Monitoring Trail Allocation in All-Optical Networks with the Random Next Hop Policy

Yangming Zhao, Shizhong Xu, Bin Wu, 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, wubin_ncl, wangxiong, wsh_keylab}@uestc.edu.cn

***Abstract*—The concept of monitoring trail (m-trail) provides a striking mechanism for fast and unambiguous link failure localization in all-optical networks. To achieve fast m-trail design in large-size networks, two efficient heuristics RCA+RCS and MTA are proposed against the optimal ILP (Integer Linear Program) model. However, RCA+RCS suffers from the disjoint trail problem which increases the required number of m-trails, and MTA always finds a deterministic solution which may not be good enough due to the limited solution space. In this paper, we propose a new heuristic RNH-MTA (Monitoring Trail Allocation with the Random Next Hop policy) to solve those issues. Similar to MTA, RNH-MTA ensures a valid optical structure of each m-trail and sequentially adds necessary m-trails to the solution, and thus is free of the disjoint trail problem. By replacing the deterministic searching in MTA using the Random Next Hop policy, RNH-MTA sets up a probabilistic model in extending each m-trail. This not only enlarges the solution space and increases the solution diversity, but also enables a controllable tradeoff between the solution quality and the running time of the algorithm. Our numerical results show the advantages of RNH-MTA over both RCA+RCS and MTA.**

Index Terms*—All-optical networks (AONs), fast link failure localization, monitoring trail (m-trail), Wavelength Division Multiplexing (WDM).

I. INTRODUCTION

WDM (Wavelength Division Multiplexing) provides an efficient way to transmit multiple high-speed wavelengths in a single fiber. As WDM networks evolve towards all-optical networks (AONs) to remove the electronic bottleneck, various bandwidth-consuming and time-critical applications (such as video conference and multimedia streaming) can be well supported with high Quality of Service (QoS) [1]. Due to the vulnerability of optical networks, network survivability is very important in ensuring service continuity and fast restoration against a link failure [2-3]. In those failure-dependent survivability schemes, a critical step is to accurately localize the failed link in a timely manner, such that it can be immediately bypassed to achieve fast service restoration [4-6].

A scheme that monitors the health status of the network and localizes the failed component is called a *monitoring scheme* [7-9]. To achieve fast failure localization, monitoring schemes can be implemented at the optical layer to avoid complex signaling [10-11]. An optical layer monitoring scheme generally adopts a set of monitoring devices called *monitors* [12-14] to check the on-off status of some optical supervisory signals. A monitor will alarm if it detects an "off" status or Loss of Light (LoL) of an optical supervisory signal. If every node is equipped with a monitor, the network management system will have to manage a large number of monitors. Due to the transparency of AONs, alarm signals may propagate in the network without electronic boundary, and thus trigger alarms in all corresponding monitors. This may lead to an "alarm storm", which limits the scalability of the network and complicates the fault management system [7-8, 15].

The recently proposed m-trail (monitoring trail) concept [15] provides an ideal optical layer fast monitoring scheme to solve the above scalability issue. A *valid* m-trail is a connected optical supervisory lightpath which can be arbitrarily routed in the network, with a dedicated monitor equipped at the end node of the m-trail. The monitor will alarm if any link on the m-trail fails. Assume that we have allocated a single m-trail t_1 in the network. It divides all the links into two subsets: those on t_1 and others not. If there is a single link failure in the network, based on the status of t_1 (i.e., whether the monitor alarms or not), we can identify which subset the failed link is in. However, the link failures in the same subset cannot be distinguished by the allocated m-trail. Such a subset is defined as an *ambiguity set* (AS). Next, assume a second m-trail t_2 is added to the network. All the links are now divided into four ASs: some on t_1 or t_2 only, some on $t_1 \cap t_2$, and others not on any m-trail. Define the process of adding an m-trail as a *round*. As more m-trails are properly added, the number of ASs will be doubled in each subsequent round, and their sizes will decrease accordingly. Finally, if an AS contains only a single link, we define the link as an *unambiguous link* (UAL), which means the link can be accurately localized.

In fact, the above mechanism follows a binary coding principle, where adding an m-trail generally divides an AS into two ASs with smaller sizes to provide finer resolution on link failure localization. The status of each m-trail can be taken as a binary bit, and that of all m-trails forms a *binary alarm code*. All links in the same AS share a common alarm code, because the corresponding link failures will disrupt the same set of m-trails and trigger alarms in the same set of monitors. If a sufficient number of m-trails are added at the optical layer, the size of each AS can be decreased to one. Then, every link will become an UAL with a unique alarm code, which leads to *fast and unambiguous link failure localization*. Since binary coding is followed, in a well-connected network, the required number of m-trails and monitors can be as small as a logarithm of the number of all the links [8, 15-16]. This greatly simplifies the fault management and makes the network more scalable.

This work was partially supported by the 973 Program (2007CB307104), NSFC Fund (60972030), the Fundamental Research Funds for the Central Universities (ZYGX2010X001, ZYGX09J007) and Ph.D. Programs Foundation of Ministry of Education of China (20090185120013).

978-1-61284-0833-6/12/$26.00 ©2012 IEEE

Motivated by the importance of m-trails and the fact that ILP (Integer Linear Program) design [15] is not scalable and even intractable for large-size networks, this paper focuses on m-trail allocation in all-optical networks using a heuristic approach. Although two heuristics RCA+RCS (Random Code Assignment plus Random Code Swapping) [16] and MTA (Monitoring Trail Allocation) [17] have been proposed in the literatures, we show that the former suffers from a *disjoint trail problem*, and the latter is limited by its deterministic searching and the relatively small solution space (reviewed in Section II).

This paper contributes to the literatures by proposing a new heuristic RNH-MTA (Monitoring Trail Allocation with the Random Next Hop policy). Although RNH-MTA takes a similar process as MTA [17] to sequentially add enough m-trails with valid optical structures, it breaks the deterministic framework of MTA by setting up a probabilistic model to extend each m-trail. This essentially brings about two distinct features: 1) the probability-enabled solution space in RNH-MTA is much larger than the deterministic one in MTA to render a better solution; and 2) RNH-MTA enables a tradeoff between running time and performance, which cannot be achieved in MTA.

The rest of the paper is organized as follows. Section II briefly reviews RCA+RCS and MTA. Section III proposes RNH-MTA. Numerical results are presented in Section IV. We conclude the paper in Section V.

II. Related Works

A. RCA+RCS Algorithm

The key idea of RCA+RCS [16] is to assign each link a unique random code in its first step RCA, and then sequentially shape each alarm code bit to generate one (or multiple) valid m-trail by randomly swapping the binary codes in its second step RCS. Fig.1 gives an example. In Fig.1a, decimal codes 1-8 are randomly assigned to the links by RCA, with binary translations in the brackets. Then, RCS sequentially shapes each binary bit a_0-a_3. Let's take a_0 (i.e., the Lowest Significant Bit LSB) as an example. Those links with a_0=1 are denoted by the dashed lines in Fig.1a. They do not form a valid m-trail because there are two disjoint parts. Define a *code pair* [16] as a pair of binary codes with only one bit different. RCS allows code swapping within a code pair. If we swap the code pair of links (0, 5) and (1, 2), a valid m-trail can be obtained in Fig.1b. Due to its randomized nature, RCS may swap another code pair of links (0, 1) and (2, 4). Then, there are still two disjoint parts as shown in Fig. 1c, which may be taken as two separate m-trails in the final solution. This is called the *disjoint trail problem* and it increases the required number of m-trails.

B. MTA Algorithm

Unlike RCA+RCS which ensures code uniqueness of each link in RCA and shapes m-trails in RCS, MTA ensures a valid structure for each m-trail, and sequentially adds m-trails to the solution. Initially, the set of all the links in the network is treated as a single AS. As m-trails are sequentially added, more ASs with smaller sizes are generated in each round to achieve finer resolution of link failure localization. MTA converges when every AS contains a single UAL (unambiguous link).

In the middle stage of MTA, a target m-trail is to be generated in each round by connecting some fragments. A *fragment* is a connected part of the m-trail, and is obtained in two steps. First, a subgraph is induced from each AS in the current round if the AS does not have any link traversed by other fragments. Among all the subgraphs, the node with the maximum nodal degree is chosen as the *root*. Second, starting from the root, the fragment extends hop by hop and as possible as it can according to the *maximum weight policy* to traverse some links in different ASs. This process terminates until all the next hops are UALs, or half of the links in their home ASs (if an AS contains link l, then this AS is the *home AS* of l) are covered by the fragments generated in this round. Fragments are iteratively constructed by this process, until each AS has at least one link traversed by a fragment. Then, MTA properly finds some (shortest) paths to connect those link-disjoint fragments into a valid m-trail. If MTA cannot find a proper path to connect two fragments, then the shorter one will be discarded, and the longest connected fragment will be taken as the m-trail.

A key point in MTA is the maximum weight policy, which determines how a fragment chooses its next hop. Basically, a possible next-hop link is weighted by the nodal degree of its end node that is not on the current fragment. The weight may be adjusted according to some routing preferences in extending the m-trail. After the weights are properly calculated, MTA extends the fragment to the next hop with the maximum weight.

III. RNH-MTA Algorithm

A. Motivation and Key Idea

As we have reviewed in Section II.A, RCA+RCS suffers from the disjoint trail problem. Our previous work [17] showed that this may significantly increase the required number of m-trails and monitors compared with MTA. In this paper, we follow a similar approach as MTA [17] to sequentially add m-trails with valid optical structures.

On the other hand, the solution space in MTA is limited due to the deterministic m-trail extending process. MTA adds each

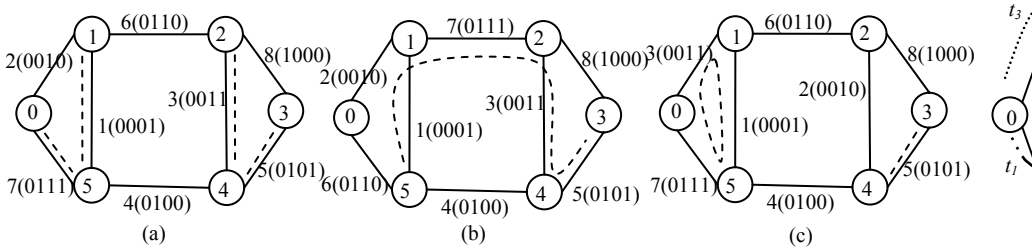


Fig.1. Mechanism of RCA+RCS and the disjoint trail problem.

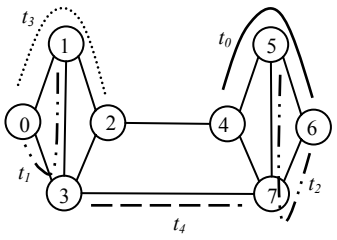


Fig.2 An example showing the limited solution space in MTA

m-trail by extending and connecting some fragments in a fixed manner, which may not always lead to the best routing of m-trails. Fig. 2 shows an example with five m-trails added in the previous rounds. In the current round, assume that MTA finds two fragments 1→3→2→1 and 5→7→4→5. According to MTA in [17] (details omitted here), we cannot find any feasible path to connect the end nodes 1 and 5 of the two fragments, and thus the m-trail constructed in the current round is either 1→3→2→1 or 5→7→4→5. Then, MTA needs at least two more m-trails to traverse all the links and meanwhile localize all the link failures. Due to the deterministic fragment extending process, if links (1, 2) and (4, 5) are found as the maximum weight next-hop (i.e., the 3rd hop) of the corresponding fragments, link (2, 4) will never be tried as the 3rd hop. In fact, other than the existing five m-trails, we only need two (instead of three or more) additional m-trails 1→3→2→4→5→7 and 1→2→4→7 to localize all link failures, if link (2, 4) (with a smaller weight) can also be tried in extending the fragments.

The above observation motivates us to design a new heuristic RNH-MTA. The key idea is to sequentially add m-trails with valid optical structures as in MTA, but set up a probabilistic model to choose the next hop of the m-trails, such that those links as (2, 4) in Fig. 2 can also be tried in extending m-trails with a certain probability. In this way, we can randomize RNH-MTA as RCA+RCS but avoid its disjoint trail problem. As a result, RNH-MTA can be re-executed for more times to explore a larger solution space (which is a merit of RCA+RCS but is not feasible in MTA due to its deterministic framework), at a cost of more computations than MTA. On the other hand, running RNH-MTA once takes the same time as MTA, which is greatly less than that of RCA+RCS [16]. The total running time depends on the number of times RNH-MTA is re-executed. Therefore, the running time is controllable and is still very small to satisfy practical engineering requirements.

B. The Main Process of RNH-MTA

The main process of RNH-MTA is shown in Fig. 3. It takes the *monitoring cost* [7-8, 15-17] in (1) as the metric to evaluate the quality of the solution, where the *cover length* is the total number of supervisory wavelength-links taken by all m-trails, and γ is a *cost ratio* for trading off between the number of monitors (same as the number of m-trails) and the cover length.

Monitoring Cost = monitor cost + bandwidth cost
$$= \gamma \times \text{number of monitors} + \text{cover length} \quad (1)$$

RNH-MTA includes a controllable total number of I main iterations indexed by k. Each main iteration produces a feasible m-trail solution for achieving fast and unambiguous link failure localization, and the final solution is chosen as the best one from all I solutions. Two parameters min_Cost and min_ACT are used to record the minimum monitoring cost and the alarm code table of the best solution found during the algorithm execution, where the *alarm code table* (ACT) lists the alarm code of each and all links. Those parameters are initialized in Step 1 of Fig. 3, where $+\infty$ denotes positive infinity and $\boldsymbol{\Phi}$ denotes a null set.

In the k^{th} main iteration, we first initialize $\boldsymbol{AS_0}$ as the set of all the links as shown in Step 2 of Fig. 3, where $\boldsymbol{AS_0}$ is a special

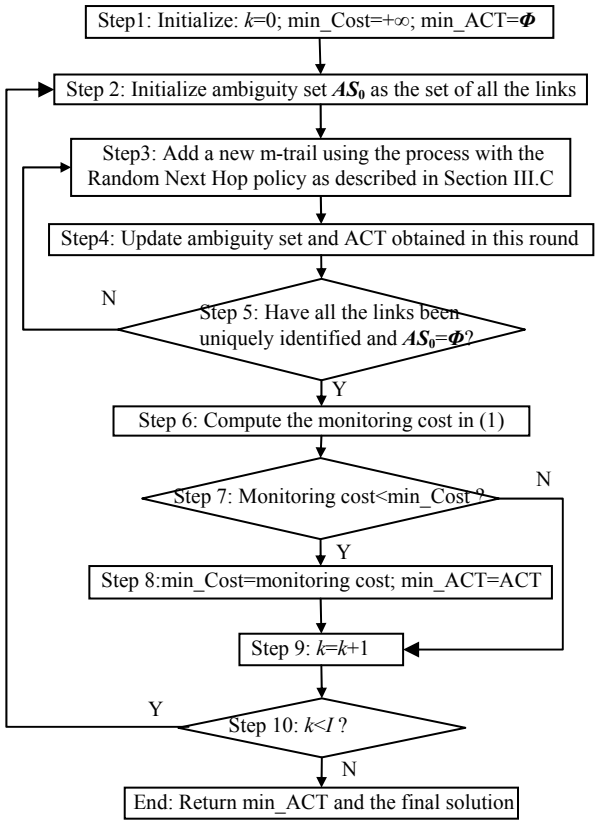


Fig.3. The main process of RNH-MTA.

ambiguity set and it always contains those links not traversed by any m-trail during the algorithm execution. Steps 3-5 constitute a round, where a new valid m-trail is added to the solution in Step 3, and the ASs and the ACT are updated in Step 4. As checked in Step 5, after a sufficient number of m-trails are added to traverse every link (i.e., $\boldsymbol{AS_0}=\boldsymbol{\Phi}$) and make it an UAL, a complete feasible solution is obtained. Then, Step 6 computes its monitoring cost, and Steps 7-8 check whether this solution is the best one to update min_Cost and min_ACT. If all I main iterations have been completed as checked in Steps 9-10, RNH-MTA terminates and returns the best solution found. Otherwise, it goes to Step 2 for the next main iteration.

C. Adding a New m-Trail Using the Random Next Hop Policy

From Fig. 3, we can see that the key step in RNH-MTA is to add a new m-trail in a particular round. Similar to MTA, an m-trail in RNH-MTA is obtained by connecting some fragments. The key difference is that RNH-MTA adopts a probability model to choose the next hop for extending the fragments.

We now focus on a particular round for adding a new m-trail, where the previous round has produced some ASs. Both fragments and the target m-trail in the current round are constructed based on those ASs. For simplicity, we say that a fragment traverses an AS if it traverses at least one link in the AS. To construct a fragment in the current round, we first identify those ASs that are not yet traversed by any other fragments constructed in this round. For example, in the construction of the first fragment, all ASs produced in the previous round are identified; in the construction of the second

fragment, only those ASs not yet traversed by the first fragment are identified, and so on. Each identified AS induces a subgraph consisting of those links in this AS. Among all the subgraphs, the node with the maximum nodal degree is chosen as the root of the target fragment, from which the target fragment starts to extend. At this point, the root is also called the extending node of the target fragment.

To understand how the target fragment extends hop by hop, we first define a *residue topology*. A residue topology is obtained by removing some links from the original network topology. The removed links are all the links that have been picked up so far in the current round for constructing the target m-trail, which include both the links on the previously constructed fragments and those before the extending node of the target fragment. If all fragments have been constructed and now RNH-MTA needs to find some paths to connect them into an m-trail, we also need to remove those connecting paths used so far.

To extend the target fragment to the next hop, each neighboring link l of the extending node in the residue topology is weighted by the nodal degree of the other end node of l. Meanwhile, the weight is adjusted based on the following rules to reflect some m-trail routing preferences: 1) if l is in $\boldsymbol{AS_0}$ (i.e., never traversed by any m-trail), it is preferred by multiplying the weight using a large predefined constant C_1; 2) if l is not the first link in its home AS that is traversed by any fragment in this round, it is not preferred by dividing the weight using a large predefined constant C_2; and 3) if l is an UAL, or a half of the links in its home AS have been traversed by all fragments in this round, then l is prohibited by setting its weight to 0. Let $\boldsymbol{L}$ be the set of all neighbouring links of the extending node in the residue topology. After the weight w_l for each link $l \in \boldsymbol{L}$ is calculated, the target fragment can be extended to l with a probability of

$$p_l = \frac{w_l}{\sum_{l \in \boldsymbol{L}} w_l}. \quad (2)$$

After the target fragment is extended by l, the extending node is updated by the other end node of l. This hop-by-hop extending process continues and the fragment extends as possible as it can, until all the next hops are UALs, or there are half of the links in their home ASs traversed by all fragments constructed so far. Then, if there are still some ASs that are not traversed by any fragment, RNH-MTA repeats to iteratively construct another fragment, until every AS is traversed by at least one fragment.

Finally, RNH-MTA adopts a similar mechanism as MTA to connect those fragments into a valid m-trail. This is achieved by finding a path in the residue topology to connect two link disjoint fragments into a longer one. Specifically, we can use Floyd-Washall algorithm [18] to compute all-pairs shortest paths in the residue topology, and sort them in an ascending order of their lengths. In the same order, sequentially check whether each path can be used to properly connect two fragments. A path is feasible if a) its length is not larger than the cost ratio γ (see (1)); b) it can properly connect the end nodes of two fragments to form a longer fragment; and c) after connected, the new fragment does not include all links of any AS, and does not traverse any directed link more than once. If only a subset of all fragments can be connected, RNH-MTA uses the longest connected fragment as the new m-trail.

D. Discussions

The most attractive feature of the m-trail mechanism is that it can significantly reduce the required number of m-trails and monitors to a logarithm of the total number of links in a well-connected network [8, 15-16]. In our previous work [17], we showed that MTA well pursues this logarithmic nature. RNH-MTA inherits this merit from MTA by always potentially dividing each AS into two with roughly equal size in the next round. This directly follows the binary coding principle which decreases the required number of m-trails/monitors to slightly above $\log_2|\boldsymbol{E}|$, where $\boldsymbol{E}$ is the set of all links in the network. Besides, RNH-MTA takes the randomized nature in RCA+RCS as a merit to enlarge the solution space over MTA, and meanwhile avoids the disjoint trail problem in RCA+RCS. If the weights of multiple possible next-hop links are close to each other, each link can be chosen as the next hop of the fragment with a certain probability, and the maximum weight link is not necessarily chosen. Combining with the fact that RNH-MTA enables multiple main iterations (see Section III.B and Fig. 3), this increases the solution diversity and quality over MTA, though at a cost of more computations. On the other hand, the probability of choosing a next-hop link is proportional to the weight of the link (see (2)), and thus m-trail routing preferences in MTA are still kept but in a probability sense.

It is easy to see that the time complexity of a main iteration in RNH-MTA is the same as that of MTA. Therefore, the overall time complexity of RNH-MTA (with I main iterations) is I times of that of MTA, i.e., $O(I|\boldsymbol{V}|^3|\boldsymbol{E}|+I|\boldsymbol{V}|^2|\boldsymbol{E}|^3)$, where $\boldsymbol{V}$ is the set of all nodes in the network. Unlike MTA, RNH-MTA enables a tradeoff between the running time and the solution quality by controlling the value of I. We can stop the algorithm if no solution improvement is observed over a certain number of iterations.

IV. Simulation Results

Simulations are carried out on a computer with Duo-Core 1.86 GHz intel CPU using C++ in visual studio 2005. Each performance point in Figs. 4-8 is obtained by averaging the results over 100 randomly generated topologies with γ=5. Due to the randomized nature of RCA+RCS and RNH-MTA, for each topology we take the best RCA+RCS result among 100 tries, and run RNH-MTA with 10 main iterations. Since the deterministic solution of MTA only depends on the network topology, MTA can only be run once.

A. Performance Comparison as Network Size Changes

Fig. 4 shows how the monitoring cost changes in the three algorithms as the network size increases. We can see that RNH-MTA always generates the minimum monitoring cost among the three algorithms. Fig. 5 focuses on how the required number of m-trails/monitors changes, where the theoretical lower bound $\lfloor \log_2 |\boldsymbol{E}| \rfloor + 1$ [8, 15-16] is also shown. From Figs. 4-5, it is obvious that both the number of required m-trails and

the monitoring cost in RCA+RCS grows much more rapidly than MTA and RNH-MTA as the network size increases. This is because RCA+RCS intrinsically suffers from the disjoint trail problem which makes the solution worsened fast in large-size networks, whereas MTA and RNH-MTA do not have this problem by ensuring a valid optical structure for each m-trail. No matter what the network size is, RNH-MTA always outperforms MTA in terms of both the monitoring cost and the required number of monitors. This is because RNH-MTA is run with 10 main iterations (i.e., I=10) to explore a larger solution space than MTA.

B. Impact of Network Connectivity

Figs .6a & 6b show how the monitoring cost and the number of m-trails/monitors change with the network connectivity in the three algorithms, respectively. Since it is impossible to explore such characteristics in various topologies with different connectivity, we only focus on two values 4 and 8 of the average nodal degree. For both values, we can see that RNH-MTA always achieves the smallest monitoring cost than the other two algorithms. As network connectivity decreases (whereas keep the same number of nodes in the network), we observe that the monitoring cost decreases as well (for γ=5), but the number of m-trails/monitors increases. This is because less links will be monitored which requires a smaller cover length, but the decrease of network connectivity limits the flexibility on freely routing m-trails, which in turn increases the required number of m-trails. We also observe that the change of network connectivity has a slightly less impact on MTA and RNH-MTA than that on RCA+RCS.

C. Running Time of the Algorithms

The performance comparisons reported in this section are obtained by running RCA+RCS 100 times, RNH-MTA with 10 main iterations (i.e., I=10), and MTA once for each topology. Since the complexity of a main iteration in RNH-MTA is the same as that of MTA, the overall running time of RNH-MTA is roughly 10 times of that of MTA. Fig. 7 compares the running time between RNH-MTA and MTA, whereas the running time of RCA+RCS is too large to be shown in the figure. However, we can use a particular network with 60 nodes and 240 links to show the point, where MTA needs 4 seconds and RNH-MTA needs 42, in contrast to 1000 seconds by RCA+RCS for 100 tries. Though RNH-MTA needs more time than MTA, its running time is controllable by changing the value of I, and also acceptable in practical engineering designs.

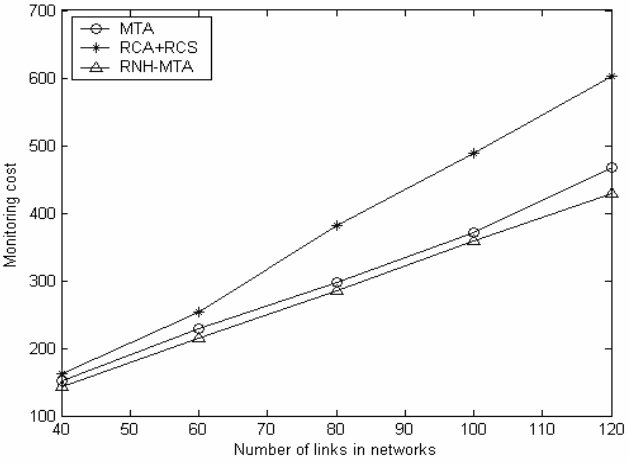


Fig. 4 Monitoring cost vs. number of nodes with an average nodal degree of 4.

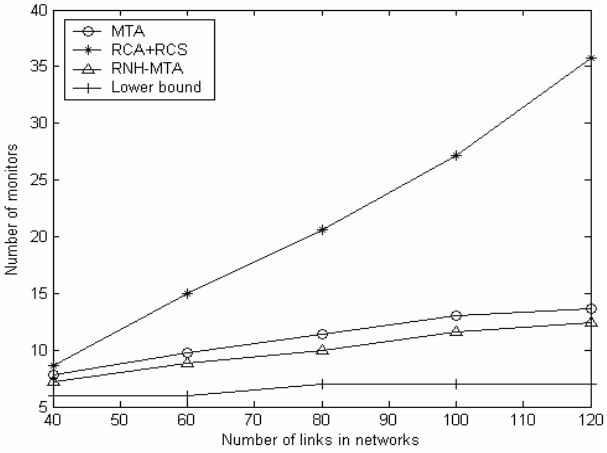
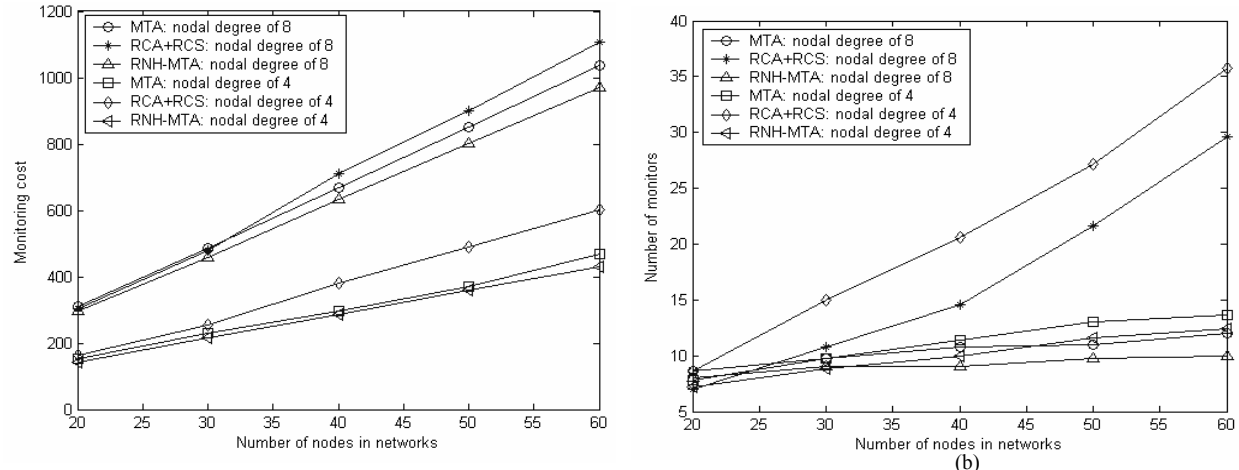


Fig. 5 Number of monitors vs. number of links with an average nodal degree of 4.



(a) (b)

Fig. 6. The impact of network connectivity

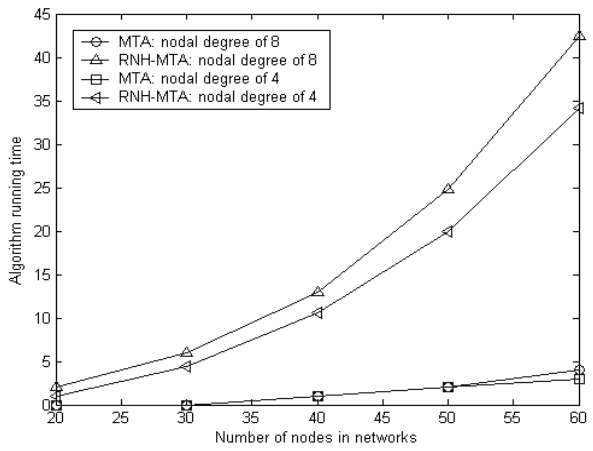


Fig. 7. Number of nodes vs. time spent by different algorithms

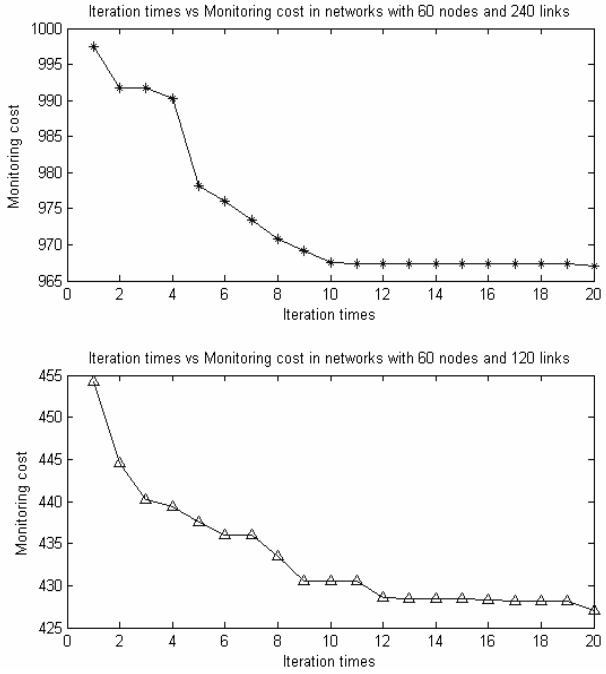


Fig. 8. Monitoring cost vs. iteration times for different connectivity.

D. Running Time / Performance Tradeoff in RNH-MTA

A larger number of main iterations (i.e., *I*) in RNH-MTA generally lead to a better solution. Fig. 8 shows such a running time/performance tradeoff using a particular example with 60 nodes and two values 4 and 8 of average nodal degree. On the other hand, we also observe that it is generally hard to significantly improve the solution further if the value of *I* is greater than 10.

V. CONCLUSION

Monitoring trail (m-trail) provides an efficient optical layer mechanism for achieving fast and unambiguous link failure localization. In this paper, we proposed a new heuristic RNH-MTA (Monitoring Trail Allocation with the Random Next Hop policy) to allocate m-trails in large-size all-optical WDM (Wavelength Division Multiplexing) mesh networks. Similar to MTA, RNH-MTA avoids the disjoint trail problem in RCA+RCS by sequentially adding valid m-trails to the solution. By replacing the deterministic solution searching in MTA using the random next hop policy, RNH-MTA sets up a probabilistic model to extend the m-trails, and takes the advantage of randomization as in RCA+RCS to enlarge the solution space over MTA. Numerical results showed that RNH-MTA outperforms both RCA+RCS and MTA, with a controllable tradeoff between the solution quality and the relatively short running time.

REFERENCES

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

[2] L. Guo, J. Cao, H. Yu and L. Li, "Path-based routing provisioning with mixed shared protection in WDM mesh networks," *IEEE/OSA J. Lightw. Tech.*, vol. 24, issue 3, pp. 1129-1141, Mar. 2006.

[3] L. Guo, X. Wang, J. Cao and L. Li, "Traffic recovery time constrained shared sub-path protection algorithm in survivable WDM networks," *Computer Networks*, vol. 52, no. 7, pp. 1492-1505, May 2008.

[4] B. Wu, K. L. Yeung and P.-H. Ho, "ILP formulations for *p*-cycle design without candidate cycle enumeration," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 284-295, Feb. 2010.

[5] B. Wu, P.-H. Ho, K. L. Yeung, J. Tapolcai and H. T. Mouftah, "CFP: cooperative fast protection," *IEEE/OSA J. of Lightwave Tech.*, vol. 28, no. 7, pp. 1102-1113, Apr. 2010.

[6] 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*, vol. 54, no. 5, pp. 716-725, Apr. 2010.

[7] 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.

[8] 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*,vol. 13, no.1, pp. 114-125.

[9] 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*, vol. 17, no. 4, pp. 1080-1093, Aug. 2009.

[10] 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.

[11] 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.

[12] Y. Hamazumi, M. Koga, K. Kawai, H. Ichino and K. Sato, "Optical path fault management in layered networks," *IEEE GLOBECOM '98*, vol. 4, pp. 2309-2314, Nov. 1998.

[13] C.-S. Li and R. Ramaswami, "Automatic fault detection, isolation, and recovery in transparent all-optical networks," *IEEE J. of Lightwave Tech.*, vol. 15, no. 10, pp. 1784-1793, Oct. 1997.

[14] S. Stanic, S. Subramaniam, H. Choi, G. Sahin and H.-A. Choi, "On monitoring transparent optical networks," *Int'l Conf. on Parallel Processing Workshops*, pp. 217-223, Aug. 2002.

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

[16] 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*, vol. 19, no. 1, pp. 275-285.

[17] Y. M. Zhao, S. Z. Xu, X. Wang and S. Wang, "A new heuristic for monitoring trail allocation in all-optical WDM networks," *IEEE GLOBECOM'10,*ONS, pp. 1-5.

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