\delta + L)(|E| - 1 - i) + \delta i + kT$.



(c) at $t = (2\delta + L)(|E| - 1 - i) + \delta i + L + kT$.

Figure 6.12: Configuration of node switch fabric of intermediate node v_i for dual-link SRLG fault localization from a single MN in ring networks. Node v_0 is the MN.

At $t = (2\delta + L)(|E| - 1 - i) + \delta i + kT$ where $k = 0, 1, \ldots$, intermediate node v_i connects two WLs from and to node v_{i-1} and two WLs from and to node v_{i+1} as shown in Figure 6.12b. At $t = (2\delta + L)(|E| - 1 - i) + \delta i + L + kT$, intermediate node v_i connects the WL from node v_{i-1} with the WL to node v_{i+1} and vice versa as shown in Figure 6.12c.

The configuration of the node switch fabric will be changed periodically between the states shown in Figure 6.12b and Figure 6.12c.

6.3.4 Numerical Results

Numerical experiments are conducted in ring networks with 3 to 12 links. Two sets of m-trails, clockwise and anti-clockwise, are derived in each network by inspection for single-link and dual-link SRLG fault localization. Then, the burst starting times from the MN along the m-trails are scheduled using the heuristic Algorithm 4.2, given in Chapter 4. The burst propagation delay δ through a unidirectional link is 2 ms. The burst length L is varied from 1 ms to 32 ms in 7 steps. The expected and real fault localization latencies according to the burst lengths in each network are shown in columns E and A of Table 6.4, respectively. The actual fault localization latencies are consistent with the theoretically expected values.

Table 6.4: Fault Localization Latency T (ms) in Ring Networks

# of	$L = 1$ ms		$L = 2$ ms		$L = 4$ ms		$L = 8$ ms		$L = 16$ ms		$L = 20$ ms		$L = 32$ ms	
Links	E	A	E	A	E	A	E	A	E	A	E	A	E	A
3	9	9	10	10	18	18	26	26	42	42	50	50	74	74
4	13	13	14	14	≤ 28	20	40	40	64	64	76	76	112	112
5	17	17	18	18	≤ 38	30	54	54	86	86	102	102	150	150
6	21	21	22	22	≤ 48	32	≤ 68	60	108	108	128	128	188	188
7	25	25	26	26	≤ 58	42	≤ 82	66	130	130	154	154	226	226
8	29	29	30	30	≤ 68	44	≤ 96	80	152	152	180	180	264	264
9	33	33	34	34	≤ 78	54	≤ 110	90	174	174	206	206	302	302
10	37	37	38	40	≤ 88	56	≤ 124	100	≤ 196	188	232	232	340	340
11	41	41	42	44	≤ 98	66	≤ 138	106	≤ 218	202	258	258	378	378
12	45	45	46	48	≤ 108	68	≤ 152	120	≤ 240	218	≤ 284	276	416	416

If any actual latency in a ring network with $|E|$ links is either upper or lower boundary value, the value in Table 6.4 can be calculated from the corresponding value in a linear network with $(|E| - 1)$ links in Table 6.2 and vice versa.

6.4 Dual-Link SRLG Fault Localization in Mesh Networks

In this section, we shall deal with dual-link SRLG fault localization in at least 2-connected all-optical mesh networks. We have shown in Section 6.3 that dual-link SRLG faults can be localized in all-optical ring networks using the sets of clockwise and anti-clockwise m-trails from a single MN. The nearer faulty link from the MN in clockwise direction is called the first faulty link and the nearer faulty link from the MN in anti-clockwise direction is called the second faulty link. The clockwise and anti-clockwise sets of m-trails will localize the first and the second faulty links, respectively. Faulty clockwise (anti-clockwise) MN adjacent link will be localized by only the clockwise (anti-clockwise) m-trail set. If a single-link that is not an MN adjacent link becomes faulty, both the m-trail sets will localize the faulty link.

Now, we can derive a set of cycles C in a mesh network such that each undirected link of the network is traversed by at least one cycle in C. Note that in a 2-connected network, a pair of link-disjoint paths exists between each pair of nodes as per Menger theorem [3]. Thus, for each undirected link, we can derive a cycle that traverse the link and the MN. The set of cycle is known as cycle cover.

In addition, each cycle in C can be viewed as an independent virtual ring network for fault monitoring purposes only. A virtual ring network can be realized using a separate pair of WLs (one WL in each direction) in each undirected link traversed by a particular cycle $c^j \in C$. The clockwise and anti-clockwise m-trails derived in the virtual ring network are called *nested m-trails*.

Theorem 6.4.1. *If each undirected link of an all-optical mesh network is traversed by at least one cycle in cycle cover C, dual-link SRLG faults can be localized unambiguously from the MN by using the sets of nested m-trails.*

Proof. Each link of the network will be traversed by at least one cycle in C and each cycle in C is considered as an independent virtual ring network. For a failure event that involves two undirected links (u, v) and (w, x), we have to consider two cases only.

Case 1: assume that both faulty links are traversed by only a single cycle in C, i.e., the faulty links (u, v) and (w, x) are traversed by a cycle $c^j \in C$. Let (u, v) and (w, x) be the first and the second faulty links of c^j, respectively. In this case, the first and the second

faulty links will be localized by the clockwise and anti-clockwise nested m-trail sets of c^j, respectively.

Case 2: assume that each faulty link is traversed by only a single cycle in C, i.e., the faulty links (u, v) and (w, x) are traversed by two cycles c^i and c^j, respectively. Now, (u, v) is the single faulty link in c^i. Thus, (u, v) will be localized as a faulty link by the clockwise, the anti-clockwise, or both nested m-trail sets of c^i. Similarly, (w, x) is the single faulty link in c^j. Thus, (w, x) will be localized as a faulty link by the clockwise, the anti-clockwise, or both nested m-trail sets of c^j.

In other cases such as multiple cycles traversing the pair of faulty links or multiple cycles traversing each faulty link but not both, the faulty links will be localized unambiguously by the sets of nested m-trails of the respective cycles. □

Theorem 6.4.2. *If the sets of nested m-trails are used to localize dual-link SRLG faults, the fault localization latency T is determined by the number of links in the largest cycle in C, i.e., by $|c^j|_{max}$.*

Proof. As each cycle in C is allocated a pair of WLs in each undirected link traversed by the cycle, the cycles in C will act like independent virtual ring networks during burst traversal along nested m-trails. Thus, burst scheduling along the sets of clockwise and anti-clockwise nested m-trails belong to cycle $c^j \in C$ can be done independent of the burst scheduling along the nested m-trails belong to any cycle $c^i \in C \backslash c^j$.

If we consider c^j as a virtual ring network, the upper bound of the fault localization latency T^j in cycle c^j will be $(2\delta + L)(|c^j| - 1) + \delta(|c^j| - 2)$ as per Theorem 6.3.4, where δ is the burst propagation delay in ms through any unidirectional link of the network and L is the burst length in ms. In other words, the upper bound of the fault localization latency in cycle c^j is determined by the number of undirected links traversed by the cycle.

Assume that the first burst in each cycle is launched at the same time in each monitoring period. Thus, the maximum fault localization latency T in the mesh network will be determine by the largest cycle in C, i.e., by $|c^j|_{max}$. □

Now, we propose a novel approach called *nested m-trail* method in Algorithm 6.4, which is self-explanatory, for dual-link SRLG fault localization in all-optical mesh networks. The

method is based on the combination of the cycle cover of mesh networks [64][72] and the m-trail [63] methodologies.

The nested m-trails will be deployed in the m-burst framework. However, instead of using one WL in each unidirectional link (u, v) if any m-trail traverses the link as per the original m-burst framework, we have to provide a pair of WLs in each undirected link (u, v) for each cycle $c^j \in C$ that is traversing the undirected link in the nested m-trail method.

Moreover, the number of monitors is not determined by the nodal degree of the MN as per the original m-burst framework. The number of monitors is determined by the number of cycles in the set to cover a mesh network. Specifically, the number of monitors required by the nested m-trail method is equal to $2 * |C|$ for dual-link SRLG fault localization in all-optical mesh network.

Algorithm 6.4: Nested M-Trail Method for Dual-link SRLG Fault Localization

Input: $G(V, E)$ and MN
Output: C, $\mathfrak{M}, \mathfrak{A}$, S, and T
begin
1 Derive a set of cycles C in mesh network G such that each cycle traverses the MN, and each link of the network is traversed by at least one cycle in C.
foreach *cycle* $c^j \in C$ **do**
2 Derive by inspection the sets of clockwise and anti-clockwise nested m-trails in cycle c^j by considering c^j as a virtual ring network. The MN has to be the source node of all nested m-trails.
Assign the nested m-trails to $\mathfrak{M}$ and $\mathfrak{A}$.
3 Determine independently the burst launching times S^j from the MN along the nested m-trails in cycle c^j.
Assign the burst launching times in S^j to S.
foreach *node* $u \in V$*: cycle* c^j *visits node* u **do**
4 Determine the burst arrival times to u and departure times from u along the nested m-trails of cycle c^j.
5 Determine fault localization latency T from the number of links in the largest cycle, i.e., $|c^j|_{max}$.
foreach *node* $u \in V$ **do**
6 Determine node switch fabric configuration schedule based on the burst arrival and departure times along the nested m-trails of each cycle visiting node u.
7 Return C, $\mathfrak{M}, \mathfrak{A}$, S, and T.

Once the cycle cover C is derived in a mesh network, we shall take the nested m-trail sets of each cycle $c^j \in C$, the burst launching times from the MN of the nested m-trail sets of c^j, and the schedule of periodic node switch fabric configuration of each node visited

by c^j from the results derived in Section 6.3 for the ring network having the same length of c^j. However, the following examples will show how we can derive relevant results by inspection.

6.4.1 M-Trail Allocation for Dual-Link Fault Localization

M-trail allocation problem in nested m-trail method has two steps: finding cycle cover of the mesh network and deriving nested m-trails from each cycle in the cycle cover set.

Example: Figure 6.13 shows a cycle cover set $C = \{c^0, c^1, c^2\}$ of the network with 4 nodes and 6 links. Now, each cycle in C is considered as an independent virtual ring network for fault monitoring purposes only.

The sets of clockwise and anti-clockwise nested m-trails $\{m^{01}, m^{02}\}$ and $\{m^{02'}, m^{01'}\}$, respectively, can be derived by inspection from cycle c^0. The nested m-trails m^{01} and m^{02} traverse unidirectional links (v_0, v_1) and (v_1, v_0), and (v_0, v_1), (v_1, v_2), (v_2, v_1) and (v_1, v_0), respectively. The nested m-trails $m^{02'}$ and $m^{01'}$ traverse unidirectional links (v_0, v_2) and (v_2, v_0), and (v_0, v_2), (v_2, v_1), (v_1, v_2) and (v_2, v_0), respectively. Similarly, the sets of clockwise and anti-clockwise nested m-trails $\{m^{12}, m^{13}\}$ and $\{m^{13'}, m^{12'}\}$, respectively, can be derived from cycle c^1 and the sets of clockwise and anti-clockwise nested m-trails $\{m^{23}, m^{21}\}$ and $\{m^{21'}, m^{23'}\}$, respectively, can be derived from cycle c^2.

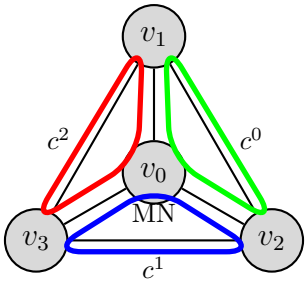


Figure 6.13: Cycle cover for dual-link SRLG fault localization in a mesh network.

The ACT of the network using the sets of nested m-trails of the cycles in C is given in Table 6.5. The set of SRLGs consists of every single-link and all combination of two links in the network. The decimal alarm code (Dec.) of each SRLG is unique.

Table 6.5: ACT for Dual-Link SRLG Fault Localization in a Mesh Network with 4 Nodes and 6 Links

SRLGs	$m^{23'}$	$m^{21'}$	m^{21}	m^{23}	$m^{12'}$	$m^{13'}$	m^{13}	m^{12}	$m^{01'}$	$m^{02'}$	m^{02}	m^{01}	Dec.
(0, 1)	1	1	0	0	0	0	0	0	0	0	1	1	3075
(0, 2)	0	0	0	0	0	0	1	1	1	1	0	0	60
(0, 3)	0	0	1	1	1	1	0	0	0	0	0	0	960
(1, 2)	0	0	0	0	0	0	0	0	1	0	1	0	10
(1, 3)	1	0	1	0	0	0	0	0	0	0	0	0	2560
(2, 3)	0	0	0	0	1	0	1	0	0	0	0	0	160
(0, 1)(0, 2)	1	1	0	0	0	0	1	1	1	1	1	1	3135
(0, 1)(0, 3)	1	1	1	1	1	1	0	0	0	0	1	1	4035
(0, 1)(1, 2)	1	1	0	0	0	0	0	0	1	0	1	1	3083
(0, 1)(1, 3)	1	1	1	0	0	0	0	0	0	0	1	1	3587
(0, 1)(2, 3)	1	1	0	0	1	0	1	0	0	0	1	1	3235
(0, 2)(0, 3)	0	0	1	1	1	1	1	1	1	1	0	0	1020
(0, 2)(1, 2)	0	0	0	0	0	0	1	1	1	1	1	0	62
(0, 2)(1, 3)	1	0	1	0	0	0	1	1	1	1	0	0	2620
(0, 2)(2, 3)	0	0	0	0	1	0	1	1	1	1	0	0	188
(0, 3)(1, 2)	0	0	1	1	1	1	0	0	1	0	1	0	970
(0, 3)(1, 3)	1	0	1	1	1	1	0	0	0	0	0	0	3008
(0, 3)(2, 3)	0	0	1	1	1	1	1	0	0	0	0	0	992
(1, 2)(1, 3)	1	0	1	0	0	0	0	0	1	0	1	0	2570
(1, 2)(2, 3)	0	0	0	0	1	0	1	0	1	0	1	0	170
(1, 3)(2, 3)	1	0	1	0	1	0	1	0	0	0	0	0	2720

6.4.2 M-Burst Launching Time Scheduling

As each cycle of the cycle cover set C will be treated as an independent virtual ring network for fault monitoring purposes in the nested m-trail method, the burst scheduling can be done using the same technique as done in ring networks.

Example: Now consider c^0 in Figure 6.13 as an independent virtual ring network. The bursts along the clockwise nested m-trails of c^0 will be launched from the MN in descending order of the lengths of the corresponding nested m-trails and bursts along the anti-clockwise nested m-trails of c^0 will be launched from the MN in ascending order of the lengths of the corresponding nested m-trails. Moreover, each pair of clockwise and anti-clockwise bursts

of c^0 having common destination node will reach the node at the same moment.

Burst β^{02} along clockwise nested m-trail m^{02} will be launched in the beginning of each monitoring period, i.e., $s^{02} = 0$. If burst $\beta^{02'}$ along anti-clockwise nested m-trail $m^{02'}$ is launched at δ, i.e., $s^{02'} = \delta$, bursts β^{02} and $\beta^{02'}$ will reach their common destination node v_2 at the same moment. To avoid collision with burst β^{02}, burst β^{01} along clockwise nested m-trail m^{01} will be launched after $2\delta + L$ ms, i.e., $s^{01} = 2\delta + L$, but burst $\beta^{01'}$ along anti-clockwise nested m-trail $m^{01'}$ can be launched immediately after burst $\beta^{02'}$ avoiding collision, i.e., $s^{01'} = \delta + L$. Hence, bursts β^{01} and $\beta^{01'}$ will reach their common destination node v_1 at the same moment.

Similarly, we can find burst launching times of the nested m-trails of cycles c^1 and c^2 by inspection.

The first burst in every cycle will be launched in the beginning of each monitoring period simultaneously; this is possible because each cycle will be formed by using a pair of WLs in each undirected link visited by the cycle. Thus, the fault localization latency T will be determine by the largest cycle in the cycle cover set C. In this example, $|c^j|_{max} = 3$. If $\delta = 2$ ms and $L = 20$ ms, $L > \delta(|c^j|-2)$. Thus, $T = (2\delta+L)(|c^j|_{max}-1)+\delta(|c^j|_{max}-2) = 50$ ms.

6.4.3 Node Switch Fabric Configuration Scheduling

Configuration of switch fabrics of the nodes traversed by each cycle in C will be done independently. The MN connects clockwise and anti-clockwise monitors with the WLs of the cycle in clockwise and anti-clockwise adjacent links, respectively. These connections will not be changed. Each node except the MN visited by the cycle will change its switch fabric configuration two times during each monitoring period: when the pair of bursts having the node as the common destination reaches the node, and after the pair of bursts leaves the node completely.

Example: Again, consider c^0 in Figure 6.13 as an independent virtual ring network. Cycle c^0 visits nodes v_0, v_1, and v_2. Node v_0 is the MN. The MN connects the incoming and outging WLs from and to undirected link (v_0, v_1) of c^0 to the cycle's clockwise monitor. The MN also connects the incoming and ougoing WLs from and to undirected link (v_0, v_2) of c^0 to the cycle's anti-clockwise monitor. In the beginning of monitoring, both nodes v_1 and v_2 connect each incoming WL of c^0 from one adjacent node to the outgoing WL of c^0 to another adjacent node.

At $2\delta + kT$ ms, where T is the fault localization latency and $k = 0, 1, \ldots$, node v_2 connects the incoming and outgoing WLs of c^0 from and to node v_0 as well as the incoming and outgoing WLs of c^0 from and to node v_1. Thus, bursts β^{02} and $\beta^{02'}$ will be able to traverse node v_2 without collision. At $2\delta + L + kT$ ms, node v_2 connects the incoming WL of c^0 from node v_0 to the outgoing WL of c^0 to node v_1 and vice versa.

At $3\delta + L + kT$ ms, node v_1 connects the incoming and outgoing WLs of c^0 from and to node v_0 as well as the incoming and outgoing WLs of c^0 from and to node v_2. Thus, bursts β^{01} and $\beta^{01'}$ will be able to traverse node v_1 without collision. At $3\delta + 2L + kT$ ms, node v_1 connects the incoming WL of c^0 from node v_0 to the outgoing WL of c^0 to node v_2 and vice versa.

Similarly, we can find the schedule for periodic change of the node switch fabric configuration of each node visited by cycles c^1 and c^2 by inspection.

6.5 Multi-Link SRLG Fault Localization in Mesh Networks

We have proposed the DMCF method in Section 5.3. The multi-link SRLG fault localization method is based on disjoint paths. In all-optical $(d + 1)$-connected mesh networks, faulty multi-link SRLG with up to d links can be localized from a single MN if each undirected link is traversed by at least $(d + 1)$ m-trails that are disjoint in other links of the network.

In Section 6.4, we propose the *nested m-trail* method in Algorithm 6.4 for dual-link SRLG fault localization in all-optical 2-connected mesh networks using cycle cover, where each link is traversed by at least one cycle.

In this section, we extend *nested m-trail* method for multi-link SRLG fault localization in all-optical mesh networks, where any SRLG can have up to d links. We shall show that d link-disjoint paths are needed for each undirected link to localize faulty multi-link SRLG with up to d links. As there are at least d link-disjoint paths from the MN to each node in a d-connected network as per Menger theorem [3], the *nested m-trail* method can be deployed in d-connected networks to localize d faulty links. Note that in *nested m-trail* method, a cycle that traverses the MN and a particular link can be viewed as 2 link-disjoint paths from the MN to the link.

Theorem 6.5.1. *If d is even and each undirected link is traversed by at least $\frac{d}{2}$ cycles that are disjoint in other links and visit the MN, faulty SRLG up to d links can be localized unambiguously from the MN using the nested m-trails.*

Proof. $\forall(u,v) \in E$: (u,v) is a faulty link. As maximum number of faulty link is d, there will be at most $(d-1)$ other faulty links that can disrupt the cycles traversing (u,v).

As each link is traversed by at least $\frac{d}{2}$ cycles that are disjoint in other links, (u,v) has at least d disjoint paths to the MN. Now, the other $(d-1)$ faulty links can disrupt only $(d-1)$ paths to the MN. Thus, (u,v) will have at least one uninterrupted path to the MN during the fault event. Hence, (u,v) will be the first, the second, or the single faulty link of at least one cycle. As a result, (u,v) will be identified unambiguously as faulty link from the MN using the nested m-trails of the cycle. □

Corollary 6.5.2. *If d is odd and each undirected link is traversed by at least one trail and $\lfloor \frac{d}{2} \rfloor$ cycles that are disjoint in other links and visit the MN, faulty SRLG up to d links can be localized unambiguously from the MN using the nested m-trails.*

Theorem 6.5.3. *If the sets of nested m-trails in virtual ring networks are used to localize multi-link SRLG faults, the fault localization latency T is determined by the number of links in the largest cycle in C, i.e., by $|c^j|_{max}$.*

Proof. Since each virtual ring network is considered as an independent network, the proof of this theorem is the same as the proof of Theorem 6.4.2. □

Corollary 6.5.4. *If the sets of nested m-trails in virtual ring or linear networks are used to localize multi-link SRLG faults, the fault localization latency T is determined by the number of links either in the largest cycle or in the largest trail of C, i.e., by $|c^j|_{max}$ or $|t^j|_{max}$.*

Next, we propose the *nested m-trail* method for multi-link SRLG fault localization in all-optical mesh networks in Algorithm 6.5, which is self-explanatory. The method is also based on the combination of the cycle cover of mesh networks [64][72] and the m-trail [63] methodologies. Algorithm 6.5 is a logical extension of Algorithm 6.4.

Algorithm 6.5: Nested M-Trail Method for Multi-Link SRLG Fault Localization

Input: $G(V, E)$, MN, and d
Output: C, $\mathfrak{M}, \mathfrak{A}$, S, and T
begin

Derive a set of cycles/trails C in mesh network G such that each undirected link is traversed by at least $\frac{d}{2}$ cycles if d is even, or by at least $\lfloor \frac{d}{2} \rfloor$ cycles and one trail if d is odd, where the trail (if any) and the cycles are mutually disjoint in other links.
Each cycle/trail in C must traverse the MN.
foreach *cycle* $c^j \in C$ **do**
    Derive by inspection the sets of clockwise and anti-clockwise nested m-trails in cycle c^j by considering c^j as a virtual ring network. The MN has to be the source node of all nested m-trails.
    Assign the nested m-trails to $\mathfrak{M}$ and $\mathfrak{A}$.
    Determine independently the burst launching times S^j from the MN along the nested m-trails in cycle c^j.
    Assign the burst launching times in S^j to S.
    foreach *node* $u \in V$*: cycle* c^j *visits node* u **do**
        Determine the burst arrival times to u and departure times from u along the nested m-trails of cycle c^j.
foreach *trail* $t^j \in C$ **do**
    Derive by inspection the set of nested m-trails in trail t^j by consider t^j as a virtual linear network. The MN has to be the source node of all nested m-trails.
    Assign the nested m-trails to $\mathfrak{M}$ and $\mathfrak{A}$.
    Determine independently the burst launching times S^j from the MN along the nested m-trails in trail t^j.
    Assign the burst launching times in S^j to S.
    foreach *node* $u \in V$*: trail* t^j *visits node* u **do**
        Determine the burst arrival times to u and departure times from u along the nested m-trails of trail t^j.
Determine fault localization latency T from the number of links in the largest cycle/trail, i.e., from $|c^j|_{max}$ or $|t^j|_{max}$.
foreach *node* $u \in V$ **do**
    Determine node switch fabric configuration schedule based on the burst arrival and departure times along the nested m-trails of each cycle/trail visiting node u.
Return C, $\mathfrak{M}, \mathfrak{A}$, S, and T.

As in dual-link SRLG fault localization, the nested m-trails will be deployed in the m-burst framework. Similarly, instead of using one WL in each unidirectional link (u, v) if any m-trail traverses the link as per the original m-burst framework, we have to provide a pair of WLs in each undirected link (u, v) for each cycle c^i or each trail t^j in C that is

traversing the undirected link.

The number of monitors is determined by the number of cycles and trails in C. Specifically, the number of monitors is equal to $\big(2 * |C \backslash t| + |C \backslash c|\big)$, where c and t are the subsets of all cycles and all trails in C, respectively.

Again, we shall take the nested m-trail sets of each cycle $c^j \in C$, the burst launching times from the MN of the nested m-trail sets of c^j, and the schedule of periodic node switch fabric configuration of each node visited by c^j from the results derived in Section 6.3 for the ring network having the same length of c^j. Similarly, for each trail $t^j \in C$, we shall take the abovementioned results derived in Section 6.2 for the linear network having the same length of t^j. However, the following examples will show how we can derive relevant results by inspection.

6.5.1 M-Trail Allocation for Multi-Link Fault Localization

As in dual-link SRLG fault localization, the first step to solve m-trail allocation problem in the nested m-trail method for multi-link SRLG fault localization is to find a set of cycles/trails C such that each undirected link of the network is traversed by at least $\frac{d}{2}$ cycles that are disjoint in other links if d is even. If d is odd, each undirected link of the network has to be traversed by at least $\lfloor \frac{d}{2} \rfloor$ cycles and one trail that are also disjoint in other links. As stated earlier, a cycle is equivalent to two trails during disjoint cycle/trail enumeration. The second step is also similar to that of dual-link SRLG fault localization: each cycle and trail in C is considered as a virtual ring and linear network, respectively, and nested m-trails are derived in the cycle and trail.

The following two examples will explain the process of finding the sets of nested m-trails by inspection from each cycle/trail in C.

SRLG up to $d = 4$ links Figure 6.14 shows a set of cycles $C = \{c^0, c^1, c^2, c^3\}$ that covers the network with 5 nodes and 8 links to localize single-link and multi-link SRLG faults. Each multi-link SRLG can have up to four links of the network that are disjoint from the MN, i.e., $d = 4$. Each MN adjacent undirected link is traversed by at least one cycle. Each undirected link of the network that is disjoint from the MN is traversed by $\frac{d}{2} = 2$ cycles that are disjoint in other links.

Now, each cycle $c^j \in C$ is considered as an independent ring network for fault monitoring purposes only. The sets of clockwise and anti-clockwise nested m-trails $\{m^{04}, m^{01}, m^{02}\}$

and $\{m^{02'}, m^{01'}, m^{04'}\}$, respectively, can be derived by inspection from cycle c^0. Nested m-trail m^{04} traverses unidirectional links (v_0, v_4) and (v_4, v_0), nested m-trail m^{01} traverses unidirectional links (v_0, v_4), (v_4, v_1), (v_1, v_4) and (v_4, v_0), and nested m-trail m^{02} traverses unidirectional links (v_0, v_4), (v_4, v_1), (v_1, v_2), (v_2, v_1), (v_1, v_4), and (v_4, v_0). Nested m-trail $m^{02'}$ traverses unidirectional links (v_0, v_2) and (v_2, v_0), nested m-trail $m^{01'}$ traverses unidirectional links (v_0, v_2), (v_2, v_1), (v_1, v_2) and (v_2, v_0), and nested m-trail m^{04} traverses unidirectional links (v_0, v_2), (v_2, v_1), (v_1, v_4), (v_4, v_1), (v_1, v_2), and (v_2, v_0).

Similarly, the sets of clockwise and anti-clockwise nested m-trails $\{m^{11}, m^{12}, m^{13}\}$ and $\{m^{13'}, m^{12'}, m^{11'}\}$, respectively, can be derived from cycle c^1, the sets of clockwise and anti-clockwise nested m-trails $\{m^{22}, m^{23}, m^{24}\}$ and $\{m^{24'}, m^{23'}, m^{22'}\}$, respectively, can be derived from cycle c^2, and the sets of clockwise and anti-clockwise nested m-trails $\{m^{33}, m^{34}, m^{31}\}$ and $\{m^{31'}, m^{34'}, m^{33'}\}$, respectively, can be derived from cycle c^3.

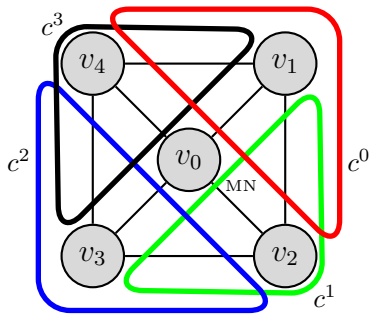


Figure 6.14: Cycle cover for multi-link SRLG fault localization in a mesh network, where $d = 4$.

The ACT of the network using the sets of nested m-trails derived from each cycle $c^j \in C$ is given in Table 6.6. The set of SRLGs consists of single-link and multi-link SRLGs. Each multi-link SRLG are disjoint from the MN. There are in total 19 SRLGs: 8 single-link SRLGs, 6 dual-link SRLGs, 4 triple-link SRLGs, and 1 quadruple-link SRLGs. The decimal alarm code of each SRLG is unique.

SRLG up to $d = 3$ links Figure 6.15 shows a set of cycles and trails $C = \{c^0, c^1, c^2, c^3, t^4, t^5, t^6, t^7\}$ that covers the network with 5 nodes and 8 links to localize multi-link SRLG faults. Each MN adjacent undirected link is traversed by two cycles that are disjoint in other links, and each of the remaining links is traversed by $\lfloor \frac{d}{2} \rfloor = 1$ cycle and 1 trail that are disjoint in other links.

Table 6.6: ACT for Multi-Link SRLG Fault Localization in a Mesh Network with 5 Nodes and 8 Links, where $d = 4$

SRLGs	$m^{33'}$	$m^{34'}$	$m^{31'}$	m^{31}	m^{34}	m^{33}	$m^{22'}$	$m^{23'}$	$m^{24'}$	m^{24}	m^{23}	m^{22}	$m^{11'}$	$m^{12'}$	$m^{13'}$	m^{13}	m^{12}	m^{11}	$m^{04'}$	$m^{01'}$	$m^{02'}$	m^{02}	m^{01}	m^{04}	Dec.
(0, 1)	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	14,680,512
(0, 2)	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	28,728
(0, 3)	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1,838,592
(0, 4)	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	229,383
(1, 2)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	1	0	0	2,484
(1, 4)	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	13,631,526
(2, 3)	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	158,976
(3, 4)	1	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10,174,464
(1, 2)(1, 4)	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	1	1	0	13,633,974
(1, 2)(2, 3)	0	0	0	0	0	0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	159,156
(1, 2)(3, 4)	1	0	0	1	1	0	1	1	0	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0	10,176,948
(1, 4)(2, 3)	1	1	0	1	0	0	1	0	0	1	1	0	1	1	0	1	0	0	1	0	0	1	1	0	13,790,502
(1, 4)(3, 4)	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	14,368,806
(2, 3)(3, 4)	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	10,185,984
(1, 2)(1, 4)(2, 3)	1	1	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	13,790,646
(1, 2)(1, 4)(3, 4)	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1	0	14,371,254
(1, 2)(2, 3)(3, 4)	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	10,186,164
(1, 4)(2, 3)(3, 4)	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	1	0	14,380,326
(1, 2)(1, 4)(2, 3)(3, 4)	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	14,380,470

Now, each cycle $c^j \in C$ is considered as an independent ring network for fault monitoring purposes only. The sets of clockwise and anti-clockwise nested m-trails $\{m^{01}, m^{02}\}$ and $\{m^{02'}, m^{01'}\}$, respectively, can be derived by inspection from cycle c^0. The nested m-trails m^{01} and m^{02} traverse unidirectional links (v_0, v_1) and (v_1, v_0), and (v_0, v_1), (v_1, v_2), (v_2, v_1) and (v_1, v_0), respectively. The nested m-trails $m^{02'}$ and $m^{01'}$ traverse unidirectional links (v_0, v_2) and (v_2, v_0), and (v_0, v_2), (v_2, v_1), (v_1, v_2) and (v_2, v_0), respectively.

Similarly, the sets of clockwise and anti-clockwise nested m-trails $\{m^{12}, m^{13}\}$ and $\{m^{13'}$, $m^{12'}\}$, respectively, can be derived from cycle c^1, the sets of clockwise and anti-clockwise nested m-trails $\{m^{23}, m^{24}\}$ and $\{m^{24'}, m^{23'}\}$, respectively, can be derived from cycle c^2, and the sets of clockwise and anti-clockwise nested m-trails $\{m^{34}, m^{31}\}$ and $\{m^{31'}, m^{34'}\}$, respectively, can be derived from cycle c^3.

Moreover, each trail $t^j \in C$ is considered as an independent linear network for fault monitoring purposes only. The set of nested m-trails $\{m^{44}, m^{41}, m^{42}\}$ can be derived by inspection from cycle t^4. Nested m-trail m^{44} traverses unidirectional links (v_0, v_4) and (v_4, v_0),

nested m-trail m^{41} traverses unidirectional links (v_0, v_4), (v_4, v_1), (v_1, v_4) and (v_4, v_0), and nested m-trail m^{42} and traverses unidirectional links (v_0, v_4), (v_4, v_1), (v_1, v_2), (v_2, v_1), (v_1, v_4) and (v_4, v_0).

Similarly, the sets of nested m-trails $\{m^{51}, m^{52}, m^{53}\}$, respectively, can be derived from cycle t^5, the sets of nested m-trails $\{m^{62}, m^{63}, m^{64}\}$, respectively, can be derived from cycle t^6, and the sets of nested m-trails $\{m^{73}, m^{74}, m^{71}\}$ can be derived from cycle t^7.

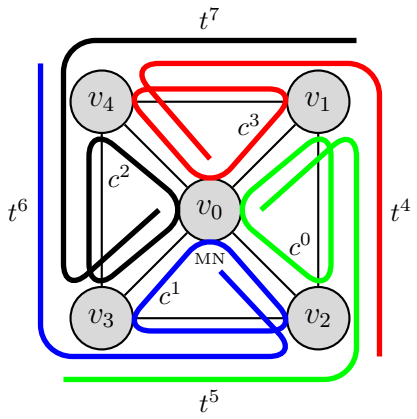


Figure 6.15: Cycle cover for multi-link SRLG fault localization in a mesh network, where $d = 3$.

The ACT of the network using the sets of nested m-trails derived from the cycles and trails in C is given in Table 6.7. The set of SRLGs consists of single-link and multi-link SRLGs. Each multi-link SRLG can have up to three links of the network, i.e., $d = 3$. There are in total 92 SRLGs: 8 single-link SRLGs, 28 dual-link SRLGs, and 56 triple-link SRLGs. The decimal alarm code of each SRLG is unique. However, due to space constraints, all multi-link SRLGs that are disjoint from the MN and all single-link SRLGs are shown in the ACT.

6.5.2 M-Burst Launching Time Scheduling

Burst scheduling for each cycle/trail will be done independently. For each cycle $c^j \in C$, the burst launching times along nested m-trails of c^j will be done using the same techniques as described in subsection 6.4.2. Now, as each trail $t^j \in C$ will be treated as an independent virtual linear network for fault monitoring purposes in the nested m-trail method, the

Table 6.7: ACT for Multi-Link SRLG Fault Localization in a Mesh Network with 5 Nodes and 8 Links, where $d = 3$

SRLGs	m^{71}	m^{74}	m^{73}	m^{64}	m^{63}	m^{62}	m^{53}	m^{52}	m^{51}	m^{42}	m^{41}	m^{44}	m^{34}	m^{31}	m^{31}	m^{34}	m^{23}	m^{24}	m^{24}	m^{23}	m^{12}	m^{13}	m^{13}	m^{12}	m^{01}	m^{02}	m^{02}	m^{01}	Dec.
(0, 1)	0	0	0	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3,719,171
(0, 2)	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	29,360,188
(0, 3)	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	234,881,984
(0, 4)	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	474,112
(1, 2)	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	3,407,882
(1, 4)	1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	134,651,904
(2, 3)	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	27,263,136
(3, 4)	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	218,106,368
(1, 2)(1, 4)	1	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	137,797,642
(1, 2)(2, 3)	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	28,573,866
(1, 2)(3, 4)	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	221,514,250
(1, 4)(2, 3)	1	0	0	1	1	0	1	0	0	1	1	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	161,915,040
(1, 4)(3, 4)	1	1	0	1	0	0	0	0	0	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	218,540,544
(2, 3)(3, 4)	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	228,592,288
(1, 2)(1, 4)(2, 3)	1	0	0	1	1	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0	162,963,626
(1, 2)(1, 4)(3, 4)	1	1	0	1	0	0	1	1	0	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	221,686,282
(1, 2)(2, 3)(3, 4)	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	229,903,018
(1, 4)(2, 3)(3, 4)	1	1	0	1	1	0	1	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	229,026,464

launching times of the bursts along nested m-trails of each trail can be derived using the same technique as done in linear networks.

Example: Now consider t^4 in Figure 6.15 as an independent virtual linear network. Assume that the bursts along the nested m-trails of t^4 are launched from the MN in descending order of the lengths of the corresponding nested m-trails.

Burst β^{42} along nested m-trail m^{42} will be launched in the beginning of each monitoring period, i.e., $s^{42} = 0$. To avoid collision with burst β^{42}, burst β^{41} along nested m-trail m^{41} will be launched after $2\delta + L$ ms, i.e., $s^{41} = 2\delta + L$. Similarly, burst β^{44} along nested m-trail m^{44} will be launched after $2\delta + L$ ms from the luanching time of burst β^{41}, i.e., $s^{44} = 4\delta + 2L$.

Similarly, we can find burst launching times of the nested m-trails of trails t^5, t^6, and t^7 by inspection.

The first burst in every cycle/trail will be launched in the beginning of each monitoring period simultaneously; this is possible because each cycle/trail will be formed by using a pair of WLs in each undirected link visited by the cycle/trail. Thus, the fault localization

latency T will be determine by the largest cycle or the largest trail in C. In the example for the SRLG with $d = 3$ links, the largest cycle in C traverses 3 links, i.e., $|c^j|_{max} = 3$. If $\delta = 2$ ms and $L = 20$ ms, $L > \delta(|c^j|_{max} - 2)$. Thus, the fault localization latency for the cycles as per Theorem 6.3.4 will be $(2\delta + L)(|c^j|_{max} - 1) + \delta(|c^j|_{max} - 2) = 50$ ms. The largest trail in C also traverses 3 links, i.e., $|t^j|_{max} = 3$. If $\delta = 2$ ms and $L = 20$ ms, $L > \delta(|t^j|_{max} - 1)$. Thus, the fault localization latency for the trails as per Theorem 6.2.8 will be $(2\delta + L)|t^j|_{max} = 72$ ms. Therefore, T will be 72 ms.

6.5.3 Node Switch Fabric Configuration Scheduling

Configuration of switch fabrics of the nodes traversed by each cycle/trail in C will be done independently. For each cycle $c^j \in C$, the node switch fabric configuration will be done using the same techniques as described in the subsection 6.4.3.

For each trail $t^j \in C$, the MN connects the outgoing and incoming WLs of the trail to a monitor, and the destination node of the trail connects the incoming and outgoing WLs of the trail. These connections will not be changed. All other nodes visited by the trail will change its switch fabric configuration two times during each monitoring period: in the beginning of each monitoring period and when the burst having the node as the destination reaches the node.

Example: Again, consider t^4 in Figure 6.15 as an independent virtual linear network. Trail t^4 visits nodes v_0, v_4, v_1, and v_2. Node v_0 is the MN, and node v_2 is the destination node of the trail. The MN connects the incoming and the outgoing WLs from and to undirected link (v_0, v_4) of t^4 to the trail's monitor. Node v_2 connects the incoming and the outgoing WLs from and to undirected link (v_1, v_2) of t^4. In the beginning of monitoring, both nodes v_1 and v_4 connect each incoming WL of t^4 from one adjacent node to the outgoing WL of t^4 to another adjacent node.

At $4\delta + L + kT$ ms, where T is the fault localization latency and $k = 0, 1, \ldots$, node v_1 connects the incoming and outgoing WLs of t^4 from and to node v_4. Thus, bursts β^{41} will be able to start returning to the MN. At $4\delta + 2L + kT$ ms, node v_1 connects the incoming WL of t^4 from node v_4 to the outgoing WL of t^4 to node v_2 and vice versa.

At $5\delta + 2L + kT$ ms, node v_4 connects the incoming and outgoing WLs of t^4 from and to node v_0. Thus, bursts β^{44} will be able to start returning to the MN. At $5\delta + 3L + kT$ ms, node v_4 connects the incoming WL of t^4 from node v_0 to the outgoing WL of t^4 to node v_1 and vice versa.

Similarly, we can find the schedule for periodic change of the node switch fabric configuration of each node visited by trails t^5, t^6 and t^7 by inspection.

6.6 ILP to Decompose Mesh Networks

The ILP finds a set of cycles/trails as a solution such that each undirected link is traversed by $\frac{d}{2}$ cycles that are disjoint in other links if d is even. One the other hand, if d is odd, each undirected link is traversed by at least one trail and $\lfloor \frac{d}{2} \rfloor$ cycles that are disjoint in other links. Each cycle/trail in an ILP solution visits the MN as well.

List of Notation

G The network, $G = (V, E)$, where V is the set of nodes and E is the set of unidirectional links of the network.

MN The monitoring node in the network, $MN \in V$.

J Predefined maximum number of allowed cycles/trails in an ILP solution.

i, j Indices of cycles/trails $i, j \in \{0, 1, 2, \ldots, J-1\}$.

r_1, r_2 Predefined cost ratios.

δ Predefined small positive constant. It is the minimum voltage increase along a cycle/trail, $|E|^{-1} \geq \delta > 0$.

id_{uv} Predefined unique ID of each undirected link (u, v), where $id_{uv} \in \{0, 1, 2, \ldots, |E|-1\}$.

N A large predefined number that represents ∞.

n A pre-calculated constant that is equal to $\lfloor \frac{d}{2} \rfloor$.

f A pre-calculated constant that is equal to 1 if d, the maximum number of links in any SRLG, is odd and 0 otherwise.

m^j Binary variable, it is equal to 1 if the jth cycle/trail is in an ILP solution and 0 otherwise.

mt^j Binary variable, it is equal to 1 if m^j is a trail and 0 otherwise.

so_u^j Binary variable, It is equal to 1 if node u is the source of the jth cycle/trail and 0 otherwise. Note that only MN can be the source of a cycle/trail.

si_u^j Binary variable, It is equal to 1 if node u is the destination of the jth cycle/trail and 0 otherwise. If the MN is also the destination of a cycle/trail, it will be a cycle.

e_{uv}^j Binary variable, it is equal to 1 if the jth cycle/trail traverses unidirectional link (u, v) and 0 otherwise.

q_{uv}^j Fractional variable, it is defined as *voltage* of the vector $u \rightarrow v$ for unidirectional link (u, v). It assumes an arbitrary positive value if the jth cycle/trail traverses link (u, v) and 0 otherwise.

z_u^j Binary variable, it is equal to 1 if the jth cycle/trail visits node u and 0 otherwise. The variables z_u^j and q_{uv}^j along with predefined constant δ help to keep each cycle/trail in an ILP solution a single connected component.

eu_{uv}^j Binary variable, it is equal to 1 if the jth cycle/trail traverses undirected link (u, v) and 0 otherwise.

cd_{uv}^{ij} Binary variable, it is equal to 1 if both the ith and the jth cycles/trails traverse undirected link (u, v). but the ith cycle/trail is link-disjoint from the jth cycle/trail in all other undirected links, and 0 otherwise.

df_{uv}^j Binary variable, it is equal to 1 if the jth cycle/trail is counted as one of the otherwise link-disjoint cycles/trails that are traversing undirected link (u, v), and 0 otherwise.

g_{uv}^j Binary variable, it is equal to 1 if the jth trail is counted as one of the otherwise link-disjoint cycles/trails that are the traversing undirected link (u, v), and 0 otherwise.

h^{ij} Binary variable, it ensures that the cycle/trail codes of the ith cycle/trail and the jth cycle/trail are different.

l Integer variable, it is the maximum length of any cycle minus 1 or the maximum length of any trail.

α^j It is the cycle/trail code of the jth cycle/trail.

ILP to Decompose Mesh Networks

The specific ILP formulation is provided as follows.

Objective:

$$Minimize\left\{ l + r_1 * \sum_j m^j + r_2 * \sum_j \sum_{\forall (u,v) \in E} e^j_{uv} \right\} \tag{6.1}$$

Subject to the following constraints:

A single source node and a single destination node are allowed for each cycle/trail in an ILP solution. Only the MN can be the source of a cycle/trail. A valid cycle/trail must traverse at least one unidirectional link.

$$so^j_{MN} = m^j, \ \sum_{u \in V} so^j_u = m^j, \ \sum_{u \in V} si^j_u = m^j, \ \forall j \tag{6.2}$$

$$\sum_{\forall (u,v) \in E} e^j_{uv} \geq m^j, \quad \forall j \tag{6.3}$$

The constraints (6.4) to (6.9) and their variables are related with cycle/trail formation and mostly taken from the method in [63]; please refer to the paper for a thorough explanation of the constraints and the variables. The indices of the cycles/trails in an ILP solution are the lowest ones. The jth cycle/trail can traverse an undirected link at most once.

$$m^j \geq m^{j+1}, \quad \forall j : j \leq J - 2 \tag{6.4}$$

$$e^j_{uv} + e^j_{vu} \leq m^j, \quad \forall (u,v) \in E : u < v, \forall j \tag{6.5}$$

The flow conservation defined for each node of the network enforces that each cycle/trail consists of connected components.

$$\sum_{(u,v) \in E} (e^j_{uv} - e^j_{vu}) = (so^j_u - si^j_u), \quad \forall u \in V, \forall j \tag{6.6}$$

If any link incident on node u is traversed by the jth cycle/trail, the node is considered visited by the cycle/trail. Voltages of the vectors corresponding to on-trail links can be

assigned non-zero values. The node voltage constraint (6.9) ensures that each cycle/trail in an ILP solution is a single connected component.

$$z_u^j \geq e_{uv}^j + e_{vu}^j, \quad \forall u \in V : (u,v) \in E, \forall j \tag{6.7}$$

$$q_{uv}^j \leq e_{uv}^j, \quad \forall (u,v) \in E, \forall j \tag{6.8}$$

$$si_u^j + \sum_{(u,v)\in E} (q_{uv}^j - q_{vu}^j) \geq \delta * z_u^j, \quad \forall u \in V, \forall j \tag{6.9}$$

We need trails if d is odd. If the MN is the destination node of the jth cycle/trail, it is indeed a cycle and otherwise it is a trail.

$$mt^j \leq f, \quad \forall j \tag{6.10}$$

$$mt^j = m^j - si_{MN}^j, \quad \forall j \tag{6.11}$$

The constraint finds the maximum of trail lengths or cycle lengths minus 1.

$$l \geq \sum_{\forall (u,v)\in E} e_{uv}^j + (mt^j - m^j), \quad \forall j \tag{6.12}$$

If the jth cycle/trail traverses undirected link (u,v) from either direction, the link is considered traversed by the cycle/trail.

$$eu_{uv}^j = e_{uv}^j + e_{vu}^j, \quad \forall (u,v) \in E : u < v, \forall j \tag{6.13}$$

Constraints (6.14) and (6.15) identify otherwise link-disjoint cycle/trail pairs that are traversing each undirected link (u,v). The ith and the jth cycles/trails can be considered link-disjoint except in link (u,v) if both the cycles/trails traverse the link. cd_{uv}^{ij} and cd_{uv}^{ji} must assume the same value.

$$cd_{uv}^{ij} \leq eu_{uv}^i, \quad cd_{uv}^{ij} \leq eu_{uv}^j, \quad cd_{uv}^{ij} = cd_{uv}^{ji}, \forall (u,v) \in E : u < v, \forall i,j : i < j \tag{6.14}$$

Moreover, the ith cycle/trail has to be link-disjoint from the jth cycle/trail in all other undirected links. Now, $\forall (u,v),(w,x) \in E : u < v \ \wedge \ w < x \ \wedge \ (u,v) \neq (w,x)$, $\forall i,j : i < j$,

$$cd_{uv}^{ij} \ + \ eu_{wx}^{i} \ + \ eu_{wx}^{j} \leq 2 \tag{6.15}$$

If the jth cycle/trail traverses undirected link (u,v), the cycle/trail can be considered as one of the otherwise link-disjoint cycles/trails traversing (u,v).

$$df_{uv}^{j} \leq eu_{uv}^{j}, \quad \forall (u,v) \in E : u < v, \forall j \tag{6.16}$$

The jth cycle/trail can be otherwise link-disjoint from either $(n+f-1)$ cycles/trails that are traversing (u,v) or none. $(n+f)$ mutually otherwise link-disjoint cycles/trails have to traverse (u,v).

$$\sum_{i:i<>j} cd_{uv}^{ij} = (n+f-1) * df_{uv}^{j}, \ \forall (u,v) \in E : u < v, \forall j \tag{6.17}$$

$$\sum_{j} df_{uv}^{j} = n+f, \quad (u,v) \in E : u < v \tag{6.18}$$

At most one of the $(n+f)$ mutually otherwise link-disjoint cycles/trails that are traversing (u,v) is a trail if d is odd.

$$g_{uv}^{j} \leq mt^{j}, \ g_{uv}^{j} \leq df_{uv}^{j}, \ (u,v) \in E : u < v, \forall j \tag{6.19}$$

$$m^{j} + g_{uv}^{j} \geq mt^{j} + df_{uv}^{j}, \ (u,v) \in E : u < v, \forall j \tag{6.20}$$

$$\sum_{j} g_{uv}^{j} \leq f, \quad (u,v) \in E : u < v \tag{6.21}$$

The cycle/trail codes are calculated based the link traversal.

$$\alpha^{j} = \sum_{\forall (u,v) \in E} 2^{id_{uv}} * eu_{uv}^{j}, \quad \forall j \tag{6.22}$$

Each cycle/trail code has to be unique. $\forall i, j : i < j$,

$$\alpha^i - \alpha^j \geq 1 - h^{ij} * (N+1) + (m^i + m^j - 2) * N \tag{6.23}$$

$$\alpha^j - \alpha^i \geq h^{ij} + (h^{ij} - 1) * N + (m^i + m^j - 2) * N \tag{6.24}$$

The objective function (6.1) aims to minimize the length of largest cycle or trail. The number of cycles/trails and usage of WLs are also minimized.

From Eq. (6.15), we have the number of constraints in the order of $\mathrm{O}(J^2|E|^2)$, where $|E|$ is the number of undirected links. However, $J \leq \lceil \frac{d}{2} \rceil |E|$. Thus, the number of constraints is in the order of $\mathrm{O}(d^2|E|^4)$. From the same equation, we have the number of variables in the order of $\mathrm{O}(d^2|E|^4)$, based on the variable cd_{uv}^{ij}.

6.7 Heuristic to Decompose Mesh Networks

The pseudo code of the proposed heuristic algorithm for finding the set of cycle/trail in mesh networks is given in Algorithm 6.6. The main objectives are to reduce the length of the largest cycle/trail in the solution set C, the number of the cycles/trails in the set, and total number of links traversed by the cycles/trails.

In line 1, the set C is initialized as empty. The outer *for* loop will consider each undirected link in turn to derive $\frac{d}{2}$ cycles if d is even or $\lfloor \frac{d}{2} \rfloor$ cycles and one trail if d is odd. The trail and the cycles have to traverse the MN and the current link under consideration, but they have to be disjoint in other links. In line 2, d link-disjoint paths from the MN to the end nodes of the current undirected link will be derived using Algorithm A.1, which is an extension of Suurballe's [50] and Bhandari's [12] algorithms for link-disjoint paths. The nested *for* loop iterates $\lfloor \frac{d}{2} \rfloor$ times. In line 3, a cycle is formed using a pair of link-disjoint paths and the current undirected link. In line 4, the cycle is added to C. If d is odd, a trail is formed using the remaining path to v and the current link under consideration in line 5. In line 6, the trail is added to C.

In line 7, all the cycles and trails are en-queued to Q in descending order of their fault localization latencies. Thus, the cycles/trails with larger fault localization latencies will be considered for removal before the cycles/trails with smaller fault localization latencies. Redundant cycles and trails will be removed in the *while* loop. In line 9, the current

cycle/trail with the maximum fault localization latency but not considered so far will be removed from Q. If each link of the network is traversed by at least $\lceil \frac{d}{2} \rceil$ cycles, or $\lfloor \frac{d}{2} \rfloor$ cycles and one trail in $C \backslash ct^j$, where the trail (if any) and the cycles are mutually disjoint in other links, ct^j will be removed from C in line 11. The set C is returned in line 12.

Algorithm 6.6: The Cycle Set Derivation Method

Input: $G(V, E)$, MN, and d
Output: C
begin
1 Initialize $C \leftarrow \phi$
foreach *undirected link* $(u, v) \in E$ **do**
2 Find $\lfloor \frac{d}{2} \rfloor$ and $\lceil \frac{d}{2} \rceil$ link-disjoint paths in G from the MN to nodes u and v, respectively.
foreach *pair of paths to nodes* u *and* v **do**
3 Form cycle c^j with the paths and (u, v)
4 $C \leftarrow c^j$
if d *is odd* **then**
5 Form trail t^j with (u, v) and the remaining path to v.
6 $C \leftarrow t^j$.
7 Assign the cycles/trails in C to a queue Q in descending order of their fault localization latencies.
8 **while** $Q \neq \phi$ **do**
9 $ct^j \leftarrow$ Extract from Q
foreach $(u, v) \in E$: (u, v) *is an undirected link* **do**
10 **if** (u, v) *is not traversed by at least* $\lceil \frac{d}{2} \rceil$ *cycles, or* $\lfloor \frac{d}{2} \rfloor$ *cycles and one trail in* $C \backslash ct^j$, *where the trail (if any) and the cycles are mutually disjoint in other links* **then**
GoTo line 8.
11 Remove ct^j from C.
12 Return C.

Now, let us derive complexity of Algorithm 6.6. The outer *for* loop iterates $|E|$ times. Finding d disjoint path from the MN to each link needs $O(d|V||E|)$ steps using Algorithm A.1 after minor modifications. Forming a cycle based on a pair of paths and a link requires $O(|E|)$ steps. Similarly, a trail formation requires $O(|E|)$. Thus, forming $\lceil \frac{d}{2} \rceil$ cycles, or $\lfloor \frac{d}{2} \rfloor$ cycles and one trail require $O(d|E|)$ steps. Thus, the outer *for* loop needs $O(d|V||E|^2)$ steps.

Each undirected link is traversed by $\lceil \frac{d}{2} \rceil$ cycles, or $\lfloor \frac{d}{2} \rfloor$ cycles and one trail. Thus, $|C| = \lceil \frac{d}{2} \rceil |E|$. Line 7 needs $O\big(\lceil \frac{d}{2} \rceil |E| \log_2(\lceil \frac{d}{2} \rceil |E|)\big)$.

The *while* loop iterates $|C| = \lceil \frac{d}{2} \rceil |E|$ times, and its inner *for* loop iterates $|E|$ times. As cycles/trails derived for each link are link-disjoint in other links of the network, the number

of cycles/trails traversing any link will be at most $|E|$. Now, to find all combination of $\lceil \frac{d}{2} \rceil$ cycles/trails, at most $O\big(|E|^{\lceil \frac{d}{2} \rceil}\big)$ steps are required. To perform pair wise AND operations in a set of cycles/trails with cardinality $\lceil \frac{d}{2} \rceil$, $O(\lceil \frac{d}{2} \rceil^2)$ steps are needed. Line 10 requires at most $O\big(\lceil \frac{d}{2} \rceil^2 |E|^{\lceil \frac{d}{2} \rceil}\big)$ steps for each link. Thus, the inner *for* loop needs $O\big(\lceil \frac{d}{2} \rceil^2 |E|^{\lceil \frac{d}{2} \rceil + 1}\big)$ steps. Accordingly, the *while* loop needs $O\big(\lceil \frac{d}{2} \rceil^3 |E|^{\lceil \frac{d}{2} \rceil + 2}\big)$. The overall complexity of the algorithm will be $O\big(d^3 |E|^{\lceil \frac{d}{2} \rceil + 2}\big)$.

6.8 Numerical Results

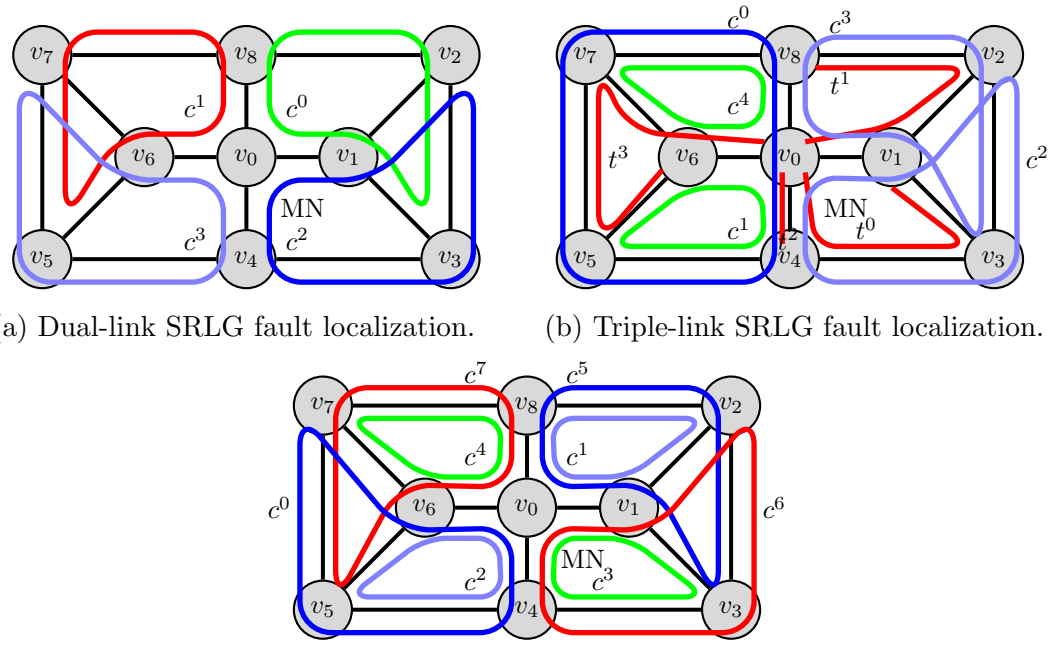
We have solved the ILP given in Section 6.6 using ILOG CPLEX 11.1. The solution of the ILP for each mesh network is a set of cycles and trails C. Two sets of nested m-trails, clockwise and anti-clockwise, are derived by inspection from each cycle in C considering the cycle as a virtual ring network. Similarly, one nested set of m-trails is derived by inspection from each trail in C considering the trail as a virtual linear network.

The burst starting times from the MN of the bursts along the nested m-trails and the fault localization latencies for the virtual ring/linear networks are taken from the results derived for the ring/linear networks with equal number of links. Note that the fault localization latencies of the linear and ring networks are shown in Table 6.2 and Table 6.4, respectively. We assume that the first burst during a monitoring period along a chosen nested m-trail of each cycle/trail is launched simultaneously. Thus, the length of the largest cycle/trail in the solution set determines the fault localization latency T of the mesh network.

The experiment considers all the SRLGs with $d = 2, 3, 4$ in each mesh network one at a time, where each SRLG with two or more links is node-disjoint from the MN. We conduct the experiment on a network with 9 nodes and 14 links. Node 0 is assigned as the MN. There are in total 59, 179, and 389 SRLGs when d is 2, 3, and 4, respectively. We assume that r_1 is 0.1 and r_2 is 0.01.

The solutions of the ILP are cycle cover sets $\{c^0, c^1, c^2, c^3\}$, $\{c^0, c^1, c^2, c^3, c^4, t^0, t^1, t^2, t^3\}$, and $\{c^0, c^1, c^2, c^3, c^4, c^5, c^6, c^7\}$ as shown in Figure 6.16 for d equal to 2, 3, and 4, respectively. On average 2.857, 4.857, and 5.143 WLs per undirected link are required for SRLG fault localization in the network for d equal to 2, 3, and 4, respectively. Then, we derive nested m-trails by inspection. Next, the ACTs of the network for $d = 2, 3, 4$ are constructed (not shown). An ACT keeps the mapping between the alarm code and the corresponding SRLG code. The SRLG code of an SRLG is determined by the m-trails traversing through the

SRLG. The SRLG code of each SRLG when d is 2, 3, or 4 is unique; thus, up to d faulty links can be localized unambiguously.



(a) Dual-link SRLG fault localization.

(b) Triple-link SRLG fault localization.

(c) Quadruple-link SRLG fault localization.

Figure 6.16: Dual-link and Multi-link SRLG fault localization in the network with 9 nodes and 14 links. Node v_0 is the MN.

For burst scheduling, we assume that the propagation delay δ through each unidirectional link is 2 ms and the burst length L is 20 ms. As the largest cycle in each case of d traverses 5 links, T is 102 ms, which is the same in each case, found by using 5 as an index in Table 6.4.

We have conducted the experiment on three more networks keeping all the assumptions the same. The results are summerized in Table 6.8.

We implement proposed heuristic in eight networks. The results are shown in Table 6.9. Like the ILP, once a set of cycles and trails are derived, the nested m-trails are derived by inspection and ACTs are formed. In all cases, the SRLG codes are unique. Then, the burst scheduling results are taken from that of linear and ring networks; T for each network is taken from Table 6.2 or Table 6.4 using the number of links in the largest cycle

Table 6.8: The Performance of the ILP Method

Networks	d	SRLG	$\|C\|$	WLs	$\frac{\text{WLs}}{\|E\|}$	$\|c^j\|_{max}$	T
6 Node, 12 links	2	40	4+0=4	32	2.667	4	76
	3	96	5+4=9	60	5.000	4	76
	4	166	8+0=8	64	5.330	4	76
7 Node, 12 links	2	40	4+0=4	34	2.833	5	102
	3	96	4+6=10	60	5.000	5	102
	4	166	6+0=6	56	4.667	5	102
8 Node, 12 links	2	48	3+0=3	36	3.000	6	128
	3	132	3+2=5	52	4.333	6	128
	4	258	6+0=6	60	5.000	6	128
9 Node, 14 links	2	59	4+0=4	40	2.857	5	102
	3	179	5+4=9	68	4.857	5	102
	4	389	8+0=8	72	5.143	5	102

or trail, respectively, as index. The fault localization latency T depends on network size and increases moderately on d.

In Table 6.10, the performance in terms of number of monitors $|M|$, WL consumption per link $\frac{WLs}{|E|}$, and fault localization latency T of the proposed method is compared with that of the previously reported methods. The results of the method in [1] deployed in the m-burst framework is shown in column H_0. The results of MCF method given in Section 5.2 and DMCF methods given in Section 5.3 are shown in the columns H_1 and H_2, respectively. Column H_{012} represents each of the previous methods. The results of the proposed method in this chapter are shown in column H_3. Though the new method needs more monitors and WLs per link, the fault localization latencies in large networks improve significantly.

6.9 Application of Nested M-Trails in Adaptive Probing

The m-trail based schemes for fault localization in all-optical networks can be viewed as non-adaptive probing schemes. A set of m-trails are pre-designed and deployed in the network to localize single-link or multi-link SRLG faults. Each m-trail can be probed

Table 6.9: The Performance of the Heuristic Method

Networks	d	SRLG	$\|C\|$	WLs	$\frac{\text{WLs}}{\|E\|}$	$\|c^j\|_{max}$	T
6 Node, 12 links	2	40	7+0=7	48	4.000	4	76
	3	96	7+5=12	78	6.500	4	76
	4	166	8+0=8	66	5.500	5	102
7 Node, 12 links	2	40	6+0=6	44	3.667	5	102
	3	96	6+3=9	62	5.167	5	102
	4	166	7+0=7	66	5.500	5	102
8 Node, 12 links	2	48	5+0=5	48	4.000	6	128
	3	132	5+3=8	66	5.500	6	128
	4	258	6+0=6	60	5.000	6	128
9 Node, 14 links	2	59	6+0=6	52	3.714	5	102
	3	179	6+4=10	80	5.714	5	102
	4	389	8+0=8	72	5.143	5	102
CERNet	2	71	7+0=7	58	3.625	5	102
	3	236	7+2=9	80	5.000	6	128
	4	566	10+0=10	106	6.625	7	154
SmallNet	2	193	11+0=11	106	4.818	6	128
	3	1162	10+9=19	168	7.636	6	128
	4	5038	17+0=17	184	8.364	7	154
NSFNet + 2 links	2	213	9+0=9	110	4.783	7	154
	3	1353	10+9=19	210	9.130	7	154
	4	6198	17+0=17	254	11.043	11	258
Bellcore + 1 link	2	282	14+0=14	132	4.552	7	154
	3	2053	13+13=26	226	7.793	8	180
	4	10908	21+0=21	266	9.172	9	206

using persistent or non-persistent optical signal, or short-length optical burst. Short-length optical bursts are used in the m-burst framework.

Now, we shall investigate briefly application of the m-burst framework in adaptive probing. Note that in adaptive probing, the set of probe or m-trails are not pre-designed. The next probe to be launched is derived based on the result found so far. If a region of the network is found healthy, the region will not be probed actively any more in the current session.

Table 6.10: The Comparative Performance of the Methods, where $d = 3$

Networks	No. of SRLGs	$\|M\|$		$\frac{\text{WLs}}{\|E\|}$		T			
		H_{012}	H_3	H_{012}	H_3	H_0	H_1	H_2	H_3
6 nodes, 12 links	96	3	19	≤ 2	6.500	108	152	88	76
7 nodes, 12 links	96	4	15	≤ 2	5.167	132	152	112	102
8 nodes, 12 links	132	3	13	≤ 2	5.500	172	228	120	128
9 nodes, 14 links	179	4	16	≤ 2	5.714	154	164	114	102
CERNet	236	5	16	≤ 2	5.000	168	258	152	128
SmallNet	1162	3	29	≤ 2	7.636	336	502	216	128
NSFNet + 2 links	1353	3	29	≤ 2	9.130	556	876	294	154
Bellcore + 1 link	2053	6	39	≤ 2	7.793	346	742	220	180

One of the important hurdles to deploy adaptive probing in all-optical networks is the large number of sequential probes needed to localize faults. The state of the art adaptive run-length probing scheme proposed in [57] needs $d \log_2 K + \frac{|E|-d+1}{K} + (d-2)$ probes to find d link failures, where maximum probing length K is derived to get approximately an equal probability of a single fault and no fault in a fiber segment along a Eulerian trail. We will not discuss how to derive exact value of K. However, $K = \frac{|E|-d+1}{d} \cdot \ln 2$ will provide the minimum number of probes.

Now, we will only show that using nested m-trail methodology, the number of adaptive probes to be launched to localize single-link or multi-link SRLG faults can be reduced. As a result, fault localization latency in adaptive probing will also be reduced. Hence, the deployment of the adaptive probing in all-optical networks for fault localization will become feasible.

In addition, application of nested m-trails in adaptive probing will reduce control complexity of setting up and launching of the probes. The bandwidth cost will be the same as that of conventional adaptive schemes; no additional monitoring hardware in any node is needed on top of the m-trail framework. Moreover, SRLG faults will be localized from a single MN.

6.9.1 Dual-Link SRLG Fault Localization in Mesh Networks

In this section, we propose a novel adaptive probing scheme in Algorithm 6.7 that uses the nested m-trail technique to localize dual-link SRLG faults in all-optical mesh networks. At first, the mesh network is decomposed into cycles. Probing will be done in each cycle

independently. Thus, probing in various region of the network can be done in parallel. As the largest cycle is upper bounded by the diameter of the network, the number of links to be probed by each independent probing process is in order of O($|D|$) instead of O($|E|$), where D and E are the diameter of the network and the number of links in the network, respectively.

Algorithm 6.7: Dual-Link SRLG Fault Localization using Adaptive Probing

Input: $G(V, E)$ and MN
Output: C and F

1 Decompose mesh network G into a set of cycles C such that each link of the network is traversed by at least one cycle, and every cycle traverses the MN.
2 Assign a pair of WLs in each link traversed by each cycle $c^j \in C$. Assign $0, \ldots, |c^j|$ clockwise consecutively as additional labels to the nodes of c^j, where the MN will have two labels 0 and $|c^j|$.
3 Each cycle $c^j \in C$ will be probed simultaneously by two probing stations (PSs) from the MN. Each PS will identify its active probing region using *lo* and *hi* node numbers of the region. Initially, *lo* is 0 and *hi* is $|c^j|$ for both the PSs. The middle node *mid* of an active probing region is $\lceil \frac{lo+hi}{2} \rceil$. The PSs P_c^j and P_a^j of cycle c^j will launch clockwise and anti-clockwise m-bursts along the cycle, respectively. The destination of each probe will be the *mid* node of the PS. Initially, *mid* of both PSs will be the same node because the entire cycle is the active probing region of each PS.
4 If both the bursts of each cycle return to the MN, there is no fault in the network. The next pair of m-bursts from the clockwise and anti-clockwise PSs will be launched immediately along each cycle.
5 If any burst of a cycle c^k does not return to the MN, link failure in the cycle is detected. The PS of each non-returning burst will become an executive probing station (EPS). If one burst does not return to the MN, one PS will become EPS now. Whenever both bursts of the cycle do not return to the MN during any subsequent round of probing, the other PS will immediately become an EPS. The faulty links will be searched in cycle c^k using the adaptive probes as described below until the faulty links are identified.
6 If there is only one EPS in the cycle currently, the EPS will update both the active regions of the PSs using the results of the last round of probing. On the other hand, if there are two EPSs in the cycle, each EPS will update its own active probing region.
7 If P_c^k is an EPS, P_c^k updates its the active probing region as follows. If clockwise burst of the cycle does not return to the MN, P_c^k updates its the active probing region by assigning its *mid* to its *hi*. Otherwise, P_c^k assigns its *mid* to its *lo*. Then, it calculates its new *mid* using *lo* and *hi* node numbers of the updated active probing region.
8 If P_a^k is an EPS, P_a^k updates its the active probing region as follows. If anti-clockwise burst of the cycle does not return to the MN, P_a^k updates its the active probing region by assigning its *mid* to its *lo*. Otherwise, P_a^k assigns its *mid* to its *hi*. Then, it calculates its new *mid* using *lo* and *hi* node numbers of the updated active probing region.
9 Then each PS will send an m-burst from the MN to probe its active probing region.
10 The above probing process (lines 6 – 9) will continue until each EPS identifies its nearer faulty link. When an active probing region of an EPS has one link, it is the faulty link.
Each EPS will assign its nearer faulty link to F.
11 Return C and F.

It is evident from the algorithm that the proposed probing scheme is equivalent to binary search. If the largest cycle has l links, we need at most $\lceil \log_2 l \rceil$ probes. Moreover, as two probing processes will be used in each cycle, only half of the links in the cycle will be searched by each probing process. Thus, only $\lceil \log_2 l \rceil - 1$ probes will be needed to localize a single fault in the cycle by each probing process. The number of adaptive probes to be launched by any probing process sequentially after fault detection to localize the faulty links will be reduced significantly. As a result, the fault localization latency T of the adaptive probing will also be reduced significantly.

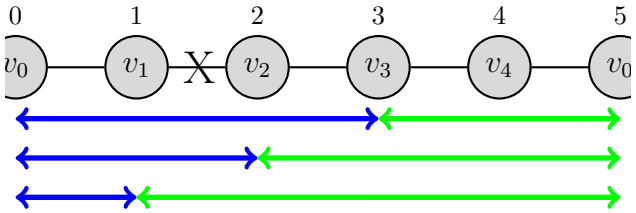
Configuration of switch fabrics of the nodes traversed by each cycle $c^j \in C$ will be done independently. The MN connects clockwise and anti-clockwise probing stations with the WLs of c^j in clockwise and anti-clockwise adjacent links, respectively. These connections will not be changed. Each node except the MN visited by c^j usually connects WLs of c^j from its clockwise adjacent node in c^j to its anti-clockwise adjacent node in c^j and vice versa. During the period a node remains the destination node of one or both the probes in c^j, it keeps WLs from and to its clockwise adjacent node in c^j connected as well as WLs from and to its anti-clockwise adjacent node in c^j connected.

Example: Consider Figure 6.16a again. The figure shows decomposition of the network with 9 nodes and 14 links. Each cycle consists of five links. Figure 6.17 shows probing sequences during one single-link and two dual-link fault events in cycle c^2 of Figure 6.16a. If adaptive proving is used in each cycle independently, we need 3 sequential probes to localize a single-link SRLG fault and a dual-link SRLG fault in the network as shown in Figure 6.17a and Figure 6.17b, respectively. However, the first pair of probes will detect the failure event as well as define initial active probing region(s). Thus, we need at most two additional sequential probes to localize the faults in the network. In Figure 6.17c, we need only one additional sequential probe to localize a dual-link SRLG fault in the network.

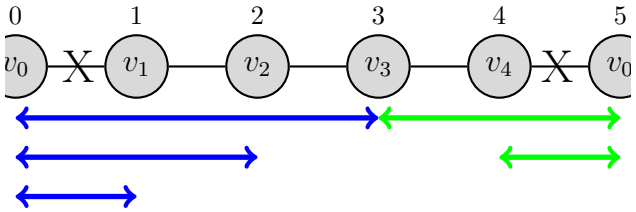
In contrast, if the adaptive proving scheme proposed in [57] is used, the value of K that gives minimum number of probes in the network is 5. Consequently, at least 7 sequential probes will be needed to localize dual-link SRLG faults.

6.9.2 Multi-Link SRLG Fault Localization in Mesh Networks

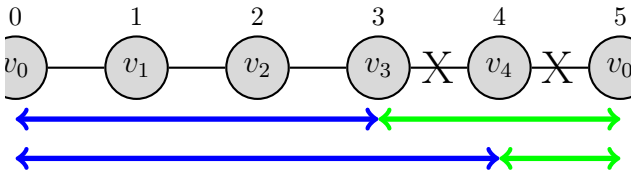
To localize d faulty links using adaptive probing in the m-burst framework, we have to ensure that each undirected link of the network is traversed by at least $\frac{d}{2}$ cycles if d is even. If d is odd, each link has to be traversed by at least $\lceil \frac{d}{2} \rceil$ cycles, or $\lfloor \frac{d}{2} \rfloor$ cycles and one trail. Thus, line 1 of Algorithm 6.7 is to be modified accordingly to deal with simultaneous d link



(a) Three clockwise and three anti-clockwise probes are required to localize the faulty link (v_1, v_2).



(b) Three clockwise and two anti-clockwise probes are required to localize the faulty links (v_0, v_1) and (v_4, v_0), respectively.



(c) Two clockwise and two anti-clockwise probes are required to localize the faulty links (v_3, v_4) and (v_4, v_0), respectively.

Figure 6.17: Application of nested m-trail in adapting probing. The clockwise and anti-clockwise probes are blue and green, respectively. Both sets of probes are sent from the MN. Faulty links are marked as (X). Three sequential probes are required to localize the single faulty link in (a), three sequential probes are required to localize both the faulty links in (b), and two sequential probes are required to localize both the faulty links in (c).

faults. The probing strategy in each cycle will remain the same as that of the dual-link SRLG fault localization discussed in subsection 6.9.1.

The numbers of required probes for dual-link and multi-link SRLG fault localization

Table 6.11: The Performance of Adaptive Schemes

Networks	d	No. of SRLGs	P_0		P_1	
			K	$\#p$	$\|c^j\|_{max}$	$\#p$
6 Node, 12 links	2	40	4	7	4	2 - 1 = 1
	3	96	3	9	4	2 - 1 = 1
	4	166	2	11	5	3 - 1 = 2
7 Node, 12 links	2	40	4	7	5	3 - 1 = 2
	3	96	3	9	5	3 - 1 = 2
	4	166	2	11	5	3 - 1 = 2
8 Node, 12 links	2	48	4	7	6	3 - 1 = 2
	3	132	3	9	6	3 - 1 = 2
	4	258	2	11	6	3 - 1 = 2
9 Node, 14 links	2	59	5	7	5	3 - 1 = 2
	3	179	3	10	5	3 - 1 = 2
	4	389	2	12	5	3 - 1 = 2
CERNet, 16 links	2	71	6	8	5	3 - 1 = 2
	3	236	3	10	6	3 - 1 = 2
	4	566	3	13	7	3 - 1 = 2
SmallNet, 22 links	2	193	8	9	6	3 - 1 = 2
	3	1162	5	12	6	3 - 1 = 2
	4	5038	4	15	7	3 - 1 = 2
NSFNet + 2 links, 23 links	2	213	8	9	7	3 - 1 = 2
	3	1353	5	12	7	3 - 1 = 2
	4	6198	4	15	11	4 - 1 = 3
Bellcore + 1 link, 29 links	2	282	10	9	7	3 - 1 = 2
	3	2053	7	13	8	3 - 1 = 2
	4	10908	5	16	9	4 - 1 = 3

are compared in Table 6.11. Columns P_0 and P_1 represent the method proposed in [57] and the method proposed in this chapter, respectively. To derive the minimum number of probes in P_0, at first we derive $K_0 = \frac{|E|-d+1}{d} \cdot \ln 2$ for each network. Then we use the next lower or higher integer number of K_0, whichever gives lower value of $\#p_0$, as K in $\#p_0 = d \log_2 K + \frac{|E|-d+1}{K} + (d-2)$. The minimum number of probes $\#p$ in the network is the closest integer number of $\#p_0$. Data for P_1 is taken from Table 6.9. It is clear from the table that the nested m-trail based adapting probing scheme needs a significantly reduced

number of sequential probes.

When d is unbounded, we can by-pass each identified faulty link and continue probing the remaining links of the cycle. However, to calculate fault localization latency, a through investigation is required. This is a topic for future work.

6.10 Conclusions

In this chapter, we have studied the m-burst framework for single-link and dual-link SRLG fault localization in all-optical linear and ring networks, respectively. We have proposed simple methods to derive the sets of m-trails, burst launching time scheduling, and node switch fabric configuration scheduling by inspection in both linear and ring networks. Though the techniques are significant achievement by themselves, they are also used for dual-link and multi-link SRLG fault localization in all-optical mesh network mainly to decrease the fault localization latency.

A novel approach for fault localization called the *nested m-trail* method is proposed. The application of nested m-trails in the m-burst framework for dual-link and multi-link SRLG fault localization in all-optical mesh networks is discussed. With the minimization of the fault localization latency as the target, an optimization problem is formulated in ILP and solved to decompose mesh networks. A heuristic algorithm for decomposition of mesh networks is also developed. Once a set of cycles and trails is derived, the m-trail allocation, the burst scheduling, and node fabric configuration scheduling are done by inspection in each cycle or trail considering the cycle or trail as a virtual ring or linear network, respectively. Extensive experiments are conducted to verify the method and to compare it with counterparts. Numerical results show that the *nested m-trail* method reduces the fault localization latency significantly. The *nested m-trail* method can also substantially reduce the number of adaptive probes.

Chapter 7

Conclusions and Future Work

7.1 Introduction

A fast and precise fault localization scheme can enhance the performance of protection/restoration schemes greatly and is essential for implementing reliable all-optical mesh networks that operate with high throughput. We have mainly aimed at developing efficient multi-link SRLG fault localization schemes that incur minimum monitoring cost and fault localization latency in this thesis.

We have applied an out-of-band monitoring technique to localize single-link and multi-link shared risk link group (SRLG) faults in all-optical mesh networks. The out-of-band monitoring techniques are based on finding the working status of the dedicated supervisory lightpaths (S-LPs) known as m-trails; any node of the network can be the transmitting and/or receiving node of an S-LP, and each S-LP is monitored at its receiving node.

To localize link faults unambiguously in the existing schemes, a set of m-trails is deployed such that each SRLG is traversed by a unique subset of the m-trails. In other words, the m-trail allocation problem is considered as a topology coding process where each SRLG is assigned a unique SRLG code based on the traversing m-trails. When an SRLG fails, each m-trail traversing any link of the faulty SRLG is disrupted; as a result, the monitor at the receiver of the affected m-trail raises an alarm, and the alarm is broadcast in the control plane. The network controller derives an alarm code based on the alarming and non-alarming monitors after collecting all the flooded alarms; finally, the controller localizes the failed SRLG by matching the alarm code with an SRLG code, usually by using a look-up table called the alarm code table (ACT).

There are three major concerns of the exiting m-trail schemes: monitor cost, bandwidth cost, and alarm dissemination delay and related uncertainty. The monitor cost is measured in terms of the number of monitors $|M|$ that is the same as the number of m-trails $|\mathfrak{M}|$. The bandwidth cost is measured in terms of the number of required wavelength channels (WLs). An m-trail needs a dedicated WL whenever it traverses a unidirectional link. The alarm dissemination delay and related uncertainty arise due to the presence of multiple monitoring nodes (MNs). As monitors are usually scattered around the network, alarm dissemination in the existing m-trail schemes is unavoidable and is a source of variable delay.

To deal with the above-mentioned concerns, we have proposed the monitoring burst (m-burst) framework in Chapter 3. The framework has one given MN, which can be any node of the network. The links of the network are traversed by a set of closed-loop m-trails such that each SRLG is traversed by a unique subset of the m-trails. Each m-trail in the m-burst framework has to traverse the MN and can be either a cycle or a trail. If the m-trail is a cycle, it can traverse any undirected link only once. On the other hand, if the m-trail is a trail, it has to traverse each undirected link from both directions. The MN launches short-length optical bursts along each m-trail to detect its on-off status. Due to the closed-loop shape of the m-trail, the launched optical bursts will be received by the MN if all the links along the m-trail are working properly, but the burst will be lost if any link along the m-trail fails. Thus, alarm dissemination is avoided altogether.

In addition, the m-burst framework needs only one WL in each unidirectional link if one or more m-trails traverse the link, because bursts traversing the link share the WL in the time domain. In other words, each m-trail in the m-burst framework is a virtual entity. Consequently, the framework needs from $|E|$ to $2|E|$ WLs for both single-link and multi-link SRLG fault localization. Moreover, the number of monitors $|M|$ and the number of m-trails $|\mathfrak{M}|$ are no longer the same in the m-burst framework. As each m-burst is launched from the single MN along a single WL in an outgoing link and the burst returns to the MN via a single WL in an incoming link, monitoring resources such as transmitters, receivers, and monitors of m-trails can also be shared. In fact, the number of required monitors $|M|$ in the m-burst framework is in the order of the nodal degree of the MN. Thus, the number of monitors $|M|$ is reduced significantly since it is fully determined by the nodal degree of the MN. Therefore, the number of m-trails $|\mathfrak{M}|$ has no significance from the perspective of the monitor cost. However, the number of m-trails $|\mathfrak{M}|$ has a major negative effect on the fault localization latency, which is the main concern in the m-burst framework.

The m-burst framework has three relevant problems: m-trail allocation, m-burst launching time scheduling, and node switch fabric configuration scheduling problems. To solve the problems, we have devised several algorithms and procedures that are described in

Section 7.2 in detail along with other contributions of the thesis.

The rest of the chapter is organized as follows. Section 7.3 presents the publications related to the thesis. In Section 7.4, future work is given. Section 7.5 concludes the chapter.

7.2 Contributions

The novel features of the thesis are given below.

In Chapter 3, a novel framework for fault localization in all-optical networks, called the m-burst framework, is proposed, and three relevant problems of the framework are formulated. The methods proposed in the subsequent chapters provide solutions to the problems. The m-burst framework needs a minimum possible numbers of monitors $|M|$ and WLs to localize single-link, or single-link and multi-link SRLG faults in all-optical networks.

In Chapter 4, an implementation of the m-burst framework for single-link SRLG fault localization in all-optical mesh networks is provided. At first, the m-trail allocation and burst scheduling problems are formulated as a joint optimization problem via an Integer Linear Program (ILP), where a recursive approach in the ILP is devised to find the burst arrival time along each m-trail at the sending end node of each unidirectional link that is traversed by the m-cycle. With the help of the burst arrival times, bursts are kept non-overlapping in each unidirectional link of the network. The burst launching times are manipulated to achieve burst separation. It is a novel approach of burst scheduling to avoid all burst-collisions. For comparison, the m-trail allocation and burst scheduling problems are formulated as separate optimization problems via two ILPs. Two heuristic algorithms are also devised to solve the problems in large networks. The burst scheduling heuristic algorithm ensures that the arrival times of each pair of bursts to the sending node of any unidirectional link are separated by L ms, where L is the burst length in ms. In the scheduling algorithm, the earliest possible burst launching time avoiding burst collision altogether in the network is found for the burst along each m-trail in the solution. Thus, the fault localization latency is minimized. The burst launching time scheduling methods developed in the ILP or heuristic in the chapter are used in the later chapters directly or indirectly.

Node switch fabric configuration scheduling related issues are also discussed in Chapter 4. It is shown that switch fabric configuration scheduling can be done using simple and deterministic calculations once m-trail allocation and burst scheduling problems are solved. Actual switch fabrics configuration repeats in each monitoring period.

In Chapter 5, an implementation of the m-burst framework for multi-link SRLG fault localization in all-optical mesh networks is presented. Two novel multi-link m-trail allocation methods are proposed: MCF and DMCF. The MCF method is based on the theory that if each healthy link is traversed by at least one m-trail that is link-disjoint from the faulty SRLG, the MN can localize the faulty SRLG unambiguously. Thus, we can localize faults using $O(|\Psi|)$ m-trails, which is equivalent to $O((|E|+1)^d)$. The m-trail allocation scheme and the burst scheduling problems are formulated as a joint optimization problem via an ILP, where the ILP takes a set of enumerated unique m-trails as the input. A heuristic algorithm is also devised to implement the m-trail allocation scheme in large networks by finding an unaffected shortest path from the MN to each link during each fault event.

To avoid m-trail enumeration in the ILP formulation and to reduce fault localization latency, an extension of the MCF m-trail allocation method is also proposed in the chapter. It is proved that if each undirected link is traversed by $(d+1)$ m-trails that are disjoint in other links, the MN can localize the faulty SRLG unambiguously. Thus, we need at most $(d+1)|E|$ m-trails. The extended m-trail allocation scheme, i.e., DMCF method, is formulated as an optimization problem via an ILP. A heuristic algorithm is also devised to implement the extended m-trail allocation scheme in large networks by finding $(d+1)$ link-disjoint shortest paths from the MN to each link.

In Chapter 6, two implementations of m-burst framework for linear and ring networks are provided. It is observed that dual-link SRLG faults can be localized in a ring network using two sets of m-trails: one clockwise and another anti-clockwise. A novel m-trail allocation technique called nested m-trails is proposed in the chapter. A mesh network is decomposed into cycles and trails such that each undirected link is traversed by $\frac{d}{2}$ cycles if d is even or by $\lfloor \frac{d}{2} \rfloor$ cycles and one trail if d is odd. Each cycle (trail) is realized as a virtual ring (linear) network by assigning two WLs in each undirected link traversed by the cycle (trail). Now, two sets (one set) of m-trails are (is) derived by inspection in each virtual ring (linear) network. The two sets (one set) of nested m-trails are deployed in the m-burst framework for the ring (linear) network. Now, fault localization in virtual ring and linear networks is done in parallel. As a result, the fault localization latency decreases significantly.

Application of nested m-trails in adaptive probing is also investigated in Chapter 6. The number of sequential probes needed by the proposed method is very low. Thus, deployment of adaptive probing for fault localization in all-optical mesh networks is now possible.

In Appendix A, we have simplified link-disjoint and node-disjoint shortest path algorithms. These algorithms find disjoint shortest paths sequentially. The next shortest path is identified by manipulating link weights and node colors of the links and the nodes, re-

spectively, of the already identified paths. Techniques such as network transformation, link direction reversal, and vertex splitting are completely avoided.

Moreover, in this thesis, we are able to decrease significantly the number of m-trails $|\mathfrak{M}|$ required theoretically for multi-link SRLG faults, where an SRLG can have at most d links. `MC-1` method proposed in [1] needs $O(|\Psi|^2)$ equivalently $O((|E|+1)^{2d})$ cycles to localize multi-link SRLG faults. The DMCF m-trail allocation method presented in Chapter 5 needs at most $(d+1)|E|$ m-trails to localize multi-link SRLG faults. The nested m-trail based m-trail allocation method presented in Chapter 6 requires at most $\frac{d}{2}|E|$ cycles if d is even, or at most $\lfloor\frac{d}{2}\rfloor|E|$ cycles and $|E|$ trails if d is odd to localize multi-link SRLG faults. For comparison, the lower bound of $|\mathfrak{M}|$ is in $\Omega\big(d\log_2(|E|+1)\big)$ because $|\Psi| \leq (|E|+1)^d$.

Finally, this thesis allows multi-link SRLG fault localization in comparatively sparse networks, where an SRLG can have at most d links. `MC-1` method proposed in [1] needs a $(d+2)$-connected mesh network to localize multi-link SRLG faults. The DMCF m-trail allocation method presented in Chapter 5 needs a $(d+1)$-connected mesh network to localize multi-link SRLG faults. The nested m-trail based m-trail allocation method presented in Chapter 6 can be deployed in a d-connected mesh network to localize multi-link SRLG faults.

7.3 Publications

The list of publications related to the thesis is given below.

1. M. L. Ali, P.-H. Ho, B.Wu, J. Tapolcai, and B. Shihada. Monitoring Burst (M-Burst)-A Novel Framework of Failure Localization in All-Optical Mesh Networks. In the proceedings of *the 8th International Workshop on Design of Reliable Communication Networks, IEEE DRCN 2011*, Krakow, Poland, October 2011. This workshop paper [9] covers mainly the materials of Sections 4.1, 4.2, and 4.3 of Chapter 4.

2. M. L. Ali, P.-H. Ho, J. Tapolcai, and B. Shihada. M-Burst: A Framework of SRLG Failure Localization in All-Optical Networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8), August 2012. This journal paper [7] covers mainly the materials of Sections 5.1 and 5.2 of Chapter 5.

3. M. L. Ali, P.-H. Ho, and J. Tapolcai. SRLG Fault Localization via M-Burst Framework. To be presented at the *IEEE ICC'13 ONS*, Budapest, Hungary, June 2013. This conference paper [6] covers mainly the materials of subsections 5.3.1, 5.3.2, and 5.3.4 of Section 5.3 of Chapter 5.

4. M. L. Ali, P.-H. Ho, J. Tapolcai, and S. Subramaniam. SRLG Fault Localization via M-Burst Framework. *To be submitted.* This paper [8] covers materials of Section 5.3 of Chapter 5.

5. M. L. Ali, P.-H. Ho, and J. Tapolcai. Fault Localization in Linear and Ring Networks. *To be submitted.* This paper [4] covers materials of Sections 6.2 and 6.3 of Chapter 6.

6. M. L. Ali, P.-H. Ho, and J. Tapolcai. SRLG Fault Localization using Nested M-trails. *To be submitted.* This paper [5] covers materials of mainly Sections 6.4 and 6.5 of Chapter 6 .

Moreover, we shall publish two more papers on adaptive probing given in Section 6.9 of Chapter 6 and disjoint path algorithms given in Appendix A.

7.4 Future Work

We have identified the following future work.

Nested m-trails: Average WL consumption of the nested m-trail technique is higher than that of the original m-burst framework: the m-burst framework needs at most 2 WLs per link. It is obvious from the numerical results that WL consumption of the nested m-trail technique can be reduced substantially because there are duplicate m-trails in the solution. Thus, further investigation is needed in this direction.

Adaptive Probing : To reduce WLs consumption or when d is unbounded, instead of setting up $\lceil \frac{d}{2} \rceil$ cycles for each undirected link, an adaptive probing scheme can be realized with a set of cycles that is a simple cycle cover of the network. Each identified faulty link in each cycle will be by-passed and probing will be continued with the remaining links of the cycle. However, to calculate fault localization latency, a through investigation is required, which is another topic for future work.

7.5 Conclusions

We have investigated single-link and multi-link SRLG fault localization in all-optical mesh networks. We have proposed the m-burst framework and several m-trail allocation algorithms. We have also devised the burst scheduling schemes and outlined the node switch fabric configuration scheduling schemes to enable link traversal by multiple bursts without any collision throughout the network.

In conclusion, we want to state that using at most a single dedicated WL per unidirectional link of an all-optical mesh network, single-link, or single-link and multi-link SRLG faults can be localized from a single MN with reasonable fault localization latency. Moreover, there still exists a trade-off between monitoring resource consumption and the fault localization latency.

APPENDICES

Appendix A

Simple Disjoint Path Algorithms

A.1 Introduction

Disjoint paths play a very important role in fault management of communication networks. In optical networks, link-disjoint and node-disjoint paths are extensively used in protection and restoration schemes. Several algorithms in this thesis are developed based on link-disjoint paths.

For each node pair, the link form of Menger's Theorem for graphs states that the maximum number of link disjoint paths, K, between the node pair is equal to the minimum number of links separating the node pair. If all the separating links are removed, all the paths between the node pair will be destroyed [3]. In other words, the number of separating links K is the cardinality of the minimum cut-set between the node pair.

We are interested in this thesis in finding k link-disjoint paths between a single monitoring node (MN) and each link of the network. The maximum number of disjoint paths K_{uv} between the MN and any undirected link (u, v) will be equal to the cardinality of the minimum cutset between the MN and the end nodes of the link. Thus, we assume that each network topology satisfies the following condition: $\forall(u, v) \in E$: (u, v) is an undirected link, $k \geq K_{uv}$. However, we will develop algorithms to find disjoint paths between a node pair in this appendix. Finding disjoint paths between the MN and an undirected link needs simple adjustment of the methods developed for a node pair.

Suurballe in [50] uses network transformation to find disjoint paths. Bhandari in [12] avoids network transformation. However, Bhandari's methods depend on direction reversal

of the links in already identified paths and vertex splitting. We have avoided these confusing network manipulations altogether.

In our methods, paths already taken are indentified using a specific link weight and node color. As a result, we have very simple algorithms to identify multiple disjoint paths between a node pair in mesh networks.

The rest of the appendix is organized as follows. Theoretical analysis on disjoint shortest paths is given in Section A.2. In Section A.3 and Section A.4, two heuristics algorithms are given for deriving link-disjoint and node-disjoint paths, respectively. Numerical experiments are conducted, and results are given in Section A.5. Section A.6 concludes the appendix.

A.2 Theoretical Analysis on Disjoint Shortest Paths

Disjoint paths are derived between a pair of nodes known as the $s-t$ pair. Each disjoint path starts from node s and terminates at node t. Let the lengths of paths p_1, $p_2, \ldots, p_k$ be l_{p_1}, $l_{p_2}, \ldots, l_{p_k}$, respectively.

To derive disjoint paths, Suurballe [49] defines an interlacing path, and Bhandari [11] informally defines an interlacing link or edge. For the purposes of this thesis only, let us formalize Bhandari's concept of an interlacing link approximately in the following definition.

Definition A.2.1. If paths p_1 and p_2 traverse unidirectional links (u, v) and (v, u), respectively, the corresponding undirected link (u, v) is called an *interlacing link* between p_1 and p_2.

Let the length or cost of the interlacing link (u, v) between p_1 and p_2 be l_{uv}. Note that instead of a single interlacing link, two paths may have an interlacing segment that consists of two or more connected interlacing links. Moreover, two paths may have more than one disjoint interlacing link or segment. However, we will use the term *interlacing link* for both an interlacing segment and an interlacing link as well as for multiple disjoint interlacing links or segments in this thesis for notational convenience.

Theorem A.2.2. *If an interlacing link (u, v) is excluded from a path pair p_1 and p_2, the path pair will be cut into four pieces such that the s adjacent portion of path p_1 (p_2) will*

remain connected with the t adjacent portion of path p_2 (p_1) at one (other) end node of (u, v). As a result, two disjoint paths between the $s-t$ node pair will be formed if p_1 and p_2 are disjoint in other parts of the network.

Proof. From inspection, let (s, a), (a, u), (u, v), (v, w), (w, t) and (s, b), (b, v), (v, u), (u, x), (x, t) be two paths from s to t, where undirected link (u, v) is an interlacing link. If the interlacing link is excluded from the path pairs, (s, a), (a, u) will be connected with (u, x), (x, t) at node u and (s, b), (b, v) will be connected with (v, w), (w, t) at node v. As a consequence, the pair of paths (s, a), (a, u), (u, x), (x, t) and (s, b), (b, v), (v, w), (w, t) becomes disjoint. □

Corollary A.2.3. *During sequential search for node-disjoint paths, if the next path p_j reaches node u that is already visited by an exiting path p_i, p_j must leave node u through a unidirectional link (u, v) such that undirected link (u, v) becomes the interlacing link between paths p_i and p_j.*

Theorem A.2.4. *If two disjoint paths are derive from the paths p_1 and p_2 by excluding the interlacing link (u, v), the total length of the disjoint path pair will be $l_{p_1} + l_{p_2} - 2 * l_{uv}$.*

Proof. From inspection, the length l_{uv} of the interlacing link (u, v) is counted in both l_{p_1} and l_{p_2}. Thus $2 * l_{uv}$ has to be deducted from $l_{p_1} + l_{p_2}$ to derive correct length of the resultant disjoint path pair. □

In this appendix, we will derive k shortest disjoint paths between nodes s and t. Thus, every path p_j in the rest of the appendix is the jth shortest path between the $s-t$ node pair, where $j \in \{1, 2, \ldots, k\}$. Moreover, the jth shortest path p_j is disjoint from every ith shortest path p_i except in the interlacing link, if any, between them, where $i \in \{1, 2, \ldots, (j-1)\}$.

Theorem A.2.5. *If p_1 is the shortest path and p_2 is the 2nd shortest path in terms of the length $(l_{p_2} - 2 * l_{uv})$, where (u, v) is the interlacing link between p_1 and p_2, the length $(l_{p_1} + l_{p_2} - 2 * l_{uv})$ of the disjoint path pair will be minimum.*

Proof. Let p_3 be the 3rd shortest path in terms the length $(l_{p_3} - 2 * l_{wx})$, where (w, x) is the interlacing link between p_1 and p_3.

As p_2 is the 2nd shortest path in terms of the length $(l_{p_2} - 2 * l_{uv})$, $(l_{p_2} - 2 * l_{uv}) \leq (l_{p_3} - 2 * l_{wx})$; otherwise, p_3 would be the 2nd shortest path in terms of length $(l_{p_3} - 2 * l_{wx})$. Hence, $(l_{p_1} + l_{p_2} - 2 * l_{uv}) \leq (l_{p_1} + l_{p_3} - 2 * l_{wx})$.

Now, let (y, z) be the interlacing link between p_2 and p_3, either one of the two disjoint paths derived from paths p_2 and p_3 by excluding the interlacing link (y, z) be p_{23}, and the length of path p_{23} be $l_{p_{23}}$.

At first, assume that there is no interlacing link between p_1 and p_{23}. As p_1 is the shortest path, $l_{p_1} \leq l_{p_{23}}$; otherwise, path p_{23} would be the shortest path. Thus, $(l_{p_1} + l_{p_2} - 2 * l_{uv}) \leq (l_{p_2} + l_{p_{23}})$. As p_2 is the 2nd shortest path in terms of the length $(l_{p_2} - 2 * l_{uv})$, $(l_{p_2} - 2 * l_{uv}) \leq l_{p_{23}}$; otherwise, path p_{23} would be the 2nd shortest path. Thus, $(l_{p_1} + l_{p_2} - 2 * l_{uv}) \leq (l_{p_1} + l_{p_{23}})$.

Now, let (a, b) be an interlacing link between p_1 and p_{23}. As p_2 is the 2nd shortest path in terms of the length $(l_{p_2} - 2 * l_{uv})$, $(l_{p_2} - 2 * l_{uv}) \leq (l_{p_{23}} - l_{ab})$; otherwise, path p_{23} would be the 2nd shortest path in terms of the length $(l_{p_{23}} - l_{ab})$. Thus, $(l_{p_1} + l_{p_2} - 2 * l_{uv}) \leq (l_{p_1} + l_{p_{23}} - l_{ab})$.

Therefore, $(l_{p_1} + l_{p_2} - 2 * l_{uv})$ is minimum. □

The result is hardly surprising. The lengths of the link-disjoint shortest paths are real numbers. If x, y, and z are the length of shortest paths and $x \leq y \leq z$, this implies that $x + y \leq x + z \leq y + z$. Let us generalize the above theorem for k disjoint paths.

Theorem A.2.6. *If p_1 is the shortest path and $\forall j \in \{2, \ldots, k\}$, p_j is the jth shortest path in terms of the length $(l_{p_j} - 2 * \sum l_{uv_j})$, where $\sum l_{uv_j}$ is the length of all interlacing links of p_j with paths p_1, …, p_{j-1}, the total length $\left(l_{p_1} + \sum_{j=2}^{k}(l_{p_j} - 2 * \sum l_{uv_j})\right)$ of k disjoint paths will be minimum.*

Proof. Proof by induction on the number of disjoint paths k. As p_1 is the shortest path between the $s-t$ node pair, length of $k = 1$ path is minimum. Now, from Theorem A.2.5, the length of $k = 2$ disjoint paths $(l_{p_1} + l_{p_2} - 2 * l_{uv_2})$ is minimum.

Let us assume that the total length $\left(l_{p_1} + \sum_{j=2}^{n}(l_{p_j} - 2 * \sum l_{uv_j})\right)$ of n disjoint paths is minimum.

Let p_{n+1} be the $(n+1)$th shortest path between s and t in terms of length $(l_{p_{n+1}} - 2 * \sum l_{uv_{n+1}})$, where $\sum l_{uv_{n+1}}$ is the length of all interlacing links of p_{n+1} with all the n disjoint

shortest paths. If p_{n+1} is included as the next disjoint shortest paths, p_{n+1} would contribute $(l_{p_{n+1}} - 2 * \sum l_{uv_{n+1}})$ to the total length of $(n+1)$ disjoint shortest paths. Now, the total length of $(n+1)$ disjoint paths that includes path p_{n+1} is $\left(l_{p_1} + \sum_{j=2}^{n}(l_{p_j} - 2 * \sum l_{uv_j})\right) + (l_{p_{n+1}} - 2 * \sum l_{uv_{n+1}})$.

Again, let p_{n+i} be the $(n+i)$th shortest path between s and t in terms of length $(l_{p_{n+i}} - 2 * \sum l_{uv_{n+i}})$, where $i > 1$ and $\sum l_{uv_{n+i}}$ is the length of all interlacing links of p_{n+i} with all the n disjoint shortest paths. If p_{n+i} is included as the next disjoint shortest paths instead of p_{n+1}, p_{n+i} would contribute $(l_{p_{n+i}} - 2 * \sum l_{uv_{n+i}})$ to the total length of $(n+1)$ disjoint shortest paths. Now, the total length of $(n+1)$ disjoint paths that includes path p_{n+i} is $\left(l_{p_1} + \sum_{j=2}^{n}(l_{p_j} - 2 * \sum l_{uv_j})\right) + (l_{p_{n+i}} - 2 * \sum l_{uv_{n+i}})$.

As p_{n+1} will contribute less than or equal to the contribution of p_{n+i} to the total length, $(l_{p_{n+1}} - 2 * \sum l_{uv_{n+1}}) \leq (l_{p_{n+i}} - 2 * \sum l_{uv_{n+i}})$; otherwise, p_{n+i} would be the $(n+1)$th shortest path. Hence, $\left(l_{p_1} + \sum_{j=2}^{n}(l_{p_j} - 2 * \sum l_{uv_j})\right) + (l_{p_{n+1}} - 2 * \sum l_{uv_{n+1}})$ will be less than or equal to $\left(l_{p_1} + \sum_{j=2}^{n}(l_{p_j} - 2 * \sum l_{uv_j})\right) + (l_{p_{k+i}} - 2 * \sum l_{uv_{k+i}})$.

Thus, $\left(l_{p_1} + \sum_{j=2}^{n}(l_{p_j} - 2 * \sum l_{uv_j})\right) + (l_{p_{n+1}} - 2 * \sum l_{uv_{n+1}})$, i.e., $\left(l_{p_1} + \sum_{j=2}^{n+1}(l_{p_j} - 2 * \sum l_{uv_j})\right)$ is minimum. □

Theorem A.2.6 indicates that the disjoint shortest paths can be searched sequentially. However, in the presence of interlacing links, a successor node will have less distance from node s than that of its predecessor if the predecessor is the other end node of a potential interlacing link.

Bhandari [13] provides a sketch for $k > 2$ to prove that interlacing will create negative weight links but not negative weight cycles during sequential search of link-disjoint shortest paths between $s-t$ node pair. As a result, Dijkstra's shortest path algorithm cannot be used to find minimum weight disjoint shortest paths. Thus, Bhandari modifies Dijkstra's algorithm to find the actual next shortest path in the presence of interlacing links.

However, Figure A.1 shows a counter example that interlacing will create negative cycles during sequential search of node-disjoint shortest paths between $s-t$ node pair. The negative cycle is formed by link $(3, 7)$, potential interlacing link $(7, 5)$, and link $(5, 3)$; total weight of the cycle is -1 because the link cost multiplier for link $(7, 5)$, γ_{75}, is -1. The negative cycle is created during search for the 3rd node-disjoint shortest paths from node 4 to node 0.

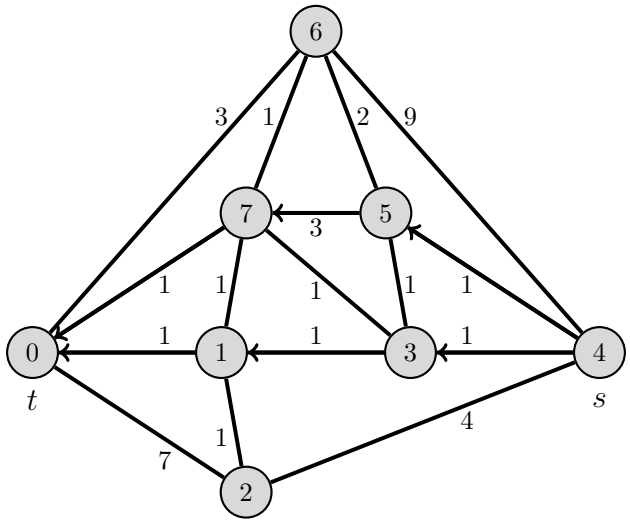


Figure A.1: An example of a negative cycle created during search of 3rd node-disjoint shortest path from node 4 to node 0. The first and the second node-disjoint shortest paths are marked with arrows. The negative cycle is $3 \rightarrow 7 \rightarrow 5 \rightarrow 3$. (the figure is adapted from the figures in Chapter 7 of the book [13])

Bhandari in [13] has proposed vertex splitting technique to find node-disjoint paths in sequence. The technique effectively prevents traversal of negative cycles during subsequent search for node-disjoint shortest paths. Instead of using vertex splitting, the method proposed in this appendix ensures that unidirectional link (u, v) is relaxed from node u *iff* v is not an ancestor of u to avoid negative cycles during subsequent search for node-disjoint shortest paths. Thus, node-disjoint shortest paths derived sequentially may or may not be optimal. Proving optimality of node-disjoint shortest paths needs further investigation. We will deal with the problem in future.

A.3 Algorithms for Link-Disjoint Shortest Paths

We propose a novel method in Algorithm A.1 to find link-disjoint shortest paths. The algorithm will find at most k link-disjoint shortest paths between $s - t$ node pair in the network G_{st}; $s - t$ node pair can be chosen arbitrarily. If t is behind the min-cut from s that has less than k links, the number of disjoint paths from s to t will be at most the

number of the links in the cut-set as per Menger Theorem [3]. The algorithm uses function Bellman-Ford-for-Link-Disjoint-Shortest-Path to find link-disjoint shortest paths one at a time.

Algorithm A.1: Link-Disjoint Path Finding Method

```
   Input: G_st(V, E), s, t, k
   Output: D
   begin
1      D ← φ, n ← 0
       while n < k do
2          p(t) ← Bellman-Ford-for-Link-Disjoint-Shortest-Path(G_st, s, t)
3          if p(t) == φ then Exit the loop.
4          Poison each unidirectional link of p(t) in G_st.
           foreach undirected link (u, v) ∈ V do
               if both unidirectional links (u, v) and (v, u) are poisoned then
5                  Normalize both (u, v) and (v, u).
6          n ← n + 1
7      Delete all non-poisoned unidirectional links from G_st.
8      Normalize all poisoned unidirectional links in G_st.
       while n > 0 do
           Find p(t) from s to t in G_st.
9          D ← p(t)
10         Delete each unidirectional link of p(t) from G_st.
11         n ← n − 1
12     Return D
```

The first *while* loop will find at most k link-disjoint shortest paths between the $s-t$ node pair in G_{st}. In line 2, next link-disjoint shortest path $p(t)$ is derive by calling the helper function Bellman-Ford-for-Link-Disjoint-Shortest-Path. The helpher function finds the next link-disjoint shortest path considering all interlacing links. At any time, if the next link-disjoint shortest path $p(t)$ is not found, there will be no more link-disjoint path between the $s-t$ node pair in G_{st}. Thus, new link-disjoint paths will not be searched any more and the first *while* loop will be terminated in line 3.

If the next link-disjoint shortest path $p(t)$ is found in G_{st}, each unidirectional link of the path is poisoned in line 4. Note that each poisoned unidirectional link in G_{st} is traversed by a link-disjoint shortest path found so far. In line 5, interlacing links will be excluded from each path pair to make them link-disjoint again as stated in Theorem A.2.2.

Once either k or the maximum number of link-disjoint shortest paths between the $s-t$ node pair are derived, all the unidirectional links not traversed by the link-disjoint paths,

which are non-poisoned unidirectional links, are deleted in line 7. Then, all the directional links traversed by the link-disjoint paths, which are poisoned unidirectional links, are normalized in line 8. At this point, G_{st} consists of only unidirectional links traversed by the link-disjoint shortest paths from node s to node t.

The second *while* loop will identify link-disjoint shortest paths in G_{st}. Each iteration of the loop finds one shortest path between nodes s and t by identifying connected links, assigns the shortest path to D, and delete each unidirectional link of the path from G_{st}. D is returned in line 12.

The helper function Bellman-Ford-for-Link-Disjoint-Shortest-Path is a simple extension of the basic Bellman-Ford algorithm to derive the next shortest path from node s to node t in G_{st}. In fact, we introduce only link cost multiplier γ_{uv} for unidirectional link (u, v) in order to find the next shortest path that is link-disjoint from the existing paths in G_{st}.

Function Bellman-Ford-for-Link-Disjoint-Shortest-Path(G_{st}, s, t)

```
   Input: G_st, s, t
   Output: p(t)
   begin
1      ∀v ∈ V, d_v ← ∞, p_v ← φ, d_s ← 0, and p(t) ← φ
       for i ← 1 to |V| − 1 do
           foreach unidirectional link (u, v) ∈ E do
2              γ_uv ← 1
3              if (v, u) is poisoned then γ_uv ← −1
4              if (u, v) is poisoned then γ_uv ← ∞
               if γ_uv ≤ 1 ∧ d_v > d_u + γ_uv * c_uv then
5                  d_v ← d_u + γ_uv * c_uv, p_v ← u
6      if d_t ≠ ∞ then Build p(t) from s to t using predecessors.
7      Return p(t)
```

In the beginning, for each node $u \in V$, the distance d_u of the node from s is ∞ and predecessor p_u of the node is ϕ. The distance of s is assigned 0. The path $p(t)$ is ϕ.

The outer *for* loop iterates $|V| - 1$ times. In each iteration, every unidirectional link of the network is relaxed. During relaxation of each unidirectional link (u, v), cost multiplier γ_{uv} for the link is updated in lines 2 to 4. At first γ_{uv} is initialized as 1 in line 2. If (v, u) is in any existing path, γ_{uv} is -1 to satisfy Theorems A.2.4 and A.2.6 for link-disjoint paths. If (u, v) link is included in the next path, it will create an interlacing link. Hence, twice the cost of the link will be deducted from the total distance of the two relevant paths. On the other hand, if (u, v) is in any existing path, the unidirectional link cannot be a part of

the next path. Hence, γ_{uv} is assigned ∞. Note that both (u, v) and (v, u) cannot be part of two previously found disjoint paths.

If a shorter path from s to v via u that is link-disjoint from the previously identified shortest paths is found, the distance of v, d_v, is updated in line 5 by using link cost c_{uv} of undirected link (u, v) and the link cost multiplier γ_{uv} of unidirectional link (u, v). The predecessor of v, p_v, is also updated in line 5. In this case, u become new predecessor of v.

An example will make clear how the values of the link cost multipliers are assigned. Let node v_1 be connected with nodes v_2, v_3, ..., v_5. Assume that p_{v_1} is v_3. If v_1 is not traversed by any existing shortest path, each link cost multiplier will be 1, i.e., $\gamma_{v_1v_2} = 1$, $\gamma_{v_1v_3} = 1$, $\gamma_{v_1v_4} = 1$, and $\gamma_{v_1v_5} = 1$. However, if v_1 is traversed by an exiting shortest path sp, link cost multipliers will assume different values. Let (v_2, v_1) and (v_1, v_5) are traversed by sp. The link cost multipliers will be $\gamma_{v_1v_2} = -1$, $\gamma_{v_1v_3} = 1$, $\gamma_{v_1v_4} = 1$, and $\gamma_{v_1v_5} = \infty$.

In line 6, if distance of t, d_t, is finite, which indicates a disjoint shortest path is found, path $p(t)$ is formed using predecessors of the nodes in the path. In line 7, $p(t)$ is returned, which may be ϕ.

Now we derive worst-case complexity of the method. The first *while* loop iterates k times. To find a shortest path, O($|V||E|$) steps are required by the helper function. Poisoning links of $p(t)$ in line 4 takes $O(|E|)$ steps. Normalization of links poisoned both way in line 5 takes $O(|E|)$ steps. Thus, the first *while* loop takes O($k|V||E|$) steps to find at most k shortest paths. Deletion of the non-poisoned links in line 7 takes $O(|E|)$ steps. Restoration of the poisoned links in line 8 also takes $O(|E|)$ steps. The second *while* loop takes $O(|E|)$ steps using depth-first search to find link disjoint paths and to form m-trail based on those paths as G_{st} at this point has links of the disjoint paths only. The overall complexity of the method is O($k|V||E|$).

A.4 Algorithms for Node-Disjoint Shortest Paths

We extend Algorithm A.1 to find node-disjoint shortest paths. The method given in Algorithm A.2 will find at most k node-disjoint shortest paths between $s - t$ node pair in the network G_{st}; $s - t$ node pair can be chosen arbitrarily. The algorithm uses function Bellman-Ford-for-Node-Disjoint-Shortest-Path to implement Corollary A.2.3 and Theorem A.2.6 in order to find node-disjoint shortest paths one at a time.

In Algorithm A.2, only line 5 and the *for* loop in line 7 are new codes. Using node coloring, the new codes identify if any existing shortest path traverses a node or not.

Initially each node is black. In line 5, each node traversed by the next shortest path is colored red. In line 8, if all the links connected with a node become normalized, the color of the node is changed back from red to black because no existing shortest path traverses the node at this point. This will happen if an interlacing link is in fact an interlacing segment. Node coloring along with interlacing will be used in the helper function to ensure that the next shortest path is node-disjoint from all the previously identified shortest paths.

Algorithm A.2: Node-Disjoint Path Finding Method

```
   Input: G_st(V, E), s, t, k
   Output: D
   begin
1      D ← φ, n ← 0, and ∀u ∈ V, color[u] ← black
       while n < k do
2          p(t) ← Bellman-Ford-for-Node-Disjoint-Shortest-Path(G_st, s, t, color)
3          if p(t) == φ then Exit the loop.
4          Poison each unidirectional links of p(t) in G_st.
5          ∀u ∈ p(t), color[u] ← red
           foreach undirected link (u, v) ∈ V do
               if both unidirectional links (u, v) and (v, u) are poisoned then
6                  Normalize both (u, v) and (v, u) in G_st.
7          foreach u ∈ p(t) do
               if color[u] is red ∧ each unidirectional link (u, v) ∈ E is not poisoned then
8                  color[u] ← black
9          n ← n + 1
10     Delete all non-poisoned unidirectional links from G_st.
11     Normalize all poisoned unidirectional links in G_st .
       while n > 0 do
           Find p(t) from s to t in G_st.
12         D ← p(t)
13         Delete each unidirectional links of p(t) from G_st.
14         n ← n − 1
15     Return D
```

Similarly, the helper function Bellman-Ford-for-Node-Disjoint-Shortest-Path is an extension of the helper function Bellman-Ford-for-Link-Disjoint-Shortest-Path to derive the next shortest node-disjoint path from node s to node t in G_{st}. In Bellman-Ford-for-Node-Disjoint-Shortest-Path, only portions of the function related with link cost multiplier and link relaxation are modified.

However, in addition to normal distance d_u from s to u and related normal predecessor p_u, each node $u \in V$ has to keep track of three more variables: alternate distance id_u from s

Function Bellman-Ford-for-Node-Disjoint-Shortest-Path(G_{st}, s, t, color)

Input: G_{st}, s, t, color
Output: $p(t)$
begin
1 $\forall u \in V,\ d_u \leftarrow \infty,\ p_u \leftarrow \phi,\ id_u \leftarrow \infty,\ ip_u \leftarrow \phi$, $ap_u \leftarrow false,\ d_s \leftarrow 0$, and $p(t) \leftarrow \phi$
for $i \leftarrow 1$ *to* $|V| - 1$ **do**
foreach *unidirectional link* $(u, v) \in E$ **do**
2 $\gamma_{uv} \leftarrow 1$
3 **if** (v, u) *is poisoned* **then** $\gamma_{uv} \leftarrow -1$
4 **else if** (u, v) *is poisoned* **then** $\gamma_{uv} \leftarrow \infty$
else if $u \neq s \wedge$ color$[u]$ *is red* $\wedge$ (u, p_u) *is not poisoned* **then**
5 $\gamma_{uv} \leftarrow 2$
if v *is not an ancestor of* u **then**
if $\gamma_{uv} \leq 1 \wedge d_v > d_u + \gamma_{uv} * c_{uv}$ **then**
6 $d_v \leftarrow d_u + \gamma_{uv} * c_{uv},\ p_v \leftarrow u,\ ap_v \leftarrow false$
if $\gamma_{uv} == 2 \wedge d_v > id_u + c_{uv}$ **then**
7 $d_v \leftarrow id_u + c_{uv},\ p_v \leftarrow u,\ ap_v \leftarrow true$
if $\gamma_{uv} == -1 \wedge id_v > d_u + \gamma_{uv} * c_{uv}$ **then**
8 $id_v \leftarrow d_u + \gamma_{uv} * c_{uv},\ ip_v \leftarrow u$
if $d_t \neq \infty$ **then**
9 Build $p(t)$ from s to t using predecessors, alternate predecessors, and alternate predecessor flags.
10 Return $p(t)$

to u through a u adjacent interlacing link, alternate predecessor node ip_u on the path from s through the u adjacent interlacing link, and alternate predecessor flag ap_u. The alternate predecessor flag ap_u will indicate whether the shortest path through an interlacing link to the predecessor of u is used for updating d_u or not. In line 1, d_u and id_u are initialized as ∞, p_u and ip_u are initialized as ϕ, and ap_u is initialized as *false*.

In line 5, link cost multiplier γ_{uv} for unidirectional link (u, v) is assigned 2 if a node $u \in V \backslash s$ is traversed by an existing shortest path sp but its predecessor p_u is not traversed by sp, and either unidirectional link (v, u) or (u, v) is not traversed by sp. This new value of the multiplier will block a normal path from s to v via u and will help to find an alternate path from s to v via u if necessary.

Again, an example will make clear how the values of the link cost multipliers are assigned. Let node v_1 that is not s be connected with nodes v_2, v_3, ..., v_5. Assume that p_{v_1} is v_3. If v_1 is not traversed by any existing shortest path, each link cost multiplier will be 1, i.e., $\gamma_{v_1 v_2} = 1$, $\gamma_{v_1 v_3} = 1$, $\gamma_{v_1 v_4} = 1$, and $\gamma_{v_1 v_5} = 1$. However, if v_1 is traversed by

an exiting shortest path sp, link cost multipliers will assume different values. Let (v_2, v_1) and (v_1, v_5) are traversed by sp. The link cost multipliers will be $\gamma_{v_1v_2} = -1$, $\gamma_{v_1v_3} = 2$, $\gamma_{v_1v_4} = 2$, and $\gamma_{v_1v_5} = \infty$. On the other hand, if p_{v_1} is v_5, (v_1, p_{v_1}) will be a poisoned link. As a result, the link cost multipliers will be $\gamma_{v_1v_2} = -1$, $\gamma_{v_1v_3} = 1$, $\gamma_{v_1v_4} = 1$, and $\gamma_{v_1v_5} = \infty$.

To avoid negative cycles, distance d_v from s to v and predecessor p_v of v can be updated if v is not an ancestor of u. In line 6, if $\gamma_{uv} \leq 1$ and a shorter path from s to v through u that is not traversing a u adjacent interlacing link is found, d_v is updated from d_u and u will be the predecessor p_v of v; in this case the alternate parent flag ap_v of v will be false. On the other hand, if $\gamma_{uv} = 2$, d_v cannot be updated from d_u because the shortest path via node u will be blocked to ensure that $p(t)$ is node-disjoint from an existing shortest path sp. However, the path through the potential interlacing link to u is perfectly valid node-disjoint path. Thus, if a shorter path from s to v through u and the potential interlacing link to u is found, distance d_v is updated using alternate distance id_u in line 7. In this case, u will be the predecessor p_v of v and alternate parent flag ap_v of v will become $true$. Thus, a lower cost blocked path to a node cannot prevent a higher cost non-blocked path through an interlacing link to the node to be used.

In line 8, alternate distance id_v from s and alternate predecessor ip_v of v are updated if unidirectional link (u, v) is a potential interlacing link and a lower value of id_v is found.

In line 9, if distance of t is finite, which indicates a disjoint shortest path is found, path $p(t)$ is formed using predecessors, alternate predecessors, and alternate predecessor flags of the nodes in the path. If alternate predecessor flag ap_u of any node u is $true$, the new shortest path from t to s through u and p_u will follow the interlacing link from p_u toward interlacing parent of p_u. In line 10, $p(t)$ is returned, which may be ϕ.

Now we derive the worst-case complexity of the method. The first *while* loop iterates k times. To find a shortest path, $O(|V|^2|E|)$ steps are required by the helper function. The basic Bellman-Ford algorithm needs $O(|V||E|)$ steps. In Bellman-Ford-for-Node-Disjoint-Shortest-Path, additional $O(|V|)$ steps are required to check if v is an ancestor of u or not before each link (u, v) is relaxed. Poisoning links of $p(t)$ in line 4 takes $O(|E|)$ steps. In line 5, $O(|V|)$ steps are needed for node coloring. Normalization of links poisoned both way in line 6 takes $O(|E|)$ steps. The *for* loop in line 7 needs $O(|V| + |E|)$ steps. Thus, the first *while* loop takes $O(k|V|^2|E|)$ steps to find at most k shortest paths. Deletion of the non-poisoned links in line 10 takes $O(|E|)$ steps. Restoration of the poisoned links in line 11 also takes $O(|E|)$ steps. The second *while* loop takes $O(|E|)$ steps using depth-first search to find link disjoint paths and to form m-trail based on those paths as G_{st} at this point has links of the disjoint paths only. The overall complexity of the method is $O(k|V|^2|E|)$.

A.5 Numerical Results

We have conducted numerical experiments to find disjoint paths between each pair of nodes in all the networks given in Appendix B, where each link weight is equal to 1, and the network given in Chapter 7 of the book [13] with all the suggested link weights. Both the algorithms are able to find the maximum possible number of disjoint paths between each node pair of the networks.

A.6 Conclusions

The algorithms developed in this appendix utilize, simplify, or extend the Ford-Fulkerson maximum flow algorithm [18], Suurballe's algorithm for shortest pairs of disjoint paths [50], and Bhandari's algorithm for disjoint paths [12] to derive the maximum number of link-disjoint or node-disjoint shortest paths between any arbitrary node pair of a network.

Appendix B

Networks Used in the Experiments

The following networks are used in various numerical experiments done for the thesis. The MN in each network is arbitrarily chosen.

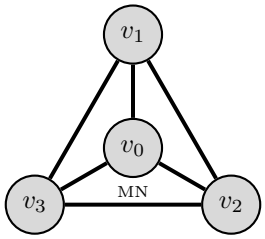


Figure B.1: Tetrahedron: A network with 4 nodes and 6 links. (Refer to [3])

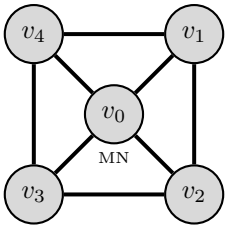


Figure B.2: Network A: A network with 5 nodes and 8 links.

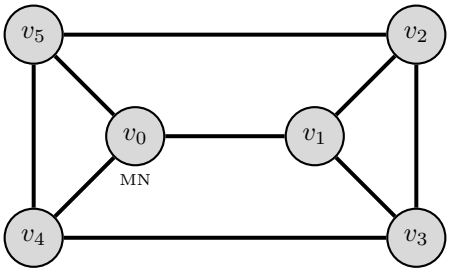


Figure B.3: Network B: A network with 6 nodes and 9 links.

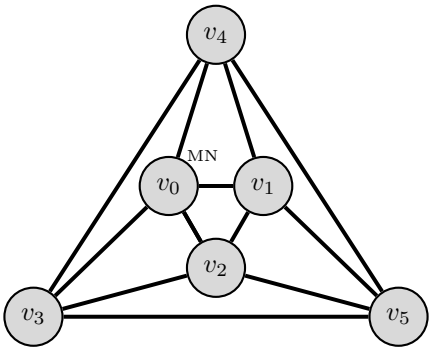


Figure B.4: Octahedron: A network with 6 nodes and 12 links. (Refer to [3])

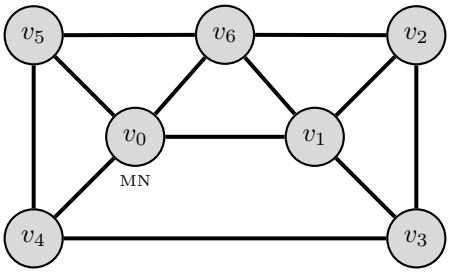


Figure B.5: Network C: A network with 7 nodes and 12 links.

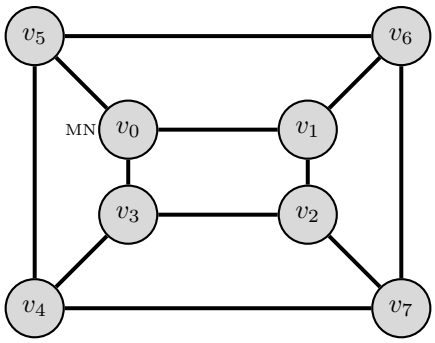


Figure B.6: Cube: A network with 8 nodes and 12 links. (Refer to [3])

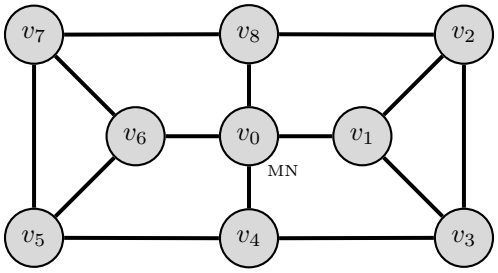


Figure B.7: Network D: A network with 9 nodes and 14 links.

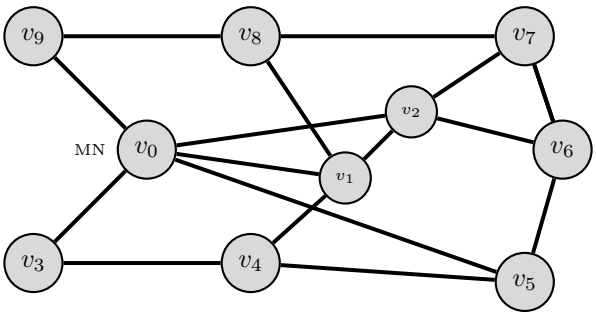


Figure B.8: CERNet: A network with 10 nodes and 16 links. (Refer to [56])

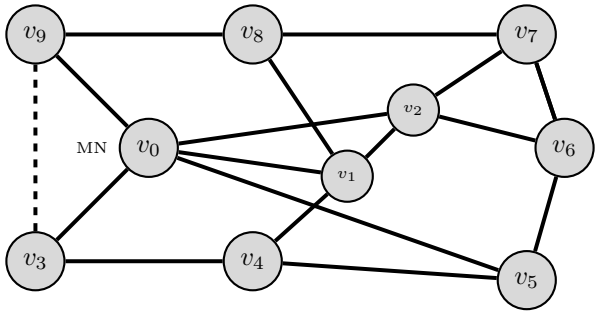


Figure B.9: CERNet + 1 link: A network with 10 nodes and 17 links. One undirected link (v_3, v_9) is added to make the network 3-connected. (Refer to [56])

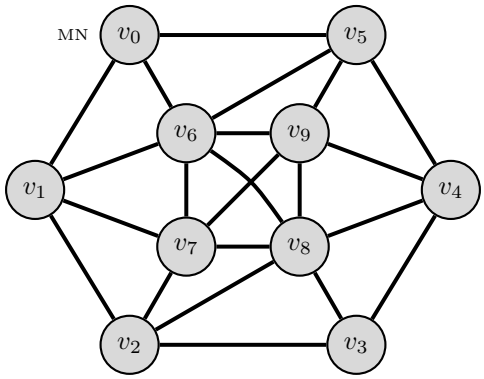


Figure B.10: SmallNet: A network with 10 nodes and 22 links. (Refer to [70])

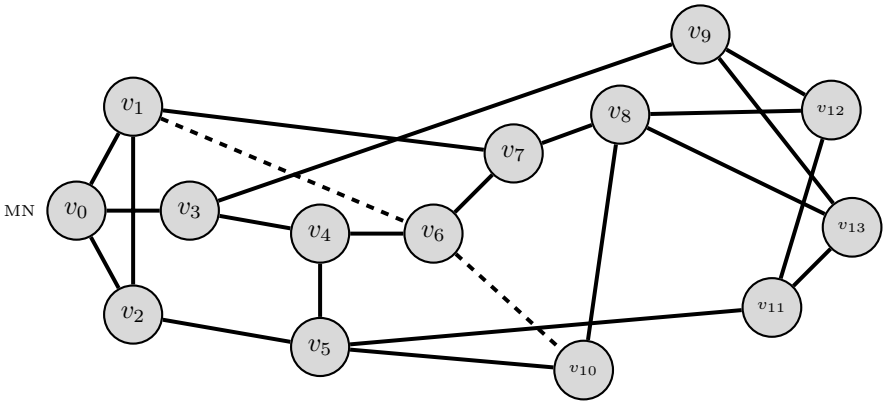


Figure B.11: NSFNet + 2 links: A network with 14 nodes and 23 links. Two undirected links (v_1, v_6) and (v_6, v_{10}) are added to make the network 3-connected. (Refer to [70])

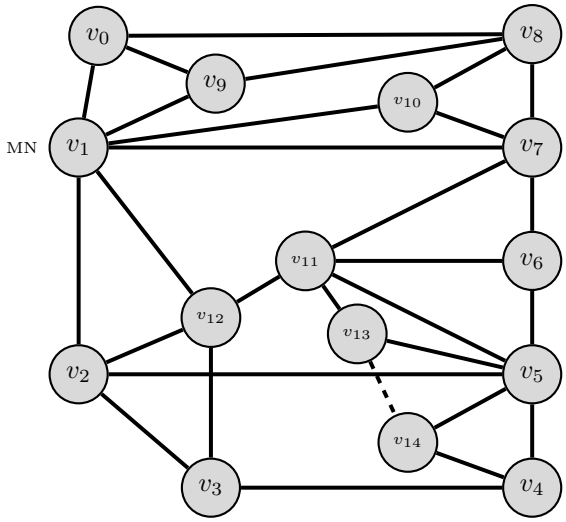


Figure B.12: Bellcore + 1 link: A network with 15 nodes and 29 links. One undirected link (v_{13}, v_{14}) is added to make the network 3-connected. (Refer to [70])

References

[1] S. Ahuja, S. Ramasubramanian, and M. Krunz. SRLG Failure Localization in All-Optical Networks using Monitoring Cycles and Paths. In the proceedings of *IEEE INFOCOM, 2008*, pages 181–185, 2008.

[2] S. Ahuja, S. Ramasubramanian, and M. Krunz. Single Link Failure Detection in All-Optical Networks using Monitoring Cycles and Paths. *IEEE/ACM Transactions on Networking (TON)*, 17(4):1080–1093, August 2009.

[3] J. M. Aldous and R. J. Wilson. *Graph and Applications: An Introductory Approach*. Springer, ISBN 1-85233-259-X, 2000.

[4] M. L. Ali, P.-H. Ho, and J. Tapolcai. Fault Localization in Linear and Ring Networks. *To be submitted*.

[5] M. L. Ali, P.-H. Ho, and J. Tapolcai. SRLG Fault Localization using Nested M-trails. *To be submitted*.

[6] M. L. Ali, P.-H. Ho, and J. Tapolcai. SRLG Fault Localization via M-Burst Framework. In *IEEE ICC'13 ONS (to be presented)*, Budapest, Hungary, June 2013.

[7] M. L. Ali, P.-H. Ho, J. Tapolcai, and B. Shihada. M-Burst: A Framework of SRLG Failure Localization in All-Optical Networks. *IEEE/OSA Journal of Optical Communications and Networking*, 4(8):628–638, August 2012.

[8] M. L. Ali, P.-H. Ho, J. Tapolcai, and S. Subramaniam. SRLG Fault Localization via M-Burst Framework. *To be submitted*.

[9] M. L. Ali, P.-H. Ho, B. Wu, J. Tapolcai, and B. Shihada. Monitoring Burst (M-Burst)-A Novel Framework of Failure Localization in All-Optical Mesh Networks. In the proceedings of *the 8th International Workshop on Design of Reliable Communication Networks, IEEE DRCN 2011*, Krakow, Poland, October 2011.

[10] 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.

[11] R. Bhandari. Optimal Diverse Routing in Telecommunication Fiber Networks. In the proceedings of *IEEE Infocom '94*, volume 3, pages 1498–1508, Toronto, On, 12-16 June 1994.

[12] R. Bhandari. Optimal Physical Diversity Algorithms and Survivable Networks. In the proceedings of *IEEE Symposium on Computer and Communication*, pages 433–441, 1-3 July 1997.

[13] R. Bhandari. *Survivable Networks: Algorithms for Diverse Routing*. Kluwer Academic Publishersr, ISBN 0-7923-8381-8, 1999.

[14] S. Bregni, G. Guerra, and A. Pattavina. State of the Art of Optical Switching Technology for All-Optical Networks. In *Communications World*, Rethymo, Greece, 2001. WSES Press.

[15] Demeester et al. Resilience in Multi-Layer Networks. *IEEE Communication Magazine*, 37(8):70–76, 1999.

[16] R. H. Deng, A. A. Lazar, and W. Wang. A Probabilistic Approach to Fault Diagnosis in Linear Lightwave Networks. *IEEE Journal on Selected Areas in Communications*, 11(9):1438–1448, December 1993.

[17] E. A. Doumith, S. A. Zahr, and M. Gagnaire. Monitoring-Tree: an Innovative Technique for Failure Localization in WDM Translucent Networks. In the proceedings of *IEEE Globecome Conference*, Miami, Florida, USA, December 2010.

[18] L.R. Ford, Jr and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[19] H.-B. Guo and G.-S. Kuo. Improvements on Fault Localization in GMPLS-based Networks. In the proceedings of *2005 Workshop on High Performance Switching and Routing, 2005 HPSR*, pages 94–99, 12–14 May 2005.

[20] A. Haddad, E. A. Doumith, and M. Gagnaire. A Meta-Heuristic Approach for Monitoring Trail Assignment in WDM Optical Networks. In the proceedings of *IFIP/IEEE RNDM Conference*, Moscow, Russia, October 2010.

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

[22] W. He, B. Wu, P.-H. Ho, and J. Tapolcai. Monitoring Trail Allocation for Fast Link Failure Localization without Electronic Alarm Dissemination. In the proceedings of *IEEE ONDM 2011*, 2011.

[23] W. He, B. Wu, P.-H. Ho, and J. Tapolcai. Monitoring Trail Allocation for SRLG Failure Localization. In the proceedings of *IEEE Globecom 2011*, Houston, Texas, USA, December 2011.

[24] I. Katzela and M. Schwartz. Schemes for Fault Identification in Communication Networks. *IEEE/ACM Transactions on Networking*, 3(6):753–764, December 1995.

[25] M. Khair. *Fault Localization for Next Generation Survivable All-Optical Networks*. PhD thesis, University of Ottawa, On, Canada, 2008.

[26] M. Khair, B. Kantarci, J. Zheng, and H. T. Mouftah. Optimization for Fault Localization in All-optical Networks. *Journal of Lightwave Technology*, 27(21), November 2009.

[27] M. Khair, J. Zheng, and H. T. Mouftah. Distributed Fault Localization for Multi-Domain All-Optical Networks with Partial Power Monitoring. In the proceedings of *IEEE Symposium on Computers and Communications (ISCC'08)*, Marrakech, Morocco, July 2008.

[28] M. Khair, J. Zheng, and H. T. Mouftah. Distributed Fault Localization for Multi-Domain All-Optical Networks without Power Monitoring. In the proceedings of *13th International Telecommunications Network Strategy and Planning Symposium (NETWORKS 2008)*, Budapest, Hungary, Sept.-Oct. 2008.

[29] M. Khair, J. Zheng, and H. T. Mouftah. Distributed Multi-Failure Localization Protocol for All-Optical Networks. In the proceedings of *13th international conference on Optical Network Design and Modeling, ONDM 2009*, pages 1–6, IEEE Press Piscataway, NJ, USA, 2009.

[30] Y. Kobayashi, Y. Tada, S. Matsuoka, N. Hirayama, and K. Hagimoto. Supervisory Systems for All-Optical Network Transmission Systems. In the proceedings of the *IEEE GLOBECOM 1996*, pages 933–937, 1996.

[31] M. Laguna. A guide to Implementing Tabu Search. *Investgación Operativa*, 4(1):5–25, April 1994.

[32] J. Lang, editor. *Link Management Protocol (LMP)*. IETF RFC 4204, October 2005.

[33] J. P. Lang and J. Drake. Link Management Protocol (LMP). In *Internet Draft, draft-lang-mpls-lmp-00.txt*. Work in Progress, 2000.

[34] J. H. Lee, N. Yoshikane, T. Tsuritani, and T. Otani. Optical Link Performance Monitoring using Extended Link Management Protocol for Transparent Optical Networks. In the proceedings of the *Optical Fiber Communication Conference, OFC 2009*, San Diego, CA, 2009.

[35] C.-S. Li and R. Ramaswami. Automatic Fault Detection, Isolation, and Recovery in Transparent All-optical Networks. *IEEE Journal of Lightwave Technology*, 15(10):1784–1793, October 1997.

[36] M. Maier. *Optical Switching Networks*. Cambridge University Press, New York, NY, USA, 2008.

[37] C. Mas, I. Tomkos, and O. K. Tonguz. Failure Location Algorithm for Transparent Optical Networks. *IEEE Journal on Selected Areas in Communications*, 23(8):1508–1519, August 2005.

[38] A. Pal, A. Mukherjee, M. K. Naskar, and M. Nasipuri. Minimal Monitor Activation and Fault Localization in Optical Networks. *Optical Switching and Networking*, 8:46–55, 2011.

[39] D. Papadimitriou and E. Mannie, editors. *Analysis of Generalized Multi-Protocol Label Switching (GMPLS)-based Recovery Mechanisms (including Protection and Restoration)*. IETF RFC 4428, March 2006.

[40] C. Qiao and M. Yoo. Optical Burst Switching (OBS) - a New Paradigm for an Optical Internet. *Journal of High Speed Networks*, 8(1):69–84, 1999.

[41] R. Ramaswami and K. N. Sivaranjan. *Optical Networks: A Practical Perspective*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2002.

[42] N. S. V. Rao. Computational Complexity Issues in Operative Diagnosis of Graph-based Systems. *IEEE Transactions Computers*, 42(4):447–457, April 1993.

[43] A. V. Sichani and H. T. Mouftah. Rolling Back Signaling Protocol - A Novel Fault Localization WDM Mesh Networks. In *China Communications*, December 2004.

[44] A. V. Sichani and H. T. Mouftah. Limited-Perimeter Vector Matching Fault-Localization Protocol for Transparent All-optical Communication Networks. *IEEE IET Communications*, 1(3):472–478, June 2007.

[45] S. Stanic, G. Sahin, H. Choi, S. Subramaniam, and H.-A. Choi. Monitoring and Alarm Management in Transparent Optical Networks. In the proceedings of *IEEE BROADNETS, 2007*.

[46] S. Stanic and S. Subramaniam. Fault Localization in All-Optical Networks with User and Supervisory Lightpaths. In the proceedings of *IEEE ICC 2011*.

[47] S. Stanic, S. Subramaniam, H. Choi, G. Sahin, and H.-A. Choi. On Monitoring Transparent Optical Networks. In the proceedings of *International Conference on Parallel Processing Workshops*, pages 217–223, August 2002.

[48] S. Stanic, S. Subramaniam, G. Sahin, H. Choi, and H.-A. Choi. Active Monitoring and Alarm Management for Fault Localization in Transparent All-optical Networks. *IEEE Transactions on Network and Service Management*, 7(2), June 2010.

[49] J. W. Suurballe. Disjoint paths in a Network. *Networks*, 4:125–145, 1974.

[50] J. W. Suurballe and R. E. Tarjan. A Quick Method for Finding Shortest Pairs of Disjoint Paths. *Networks*, 14:325–336, 1984.

[51] 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. *Journal of Lightwave Technology*, 29(10):1597–1606, 15 May 2011.

[52] J. Tapolcai, P.-H. Ho, L. Rónyai, and B. Wu. Network-Wide Local Unambiguous Failure Localization (NWL-UFL) via Monitoring Trails. *IEEE/ACM Transactions on Networking*, to appear in 2013.

[53] J. Tapolcai, L. Rónyai, and P.-H. Ho. Optimal Solutions for Single Fault Localization in Two Dimensional Lattice Networks. In the proceedings of *IEEE Infocom 2010*.

[54] J. Tapolcai, B. Wu, and P.-H. Ho. On Monitoring and Failure Localization in Mesh All-Optical Networks. In the proceedings of *IEEE InfoCom 09*, pages 1008–1016, April 2009.

[55] 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, February 2011.

[56] R. Wang, L. Xu, D. Wu, and S. Huang. An Effecive Mult-Fault Localization Algorithm for Optical Networks. In the proceedings of *2009 Third International Symposium on Intelligent Information Technology Application Workshops*, 2009.

[57] Y. Wen, V. W. S. Chan, and L. Zheng. Efficient Fault-Diagnosis Algorithms for All-Optical WDM Networks with Probabilistic Link Failures. *IEEE/OSA Journal of Lightwave Technology*, 23(10):3358–3371, October 2005.

[58] Y. Wen, V. W. S. Chan, and L. Zheng. Efficient Fault Detection and Localization for All-Optical Networks. In the proceedings of *GlobeCom06*, San Francisco, CA, December 2006.

[59] Y. Wen, V. W. S. Chan, and L. Zheng. Efficient Fault Diagnosis for All-Optical Networks: An Information Theoretic Approach. In the proceedings of *ISIT06*, Seattle, WA, July 2006.

[60] 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 the proceedings of *IEEE Global Telecommunication Conference (GLOBEOCOM) 2010*, pages 1–5, 2010.

[61] B. Wu, P.-H. Ho, J. Tapolcai, and X. Jiang. A Novel Framework of Fast and Unambiguous Link Failure Localization via Monitoring Trails. In the proceedings of *IEEE Infocom 2010 WIP*, 2010.

[62] B. Wu, P.-H. Ho, and K. L. Yeung. Monitoring Trail: a New Paradiam for Fast Link Failure Localization in WDM Mesh Networks. In the proceedings of *IEEE GLOBECOM 08*, 2008.

[63] B. Wu, P.-H. Ho, and K. L. Yeung. Monitoring Trail: on Fast Link Failure Localization in All-Optical WDM Mesh Networks. *IEEE/OSA Journal of Lightwave Technology*, 27(18):4175–4185, September 2009.

[64] B. Wu and K. L. Yeung. M^2-CYCLE: an Optical Layer Algorithm for Fast Link Failure Detection in All-Optical Mesh Networks. In the proceedings of *IEEE GLOBECOM '06*, pages 1–5, December 2006.

[65] B. Wu and K. L. Yeung. Monitoring Cycle Design for Fast Link Failure Detection in All-Optical Networks. In the proceedings of *IEEE GLOBECOM 07*, November 2007.

[66] B. Wu, K. L. Yeung, and P.-H. Ho. Monitoring Cycle Design for Fast Link Failure Localization in All-Optical Networks. *IEEE/OSA Journal of Lightwave Technology*, 27(10):1392–1401, May 2009.

[67] R. Yadav and R. R. Aggarwal. Survey and Comparison of Optical Switch Fabrication Techniques and Architectures. *Journal of Computing, ISSN 2151-9617*, 2(4), April 2010.

[68] H. Yan, R.-Y. Wang, Q.-J. Mao, and D.-P. Wu. A Fast Multi-fault Localization Mechanism for Multi-domain All-optical Networks. In the proceedings of *2010 3rd International Conference on Advance Computer Theory and Engineering (ICACTE)*, 2010.

[69] J. Y. Yen. Finding the K Shortest Loopless Paths in a Network. *Management Science*, 17(11):712–716, July 1971.

[70] H. Zeng, C. Huang, and A. Vukovic. Spanning-Tree based Monitoring-Cycle Construction for Fault Detection and Localization in Meshed AONs. In the proceedings of *IEEE international Conference on Communications 2005*, Seoul, Korea, 16-20 May 2005.

[71] H. Zeng, C. Huang, and A. Vukovic. A Novel Fault Detection and Localization Scheme for Mesh All-Optical Networks based on Monitoring-Cycles. *Photonic Network Communications*, 11(3):277–286, May 2006.

[72] H. Zeng, C. Huang, A. Vukovic, and Savoie M. Fault Detection and Path Performance Monitoring in Meshed All-Optical Networks. In the proceedings of *IEEE GLOBECOM '04*, 2004.

[73] H. Zeng and A. Vukovic. The Variant Cycle-Cover Problem in Fault Detection and Localization for Mesh All-Optical Networks. *Photonic Network Communications*, 14(2):111–122, 2007.

[74] H. Zeng, A. Vukovic, and C. Huang. A Novel End-to-End Fault Detection and Localization Protocol for Wavelength-Routed WDM Networks. In the proceedings of *SPIE 2005, Photonics North*, pages 1–8, Toronto, ON, Canada, 5970 (59702D), September 2005.