This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

A New Heuristic for Monitoring Trail Allocation in All-Optical WDM Networks

Yangming Zhao , Shizhong Xu, Xiong Wang and Sheng Wang
School of Communication and Information Engineering
University of Electronic Science and Technology of China, Chengdu, P. R. China, 610054
E-mail: {zhaoyangming, xsz, wangxiong, wsh_keylab}@uestc.edu.cn

***Abstract*—We study the m-trail (monitoring trail) allocation problem in all-optical WDM mesh networks for achieving fast and unambiguous link failure localization. The existing ILP is not feasible for solving the problem in large-size networks. A heuristic RCA+RCS can find feasible solutions in a shorter running time, but it is a randomized algorithm. More importantly, RCA+RCS suffers from the disjoint trail problem which dramatically increases the number of required monitors in large-size networks. In this paper, we propose a new heuristic MTA (Monitoring Trail Allocation) to solve the problem. MTA avoids those issues in RCA+RCS, and achieves an efficient tradeoff between monitor cost and bandwidth cost. Compared with RCA+RCS, MTA greatly shortens the running time and achieves a much higher solution quality. We also show that MTA provides a flexible framework to enable multiple possible variations for future study.**

***Index Terms*—Failure localization, monitoring trail (m-trail), optical networks, Wavelength Division Multiplexing (WDM).**

I. INTRODUCTION

All-optical WDM networks have played a vital role in supporting various Internet applications with high QoS [1]. In WDM networks, hundreds of wavelengths can be multiplexed onto a single fiber for parallel data transmission. This improves data transmission efficiency and cuts down the network cost as well. But, network survivability becomes a great concern, as even a very short service downtime can lead to huge data loss. To minimize data loss, it is important to have a fast monitoring scheme to localize the failed link, such that it can be bypassed and the disrupted traffic can be recovered in a timely manner.

In upper-layer monitoring schemes [2], fault localization time is generally long due to the complex signaling. A scheme at the optical layer [3-8] can greatly simplify the required signaling to render a much shorter localization time. In this paper, we focus on an optical layer monitoring scheme.

Recently, several optical layer monitoring schemes using monitoring cycles (or m-cycles) [3-4] and trails (or m-trails) [5] were proposed. A common feature of those schemes is to pre-cross-connect some dedicated supervisory wavelengths into a set of m-cycles or m-trails. Each m-cycle or m-trail is a supervisory lightpath, and an m-trail differs from an m-cycle by taking an arbitrary (open or closed) path, instead of only a closed path as in an m-cycle. Since m-cycle is a special case of m-trail [5], it is also called m-trail hereafter. Any link failure on an m-trail will disrupt the optical supervisory signal transmitted in the m-trail, and thus trigger an alarm in a dedicated monitor equipped at the sink of the m-trail. In this way, one monitor can be shared to monitor the health of all links on the m-trail.

To accurately localize all possible single link failures, each link must be traversed by a distinct set of m-trails. Then, a link failure will trigger a unique alarm code to unambiguously identify a specific link failure, where the on-off status of each alarm is indicated by a binary bit. In essence, this approach codes each link using a binary system, subject to the monitoring structure and the network topology constraints [4-6]. In a well-connected network, the number of required monitors is close to $\log_2|\boldsymbol{E}|$ where $\boldsymbol{E}$ is the set of all the links [5-7]. This greatly reduces the number of monitors compared with the link-based monitoring schemes. Then, fault management can be greatly simplified by managing only a small set of monitors.

The ILP (Integer Linear Program) in [5] needs a long running time to find an optimal m-trail solution. To shorten the running time, a two-step heuristic RCA (Random Code Assignment) + RCS (Random Code Swapping) is proposed in [7]. In RCA, each link is randomly assigned a unique code. RCA aims at ensuring the code uniqueness in advance. Then, RCS analyzes each bit in the codes to generate one or several m-trails. Due to the randomized nature of this algorithm, its performance highly depends on RCA. Besides, RCS may generate multiple disjoint m-trails in analyzing each bit (i.e., the *disjoint trail problem*), which increases the number of m-trails and monitors.

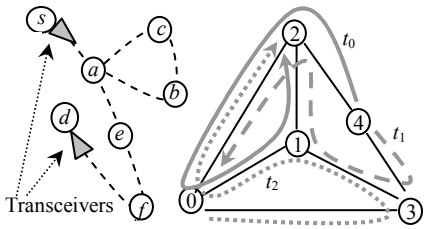
In this paper, we propose a heuristic MTA (Monitoring Trail Allocation) which removes the randomness and the disjoint trail problem in RCA+RCS [7]. Instead of ensuring code uniqueness first as in RCA+RCS, our idea is to ensure a valid m-trail structure first, and then sequentially add necessary m-trails to achieve unambiguous link failure localization. Theoretical analysis and simulation results show that MTA can generate a better solution in a much shorter time than RCA+RCS.

The rest of the paper is organized as follows. Section II briefly reviews the m-trail concept. Section III discusses the existing RCA+RCS algorithm [7]. MTA is proposed in Section IV. Section V presents simulation results to compare MTA with RCA+RCS. We conclude the paper in section VI.

II. M-TRAIL CONCEPT

A monitoring trail (m-trail) is a supervisory lightpath which can traverse a directed link once and a node multiple times (see Fig. 1a). If a link on an m-trail fails, the monitor at the sink of the m-trail will alarm due to loss of the supervisory signal. Fig.1b shows an example with three m-trails $\{t_0, t_1, t_2\}$. If link (0, 1) fails, monitors on t_0 and t_2 will alarm to generate an alarm code $[a_2, a_1, a_0]=[1, 0, 1]$, where $a_j=1$ means that the monitor on t_j

This work was partially supported by the 973 Program (2007CB307104), NSFC Funds (60972030, 60872032), Fundamental Research Funds for the Central Universities and Ph.D. Programs Foundation of Ministry of Education of China.



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. 1. Fast link failure localization using m-trails.

alarms and a_j=0 otherwise. With the predefined alarm code table (ACT) in Fig.1c, the failed link (0, 1) can be localized. Other link failures can be uniquely identified in the same way.

The objective is to minimize the monitoring cost in (1) [5]:

Monitoring Cost = monitor cost + bandwidth cost

$$= \gamma \times \text{number of monitors} + \text{cover length} \quad (1)$$

where the cover length is the total number of wavelength-links traversed by all m-trails, and the *cost ratio* γ weights the cost of a monitor over that of a supervisory wavelength-link.

III. RCA+RCS ALGORITHM

RCA+RCS [7] is the only heuristic proposed so far for m-trail design. In its first step RCA, each link is assigned a unique random code. The key spirit is to ensure unique identification of each link first. Basically, all links with alarm code bit a_j=1 should be traversed by m-trail t_j (see Fig. 1c). Due to the randomized nature of RCA, it cannot ensure a proper m-trail structure (i.e., a single connected supervisory lightpath). Then, RCS (Random Code Swapping) is used to shape each t_j (or each alarm code bit a_j for simplicity) into a valid m-trail.

The above idea is illustrated in Fig.2. In Fig.2a, a set of decimal codes 1-8 is randomly assigned to the links by RCA, with binary translations in the brackets. Then, RCS sequentially shapes each bit a_0-a_3. Let's take a_0 as an example, where those links with a_0=1 are denoted by the dashed lines in Fig.2a. Define a *code pair* [7] as a pair of binary codes with only one bit different. If we swap the code pair of links (0, 5) and (1, 2), a valid m-trail can be obtained in Fig.2b. However, RCS may swap another code pair of links (0, 1) and (2, 4). Then, two separate m-trails will be generated as shown in Fig. 2c (i.e., the disjoint trail problem as defined in Section I).

Another issue in RCA+RCS is that the solution quality depends on the randomized RCA. More times we try to re-execute the algorithm, better result we may get with high possibility. Sometimes, simultaneously swapping multiple code pairs can improve the solution, but is not allowed in RCS.

IV. MTA ALGORITHM

A heuristic for m-trail design may follow two possible approaches. The first one is to ensure code uniqueness of each link first and then shape the m-trail structures, as in RCA+RCS. The second one is to ensure valid m-trail structures first, and then sequentially add m-trails to achieve unambiguous link failure localization. MTA adopts the second approach.

A. Key Definitions

Temporary Code: MTA sequentially adds m-trails to the solution. Before the final solution is obtained, the codes assigned to the links are called temporary codes. They are determined by the m-trails that have already been allocated so far. Since unambiguous link failure localization is not yet achieved due to the insufficient number of m-trails allocated, multiple links may have the same temporary code.

Ambiguity Set ($\boldsymbol{AS_c}$): An ambiguity set $\boldsymbol{AS_c}$ contains at least two links with the same temporary code c. If a link l is in an $\boldsymbol{AS_c}$ (i.e., $l \in \boldsymbol{AS_c}$), we call $\boldsymbol{AS_c}$ the *home ambiguity set* of l. In Fig.3, we have two ambiguity sets $\boldsymbol{AS_1}$={(0,1), (1,2)} and $\boldsymbol{AS_2}$={(0,3), (2,3)}, and the home ambiguity set of link (0, 1) is $\boldsymbol{AS_1}$.

Unambiguous Link (UAL): Whenever a link can be uniquely identified from others by its current temporary code c (c>0), it becomes an unambiguous link (UAL). In MTA, once a link becomes an UAL, it remains as an UAL no matter how many other m-tails will be added to the solution in the future.

Fragment: MTA may need to connect multiple links or short paths to produce a new m-trail. Each of them is defined as a fragment. Let $\boldsymbol{F_j}=\{\boldsymbol{f_k^j}\}$ be a set of fragments used to generate an m-trail t_j where $\boldsymbol{f_k^j}$ is the k^{th} fragment. If a link l is a part of $\boldsymbol{f_k^j}$, we denote the relationship by $l \in \boldsymbol{f_k^j}$, $l \in \boldsymbol{F_j}$ and $\boldsymbol{f_k^j} \subseteq \boldsymbol{F_j}$.

Residue Topology: Let $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ be the network topology where $\boldsymbol{V}$ is the set of all the nodes and $\boldsymbol{E}$ is the set of all the links. For a given set of links $\boldsymbol{L} \subseteq \boldsymbol{E}$, a residue topology $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})-\boldsymbol{L}$ is obtained by removing $\boldsymbol{L}$ from $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$.

B. MTA Mechanism

Fig. 4 shows the main flowchart and Fig.5 gives the details of MTA. In Step 1, we put all the links in a special ambiguity set $\boldsymbol{AS_0}$, which is a set of links that have not yet been traversed by any allocated m-trails. When MTA ends up in Step 4 (see Fig. 5), $\boldsymbol{AS_0}$ must be a null set $\boldsymbol{\Phi}$. This is because each link must be traversed by at least one m-trail in the final solution. Otherwise, a failure at this link cannot be detected by any m-trail [5].

In Fig. 5, Step 2 adds a new m-trail t_j to the solution based on a set of fragments $\boldsymbol{F_j}=\{\boldsymbol{f_k^j}\}$. $\boldsymbol{F_j}$ is found in Step 2.1 and it generally consists of some link-disjoint fragments. In Step 2.2, the end nodes of those link-disjoint fragments are connected using some shortest paths in the residue topology $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})-\boldsymbol{F_j}$ to

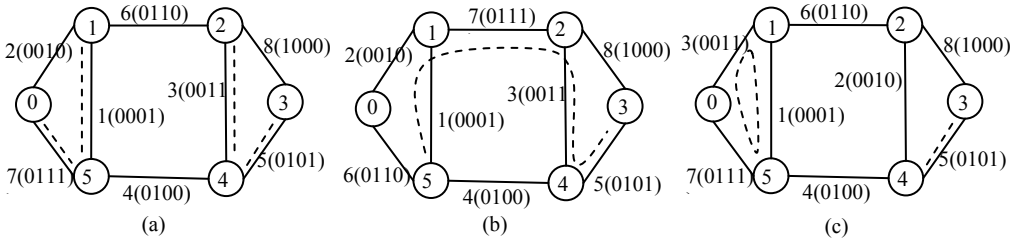


Fig.2. Mechanism of RCA+RCS and the disjoint trail problem.

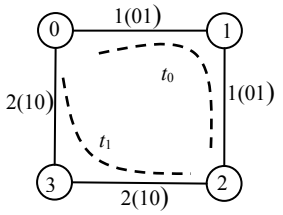


Fig.3 Example of ambiguity sets

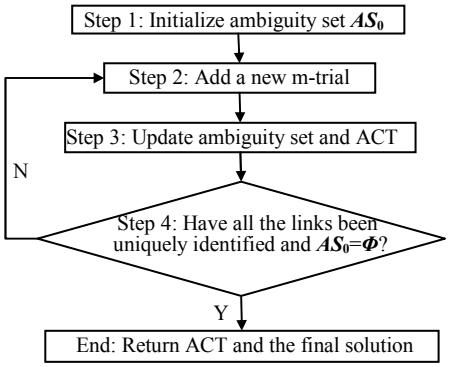


Fig.4. The main flowchart of MTA.

produce a valid m-trail structure. Since several link-disjoint fragments may traverse a common node, the m-trail generated may also pass through a node multiple times.

In particular, Step 2.1.1 initializes F_j to a null set Φ, and Step 2.1.2 finds an extending node r_k from which a particular fragment f_k^j stems. Step 2.1.3 weights the neighbouring links of r_k, and fragment f_k^j is extended in Step 2.1.4 according to the weights. Note that the maximum nodal degree is always pursued in choosing r_k and weighting the neighbouring links, as well as extending f_k^j. This gives more possible routes in extending f_k^j, and also increases the possibility that more links can share the same m-trail t_j which reduces the number of m-trails. Also note that the weight w_l is adjusted in Step 2.1.3. Specifically, if f_k^j is extended using a link in AS_0, it is promoted by increasing w_l (i.e., $w_l \times C_1 \rightarrow w_l$); If f_k^j is extended using a link which is not the first one in an ambiguity set, the m-trail t_j cannot distinguish this link from other links on f_k^j which are in the same ambiguity set, and this is not preferred by decreasing w_l (i.e., $w_l / C_2 \rightarrow w_l$). In short, MTA prefers to let t_j pass through those links not yet traversed by any m-trail, and extend fragments and m-trails using links from different ambiguity sets. If a link has become a UAL, or its home ambiguity set has more than half links in F_j, this link is not used to extend f_k^j by setting its weight to zero.

Steps 2.1.3 and 2.1.4 form an inner loop to iteratively extend a fragment f_k^j. If no more links with a positive weight can be used to extend f_k^j, or the length of f_k^j has reached $|E|/2$, then the process of extending f_k^j completes, and MTA turns to the outer loop (Steps 2.1.2-2.1.5) to construct the next fragment in F_j. F_j construction completes when every ambiguity set has at least one link in F_j.

Since F_j may consist of several link-disjoint fragments, Step 2.2 in Fig. 5 is to connect them using some shortest paths to ensure a valid m-trail structure. Shortest paths are computed in the residue topology $G(V, E)-F_j$ using Floyd-Washall algorithm [9], and are sequentially checked in an ascending order of their lengths to see if it can properly connect two fragments $f_a, f_b \subseteq F_j$, subject to constraints a)-d) in Step 2.2. Constraint a) means that it is not cost-efficient to use a shortest path SP_{uv} longer than γ hops to save one monitor by connecting f_a and f_b (see (1)); Constraint b) says that SP_{uv} must be able to properly connect f_a and f_b into a longer fragment; Constraint c) specifies that the new connected fragment $SP_{uv} \cup f_a \cup f_b$ must not fully cover all links in an ambiguity set (otherwise it cannot divide those links into two distinguishable subsets); Constraint d) ensures that any directed link is not traversed by an m-trail more than once. Since Step 2.2 may not be able to connect all fragments into a single m-trail t_j, we set t_j as the longest fragment in the updated F_j.

After a new m-tail t_j is added in Step 2, we can update the ambiguity sets and ACT in Step 3. If all the links have become UAL and $AS_0=\Phi$, stop the algorithm and return ACT; Otherwise

MTA Algorithm

Input: a network topology $G(V, E)$ and a cost ratio γ.

Output: a set of m-trails for unambiguous single link failure localization.

Step 1: Initialize ambiguity set AS_0:

Define AS_0 as a set of links that have never been traversed by any m-trail. Set $E \rightarrow AS_0$ (i.e., initialize AS_0 to the set of all the links E). Mark all the links in E as non-UAL. Let j be the index of the m-trails to be added. Set $0 \rightarrow j$.

Step 2: Add a new m-trail t_j:

2.1) Find a fragment set F_j for t_j:

2.1.1) Initialize F_j :

Define $F_j = \{f_k^j\}$ as a set of fragments f_k^j that should be chosen from $G(V, E)$ to produce t_j. Set $0 \rightarrow k$ and initialize F_j to a null set Φ by setting $\Phi \rightarrow f_k^j$ and $\Phi \rightarrow F_j$.

2.1.2) Find an extending node r_k for f_k^j:

For each AS_c generated so far (including AS_0), if $AS_c \cap F_j = \Phi$, construct a subgraph of $G(V, E)$ from AS_c. In all the subgraphs obtained, find an extending node r_k of f_k^j as the node with the maximum nodal degree (ties are broken by randomly choosing one from multiple peers).

2.1.3) Weight neighbouring links of r_k :

In residue topology $G(V, E)-F_j$, weight each neighbouring link l of r_k using the nodal degree w_l of the other end node of l. Let C_1 and C_2 be two predefined large constants, and AS be the home ambiguity set of l (i.e., $l \in AS$). If $AS=AS_0$, set $w_l \times C_1 \rightarrow w_l$. If there is another link $l' \in AS \cap F_j$, set $w_l / C_2 \rightarrow w_l$. If l is an UAL, or $|AS \cap F_j| \geq |AS|/2$, set $0 \rightarrow w_l$.

2.1.4) Extend fragment f_k^j:

In residue topology $G(V, E)-F_j$, let l_{max} be the neighbouring link of r_k with the maximum positive weight w_l as calculated in Step 2.1.3 (ties are broken by randomly choosing one from multiple peers), and r'_k be the other end node of l_{max}. Set $r'_k \rightarrow r_k$ and $l_{max} \cup f_k^j \rightarrow f_k^j$ (thus l_{max} also becomes a link $l_{max} \in F_j$). Repeat Steps 2.1.3 and 2.1.4 until no more links with positive weight can be further added to f_k^j, or $|f_k^j| = |E|/2$.

2.1.5) Loop control to construct another fragment in F_j:

If $AS_c \cap F_j \neq \Phi$ for all AS_c, go to Step 2.2; Otherwise set $k+1 \rightarrow k$ and go to Step 2.1.2 to construct the next fragment.

2.2) Connect link-disjoint fragments in F_j to produce t_j:

Let SP_{uv} be the shortest path between nodes u and v in the residue topology $G(V, E)-F_j$. Use Floyd-Washall algorithm to compute SP_{uv} between all node pairs $u, v \in G(V, E)-F_j$, and sort those shortest paths in an ascending order of their lengths. In the same order, sequentially check whether each SP_{uv} can be used to properly connect two link-disjoint fragments (say f_a^j, f_b^j) in F_j. If

a) $|SP_{uv}| \leq \gamma$,

b) SP_{uv} can properly connect f_a^j and f_b^j into a new fragment,

c) $AS_c \not\subset (SP_{uv} \cup f_a^j \cup f_b^j)$ for all AS_c, and

d) $SP_{uv} \cap (f_a^j \cup f_b^j) = \Phi$,

set $SP_{uv} \cup f_a^j \cup f_b^j \rightarrow f_a^j$ and $F_j - f_b^j \rightarrow F_j$. After all SP_{uv} have been checked, set t_j as the longest fragment in F_j and go to step 3.

Step 3: Update ambiguity sets and ACT:

Re-compute the temporary code of each link based on the m-trails $\{t_0, t_1, \ldots, t_j\}$ allocated so far. Put any link with a temporary code c into AS_c and those links not traversed by any m-trail into AS_0. Mark each link with a distinct temporary code as an UAL. Update ACT accordingly.

Step 4: Loop control for unambiguous link failure localization:

If every link in E has become an UAL and $AS_0=\Phi$, stop and return ACT; Otherwise, set $j+1 \rightarrow j$ and go to Step 2 to add another m-trail.

Fig. 5 Details of MTA algorithm.

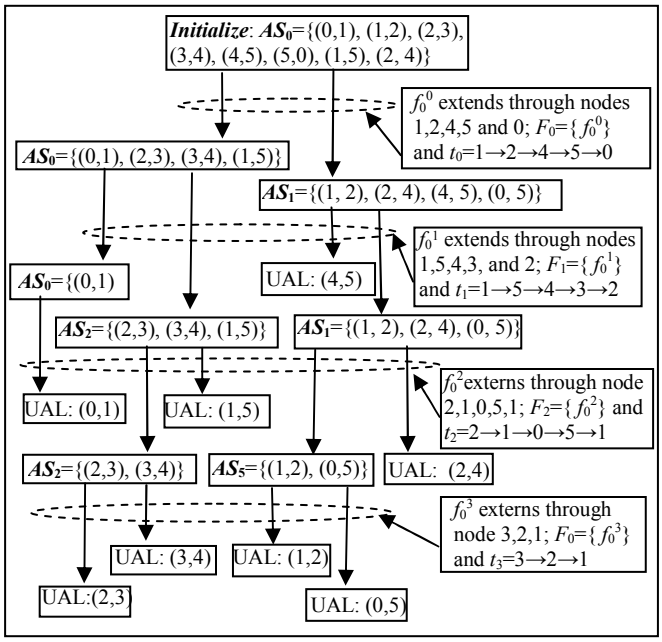


Fig.6. An example of MTA execution based on the network in Fig. 2 with γ=5 and C_1=C_2=10.

go to Step 2 to add another m-tail until unambiguous link failure localization is achieved, as stipulated in Step 4.

Fig. 6 shows an example of MTA based on the network in Fig. 2, where we set γ=5 and C_1=C_2=10. Compared with the optimal ILP result, which requires a monitoring cost of 32 (4 monitors and 12 cover length), MTA requires 34 (4 monitors and 14 cover length) which is within a gap of only 6.25% to optimality.

C. Theoretical Analysis and Discussions

Compared with conventional monitoring schemes which require at least $O(\boldsymbol{E})$ monitors, the m-trail scheme only requires $O(\log_2|\boldsymbol{E}|)$ m-trails/monitors (in well-connected networks) [5-7]. MTA has indeed pursued this logarithmic nature. MTA ensures that at least one link is chosen from each ambiguity set to construct a new m-trail t_j (see Step 2.1.5 in Fig. 5), and t_j must not fully cover all links in any ambiguity set (see constraint c) in Step 2.2). It means that each m-trail t_j intends to divide every ambiguity set into two distinguishable subsets (if all fragments $\boldsymbol{f_k^j}$ can be properly connected into t_j in Step 2.2). Besides, MTA intends to divide each ambiguity set into two subsets with almost equal size. This is because MTA will never use more than half links in any ambiguity set to extend a fragment (see Step 2.1.3), and it connects them using some other links. So, an m-trail intends to split each ambiguity set half to half. This directly follows the binary coding principle which makes the required number of m-trails/monitors close to $\log_2|\boldsymbol{E}|$.

In addition to potentially saving monitors as analyzed above, MTA also intends to save supervisory wavelengths. Examples include that MTA does not extend a fragment using any UAL (but the UAL can still be used to connect fragments in Step 2.2), and shortest paths are used to connect fragments, etc. Besides, the tradeoff between monitor cost and bandwidth cost is also taken into account, because MTA never uses a shortest path more than γ hops to connect two fragments for saving one monitor, as formulated in constraint a) in Step 2.2.

Unlike the randomized RCA+RCS [7], MTA runs in a deterministic manner. It does not need to be re-executed many times for a good solution, and this greatly shortens the running time. The time complexity of MTA is dominated by Step 2.2, where it needs at most $O(|\boldsymbol{V}|^3)$ time to compute all-pairs shortest paths, $O(|\boldsymbol{V}|^2\log|\boldsymbol{V}|)$ time to sort those shortest paths, and $O(|\boldsymbol{V}|^2|\boldsymbol{E}|^2)$ time to connect fragments. This results in a time complexity of $O(|\boldsymbol{V}|^3+|\boldsymbol{V}|^2|\boldsymbol{E}|^2)$ to generate an m-trail. Since a solution may consist of at most $O(|\boldsymbol{E}|)$ m-trails, in the worst case the time complexity of MTA is at most $O(|\boldsymbol{V}|^3|\boldsymbol{E}|+|\boldsymbol{V}|^2|\boldsymbol{E}|^3)$.

By ensuring valid m-trail structures and sequentially adding them to the solution, MTA removes the disjoint trail problem in RCA+RCS. In large-size networks, this saves a lot of m-trails, and makes the design more scalable in terms of minimizing the number of monitors and the fault management efforts.

MTA provides a flexible framework where one can easily try many variations in different steps. For example, we can easily modify MTA by randomly choosing half links from each ambiguity set and connecting them using shortest paths. Then, the algorithm will be randomized which can gradually improve the solution by many times of tries. Another example is that, if distance-related costs instead of hop-counts are used as the link cost metric, we can easily modify the weighting in Step 2.1.3 to take it into account, which is very hard to achieve in RCA+ RCS. We will studies similar issues in our future work.

V. Simulation Results

The simulation is carried out on a computer with Duo-Core 1.86 GHz intel CPU using C++ in visual studio 2005. To compare MTA with RCA+RCS [7], we set γ=5000 as in [7] to minimize the number of m-trails. In Figs. 7-9, each performance point is obtained by averaging the results over hundreds of randomly generated topologies. Due to the randomized nature of RCA+RCS, we report its best result over 100 tries for each topology whereas MTA is run only once.

A. Performance Comparison as Network Size Changes

Figs. 7-8 show how the number of m-trails changes in both algorithms as the network size increases. Fig. 8 also shows the theoretical lower bound $\lfloor \log_2|\boldsymbol{E}| \rfloor+1$ [5]. When the network size is relatively small, the performance of RCA+RCS and MTA is very close to the lower bound, and RCA+RCS may even outperform MTA. In this small-size network scenario, RCA+RCS can well explore the solution space without trying many times, but we may turn to the ILP for optimal solution. As the network size increases, the number of required m-trails in RCA+RCS grows much more rapidly than MTA. This is because RCA+RCS intrinsically suffers from the disjoint trail problem which makes the situation worsened fast in large-size networks, whereas MTA does not have this problem by ensuring a valid connected optical structure for each m-trail added to the solution. Of course, the performance of RCA+RCS may be slightly improved by re-executing the algorithm more (than 100) times, but that also means a much longer running time.

Since we have to re-execute RCA+RCS 100 times for each topology to get the performance shown in the figures, the running time of RCA+RCS is much longer than that of MTA. In a network with 60 nodes and an average nodal degree of 8, MTA only needs 3-4 seconds, whereas RCA+RCS needs more than 900 seconds to generate a much poorer solution. If the number

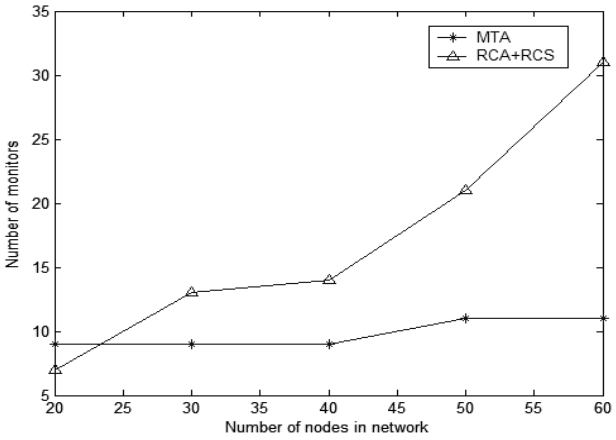


Fig.7. Number of monitors vs. number of nodes with an average nodal degree of 8.

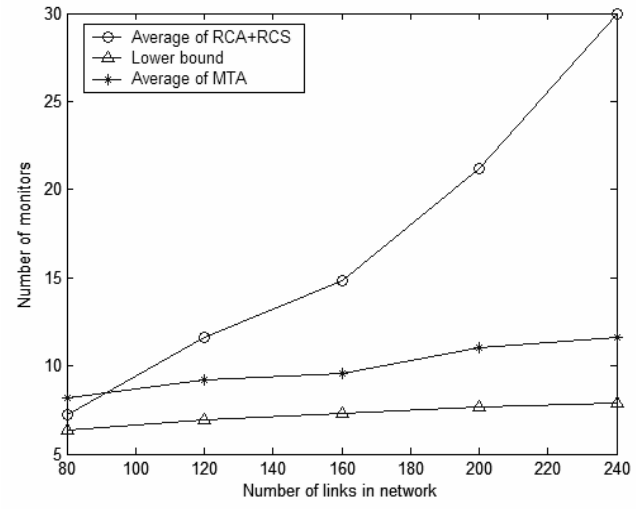


Fig.8. Number of monitors vs. number of links with an average nodal degree of 8.

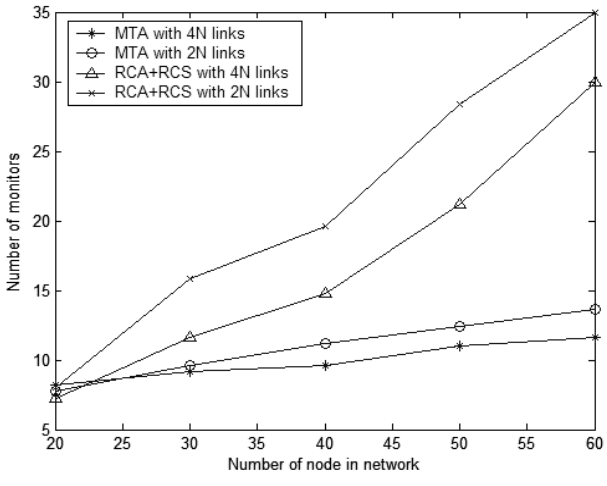


Fig.9. The impact of network

of nodes is reduced to 30, the running time of MTA will be less than 1 second but RCA+RCS still needs 150 seconds.

B. The Impact of Network Connectivity

The number of m-trails required may change from $\lceil |\boldsymbol{E}|/2 \rceil$ for a ring topology to $6+\lceil \log_2(|\boldsymbol{E}|+1) \rceil$ for a fully meshed topology [7]. Due to the diversity of the networks, it is impossible to fully explore how this number is exactly affected by various topologies with different network connectivity. But, we can use Fig.9 to show how MTA and RCA+RCS are affected, where two values 8 and 4 of average nodal degree are considered. As the average nodal degree decreases, more m-trails are required although there are less links in the network. This is because m-trails cannot be very freely routed in a sparsely connected network, which greatly limits the advantage of using m-trails. In Fig. 9, MTA always outperforms RCA+RCS in large-size networks. Besides, network connectivity has a slightly less impact on MTA than on RCA+RCS.

VI. CONCLUSION

We studied fast and unambiguous link failure localization in all-optical WDM mesh networks using the optical layer m-trail structures. An efficient heuristic MTA (Monitoring Trail Allocation) was proposed to find high-quality m-trail solutions in large-size networks with a very short running time. MTA minimizes the monitoring cost with operable tradeoff between monitor cost and bandwidth cost. Unlike the randomized RCA+RCS, MTA runs in a deterministic manner. MTA first ensures a valid structure of each m-trail, and then sequentially adds necessary m-trails to achieve unambiguous link failure localization. By avoiding the disjoint trail problem in RCA+RCS, MTA achieves a much smaller monitoring cost. Our numerical results demonstrated the superior performance of MTA over RCA+RCS in both solution quality and running time.

REFERENCES

[1] 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.

[2] 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," *IEEE GLOBECOM '02*, vol. 3, pp. 2676-2680, Nov. 2002.

[3] 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.

[4] 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.

[5] B. Wu, P.-H. Ho and K. L. Yeung, "Monitoring trail: on fast link failure localization in WDM mesh networks," *IEEE/OSA Journal of Lightwave Technology*, vol. 27, no. 18, pp. 4175-4185, Sept. 2009.

[6] B. Wu, P.-H. Ho, K. L. Yeung, J. Tapolcai, and H. T. Mouftah, "Optical layer monitoring schemes for fast link failure localization in all-optical networks," *IEEE Communications Surveys and Tutorials*, to appear.

[7] J. Tapolcai, Bin Wu, Pin-Han Ho, and L. Ronyai, "A novel approach for failure localization in all-optical mesh networks," *IEEE/ACM Transactions on Networking*, to appear.

[8] 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.*

[9] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice-Hall, 1993.