Optical Layer Monitoring Schemes for Fast Link Failure Localization in All-Optical Networks

Bin Wu, Pin-Han Ho, Kwan L. Yeung, János Tapolcai and Hussein T. Mouftah

***Abstract*—Optical layer monitoring and fault localization serves as a critical functional module in the control and management of optical networks. An efficient monitoring scheme aims at minimizing not only the hardware cost required for 100% link failure localization, but also the number of redundant alarms and monitors such that the network fault management can be simplified as well. In recent years, several optical layer monitoring schemes were reported for fast and efficient link failure localization, including simple, non-simple monitoring cycle (m-cycle) and monitoring trail (m-trail). Optimal ILP (Integer Linear Program) models and heuristics were also proposed with smart design philosophy on flexibly trading off different objectives. This article summarizes those innovative ideas and methodologies with in-depth analysis on their pros and cons. We also provide insights on future research topics in this area, as well as possible ways for extending the new failure localization approaches to other network applications.**

***Index Terms*—Fast link failure localization, ILP (Integer Linear Program), monitoring cycle (m-cycle), monitoring trail (m-trail), WDM (Wavelength Division Multiplexing).**

I. Introduction

ALTHOUGH the problem of fault detection and localization has been widely studied in general communication networks [1-5], it continues to attract extensive research attentions in optical networks [6-19] as WDM (Wavelength Division Multiplexing) technology is widely deployed in the past decade. Due to the high-speed nature and the vulnerability of WDM-based all-optical networks, fast monitoring and fault localization schemes play a vital role in immediate traffic recovery against a particular component failure. A monitoring scheme monitors the health of the network and helps to localize a component failure, such as a fiber-cut [20-21] which is the most common failure in optical networks. According to Bellcore statistics, the rate of fiber-cuts is about 1-5 cuts per 1000 km each year. If a fiber carries 160 wavelength channels and each operates at 10 Gbps (OC-192), a fiber-cut will result in 1.6 Tbps data loss [22-23]. This leads to great economic damage, as our daily commercial, social and cultural activities have tremendously relied on Internet which is built on top of the optical backbone [20, 24-25]. If a link failure can be accurately localized in a timely manner, the disrupted traffic will be promptly rerouted to bypass the failed link [26]. Accordingly, a fast monitoring scheme helps to minimize service downtime and data loss as well as economic damage.

Basically, either upper layer protocols or optical layer schemes can work alone for fault monitoring. They can also work together in a cross-layer manner. The choice generally depends on the tradeoff between the desired hardware cost and the fault detection time, and it may differ from case to case in practical network implementations. Examples include fault management mechanisms in some routing protocols such as IS-IS (Intermediate System-Intermediate System) and OSPF (Open Shortest Path First) [4], or cross-layer designs [7]. An information theoretic approach is also reported in [8-10], where link failures are localized by analyzing the syndromes (i.e. measurement results) of a minimum number of probe signals. The probe signals are sent onto a set of predetermined lightpaths for fault diagnosis purpose, where a *lightpath* is an all-optically connected path using a wavelength channel on each link along the path. Compared with optical layer monitoring schemes, upper layer protocols need less hardware support but more signaling efforts for fault monitoring. As a result, optical layer monitoring schemes generally respond much faster to a failure event, and thus is preferred in achieving fast link failure localization.

In an optical layer monitoring scheme, a link failure is detected and localized simply based on the on-off status of some supervisory optical signals. This requires additional wavelength channels to transmit the supervisory optical signals, and some special devices called *monitors* [11] to check the on-off status and generate alarms upon a failure event. This hardware cost is necessary for achieving fast link failure localization at the optical layer, and is to be minimized as a major design objective. Meanwhile, reducing the required number of monitors has more significant importance, because it leads to less fault management efforts by managing only a small set of monitors. Due to the transparency (i.e., the all-optical property) of the network, an upstream link failure generally triggers redundant alarms in the monitors equipped at the downstream nodes. It is

Bin Wu and Pin-Han Ho are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, N2L 3G1 (e-mail: b7wu@uwaterloo.ca, pinhan@bbcr.uwaterloo.ca).

Kwan L. Yeung is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong (e-mail: kyeung@eee. hku.hk).

János Tapolcai is with the Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Budapest, Hungary (e-mail: tapolcai@tmit.bme.hu).

Hussein T. Mouftah is with the School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, ON, Canada, K1N 6N5 (e-mail: mouftah@site.uottawa.ca).

reported that a single fiber-cut with only 16 disrupted wavelengths can lead to hundreds of alarms in the network [12]. This not only increases the management cost of the control plane, but also makes the failure localization difficult. Therefore, minimizing the number of required monitors (but without losing the accuracy of the failure localization) can greatly simplify fault management and make the network more scalable. By jointly considering the costs of monitors and supervisory wavelength channels, a *monitoring cost* can be defined (see (1) in Section II.A) which provides a performance metric in comparing different optical layer monitoring schemes.

We assume a single link failure in the network. With the design objective of minimizing the monitoring cost, the goal of a fast monitoring scheme is to accurately localize each possible link failure (i.e., 100% link failure localization). At present, optical layer monitoring schemes generally adopt link-based monitors, where each individual link needs a dedicated monitor. Obviously, 100% link failure localization can be easily ensured in this case, but the number of required monitors equals to the number of links in the network. To reduce the amount of monitoring resources and management efforts, a common approach is to predefine a set of supervisory lightpaths and assign one monitor to each of them. As such, one monitor is capable of generating alarm upon any link failure on the supervisory lightpath [8-10, 13-19]. If multiple supervisory lightpaths pass through the failed link, each of the monitors associated with the supervisory lightpaths will generate an alarm due to the disruption of the supervisory optical signal. Those alarm signals can be denoted by a *binary alarm code*, where each binary bit indicates whether the corresponding monitor alarms or not. If the set of supervisory lightpaths are properly allocated such that each link failure will trigger alarms in a unique set of monitors, then the failure can be localized by identifying the unique alarm code.

In principle, the above mechanism can be interpreted into a one-to-one mapping where each link failure is mapped onto a unique alarm code. It is equivalent to coding each link in the network, subject to a series of constraints including 100% link failure localization, cost minimization, as well as network topology and monitoring structure constraints (i.e., each supervisory lightpath must be properly set up to transmit the supervisory optical signal). Due to such a coding mechanism, it is possible to use only a few bits (or monitors) to monitor the health status of a large number of links. This is the key point why 100% link failure localization can be achieved with a dramatically reduced number of monitors (compared with the conventional link-based monitoring scheme).

In the past, how to design the monitoring structure was a key concern. Although it is now clear that the monitoring structure should be constructed as supervisory ligthpaths, other structures such as monitoring cycle (or m-cycle) [15] also attracted a lot of research interests in previous studies [13-17]. As far as we know, the concept of simple m-cycle is first proposed in [15], where a set of simple cycles are found to cover the network topology and provide supervisory wavelengths for fault monitoring. A simple m-cycle is an optical loopback of supervisory wavelengths and it passes through each on-cycle node exactly once. It is implemented by pre-cross-connecting a supervisory wavelength on each on-cycle link. To design simple m-cycle solutions, heuristic algorithms and ILP (Integer Linear Program) models [14-16] are proposed.

The m-cycle concept is also extended to non-simple m-cycle in [17]. In contrast to a simple m-cycle, a non-simple m-cycle is allowed to pass through a node multiple times. Generally, non-simple m-cycles can better explore mesh connectivity of a network than simple m-cycles due to the more flexible monitoring structure.

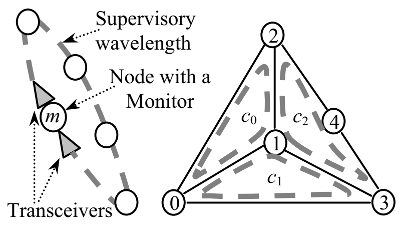
Recently, a new concept of monitoring trail (or m-trail) was proposed [18-19]. It differs from simple and non-simple m-cycles by removing the cycle constraint, and thus an m-trail can be taken as an acyclic supervisory lightpath with an associated monitor equipped at the destination node of the m-trail. Similar to a non-simple m-cycle, an m-trail can pass through a node multiple times. In fact, it is shown in [27] that simple cycle is a special case of non-simple cycle, and both simple and non-simple cycles are special cases of trail (i.e., they can be treated as closed trails). In general, m-trail provides the most flexible monitoring structure in exploring mesh connectivity of the network, and thus achieves the minimum monitoring cost compared with other optical layer monitoring schemes. On the other hand, ILP-based optimal design of m-trails [18] needs huge computation and thus is not scalable. To this end, an efficient heuristic is proposed in [19]. Since minimizing the number of required monitors is very important, multiple papers [14, 19] also analyze the bound on the number of monitors based on some special topologies such as ring and fully-meshed topologies.

The remaining part of this article summarizes the most up-to-date research progress on optical layer monitoring schemes. The concepts of simple, non-simple m-cycles and m-trails are surveyed and the design philosophy of the corresponding algorithms is studied. We also analyze the pros and cons of each scheme, and provide some hints for pending issues and future research topics in this area.

II. Simple m-Cycle

A. Concept of Simple m-Cycle

Fig. 1a shows the monitoring structure of a simple m-cycle, which passes through each on-cycle node exactly once. It is a



Link	c_2	c_1	c_0	Decimal
(0,1)	0	1	1	3
(0,2)	0	0	1	1
(0,3)	0	1	0	2
(1,2)	1	0	1	5
(1,3)	1	1	0	6
(2,4)	1	0	0	4
(3,4)	1	0	0	4

(a) Simple m-cycle (b) Solution (c) Alarm code table

Fig. 1. Fast link failure localization based on simple m-cycles.

loop-back optical pre-cross-connection of supervisory wavelengths with a pair of optical transceivers and a dedicated monitor. If any link on the m-cycle fails, optical supervisory signal in the m-cycle will be disrupted. Accordingly, the monitor detects the off-status of the supervisory signal and generates an alarm.

Assume that an m-cycle solution consists of M m-cycles $\{c_0, c_1, \ldots, c_{M-1}\}$. Upon a particular link failure, optical supervisory signals in all m-cycles passing through the failed link will be disrupted, and the corresponding monitors will alarm. A binary alarm code $[a_{M-1}, \ldots, a_1, a_0]$ is thus obtained, where a_i=1 means that the monitor on m-cycle c_i alarms and a_i=0 otherwise. Fig. 1b shows an example with three m-cycles $\{c_0, c_1, c_2\}$. If link (0, 1) fails, the monitors on c_0 and c_1 will alarm to generate an alarm code [0, 1, 1]. Then, the failure can be localized by referring to the predefined alarm code table in Fig. 1c. Note that in the middle part of the alarm code table, each row gives a binary alarm code for a particular link failure, and each column matches an m-cycle which passes through those links with a corresponding "1" entry in this column.

If a path consists of at least two links and any intermediate node on it has a nodal degree of two, we call the path a *segment* (such as 2−4−3 in Fig. 1b). A cycle-based monitoring scheme cannot distinguish individual failures on the same segment (e.g. the two failures at links (2, 4) and (3, 4) in Fig. 1b). This is because all the links on the same segment must be traversed by the same set of m-cycles. Generally, if a pair of links forms a cut [28] of the network topology, the corresponding link failures cannot be distinguished by any cycle-based monitoring scheme. To achieve 100% link failure localization, extra link-based monitors and supervisory wavelengths are required. In Fig. 1b, an extra link-based monitor and a supervisory wavelength can be added to either (2, 4) or (3, 4) and thus all individual link failures will be distinguishable. Although the total number of monitors is increased from 3 to 4, it is still less than 7, as required by a pure link-based monitoring scheme.

Let the *length* of an m-cycle be the number of links it passes through, or equivalently the number of supervisory wavelength-links it requires. The total length of all cycles in an m-cycle solution is called the *cover length*. Define the *cost ratio* r as the ratio of a monitor cost to the cost of a supervisory wavelength-link. The cost of a monitor may denote only the hardware cost of the monitor. But, if we want to take the quantified fault management cost into account, it can also be amortized into the cost of each monitor. Then, the *monitoring cost* can be defined as

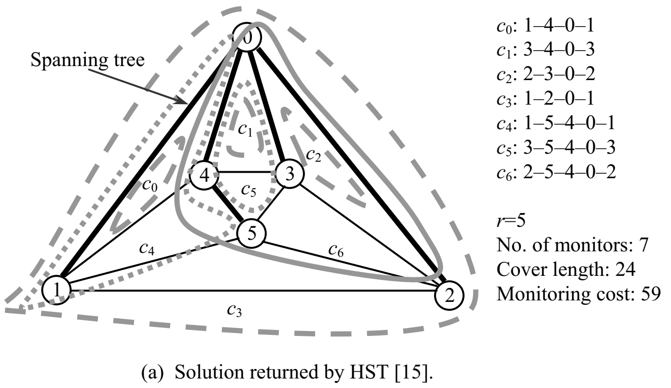
$$\text{Monitoring cost} = r \times \text{number of monitors} + \text{cover length}. \quad (1)$$

Assume r=5. For the solution in Fig. 1b with three simple m-cycles, the cover length is 10 and the monitoring cost is 5 × 3 + 10 = 25. If we need to achieve 100% link failure localization (default hereafter) with one more link-based monitor and supervisory wavelength-link, the monitoring cost will be increased to 25 + 5 + 1 = 31.

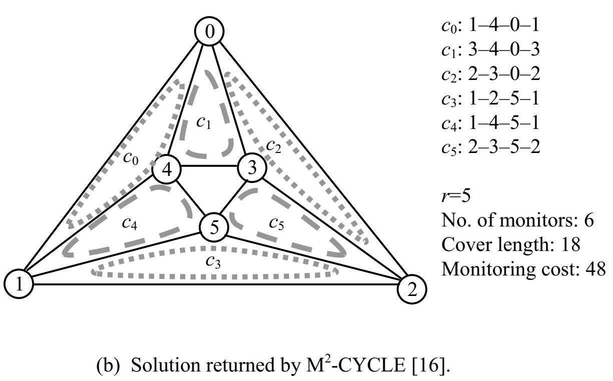
B. Simple m-Cycle Design

A Heuristic Spanning Tree (HST) algorithm is proposed in [15] for simple m-cycle design. HST includes two steps. In the first step, a spanning tree is constructed. The tree roots at the node with the maximum node degree, and always extends at the node with the maximum number of neighbors not yet included in the tree, until a spanning tree is built. Let node 0 in Fig. 2a be the root. All links incident on node 0 are first added to the tree. At this point, all the neighbors of nodes 0 and 5 have already been included in the tree, and each of the nodes 1, 2, 3 and 4 has a common neighbor 5 not yet included in the tree. Assume that the tree extends at node 4. Then, link (4, 5) is added and a spanning tree is obtained. Those links in the spanning tree are called *trunks* (denoted by bold lines), and other links are called *chords*. In the second step, HST generates a simple m-cycle from each chord, where all other links on the m-cycle must be trunks. For example, in Fig. 2a the simple m-cycle generated from chord (1, 4) is c_0: 1-4-0-1 and that from chord (1, 5) is c_4: 1-5-4-0-1.

Another algorithm M^2-CYCLE (minimum-length m-cycle) [16] always constructs m-cycles with the minimum cycle length, as shown in Fig. 2b. M^2-CYCLE also includes two steps. In the first step, the algorithm constructs a set of m-cycles to cover the network topology (i.e., any link in the network must be passed through by at least one m-cycle). The set of m-cycles are constructed one by one, where the algorithm repeatedly picks



(a) Solution returned by HST [15].



(b) Solution returned by M^2-CYCLE [16].

Fig. 2. Heuristics for simple m-cycle design.

up a link uncovered so far and then finds a minimum-length m-cycle to cover it. Based on the set of m-cycles obtained, M^2-CYCLE carries out a refinement process in the second step to generate the final solution. In particular, if two link failures still cannot be distinguished and they do not form a cut of the network topology, M^2-CYCLE adds an additional m-cycle to distinguish them. On the other hand, some m-cycles may be redundant, which can be removed from the solution without decreasing the accuracy of the link failure localization. Such redundant m-cycles can be identified and removed in the refinement process.

It is proved in [16] that M^2-CYCLE always outperforms HST. This can be intuitively understood from three aspects: 1) in an HST solution, all the on-cycle links of an m-cycle must be trunks in the spanning tree, except one chord from which the m-cycle is generated. Accordingly, m-cycles in HST generally have larger cycle lengths than those minimum-length m-cycles generated by M^2-CYCLE; 2) the spanning tree in HST is shared to construct every m-cycle and thus the trunks are highly utilized. This is against the spirit that the set of m-cycles are used to distinguish as many link failures as they can, and thus should pass through different links as possible as they can to provide a better resolution in link failure localization; and 3) due to the spanning tree based mechanism, all m-cycles generated by HST are not redundant. Instead, M^2-CYCLE may generate some redundant m-cycles but will eventually remove them from the final solution. For example, the first step in M^2-CYCLE may generate a redundant cycle 3-4-5-3 in Fig. 2b, along which all the three possible link failures can be distinguished by other surrounding m-cycles c_1, c_4 and c_5. Such kind of redundant cycles are defined as *inside tracks* [16]. In a large-size network, there could be a lot of similar inside tracks and all of them will be removed in the second step of M^2-CYCLE. As a result, M^2-CYCLE may require much less m-cycles than HST.

Besides HST and M^2-CYCLE, the work in [14] provides an ILP and a heuristic to generate monitoring solutions consisting of simple m-cycles and paths, but with different design objectives (e.g., the ILP in [14] minimizes the cover length under the constraint of a single monitoring location).

Compared with non-simple m-cycle and m-trail design, simple m-cycle design requires the least amount of running time, but it generally produces the largest monitoring cost for a given network.

III. Non-Simple m-Cycle

A. *Concept of Non-Simple m-Cycle*

Similar to a simple m-cycle, a non-simple m-cycle is also a loopback optical pre-cross-connection of supervisory wavelengths as shown in Fig. 3. The difference is that a non-simple m-cycle can pass through some nodes multiple times, and thus could have different optical pre-cross-connection patterns of supervisory wavelengths. In Fig. 3, some dotted arrows are used to indicate the pre-cross-connection patterns of the non-simple m-cycles. We can see that the two non-simple m-cycles have the same set of

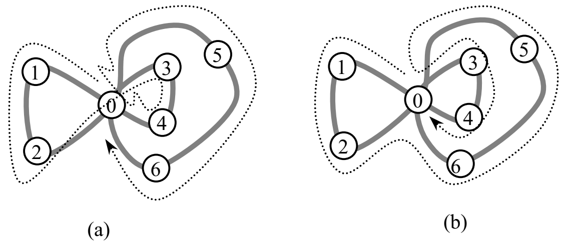


Fig. 3. Non-simple m-cycles.

on-cycle links but different optical pre-cross-connection patterns. However, such a difference in m-cycle implementation will not affect the monitoring result, which is only based on the on-off status of the supervisory optical signal but is independent of the pre-cross-connection pattern of the m-cycle. As a result, the two non-simple m-cycles in Fig. 3 can be practically treated as the same.

Compared with a simple m-cycle, a non-simple m-cycle intends to consume more supervisory wavelength-links due to its more complex optical structure. On the other hand, this also means that more links can share the same monitor, which helps to reduce the required number of monitors. In principle, it is possible that multiple simple m-cycles are combined into a single non-simple m-cycle by rearranging the pre-cross-connection pattern at the intersection nodes. Although non-simple m-cycles still keep the cyclic optical loopback structure, they are more flexible than simple m-cycles in exploring the mesh connectivity of a network [27] due to the weaker constraint on the monitoring structure.

B. *Non-Simple m-Cycle Design*

Compared with simple m-cycle design, non-simple m-cycle design is much more complex. Due to the flexible monitoring structure of non-simple m-cycles, it is not easy to invent a heuristic similar to HST or M^2-CYCLE for non-simple m-cycle design. We also notice that some researchers use ILP (Integer Linear Program) to design simple m-cycles [14], where a cycle enumeration approach is used to find a set of simple cycles in the network before they are fed into the ILP as candidate cycles for optimization. Similar approaches based on candidate cycle enumeration are not practical in non-simple m-cycle design, because the number of all possible non-simple cycles in a network is much more than that of simple cycles, and both numbers grow exponentially with the network size.

A new approach to design non-simple m-cycles is based on ILP but without candidate cycle enumeration. In other words, cycles are directly formulated in the ILP. As formulated in (2), cycles can be defined by requiring each node in the network to have an even number of on-cycle links incident on it [29].

$$\sum_{(u,v)\in \boldsymbol{E}} e^j_{uv} = 2z^j_u, \quad \forall u \in \boldsymbol{V},\ \forall j. \tag{2}$$

In (2), $\boldsymbol{E}$ is the set of all links in the network and $\boldsymbol{V}$ is the set of all nodes. e^j_{uv} is a binary variable where $e^j_{uv} = 1$ indicates that a particular cycle j passes through link (u, v) (0 otherwise). If z^j_u is defined as a binary variable, there could be at most two on-cycle

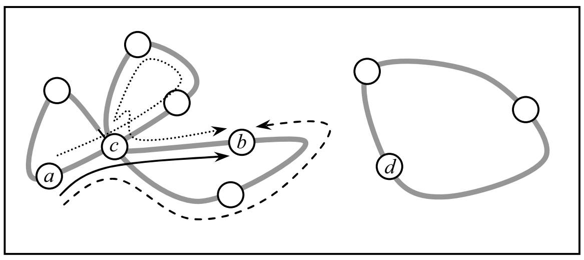


Fig. 4. Multiple disjoint cycles generated by (2).

links of cycle j incident on a node u. This defines simple cycles. On the other hand, if we define z^j_u as a general integer variable, then cycle j can pass through a node multiple times and thus non-simple cycles are enabled.

However, an issue in the above cycle formulation is that multiple disjoint cycles can be generated by (2), although we intend to formulate only a single cycle. As shown in Fig. 4, every node has an even number of on-cycle links incident on it, but we get two disjoint cycles without any common node or link. In fact, multiple disjoint cycles can be generated, and the exact number of cycles is still unknown.

In the optimal design of m-cycles, the above problem is not allowed because we need to accurately count the number of monitors required, which equals to the number of m-cycles. Therefore, it is necessary to ensure that only a single cycle is generated at a time. This is achieved by adding additional constraints in the ILP to exclude all other cycles but keep only one at a time. In particular, a flow-based analysis is carried out in [17], where each node pair in the network is sequentially checked to see whether a flow can exist between the two nodes. The flow must go along only the on-cycle links, and it must obey flow conservation [28] at all nodes except at the node pair currently being checked (which serve as the source and the sink of the flow). In Fig. 4, we can see that there are multiple possible flows between nodes a and b as indicated by the arrows, and thus the two nodes are on the same cycle. On the contrary, we cannot find any flow between nodes a and d because they are on different cycles.

Based on the above flow analysis, the ILP ensures the existence of at least one flow for any pair of on-cycle nodes, or equivalently a single cycle is generated at a time. Note that the flow-based analysis provides a logical approach and it is independent of the pre-cross-connection pattern of the m-cycle. For example, it does not matter which one of the three flows in Fig. 4 exists between nodes a and b, because we only concern about whether a flow can exist or not.

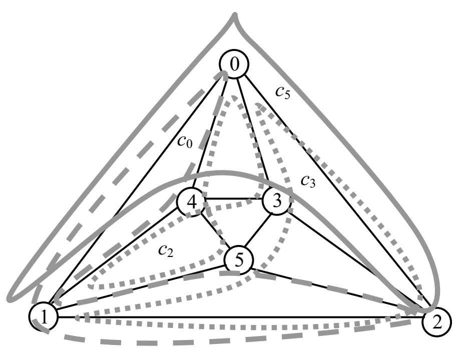
In the ILP-based approach, another key issue is to formulate a distinct alarm code for each distinguishable link failure. To achieve this, *decimal alarm codes* are invented in [17] which are decimal translations of the corresponding binary alarm codes, as shown by the last column in Fig. 1c. Decimal alarm codes can greatly help to reduce the complexity of the ILP-based design, because we can directly check whether two decimal alarm codes are equal or not without bit-wise comparison (which is required in the binary alarm code scenario).

Although ensuring unequal decimal alarm codes is a nonlinear task and thus is not easy to be formulated in an ILP, we can find multiple ways to tackle this problem. The idea in [17] is to let each decimal alarm code take a value from the set of all possible candidate alarm codes, and meanwhile avoid assigning the same value to two different link failures. Assume that J denotes the maximum number of m-cycles in the solution (which needs to be properly estimated as in [17]). Let a_{uv} be the decimal alarm code of link (u, v), and y^k_{uv} be a binary variable indicating whether a decimal value k is assigned to link (u, v) ($y^k_{uv} = 1$) or not ($y^k_{uv} = 0$). Then, the following set of constraints (3)-(6) ensures a distinct decimal alarm code for each link failure (assume that all link failures are distinguishable).

$$\alpha_{uv} = \sum_j 2^j \times e^j_{uv}\,, \qquad \forall (u,v) \in \boldsymbol{E}\,; \tag{3}$$

$$\alpha_{uv} = \sum_{k=1}^{2^J-1} k \times y^k_{uv}\,, \qquad \forall (u,v) \in \boldsymbol{E}\,; \tag{4}$$

$$\sum_{k=1}^{2^J-1} y^k_{uv} = 1\,, \qquad \forall (u,v) \in \boldsymbol{E}\,; \tag{5}$$



c_0: 0–1–2–5–1–4–0
c_2: 0–3–4–1–5–4–0
c_3: 0–2–1–5–3–0
c_5: 0–1–4–3–2–0

r=5
No. of monitors: 4
Cover length: 22
Monitoring cost: 42
J=6
Running time: 30775.22 sec
Gap-to-optimality: 0% (optimal)

Link	c_5	c_4	c_3	c_2	c_1	c_0	Decimal
(0,1)	1	0	0	0	0	1	33
(0,2)	1	0	1	0	0	0	40
(0,3)	0	0	1	1	0	0	12
(0,4)	0	0	0	1	0	1	5
(1,2)	0	0	1	0	0	1	9
(1,4)	1	0	0	1	0	1	37
(1,5)	0	0	1	1	0	1	13
(2,3)	1	0	0	0	0	0	32
(2,5)	0	0	0	0	0	1	1
(3,4)	1	0	0	1	0	0	36
(3,5)	0	0	1	0	0	0	8
(4,5)	0	0	0	1	0	0	4

Fig. 5. Optimal non-simple m-cycle solution.

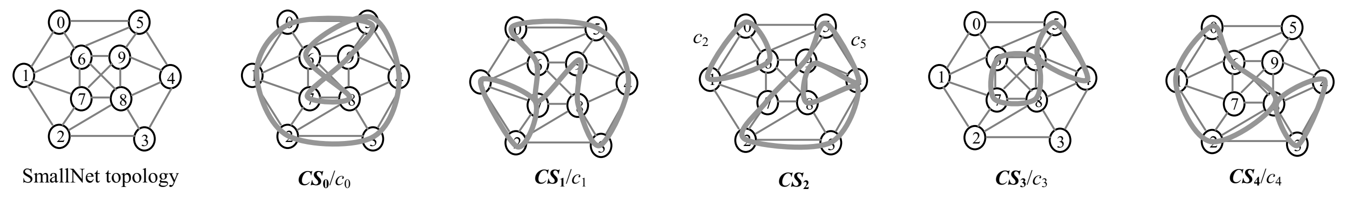
$$\sum_{(u,v)\in \boldsymbol{E}} y_{uv}^{k} \le 1\,,$$

$$\forall\, k \in \{1, 2, \ldots, 2^J-1\}. \qquad (6)$$

In particular, constraint (3) translates a binary alarm code into a decimal one; constraint (4) formulates a decimal alarm code into a combinatorial sum of all possible candidate alarm codes $\{1, 2, \ldots, 2^J-1\}$; constraint (5) specifies that one and only one value in the candidate set $\{1, 2, \ldots, 2^J-1\}$ must be assigned to a specific link failure; finally, constraint (6) says that no two link failures can take the same decimal alarm code. Note that if two links form a cut of the network topology, any cycle-based monitoring scheme cannot distinguish the two link failures. Then, the ILP needs to be slightly modified as detailed in [17]. To achieve 100% link failure localization in this case, a common approach is to add additional link-based monitors [15-17], as we also discussed earlier in the example in Fig. 1.

To achieve optimal design of m-cycles with the minimum monitoring cost, the objective function is formulated as follows.

$$\text{minimize}\left\{ r\sum_{j} m^{j} + \sum_{j}\sum_{(u,v)\in \boldsymbol{E}} c_{uv} e_{uv}^{j} \right\}. \qquad (7)$$



(a) SmallNet topology (with 10 nodes and 22 links) and m-cycles/cycle sets.

Link	CS_7	CS_6	CS_5	CS_4	CS_3	CS_2	CS_1	CS_0	Decimal	Link	CS_7	CS_6	CS_5	CS_4	CS_3	CS_2	CS_1	CS_0	Decimal
(0,1)	0	0	0	1	0	1	0	1	21	(4,5)	0	0	0	0	1	1	1	1	15
(0,5)	0	0	0	0	0	0	1	1	3	(4,8)	0	0	0	1	0	1	0	0	20
(0,6)	0	0	0	1	0	1	1	0	22	(4,9)	0	0	0	0	1	1	0	0	12
(1,2)	0	0	0	1	0	0	1	1	19	(5,6)	0	0	0	0	0	0	0	1	1
(1,6)	0	0	0	0	0	1	0	0	4	(5,9)	0	0	0	0	1	1	0	1	13
(1,7)	0	0	0	0	0	0	1	0	2	(6,7)	0	0	0	0	1	0	1	0	10
(2,3)	0	0	0	0	0	1	0	1	5	(6,8)	0	0	0	1	0	0	0	1	17
(2,7)	0	0	0	0	0	1	1	0	6	(6,9)	0	0	0	0	1	0	0	0	8
(2,8)	0	0	0	1	0	0	0	0	16	(7,8)	0	0	0	0	1	0	0	1	9
(3,4)	0	0	0	1	0	1	1	1	23	(7,9)	0	0	0	0	0	1	1	1	7
(3,8)	0	0	0	1	0	0	1	0	18	(8,9)	0	0	0	0	1	1	1	0	14

(b) Alarm code table based on $\boldsymbol{CS_j}$.

Heuristic ILP:

J: 8
γ: 0
Running time: 152.14 sec
Gap-to-"optimality": 4.53%
No. of monitors: 6
Cover length: 49

HST [15]:

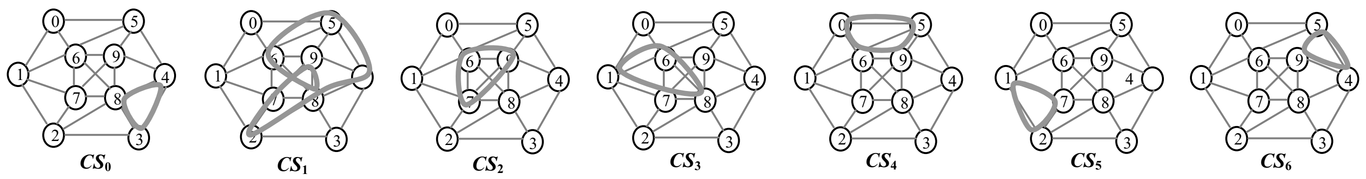
No. of monitors: 13
Cover length: 43

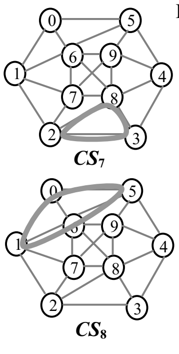
M²-CYCLE [16]:

No. of monitors: 12
Cover length: 36

(c) Comparison with existing algorithms.

Fig. 6. m-cycle design based on the heuristic ILP for SmallNet with J=8 and γ=0.





Heuristic ILP model:

J: 9
γ: 1024
Running time: 1795.15 sec
Gap-to-"optimality": 2.88%
No. of monitors: 9
Cover length: 35

M²-CYCLE [16]:

No. of monitors: 12
Cover length: 36

Link	CS_8	CS_7	CS_6	CS_5	CS_4	CS_3	CS_2	CS_1	CS_0	Decimal	Link	CS_8	CS_7	CS_6	CS_5	CS_4	CS_3	CS_2	CS_1	CS_0	Decimal
(0,1)	1	0	0	0	0	0	0	0	0	256	(4,5)	0	0	1	0	0	0	0	1	0	66
(0,5)	1	0	0	0	1	0	0	0	0	272	(4,8)	0	0	0	0	0	0	0	1	1	3
(0,6)	0	0	0	0	1	0	0	0	0	16	(4,9)	0	0	1	0	0	0	0	0	0	64
(1,2)	0	0	0	1	0	0	0	0	0	32	(5,6)	1	0	0	0	0	0	0	1	0	258
(1,6)	1	0	0	0	0	1	0	0	0	264	(5,9)	0	0	1	0	1	0	0	0	0	80
(1,7)	0	0	0	1	0	1	0	0	0	40	(6,7)	0	0	0	0	0	0	1	0	0	4
(2,3)	0	1	0	0	0	0	0	0	0	128	(6,8)	0	0	0	0	0	1	0	1	0	10
(2,7)	0	0	0	1	0	0	0	1	0	34	(6,9)	0	0	0	0	1	0	1	0	0	20
(2,8)	0	1	0	0	0	0	0	1	0	130	(7,8)	0	0	0	0	0	1	0	0	0	8
(3,4)	0	0	0	0	0	0	0	0	1	1	(7,9)	0	0	0	0	0	0	1	1	0	6
(3,8)	0	1	0	0	0	0	0	0	1	129	(8,9)	0	0	0	0	0	0	0	1	0	2

Fig. 7. m-cycle design based on the heuristic ILP for SmallNet with J=9 and γ=1024.

Basically, objective (7) is a direct translation of (1). In (7), m^j=1 means that cycle j is an m-cycle in the solution and m^j=0 otherwise. c_{uv} is the cost of adding one supervisory wavelength to link (u, v) (c_{uv}=1 if hop-count is used as the cost metric). So the first term in (7) denotes the number of monitors, and the second term denotes the cover length. Note that HST and M^2-CYCLE heuristics can only generate simple m-cycle solutions without any tradeoff between the number of monitors and the cover length. Such a tradeoff is enabled in the ILP-based approach by adjusting the value of r. Fig. 5 shows the optimal non-simple m-cycle solution for the same network in Fig. 2. We can see that only four m-cycles are required. Since we have preset a value of J=6 which is larger than necessary, c_1 and c_4 are empty m-cycles which do not pass through any link. Compared with the simple m-cycle solutions in Fig. 2, both the number of required monitors and the monitoring cost in Fig. 5 are much smaller.

However, ILP-based optimal design of non-simple m-cycles needs a very long running time as indicated in Fig. 5, and thus is not scalable at all. This is because the optimal ILP needs to carry out the flow-based analysis, as well as ensuring a unique alarm code for each distinguishable link failure as formulated in (3)-(6).

To reduce the running time, a heuristic is proposed in [17], which still adopts the ILP-based approach but the flow-based analysis is removed. As a result, cycles are generated only based on (2), and multiple disjoint cycles may coexist at the same time (defined as a *cycle set* $\boldsymbol{CS}_j$). Accordingly, the heuristic cannot accurately count the number of m-cycles and monitors. The objective of the heuristic is to minimize the sum of all the decimal alarm codes plus the cover length. As we can see from an alarm code table, if the sum of all the decimal alarm codes is minimized, the required number of bits in the binary alarm codes can be suppressed, which has a direct effect in reducing the required number of m-cycles and monitors. As a side effect, this also helps to reduce the total number of 1s in the binary alarm codes, which is equivalent to reducing the cover length. Generally, the former effect is much greater than the latter [17]. So, the ILP-based heuristic uses the sum of all the decimal alarm codes as a heuristic measure for monitor cost. Similar to (7), the cover length can be included in the objective function, and a *heuristic cost ratio* γ can also be applied to provide a way for achieving tradeoff between the monitor cost and the cover length, although such a tradeoff control is not as accurate as that in the optimal ILP model. By removing the flow-based analysis, the ILP can be greatly simplified with much shorter running time, but optimal solutions are not ensured.

Figs. 6 and 7 show two solutions generated by the heuristic ILP [17] for the same SmallNet topology [15], where the value of the heuristic cost ratio γ is varied in the two figures to achieve a tradeoff between the monitor cost and the cover length.

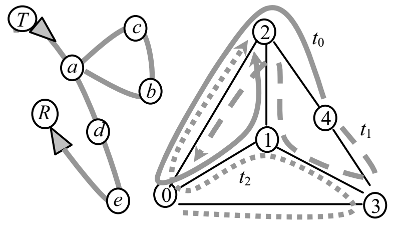
IV. Monitoring Trail (m-Trail)

A. Concept of m-Trail

Although non-simple m-cycle is much more flexible than simple m-cycle in exploring the mesh connectivity of a network, it is still subject to the cycle constraint on the monitoring structure. This limits the flexibility in coding each link with a distinct alarm code. An apparent observation is that, without the aid of link-based monitoring, a cycle-based monitoring scheme cannot distinguish two link failures if the two links form a cut of the network topology. If link-based monitoring is used, although 100% link failure localization can always be ensured, the m-cycle design algorithms discussed earlier still fail in carrying out a joint design by optimizing the allocation of both m-cycles and link-based monitors at the same time.

In fact, the monitoring structure is not necessarily cycle-based. By removing the cycle constraint, the monitoring structure can be arbitrary trails which are general supervisory lightpaths. Such a monitoring structure is called monitoring trail, or m-trail [18-19]. Similar to a non-simple m-cycle, an m-trail can pass through a node multiple times, and different pre-cross-connection patterns of the supervisory wavelengths do not affect the monitoring result.

Fig. 8a shows the structure of an m-trail, where the two triangles denote a pair of optical transceivers to support the supervisory optical signal, and a dedicated monitor is equipped at the sink node R. If any link on the m-trail fails, the monitor detects the off-status of the supervisory optical signal and generates an alarm. Fig. 8b shows an m-trail solution for the same network in Fig. 1b, with the alarm code table in Fig. 8c. Compared with the simple m-cycle solution in Fig. 1b which requires three simple m-cycles and one link-based monitor to achieve 100% link failure localization, only three m-trails and monitors are required in Fig. 8b. Notably, the two links (2, 4) and (3, 4) form a cut of the network topology, but the corresponding link failures can be distinguished by two m-trails



Link	t_2	t_1	t_0	Decimal
(0,1)	1	0	1	5
(0,2)	1	1	1	7
(0,3)	1	0	0	4
(1,2)	0	1	1	3
(1,3)	1	1	0	6
(2,4)	0	0	1	1
(3,4)	0	1	0	2

(a) m-trail (b) An m-trail solution (c) Alarm code table

Fig. 8. Fast link failure localization based on m-trails.

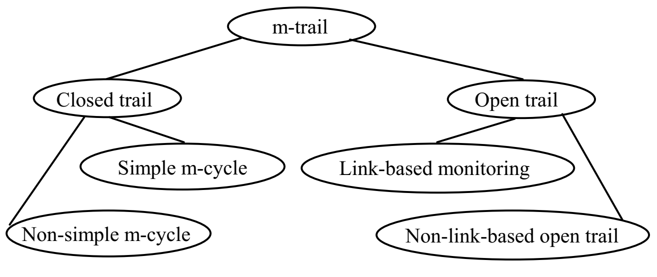


Fig. 9. m-trail is a generalization of all monitoring structures.

t_0 and t_1 due to their acyclic monitoring structure.

The concept of m-trail generalizes all the previously studied monitoring structures, including simple, non-simple m-cycles and link-based monitoring. In particular, simple and non-simple m-cycles can be treated as closed trails, and link-based monitoring is also a special case of m-trail. As a supervisory lightpath, an m-trail can also take the general structure of non-link-based open trail. Fig. 9 summarizes the relationship among different monitoring structures studied so far.

B. M-Trail Design

We first consider the optimal design of m-trails using an ILP approach. As shown in Fig. 10, we use *on-trail vectors* (vectors for short) to denote the supervisory wavelengths of an m-trail t_j. A vector $u \rightarrow v$ denotes a supervisory wavelength of t_j on link (u, v), with the supervisory optical signal transmitted from node u to node v. Each m-trail t_j has a unique source-sink (i.e., *T-R*) node pair. Let Δ_u be the difference of the number of outbound and inbound vectors at node u. For an open trail, we have $\Delta_T=1$, $\Delta_R=-1$, and $\Delta_u=0$ for $u \neq T$ and $u \neq R$. In other words, the vectors must obey flow conservation at each node, except at source T and sink R. For a closed trail, T and R denote the same node (we still use the term "*T-R* node pair" for simplicity), and we have $\Delta_u=0$ for each node u in the network.

Similar to the ILP-based non-simple m-cycle design [17], it is incomplete to formulate an m-trail only based on the above flow conservation property, because multiple disjoint cycles may be generated at the same time. As we can see in Fig. 10a, although flow conservation is not violated, we get an open trail and a disjoint cycle.

To generate a single m-trail t_j at a time, a voltage analysis is carried out [18]. The purpose of the voltage analysis is to complement the above flow-conservation-based m-trail formulation, such that only a single m-trail is formulated by excluding all other possible disjoint cycles. In particular, a positive *voltage* value is defined for each vector $u \rightarrow v$ on t_j, as denoted by a fraction next to each vector in Fig. 10. If a link (u, v) is not traversed by t_j, by default the corresponding voltage value is 0. For any node traversed by t_j except sink R, the sum of the voltage values of its outbound vectors must be larger than that of its inbound vectors. This is called the *voltage constraint*. Voltage value only provides an analytical tool to support our voltage analysis, and it does not have any physical meaning in real world. Accordingly, the specific voltage values are not important as long as the voltage constraint is obeyed. Note that the voltage constraint does not apply to sink R. If an m-trail t_j traverses any node at most once as shown in Figs. 10a & 10b, the voltage values of the vectors must keep increasing along t_j, except at sink R where a voltage decrease may occur. In Fig. 10a, we can find a feasible set of voltage values for those vectors on the open trail. Since the cycle does not pass through sink R, the voltage values of those on-cycle vectors must keep increasing along the cycle. Then, a voltage conflict will occur due to the cyclic structure, and the voltage constraint cannot be satisfied at every node (e.g. at node d).

Based on the above mechanism, the ILP can exclude all other disjoint cycles but generate a single m-trail at a time. On

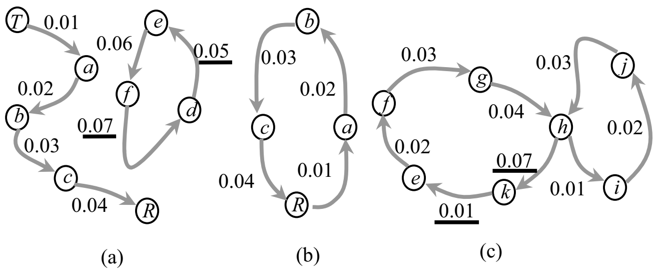


Fig. 10. Formulating a single m-trail using the voltage constraint.

the other hand, if a closed trail (i.e., a cycle) passes through sink R as shown in Fig. 10b, voltage conflict will not occur because the voltage constraint does not apply to sink R. Since the ILP ensures a single *T-R* node pair for each m-trail, only the trail passing through the unique *T-R* node pair is kept and all other disjoint trails will be excluded. Note that the voltage constraint can also be applied to the case where some nodes are traversed multiple times by an m-trail [18], as illustrated in Fig. 10c. With the above voltage analysis, the ILP can accurately count the number of m-trails in the solution. Accordingly, the total cost of all monitors can be properly formulated.

To assign a distinct alarm code to each possible link failure, the concept of decimal alarm codes is used to avoid bit-wise comparisons in the corresponding binary alarm codes. Although the set of constraints (3)-(6) can be directly used in m-trail design, a huge number of variables will be involved. As we can see in constraints (4)-(6), y^k_{uv} will introduce $(2^J-1)\times\|\boldsymbol{E}\|$ variables to the ILP solver, where J is the maximum number of m-cycles or m-trails in the solution and $\|\boldsymbol{E}\|$ is the total number of links in the network. This number increases exponentially with J, and thus the ILP running time will be dramatically increased if the network size is large.

To reduce the required number of ILP variables, a smarter way is invented in [18] to ensure a distinct alarm code for each link failure. Specifically, a binary variable f_{uv}^{xy} is defined to indicate the inequality between two decimal alarm codes α_{uv} and α_{xy}. $f_{uv}^{xy}=1$ means $\alpha_{uv}>\alpha_{xy}$ and $f_{uv}^{xy}=0$ means $\alpha_{uv}<\alpha_{xy}$. Let $(u,v),(x,y)\in\boldsymbol{E}:(u,v)\neq(x,y)$ denote two distinct links (u, v) and (x, y). With a predefined small positive constant β, the following constraints (8)-(9) can efficiently ensure $\alpha_{uv}\neq\alpha_{xy}$. Otherwise, the two constraints cannot hold at the same time, no matter whether the binary variable f_{uv}^{xy} takes the value of 0 or 1.

$$\beta+\beta\left(\alpha_{uv}-\alpha_{xy}\right)\leq f_{uv}^{xy}, \quad \forall(u,v),(x,y)\in\boldsymbol{E}:(u,v)\neq(x,y); \tag{8}$$

$$\beta+\beta\left(\alpha_{xy}-\alpha_{uv}\right)\leq 1-f_{uv}^{xy}, \quad \forall(u,v),(x,y)\in\boldsymbol{E}:(u,v)\neq(x,y). \tag{9}$$

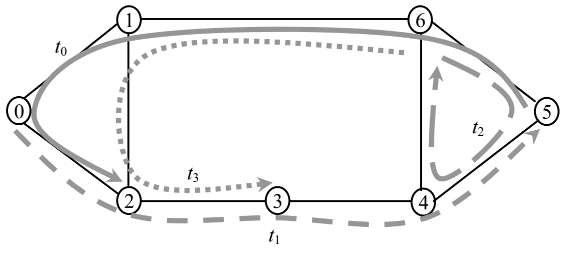
In constraints (8)-(9), the specific value of β is not important, as long as it is a positive value small enough to ensure $\beta+\beta\times|\alpha_{uv}-\alpha_{xy}|\leq 1$. For example, if we allow 9 m-trails in the solution, the candidate set of all the decimal alarm codes is $\{1, 2, \ldots, 2^9-1\}$, and β can be predefined in $0<\beta\leq 1/512$. In contrast to the $(2^9-1)\times\|\boldsymbol{E}\|$ variables (i.e., y^k_{uv}) in (4)-(6), the number of variables (i.e., f^{xy}_{uv}) in (8)-(9) is dramatically reduced to $\|\boldsymbol{E}\|\times(\|\boldsymbol{E}\|-1)/2$.

The objective function in m-trail design is similar to (7) and is formulated by minimizing the monitoring cost in (1). We will not give the details of the ILP in this article, and interested readers may refer to [18]. With the ILP-based approach, an efficient tradeoff between the monitor cost and the cover length can be achieved by adjusting the value of the cost ratio r. Since m-trail is the most general monitoring structure and it takes all the previously studied monitoring structures as special cases (see Fig. 9), an m-trail solution can always ensure 100% link failure localization for any network topology. On the other hand, optimal design of m-trails ensures the true optimality of the monitoring resource allocation, because all possible monitoring structures are jointly optimized.

Fig. 11 shows an optimal m-trail solution generated by the ILP [18] for a simple network topology. Clearly, m-trail t_2 is a closed trail, or an m-cycle. Fig. 12 shows an m-trail solution for the SmallNet Topology [15]. In both examples, J is predefined larger than necessary. The final solution contains less than J m-trails, and others are empty trails which do not pass through any link. For example, t_3, t_5 and t_6 in Fig. 12 are empty trails. Basically, empty trails can be removed from the alarm code table by rearranging the value of all the decimal alarm codes accordingly.

Since the ILP-based approach is not scalable, a two-step heuristic is proposed in [19] for scalable m-trail design in large-size networks. The first step is called Random Code Assignment (RCA), where a unique (temporary) alarm code is assigned to each link failure in a random manner, with the key spirit of ensuring 100% link failure localization first. The second step is an iterative process called Random Code Swapping (RCS), which shapes the monitoring structures obtained from RCA into m-trails.

In RCS, each bit in all binary alarm codes is sequentially analyzed to check whether one or a set of disjoint m-trails are formed or not. Note that an m-trail must obey flow conservation at all nodes except at the source T and the sink R, but the monitoring structures obtained from RCA may not satisfy this constraint. Accordingly, RCS initiates an iterative process to find those nodes with odd degree of on-trail vectors. By

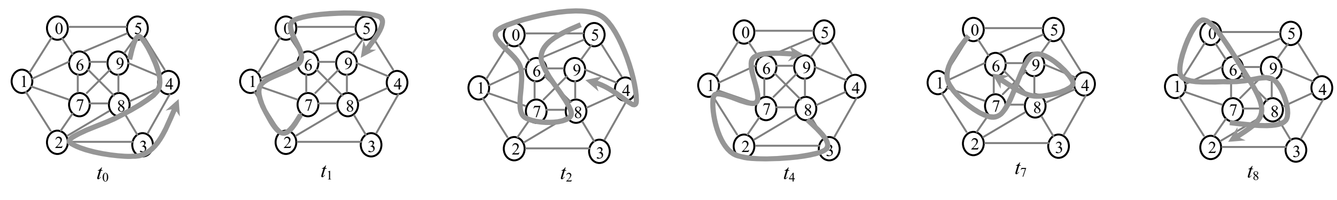


Link	t_3 t_2 t_1 t_0	Decimal	Link	t_3 t_2 t_1 t_0	Decimal
(0,1)	0 0 0 1	1	(3,4)	0 0 1 0	2
(0,2)	0 0 1 1	3	(4,5)	0 1 1 0	6
(1,2)	1 0 0 0	8	(4,6)	0 1 0 0	4
(1,6)	1 0 0 1	9	(5,6)	0 1 0 1	5
(2,3)	1 0 1 0	10			

Predefined parameters (see [18] for details): J=12, γ=5, λ=0.01, β=0.0001, B=34

Results: ILP running time=12.70 sec, Gap to optimality=0% (Optimal), Number of m-trails=4, Monitoring cost=34

Fig. 11. Optimal m-trail solution for a simple network topology.



Link	t_8	t_7	t_6	t_5	t_4	t_3	t_2	t_1	t_0	Decimal	Link	t_8	t_7	t_6	t_5	t_4	t_3	t_2	t_1	t_0	Decimal
(0,1)	1	1	0	0	0	0	0	0	0	384	(4,5)	0	0	0	0	0	0	1	0	1	5
(0,5)	0	0	0	0	0	0	1	1	0	6	(4,8)	0	1	0	0	0	0	0	0	1	129
(0,6)	1	0	0	0	0	0	1	1	0	262	(4,9)	0	1	0	0	0	0	1	0	0	132
(1,2)	0	0	0	0	1	0	0	1	0	18	(5,6)	0	0	0	0	0	0	1	0	0	4
(1,6)	1	0	0	0	0	0	0	1	0	258	(5,9)	0	0	0	0	0	0	0	1	1	3
(1,7)	0	1	0	0	1	0	0	0	0	144	(6,7)	0	0	0	0	1	0	1	0	0	20
(2,3)	0	0	0	0	1	0	0	0	1	17	(6,8)	1	1	0	0	0	0	1	0	0	388
(2,7)	0	0	0	0	0	0	0	1	0	2	(6,9)	1	0	0	0	1	0	0	0	0	272
(2,8)	1	0	0	0	0	0	0	0	1	257	(7,8)	1	0	0	0	0	0	1	0	0	260
(3,4)	0	0	0	0	0	0	0	0	1	1	(7,9)	0	1	0	0	0	0	0	0	0	128
(3,8)	0	0	0	0	1	0	0	0	0	16	(8,9)	1	0	0	0	0	0	0	0	0	256

Predefined parameters (see [18] for details):

J=9
r=5
δ=0.01
β=0.001
B=69

Result:

Solution time=761.84 sec
Gap to optimality=1.43%
Number of m-trails=6
Monitoring cost=70

Fig. 12. m-trail solution for the SmallNet topology with 10 nodes and 22 links.

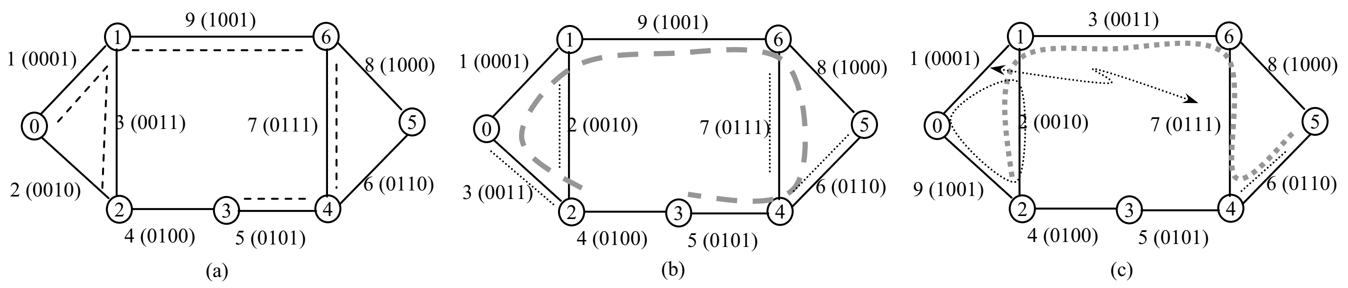


Fig. 13. Heuristic m-trail design algorithm based on RCA and RCS.

properly swapping alarm codes between some pairs of links based on a bit-wise analysis, RCS gradually reduces the number of nodes with odd degree of on-trail vectors, until an m-trail (or an m-trail and a set of disjoint m-cycles) is formed, which means there are at most two nodes in the network with odd degree of on-trail vectors. This process is iteratively carried out bit-by-bit from the LSB (Least Significant Bit) of the binary alarm codes to the MSB (Most Significant Bit), until all the monitoring structures are shaped into m-trails and m-cycles.

When swapping two alarm codes between a pair of links, the algorithm ensures that all the previously formed m-trails (i.e., those binary bits analyzed so far) will not be changed. Whenever it is found that the number of bits (i.e., the length of a binary alarm code, or equivalently the number of monitors) is not sufficient for ensuring 100% link failure localization, a new bit is added as the MSB of the binary alarm codes. This increases the length of the binary alarm codes by one, but doubles the size of the candidate alarm code set such that more candidate alarm codes can be used for swapping. Eventually, 100% link failure localization, which is initially ensured in RCA, will be kept throughout the RCS process, and all monitoring structures will be shaped into m-trails by RCS. In short, RCA carries out an initial alarm code assignment to ensure 100% link failure localization, and RCS aims at shaping the monitoring structures into m-trails by swapping some alarm codes based on the flow conservation constraint.

The above idea is illustrated by a simple example in Fig. 13, where the network topology is the same as that in Fig. 11. Fig. 13a shows the result of RCA, where a set of decimal alarm codes 1-9 is randomly assigned to the links with their binary translations in the brackets. The LSBs of all the binary alarm codes are first analyzed, where a "1" denotes an on-trail vector and a "0" otherwise. The resulting vectors are indicated by the dashed lines in Fig. 13a. We can see that four nodes 0, 1, 2 and 3 have odd number of dashed lines incident on each. If we swap the alarm codes between two links (0, 2) and (1, 2), the updated alarm code assignment is shown in Fig. 13b. The number of nodes with odd degree of vectors is reduced from 4 to 2, and the structure consisting of the dashed lines in Fig. 13a is shaped into the broad-brush m-trail in Fig. 13b. Since an m-trail is obtained, the analysis on the LSBs of the binary alarm codes is complete. Then, RCS turns to check the second bit, where the vectors indicated by "1" are denoted by the dotted lines in Fig. 13b.

At this point, four nodes 0, 1, 5 and 6 have odd number of dotted lines incident on each. If we swap the alarm codes between two links (0, 2) and (1, 6), the updated alarm code assignment is shown in Fig. 13c. This time, the structure consisting of the dotted lines in Fig. 13b is shaped into the broad-brush m-trail in Fig. 13c. Note that we can swap the alarm codes of links (0, 2) and (1, 6), because the LSBs of their binary alarm codes are the same. In RCS, if we want to swap the alarm codes of two link failures, all the previously analyzed bits of the two binary alarm codes must be the same, such that the swapping will not change the m-trails obtained so far. Otherwise, we have to add another bit as the new MSB to provide more candidate alarm codes for swapping.

An issue in RCS is that multiple disjoint m-trails may be generated when we analyze a particular bit in the binary alarm codes. For the example in Fig. 13, when we analyze the second bit, the alarm code 1 (0001) of link (0, 1) and 7 (0111) of link (4, 6) also have the same LSB of "1". As a result, we can also swap the alarm codes of links (0, 1) and (4, 6) as indicated by the dotted arrow in Fig. 13c, instead of swapping the alarm codes between links (0, 2) and (1, 6) as we have done earlier. If so, the number of nodes with odd degree of vectors will also be reduced from 4 (i.e., nodes 0, 1, 5 and 6 as indicated by the dotted lines in Fig. 13b) to 2 (i.e., nodes 4 and 5 as indicated by the dotted lines in Fig. 13c). Instead of having a single m-trail as shown by the dotted broad-brush line in Fig. 13c, a link-based m-trail at link (4, 5) and a disjoint m-cycle 0-1-2-0 will be generated, as shown by the (regular) dotted lines in Fig. 13c.

V. Bounds on the Number of Monitors

As formulated in (1), the monitoring cost can be defined as a weighted sum of the monitor cost (denoted by the number of monitors) and the cover length. On the other hand, the required number of monitors has attracted more research attentions [14, 17, 19] than the cover length. In addition to decreasing the hardware cost, reducing the required number of monitors can greatly simplify the fault management and thus make the network more scalable.

In simple m-cycle design, the required number of monitors increases linearly with the network size. For example, the spanning tree based algorithm HST [15] generates a simple m-cycle from each chord. Let $\boldsymbol{E}$ be the set of all the links and $\boldsymbol{V}$ be the set of all the nodes in the network. Because a network has

$\|V\|-1$ trunks and $\|E\|-\|V\|+1$ chords, an HST solution contains exactly $\|E\|-\|V\|+1$ m-cycles, plus additional link-based monitors if required for achieving 100% link failure localization. Although the number of required monitors is generally less than $\|E\|$ (which is required by the conventional link-based monitoring), it still increases linearly with the network size.

The work in [17] proposes the ILP-based approach and the concept of non-simple m-cycles. Besides introducing an efficient tradeoff between the monitor cost and the cover length (which cannot be achieved in HST and M^2-CYCLE), a more significant contribution of [17] is that the required number of monitors is dramatically reduced. In [17], the m-cycle design problem is translated into the binary coding of individual link failures, subject to the network topology and the cycle constraints. Since each m-cycle matches one bit in the binary alarm codes, the required number of m-cycles is dramatically reduced from $\boldsymbol{O}(\|E\|-\|V\|+1)$ to $\boldsymbol{O}(\log_2\|E\|)$ (note that only the number of m-cycles is considered, and some necessary link-based monitors for achieving 100% link failure localization is not counted here). This is due to the more flexible monitoring structure of non-simple m-cycles, which greatly weakens the cycle constraint compared with simple m-cycles, and thus gives more flexibility to the binary coding of the link failures.

However, the cycle constraint is still not removed in non-simple m-cycle design. The monitoring structure is still limited to either simple/non-simple m-cycles or link-based monitors, and other open trails are not considered. Besides, the monitoring resource allocation is not jointly optimized by considering both m-cycles and link-based monitors at the same time. All the above limitations prevent the number of required monitors from being reduced further. With the concept of m-trails proposed in [18], all those limitations are removed, and thus the minimum number of required monitors can be achieved for 100% link failure localization.

However, this does not mean that the number of monitors required by an m-trail solution is always in $\boldsymbol{O}(\log_2\|E\|)$. Note that in non-simple m-cycle design, $\boldsymbol{O}(\log_2\|E\|)$ only counts the number of m-cycles. If we count the total number of required monitors, some additional link-based monitors still need to be considered. Due to the diversity of the network topologies, in some extreme cases the required number of link-based monitors could be very huge for achieving 100% link failure localization.. For example, in a ring topology, an m-cycle design is not feasible at all, and every link needs a link-based monitor if the general m-trail structure is not adopted. If m-trail design is considered, it is proved in [19] that $\lceil \|E\|/2 \rceil$ monitors are always sufficient for 100% link failure localization in any two-connected network, and it is also the necessary number of monitors for a ring network. Obviously, this is much larger than $\boldsymbol{O}(\log_2\|E\|)$. On the other hand, solutions generated from the ILP indicate that exactly $\log_2\|E\|$ m-trails and monitors are sufficient for achieving 100% link failure localization in a fully-meshed topology, but it is very hard to find a theoretical proof for this observation. Instead, an upper bound of $6+\lceil \log_2(\|E\|+1) \rceil$ monitors is proved in [19] for achieving 100% link failure localization in a fully-meshed topology.

A general observation is that the required number of monitors decreases as the average node degree increases. Although increasing the average node degree leads to more links in the network and thus intuitively more m-trails and monitors may be required, it actually increases the network connectivity and thus relaxes the topology constraint on coding the link failures. Based on the heuristic solutions obtained in various randomly generated networks, it is shown in [19] that for most networks with moderate node degree, the required number of m-trails and monitors is only slightly above $\log_2\|E\|$. As a result, an m-trail based monitoring scheme can greatly simplify the fault management in a large-size network.

VI. Future Research Topics

Although our experiments indicate that a fully-meshed network needs exactly $\log_2\|E\|$ m-trails and monitors to achieve 100% link failure localization, finding a theoretical proof for this observation is still an open problem. Due to the diversity of the network topologies, more results other than those observed in [19] are expected on how the required number of monitors changes with the network topologies. Besides, it would be an interesting supplement to take the cover length into similar studies.

In this article, we showed that m-trail provides the most general optical layer monitoring structure by including all the previously studied structures. However, non-simple m-cycle design may be desired in some applications, especially when we hope that the same node can send out the supervisory optical signal and meanwhile check the content or the quality of the loopback optical signal. Compared with an m-cycle, an open m-trail separates the optical transmitter and receiver at two different nodes. This may introduce additional cost, but it can be easily incorporated into the optimization process by slightly modifying the ILP. Besides, in the monitoring schemes we studied, monitors can be equipped at an arbitrary set of nodes, and thus there is no additional constraint on the monitoring sites. In practice, we may hope that the set of monitors can be equipped at some given nodes, or at a centralized manager to facilitate the alarm code collection and distribution. We noticed that similar engineering implementations have been considered in [14]. Such requirements can also be formulated in our ILP by adding some extra constraints.

Note that all the schemes reviewed in this article consider only a single link failure in the network. Based on the monitoring structures and design approaches discussed in this article, a direct extension is to consider fault localization in optical networks under multiple failures or SRLG (Shared Risk Link Group) constraints. In fact, some researchers have initiated similar work. In particular, the work in [13] considers failure localization of SRLG with up to k links, where the monitor/monitors can be placed at a single or multiple locations. Some necessary and sufficient conditions on monitor placement and network connectivity, as well as the minimum number of required monitors, are derived for unique SRLG failure localization. A heuristic and an ILP model based on simple cycle enumeration are also formulated in [13]. But, the work in [13] only considers simple m-cycles and paths.. It would be interesting to see how non-simple m-cycles and m-trails can be applied in similar applications. Since fault localization is a common concern in other networks such as Internet and wireless networks, we also expect that the concepts and methodologies studied in this article can be extended to solve similar fault localization problems in those networks.

References

[1] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165-194, Nov. 2004.

[2] I. Katzela and M. Schwartz, "Schemes for fault identification in communication networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 753-764, Dec. 1995.

[3] A. T. Bouloutas, S. Calo and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Transactions on Communications*, vol. 42, no. 2-4, pp. 523-533, Apr. 1994.

[4] M. Goyal, K. K. Ramakrishnan, and W.-C. Feng, "Achieving faster failure detection in OSPF networks", in *Proc. IEEE ICC'03*, May 2003, vol. 1, pp. 296-300.

[5] M. AI-Kuwaiti, N. Kyriakopoulos and S. Hussein, "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability," *IEEE Communications Surveys & Tutorials,* vol. 11, no. 2, 2nd Quarter 2009. pp. 106 – 124.

[6] C. Mas, P. Thiran and J.-Y. Le Boudec, "Fault localization at the WDM layer," *Photonic Network Communications*, vol. 1, no. 3, pp. 235-255, Nov. 1999.

[7] C. Assi, Y. Ye, A. Shami, S. Dixit and M. Ali, "A hybrid distributed fault-management protocol for combating single-fiber failures in mesh-based DWDM optical networks," in *Proc. IEEE Globecom'02*, Nov. 2002, vol. 3, pp. 2676-2680.

[8] Y. G. Wen, V. W. S. Chan and L. Z. Zheng, "Efficient fault diagnosis algorithms for all-optical WDM networks with probabilistic link failures," *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 10, pp. 3358-3371, Oct. 2005.

[9] N. J. A. Harvey, M. Patrascu, Y. G. Wen, S. Yekhanin and V. W. S. Chan, "Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs," in *Proc. IEEE InfoCom'07*, May 2007, pp. 697-705.

[10] Y. G. Wen, V. W. S. Chan and L. Z. Zheng, "Efficient fault diagnosis for all-optical networks: an information theoretic approach," in *Proc. 2006 IEEE International Symposium on Information Theory*, Jul. 2006, pp. 2919-2923.

[11] S. Stanic, S. Subramaniam, H. Choi, G. Sahin and H.-A. Choi, "On monitoring transparent optical networks," in *Proc. Int'l Conf. on Parallel Processing Workshops*, Aug. 2003, pp. 217-223.

[12] M. W. Maeda, "Management and control of transparent optical networks," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1008-1023, Sept. 1998.

[13] S. Ahuja, S. Ramasubramanian and M. Krunz, "SRLG failure localization in all-optical networks using monitoring cycles and paths," in *Proc. IEEE InfoCom'08*, Apr. 2008, pp. 700-708.

[14] 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*, to appear.

[15] 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*, vol. 11, no. 3, pp. 277-286, May 2006.

[16] B. Wu and K. L. Yeung, "M^2-CYCLE: an optical layer algorithm for fast link failure detection in all-optical mesh networks," in *Proc. IEEE GLOBECOM '06*, Dec. 2006, pp. 1-5.

[17] 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*, vol. 27, no. 10, pp. 1392-1401, May 2009.

[18] 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*, vol. 27, no. 18, pp. 4175-4185, Sept. 2009.

[19] J. Tapolcai, B. Wu and P.-H. Ho, "On monitoring and failure localization in mesh all-optical networks," in *Proc. IEEE InfoCom'09*, Apr. 2009, pp. 1008-1016.

[20] M. S. Kiaei, C. Assi and B. Jaumard, "A survey on the *p*-cycle protection method," *IEEE Communications Surveys & Tutorials,* vol. 11, no. 3, 3rd Quarter 2009. pp.53-70.

[21] A. Haider and R. Harris, "Recovery techniques in next generation networks," *IEEE Communications Surveys & Tutorials,* vol. 9, no. 3 3rd Quarter 2007. pp. 2 – 17.

[22] B. Mukherjee, *Optical WDM Networks*. New York: Springer, 2006.

[23] P.-H. Ho and H. T. Mouftah, "Shared protection in mesh WDM networks," *IEEE Communications Magazine*, vol. 42, no. 1, pp. 70-76, Jan. 2004.

[24] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang and C.-N. Chuah, "Fast local rerouting for handling transient link failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 359-372, Apr. 2007.

[25] T.-H. Wu, "Emerging technologies for fiber network survivability," *IEEE Communications Magazine,* vol. 33, no. 2, Feb. 1995. pp. 58 - 59, 62-74.

[26] J. Li and K. L. Yeung, "A novel two-step approach to restorable dynamic QoS routing," *IEEE Journal of Lightwave Technology*, vol. 23, no. 11, pp. 3663-3670, Nov. 2005.

[27] B. Wu, K. L. Yeung and P.-H. Ho, "ILP formulations for non-simple *p*-cycle and *p*-trail design in WDM mesh networks," *Elsevier Computer Networks*, to appear.

[28] R. Diestel, *Graph Theory*, 2nd ed. New York: Spring-Verlag, 2000.

[29] B. Wu, K. L. Yeung and P.-H. Ho, "ILP formulations for *p*-cycle design without candidate cycle enumeration," *IEEE/ACM Transactions on Networking*, to appear. Available at http://www.eee.hku.hk/research/doc/tr/TR2008001_IFDCC.pdf.

Bin Wu received the B.Eng. degree from Zhe Jiang University, Hangzhou, China, in 1993, M.Eng. degree from University of Electronic Science and Technology of China, Chengdu, China, in 1996, and PhD degree from the University of Hong Kong, Hong Kong, in 2007. During 1997-2001, he served as the department manager of TI-Huawei DSP co-lab in Huawei Tech. Co. Ltd, Shenzhen, China. Currently he is a postdoctoral research fellow at the University of Waterloo, Waterloo, Canada.

Pin-Han Ho received his B.Sc. and M.Sc. degrees from the Electrical and Computer Engineering, Department of National Taiwan University in 1993 and 1995, respectively. He started his Ph.D. studies in 2000 at Queen's University, Kingston, Ontario, Canada, focusing on optical communications systems, survivable networking, and QoS routing problems. He finished his Ph.D. in 2002, and joined the Electrical and Computer Engineering Department at the University of Waterloo as an assistant professor in the same year. He is the author/co-author of more than 100 refereed technical papers and book chapters, and the co-author of a book on optical networking and survivability. He is the recipient of the Distinguished Research Excellence Award in the ECE Department at the University of Waterloo, the Early Researcher Award in 2005, the Best Paper Award at SPECTS '02 and the ICC '05 Optical Networking Symposium, and the Outstanding Paper Award in HPSR '02.

Kwan L. Yeung was born in 1969. He received his B.Eng. and Ph.D. degrees in Information Engineering from The Chinese University of Hong Kong in 1992 and 1995, respectively. He joined the Department of Electrical and Electronic Engineering, The University of Hong Kong in July 2000, where he is currently an Associate Professor, and the Information Engineering Program Co-Director. Before that, he has spent five years in the Department of Electronic Engineering, City University of Hong Kong as an Assistant Professor. During the summer of 1993, Dr. Yeung served with the Performance Analysis Department, AT&T Bell Laboratories (now Bell Labs, Lucent Technologies), Holmdel, USA, as a Member of Technical Staff. Dr. Yeung's research interests include next-generation Internet, packet switch/router design, all-optical networks and wireless data networks. He has obtained two patents and published over 120 papers in international journals and conferences since 1993.

János Tapolcai received his M.Sc. ('00 in Technical Informatics), and Ph.D. ('05 in Computer Science) degrees in Technical Informatics from Budapest University of Technology and Economics (BME), Budapest, Hungary. Currently he is an Associate Professor at the High-Speed Networks Laboratory at the Department of Telecommunications and Media Informatics at BME. His research interests include applied mathematics, combinatorial optimization, linear programming, linear algebra, routing in circuit switched survivable networks, availability analysis, grid networks, and distributed computing. He has been involved in a few related European and Canadian projects (IP NOBEL; NoE e-Photon/ONe; BUL). He is an author of over 30 scientific publications, and is the recipient of the Best Paper Award in ICC'06. He is a member of the IEEE.

Hussein T. Mouftah joined the School of Information Technology and Engineering (SITE) of the University of Ottawa in September 2002 as a Canada Research Chair Professor (Tier 1). He has been with the Department of Electrical and Computer Engineering at Queen's University since 1979, where prior to his departure in August 2002 he was a full professor and department associate head, after three years of industrial experience mainly at Bell Northern Research of Ottawa (now Nortel Networks). He has spent three sabbatical years at Nortel (1986–1987, 1993–1994, and 2000–2001) conducting research in the areas of broadband packet switching networks, mobile wireless networks, and quality of service over the optical Internet. He served as Editor-in-Chief of *IEEE Communications Magazine* (1995–1997), IEEE Communications Society Director of Magazines (1998–1999), Chair of the Awards Committee (2002-2003), Director of Education (2006-2007), and Member of the Board of Governors (1997-1999 and 2006-2007). He is also the founding Chair of two of IEEE Communications Society Technical Committees (TCs): Optical Networking TC (2002-2004) and Ad Hoc and Sensor Networks TC (2005-2007). He has been a Distinguished Speaker of the IEEE Communications Society (2000-2007). He is author or co-author of 6 books and more than 850 technical papers and 10 patents. He was the recipient of the 1989 Engineering Medal for Research and Development of the Association of Professional Engineers of Ontario (PEO). He has also received 8 Outstanding/Best Paper Awards, the IEEE Canada Outstanding Service Award (1995), and the CSIM Distinguished Service Award of the IEEE Communications Society (2006). In 2004 Dr. Mouftah received the IEEE Communications Society Edwin Howard Armstrong Achievement Award and the George S. Glinski Award for Excellence in Research from the Faculty of Engineering, University of Ottawa. In 2006 he was honoured with the IEEE McNaughton Gold Medal and the Engineering Institute of Canada Julian Smith Medal. In 2007 he was the recipient of the Royal Society of Canada (RSC) Thomas W. Eadie Medal. Most recently, Dr. Mouftah received the University of Ottawa 2007-2008 Award for Excellence in Research and the ORION Leadership Award of Merit (2008). Dr. Mouftah is a Fellow of the IEEE (1990), Fellow of the Canadian Academy of Engineering (2003), Fellow of the Engineering Institute of Canada (2005) and Fellow RSC: The Academies of Canada (2008).