In Proceedings of the 8th International Conference on Information Technology (CIT 2005), Bhubaneswar, India, 20-23 December 2005.

Shared Path Protection in DWDM Mesh Networks

Debasis Mandal, Bivas Mitra
Department of Computer Science & Engineering
Haldia Institute of Technology
P.O.: H.I.T., Midnapore (East), WB 721657, India
Email debasis.mandal@gmail.com, bivas_mitra@yahoo.com

ABSTRACT
Failure of even a single fiber link in DWDM mesh networks can cause severe data and revenue loss due to failure of several lightpaths of high bandwidth. This paper investigates this problem using shared path protection technique. First, we propose a link disjoint path finding heuristic, Multiple Active Path Search (MAPS) and show that it works even under *trap topologies* where conventional algorithms fail. Then using it as routing heuristic, we assign wavelengths to each connection request as they arrive, to design RWA algorithms. Apart from this static routing, we also design an Adaptive Routing (AS) heuristic, which operates in case no wavelengths are available for the disjoint path pair found by MAPS. The numerical results obtained for random network topologies with static random traffic demands show that both fixed and adaptive routing algorithms outperform conventional algorithm, giving lower blocking probability in terms of traffic and maximizing one-hop protected traffic.

Keywords
Physical topology, lightpath, wavelength division multiplexing (WDM), survivability, shared path-protection, routing and wavelength assignment (RWA).

1. INTRODUCTION
Survivability of a network refers to the network's capability to provide continuous service in the presence of failure. A network failure may be mainly due to link or node failure. Since most modern node devices have built-in redundancy that greatly improves their reliability, failure of fiber links is more of a concern [7] as they pass through different atmospheric conditions (like, under oceans) where manual recovery is more challenging. Again, since protection at electronic layer (ATM, IP) is more time-consuming, optical layer provides resource and time effective fault-tolerance even to upper unprotected layers [5]. So we restrict to survivability of link failure in optical layer in this paper.

In survivable WDM, all the lightpaths using the faulty fiber link are switched to the corresponding backup path. In *protection*, these backup resources are pre-computed and reserved for each connection until it fails. Protection can be of two types based on the process of backup: 1) In *link protection*, a link-disjoint backup lightpath (BP) is reserved for each individual link in working or active lightpath (AP). 2) *Path protection* refers to backup lightpath reservation for each active lightpath. Path protection usually has lower resource requirements and lower end-to-end propagation delay for the recovered route than link protection [7]. Again, if protection is *dedicated*, each node or link can be reserved as a backup resource for at most one connection. In *shared protection*, they can be reserved as a backup resource for multiple failure-disjoint connections. Dedicated protection requires more network resources but is simpler to implement, while shared protection is more resource efficient but requires increased switching time, complex signaling and network management [5][7].

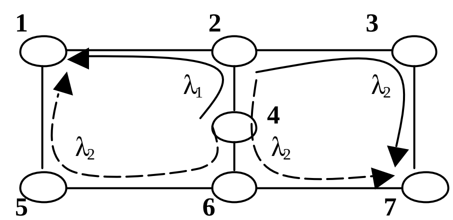

Figure 1. Shared Path Protection for single link-failure

In figure 1, lightpaths 4-2-1and 2-3-7 act as APs, the corresponding BPs being 4-6-5-1 and 2-4-6-7 respectively. As shown, the backup paths share the same wavelength λ_2 between them, as the two APs do not fail simultaneously, causing no interference between BPs.

1.1 Motivation and Organization of the Paper
In this paper, we address the Shared Path Protected Lightpath Problem (SPPLP) [2] against single-fiber failure (it's the predominant form of failure [5]) in mesh network. In [7], it is proven that irrespective of total cost of AP and BP, the problem is NP-complete. In section 2, we formulate the problem and specify constraints for shared path protection. Section 3 states the popular Active Path Search (APF) heuristic to find disjoint pair of paths and identifies its limitations. Then we present a new heuristic, Multiple Active Paths Search (MAPS) that guarantees to find disjoint paths pair between a pair of source and destination, if exist, and a set of lemmas supporting the heuristic. Section 4 presents three RWA heuristic algorithms based on different disjoint path heuristics: APF, MAPS and Adaptive Search. Section 5 evaluates the performance of our heuristics via simulation and provides numerical results. Lastly section 6 concludes this work with a discussion of its main contributions and future works.

1.2 Related Previous Work
For *single* wavelength networks, a feasible solution can be found using Suurballe's algorithm [4] in polynomial time

($O(n^2 \log n)$). The total cost of the resulting two link-disjoint lightpaths is minimal among all such path pairs. For networks with multiple wavelengths, we can apply this algorithm on every wavelength to find the same lightpath on the same wavelength, in polynomial time. However, if such paths do not exist, the problem becomes NP complete, which is to find two link-disjoint lightpaths on two different wavelengths. It has been studied in [6][7].

2. PROBLEM STATEMENT AND CONSTRAINTS

A WDM network is modeled by an undirected connected graph G(V,E) where V is the set of N nodes (combination of access station and OXC) and E is the set of L fiber optic links ($L \geq N-1$). Each fiber can carry W wavelengths $\Lambda=\{\lambda_1, \lambda_2, \ldots, \lambda_W\}$. The relative traffic demand is given by a $N \times N$ ($|V|=N$) matrix T where $(i,j)^{th}$ element x_{ij} represents the long term average traffic demand from node i to node j. The traffic is assumed to be static, i.e. the set of requests is known previously [7]. Though the average traffic demand may be asymmetric, we generate a symmetric traffic matrix where $t_{ij} = \max(x_{ij}, x_{ji})$. Also we assume diagonal elements to be zero considering no extra traffic sinks or generates at nodes.

In our RWA algorithm, the AP l_w and BP l_b satisfy the following *shared path protection constraints* with respect to the existing lightpaths [2].

For same request:

C.1 l_w and l_b are link disjoint.

C.2 l_w and l_b may have same or different wavelengths.

For different requests i and j:

C.3 l^i_b and l^j_b can share both links and wavelengths on the common link they traverse.

C.4 l^i_w and l^j_b can share only links but not any wavelength on the common link they traverse.

C.5 l^i_w and l^j_w do not share any link between them;

C.6 l^i_w and l^j_w can have same or different wavelengths.

Now we state our problem SPPLP as follows [2]: Given a WDM network G, traffic matrix T and a set of wavelengths Λ in each link. Route each incoming request following shared-path protection constraints to minimize blocking probability of traffic, thus maximizing one-hop protected traffic (total traffic handled by one optical hop without opto-electronic bottleneck throughout the network with different wavelengths). We want to maximize one-hop traffic to minimize the size of buffer at nodes and access of SONET layer needed for multi-hops.

3. HEURISTICS FOR FINDING LINK-DISJOINT PATHS

In this section, we assume any connection request that originates and terminates at the same node pair will follow the same route of either AP or BP. First we provide the conventional APF heuristic with its pitfalls.

3.1 Active Path First (APF)

Algorithm 1:

Step 1: **AP** is the shortest path between <s,d> in graph G.

Step 2: After removing the AP links, find the shortest path between <s,d> again. If found, this is **BP**. Otherwise, disjoint pair is not possible between <s,d>. Exit.

Limitations of APF Heuristic

Although APF is simple to implement, runs fast, the solution fails to find AP-BP pair even if there is some, in so-called *trap topologies* [7]. The reasons are two-folds: 1) It tries to find only shortest path between <s,d> as AP. But for this, there may not exist any BP. 2) Among many possible shortest paths, it randomly selects one as AP which has no BP. For example: In figure 2(a), say, <s,d> are <4,3>. If path 4-5-2-3 is selected randomly as AP, APF can't find any disjoint path to it (figure 2(b)), even if 4-1-2-3 and 4-5-6-3 exist as AP-BP pair.

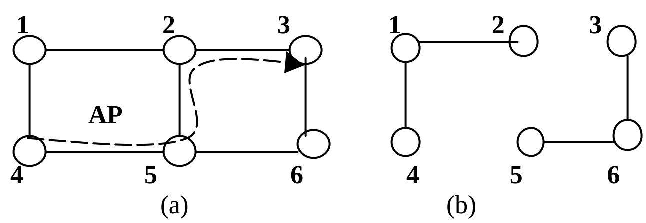

Figure 2. Trap topologies (a) before and (b) after applying APF

3.2 Multiple Active Paths Search (MAPS)

Now we propose an algorithm called MAPS, which runs successfully, even for *trap topologies*. Here we not only restrict ourselves to the shortest path as AP, but also search for other possible APs with increasing length for which BP may exist. Thus, our algorithm gives the minimum AP for which BP exists (corresponding BP is also minimum). The heuristic stops when any AP-BP pair is found or no such pair exists. In this algorithm, we use a variable ***cost*** of a path, mathematically defined as, *cost* = $a + b \times M$, where 'a' is the number of edges of the graph which are never visited to find AP between <s,d>; 'b' is the number of edges which are visited at least once to find AP between <s,d> and hence assigned with a *big weight* 'M' such that $i \times M \geq j \times M$, for positive integers $i \geq j$.

Algorithm 2:

Step 1: Variable *cost* is initialized to zero and each edge of G is assigned a unit weight.

Step 2: Find **AP** which is the path with minimum cost between <s,d> in graph G.

Step 3: After removing the AP links, find the path with minimum cost between <s,d> again. If it's found, this is **BP**. Hence exit. Otherwise, go to step 4.

Step 4: *Find new AP:* Restore AP links in the graph G and assign weight M to each such link. Then find the cost of the AP i.e. ***cost (AP)*** and compare it against ***cost***.

If (***cost*** ≥ ***cost (AP)***), then disjoint pair between <s,d> in G is not possible. Hence Exit. Otherwise, assign ***cost (AP)*** to ***cost*** and go to step 2 to search for the new AP for which disjoint path can be found.

Illustrative Example

In previous case (figure 2), MAPS assigns M to the AP links (figure 3(a)) and then ***cost (AP)*** is computed, which is 3M. Since ***cost*** (0, initially) is less than 3M, another AP is found by step 2, which is 4-1-2-3, say (figure 3(a)). Now, deleting these links gives 4-5-6-3 as BP (figure 3(b)).

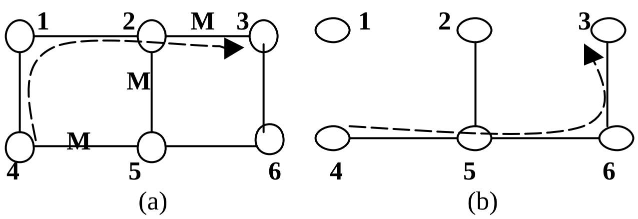

Figure 3. Application of MAPS upon Trap Topology

3.2.1 Correctness of MAPS

Now, we propose a set of lemmas, which support the correctness of the above heuristic.

Lemma 1:

Assume that a graph G(V,E) (|V|=N) contains a link disjoint path pair. Now MAPS can find that link disjoint path pair if M ≥ **N**, where M is defined as above.

Proof: We consider the following general graph G(V,E) having N nodes. Assume that the link disjoint paths between node pair <s,d> are P1=<s, n_1,….n_i,…n_{N-2}, d>and P2=<s, n'_1,….n'_i,…n'_{N-2}, d>. Suppose step 2 of MAPS finds an Active Path PT between <s,d> (Here we are assuming that PT contains a single link <s,d>. Later we generalize the concept for multiple edges in PT). After removing edges from G, let <s,d> node pair becomes disconnected in step 3. Now we want to prove that in step 4, our algorithm will be able to find a new path P1 that will be our new AP.

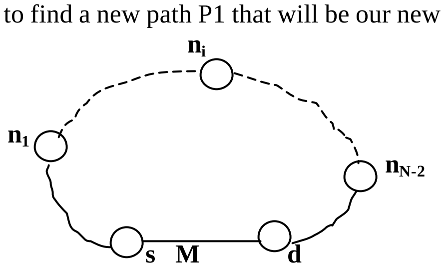

Figure 4. A general graph having N nodes

In step 4, the link PT=<s,d> is assigned with weight M (Figure 4). So the cost of the path PT=M. Another candidate path between <s,d> is P1=<s, n_1,…n_i,…n_{N-2}, d> which has at most N nodes (For any graph, N is the maximum possible nodes visited by any path). So path P1 has maximum (N-1) edges, each of unit cost. Thus, cost of P1 becomes at most

$cost_{max}$ (P1) = N-1, ………….(1) (Assuming P1 and PT link disjoint)

i.e. in general, cost(P1) ≤ N-1. ………….(2)

Also, cost (PT) = M. ………….(3)

To prove the necessary condition, i.e. M ≥ N, we assume, the shortest path algorithm selects P1, implying, cost (PT) > $cost_{max}$ (P1). Therefore from (1) & (3),

M > N-1

i.e. M ≥ N.

If PT contains multiple edges, then also it is obvious that cost (PT) > $cost_{max}$ (P1).

If PT and P1 are not link disjoint and x edges are common between PT and P1 and p and q are the disjoint edges in the paths PT and P1 respectively, then cost(PT)=(x+p)×M and cost(P1)=x×M + q.

It is quite obvious that cost (PT) > $cost_{max}$ (P1).

Now after removing Active Path P1 from G, MAPS will assign P2 as Backup Path.

Therefore MAPS can find disjoint path pair, (if exists) if M≥N.

Corollary 1: In a connected graph G, no link disjoint path pair between <s,d> will be found iff all paths between <s,d> have some common edge(s).

Lemma 2:

In a connected graph G, there must be some common edges(s) among all possible paths between <s,d> iff ***cost*** $_i$ ≥ ***cost*** $_i$ ***(AP),*** at some iteration i.

Proof: To prove the necessary condition, i.e. **cost** $_i$ ≥ **cost** $_i$ **(AP)**, assume there are common edge(s) among all possible paths between <s,d>. From corollary 1, no disjoint pair is possible. Clearly, two possible cases arise:

Case 1: No two paths are of same length. Our algorithm will keep on finding new paths and assigning M to the AP edges, until the algorithm fails to get any new AP. No new AP exists when all edges of all possible paths between <s,d> are assigned with M. Then from matrix property, [M×y_{ij}] = M×[y_{ij}], the old AP is repeated at the next iteration, say i, by the shortest path algorithm. Now, according to our algorithm, ***cost (AP)*** keeps on increasing at next iteration due to the assignment of M and shortest path algorithm. Therefore, due to the repetition, new AP has lower cost, implying,

$$cost_i(AP) < cost_{i-1}(AP)$$

$$\Rightarrow cost_i(AP) < cost_i$$

Case 2: There exist at least two paths having same length. Then, these two paths will be selected by our algorithm in two successive iterations, say (i-1) and i. At iteration i, ***cost*** $_i$ ***= cost*** $_i$ ***(AP).***

To prove the sufficient condition, let us assume, cost $_i$ ≥ cost $_i$ (AP). Again, we consider two cases:

Case 1: ***cost*** $_i$ ***> cost*** $_i$ ***(AP).*** Since cost (AP) keeps on increasing in every next iteration, the above case arises only when an old path is selected at iteration i. Then it is obvious there must be some common edge(s) among all possible paths between <s,d>.

Case 2: ***cost*** $_i$ ***= cost*** $_i$ ***(AP).*** This implies, $cost_{i-1}$(AP) =cost $_i$ (AP), i.e. two successive APs have same costs (from lemma 1). These two paths are not obviously disjoint; otherwise they would have been selected earlier in step 3 of MAPS. This is same for all other paths of equal lengths.

Thus all paths between <s,d> have some common link.

Corollary 2: No link-disjoint disjoint pair is possible between <s,d> in a graph G, iff ***$cost_i$* ³ *$cost_i$ (AP),*** at some iteration i.

3.2.2 Complexity Analysis of MAPS

Our heuristic does not run on polynomial time theoretically (share path protection for multiple wavelengths is a NP-complete problem). But also note that MAPS *does not* check all possible paths (can be proved) for AP between source-destination pair. For each iteration, this heuristic runs in $O(n^2)$, where 'n' is the number of nodes in a graph. So if it takes 'k' iterations on an average, it has average time complexity $O(k.n^2)$. Theoretically, value of 'k' might be large, but in reality, value of 'k' is significantly low (for large n, k<<n) for initial requests when the graph is dense. When the graph becomes less dense towards the last requests, there may be some all-paths checking between <s,d>; but then number of paths between <s,d> reduces to large extent due to the sharing nature of our algorithm. So we claim that our algorithm works significantly well in practice.

4. ROUTING AND WAVELENGTH ASSIGNMENT

Our fault tolerant RWA algorithms have two different phases: 1) *Routing heuristic* i.e. finding two link disjoint routes between node pair to establish active and backup path 2) *Wavelength assignment heuristic* i.e. assigning wavelengths to each of those active and backup path.

We use a W×L matrix called *link-state matrix* for the physical topology G(V,E). The state of a link $i \in E$ can be specified by the column vector $\sigma_i = (\sigma_i(1), \sigma_i(2), \ldots, (\sigma_i(W))^T$, where $\sigma_i(j)=1$ if wavelength λ_j is allocated in the link i by some lightpath and $\sigma_i(j)= 0$, otherwise. So the state of the network is given by the link-state matrix $U= (\sigma_1, \sigma_2, ..., \sigma_L)$ [1]. Initially, all wavelengths are available in each link. So we initialize the U matrix by 0. When a connection request arrives, RWA algorithm searches for AP & BP pair such that for each path $P=<i_1, i_2, \ldots, i_l>$ between <s,d> of the request, $\sigma_{i_k}(j)=0$ for all k=1, 2, …, l and some j. The search order is also fixed in *priori*, i.e., $\lambda_1, \lambda_2,\ldots, \lambda_W$ [1].

To implement shared path protection scheme, we have two separate graphs G_{AP} & G_{BP} on which AP and BP will be found. We also consider two state matrices U_{AP} and U_{BP}, to maintain the wavelength states of active path and backup path respectively. We sort the traffic matrix T in descending order. Let *Req* be the ordered set of source-destination pairs so that $Req=\{(i_1,j_1),(i_2,j_2),\ldots\ldots,(i_n,j_n)$**:** $T[i_k,j_k] \geq T[i_{k+1},j_{k+1}]$, $1 \leq i_k,j_k \leq n$, $1 \leq k \leq n-1\}$. Connection requests for establishing lightpaths will be generated according to this ordered set *Req*, to maximize one-hop traffic.

We present two RWA algorithms based on *static* routing (AP and BP are pre-computed based on physical topology) and last with *adaptive* routing heuristics:

1. RWA with APF
2. RWA with MAPS
3. RWA with Adaptive-Search (AS).

Initially $G_{AP} = G_{BP} = G$, for all the algorithms.

4.1 RWA with APF as Routing Heuristic

Algorithm 3:

Step 1: Request Arrival: Let a connection request arrives (from the connection request set *Req*) to establish a lightpath between <s,d>. First check, whether <s,d> are connected in G_{AP}. If they are not connected, then connection request is blocked. Exit.

Otherwise, go to next step.

Step2: Routing: Find disjoint path pair AP and BP between <s,d> using Procedure *disjoint_APF*. If not found, then connection request is blocked. Exit.

Else go to step 3.

Step 3: **Wavelength Assignment:**

3.a) *for AP:* For each wavelength λ_i ($1 \leq i \leq W$), check U_{AP} whether λ_i is available for all links of AP. If available, then allocate λ_i for AP. Update U_{AP} and U_{BP} accordingly and go to 3.b.

If no such i is available, request is blocked. Exit.

3.b) *for BP:* Do the same as 3.a for BP using U_{BP} and allocate λ_j ($1 \leq j \leq W$) for BP and update U_{AP}, if j is the available wavelength. Then return AP with wavelength i and BP with wavelength j.

If no such j is available, restore old U_{AP} and U_{BP} before this request and current request is blocked. Exit.

Procedure *disjoint_APF:*

Step 1: First check whether both s and d have degrees less than 2 in the original graph G. If true then exit, as no disjoint pair is possible. Else go to next step.

Step 2: Find AP: Now apply shortest path algorithm in G_{AP} to find an AP between <s,d>. Then remove these AP-links from G_{AP} and G_{BP}.

Step 3: Find BP, if possible:

3.a) Check the connectivity of the remaining G_{BP} between <s,d>. If they are connected, then find the shortest path between <s, d>. This is BP. If not connected, go to Step 4.

3.b) So AP and BP are found between <s,d>. Restore AP-links in G_{BP} and return AP and BP.

Step 4: If BP not found for above AP: Restore AP-links in G_{AP} and G_{BP}. Exit.

4.2. RWA with MAPS as Routing Heuristic

Algorithm 4:

Same as Algorithm 3 except that in step 2 during routing, we use Procedure *disjoint_MAPS* to find disjoint path pair instead of Procedure *disjoint_APF*.

Procedure *disjoint_MAPS*:

Step 1: Variable *cost* is initialized to zero. Assign unit weight to all edges in G_{AP} and G_{BP}. Then check whether both s and d have degrees less than 2 in the original graph G. If yes, then exit, as no disjoint pairs are possible.

Otherwise, go to next step.

Step 2: **Find AP:** Now apply shortest path algorithm on G_{AP} to find an AP between <s,d>. Then remove these AP-links from G_{AP} and G_{BP}.

Step 3: **Find BP, if possible:**

3.a) Check the connectivity of the remaining G_{BP} between <s,d>. If they are connected, then find shortest path between <s, d>. This is BP. If not connected, go to Step 4.

3.b) So AP and BP are found between <s,d>. Restore AP-links in G_{BP} and return AP and BP.

Step 4: If BP not found for above AP:

4.a) Restore AP-links in G_{AP} and G_{BP}. Assign a big weight M to AP-links in G_{AP} and find the *cost* of the AP i.e. *cost (AP)*. Then compare it against *cost*.

4.b) If (*cost* ≥ *cost (AP)*), then disjoint pair between <s,d> is not possible. Exit. Otherwise,

4.b.i) Assign *cost (AP)* to *cost*.

4.b.ii) Go to step 2 to search for the new alternative AP for which disjoint-pair can be found.

4.3 RWA with Adaptive-Search (AS)

In algorithm 4, if wavelengths cannot be assigned to the AP-BP pair returned by *disjoint_MAPS*, the connection request is blocked. But some link disjoint path pair still may exist for which wavelengths are available. In *adaptive routing*, the routes are chosen dynamically from the available resources in the network state at the time of connection establishment [1].

Algorithm 5:

If the algorithm 4 fails in step 3 due to unavailability of wavelengths, perform the Procedure *Adaptive_Search*.

Procedure *Adaptive_Search:*

Step 1: **Find AP and corresponding wavelength:**

1.a) For each available wavelength λ_k ($1 \le k \le W$) in link-state matrix U_{AP}, generate a graph $G^k_{AP}(V, E_k)$ where E_k is the set of links for available λ_k. Then go to 1(b).

If no such graph is generated for all λ_k, request is blocked due to wavelengths constraint. Exit.

1.b) Get the minimum-cost-path p_k between <s,d> for each G^k_{AP}. Say, $P = \cup p_k$, ($1 \le k \le W$). Find minimum-cost-path p_i from P, ($1 \le i \le W$). Then p_i is AP with wavelength λ_i. Go to Step 2.

Step 2: Find BP and corresponding wavelength:

2.a) For each available wavelength λ_k ($1 \le k \le W$) in state matrix U_{BP}, generate a graph $G^k_{BP}(V, E_k\text{-AP})$ where E_k is the set of links for available λ_k. Then go to 2(b).

If no such graph is generated for all λ_k, request is blocked due to wavelengths constraints. Exit.

2.b) Get the minimum-cost-path p_k between <s,d> for each G^k_{BP}. Say, $P' = \cup p_k$, ($1 \le k \le W$). Find minimum-cost-path p_j from P′, ($1 \le j \le W$). Then p_j is BP with wavelength λ_j. Return AP p_i and BP p_j with wavelengths λ_i and λ_j respectively.

Step 3: For next request, eliminate links in p_i (AP) from U_{AP} and U_{BP} for λ_i. Also eliminate links in p_j (BP) from U_{AP} for λ_j.

5. SIMULATION RESULTS

We evaluate the performance of various algorithms discussed in this paper with the help of simulation on the graphs of various sizes and densities. Our density gradations of random graphs are as follows: Graph of n number of nodes with density δ has $\lceil \delta \times (n-1) \rceil$ number of edges placed randomly all over the graph. First, we show the results of the averages of 25 graphs for each case (Table 1), comparing conventional APF algorithm with our MAPS. We observe that they give almost same result when the graph is dense. As the density of the graph becomes low, the improvement of MAPS becomes significant than APF.

Table 1. Comparison between APF and MAPS routing algorithms

No. Of nodes (n)	**Graph Density** (d)	**Link Disjoint Paths Discovered (%)**	
		APF	**MAPS**
30	2	84.94	84.94
30	3	97.60	97.60
30	4	100.0	100.0
40	2	85.00	85.06
40	3	98.78	98.81
40	4	99.40	99.40
50	2	85.94	85.96
50	3	96.35	96.37
50	4	99.68	99.68
60	2	82.41	82.47
60	3	97.62	97.62
60	4	99.86	99.86
70	2	87.17	87.35
70	3	98.21	98.22
70	4	99.66	99.66
80	2	84.93	84.93
80	3	97.70	97.75
80	4	99.93	99.93

In the second phase, simulations are developed to evaluate the performances of three RWA heuristic algorithms for shared path protection:

Algo 1: RWA with traditional APF

Algo 2: RWA with MAPS

Algo 3: RWA with Adaptive Search (AS)

We generate the long-term average traffic demand matrix randomly for each case where the relative traffic between two nodes varies from 1 to 100 units. As we see from Figure 5 and 6, performances of algorithm 2 & 3 are always superior to algorithm 1. Even algorithm 3 outperforms algorithm 2 especially when the number of wavelengths is relatively large compared to number of nodes. When density of the graph is low, algorithm 2 gives better result than algorithm 3 probably due to following reason: The algorithm 3 does not work separately for routing and wavelength assignment, but it works on the ‘state matrix’ data structure. Thus it satisfies previous requests even after failure of MAPS, but potentially blocks many future requests by that. So, in that case the overall blocking probability increases.

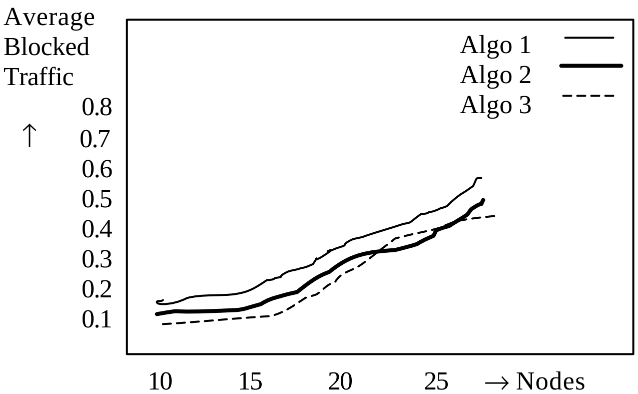

Figure 5. Average blocked traffic Vs nodes for the various algorithms discussed in section 4 with graph density δ=3 and no of wavelengths per link W=6

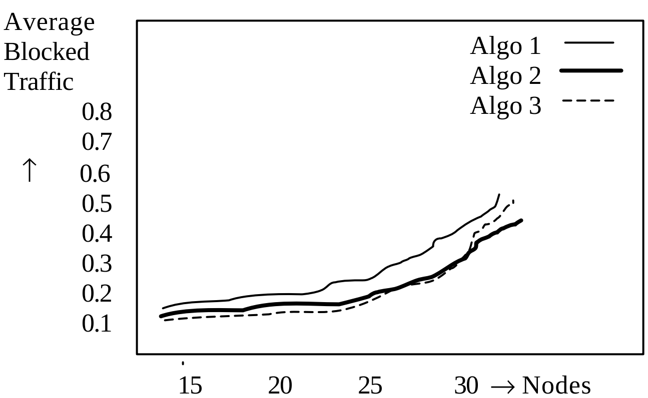

Figure 6. Average Blocked Traffic Vs nodes for the various algorithms discussed in section 4 with graph density δ=4 and no of wavelengths per link W=12

6. CONCLUSION AND FUTURE WORKS

In this paper, we studied the DWDM shared path protection problem. We presented a link disjoint path finding algorithm called Multiple Active Paths Search (MAPS), which takes care of the trap topologies, where the traditional Active Path First (APF) heuristic fails. It also outperforms the APF algorithm specially when the density of the graph is low. Finally we proposed a set of fault tolerant routing and wavelength assignment (RWA) algorithms using fixed and Adaptive Search (AS) routing and showed significant improvement in average blocked traffic than RWA algorithm with traditional APF for routing.

In this paper we investigated the problem of single link failure in DWDM networks. Fault recovery for multiple link failure is still unexplored. Pre-designed path protection scheme is efficient in terms of fault recovery time but it is inefficient in terms of resource utilization. Dynamic restoration may provide a viable alternative to it, but little research has been done on the design of rapid restoration schemes that can provide any level of service guarantee.

ACKNOWLEDGMENTS

The authors would like to thank the teaching and non-teaching staff members of the Computer Science & Engineering department of Haldia Institute of Technology. They also thank Prof S. Ghose, Prof I. Sengupta, IIT Kharagpur and Dr. R. Datta, NERIST who introduced them to the problems of WDM networks.

REFERENCES

[1] A. Mokhtar, M. Azizoglu, “Adaptive Wavelength Routing in All-Optical Networks”, IEEE/ACM Transactions on Networking, Vol-6, No.2 April 1998.

[2] C. Ou, J. Zhang, H. Zang, L. Sahasrabuddhe, B. Mukherjee, “Near-Optimal Approaches for Shared Path Protection in WDM Mesh Networks”, ICC 2003 – IEEE International Conference on Communications, Vol 26, No 1, May 2003, pp 1320-1324.

[3] D. Zhou, S. Subramaniam, “Survivability in Optical Networks”, IEEE Network, Vol 14, No 6, pp 16-23, Nov/Dec. 2000.

[4] J. W. Suurballe, R. E. Tarjan, “A Quick Method for finding Shortest Pairs of Disjoint Paths”, Networks, Vol.-14, pp 325-336, 1984.

[5] L. Sahasrabuddhe, S. Ramamurthy, B. Mukherjee, “Survivable WDM Mesh Networks”, IEEE/OSA Journal of Lightwave Technology, Vol-21, No-4, pp 870-883, April-2003.

[6] R Anderson, F. Chung, A. Sen, G. Xue, “On Disjoint Path Pairs with Wavelength Continuity Constraint in WDM Networks”, IEEE INFOCOM 2004, pp 524-535.

[7] S. Yuan, Jason p. Jue, “Dynamic Lightpath Protection in WDM Mesh Networks under Wavelength Continuity Constraints”, IEEE Globecom, 2004.