# Path Computation Algorithms for Dynamic Service Provisioning in SDH Networks

R.Madanagopal*, N.Usha Rani†, Timothy A. Gonsalves*
*Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai, India
Email: {madan,tag}@lantana.tenet.res.in
†NMSWorks Software Pvt. Ltd.
Chennai, India
Email: usha@nmsworks.co.in

***Abstract*— Synchronous Digital Hierarchy (SDH) is a time division multiplexing technology widely used in transport networks to provide bandwidth services. Dynamic service provisioning refers to the arrival of service requests one-by-one randomly with no prior information on future requests. This requires the use of on-line algorithms which automatically compute the path to be taken to satisfy the given service request. This problem involves a tradeoff between minimizing the number of requests that are rejected and minimizing the total bandwidth that is utilized. Many earlier works have addressed path computation algorithms, but they treat each link as having some integer units of bandwidth. They do not take into account the multiplexing structure defined by SDH which imposes restrictions on the allocation of bandwidth and the fact that higher order trails (logical connections) have to be established to support any bandwidth requirement. In this work, these factors are considered in the path computation algorithms. The network is treated as a graph containing physical links and logical trails and weights are assigned to them before computing a path with the least cost. Weights are assigned such that the trails are given higher preference to physical links so that existing trails are used wherever possible. This avoids unnecessary creation of new trails. The performance is evaluated for different values of weights. An improvement in the form of dynamically adjusting the weights of links and trails is done and its performance is shown to be better than having constant weights.**

## I. Introduction

In the current environment of high speed communications, network operators and service providers are constantly flooded with bandwidth requirements. This has led to the development of different transport technologies like PDH, SONET/SDH, WDM/DWDM, ATM, DSL etc. Also, the need for providing differentiated services with different Quality of Service (QoS) requirements is being seen as a good opportunity by service providers to increase their revenue[1]. In this scenario, it is envisaged that there will be a large number of service requests with different QoS requirements which implies there will be a constant change in configuration of the network. This is compounded by the fact that the service providers will try to use the available network resources efficiently before investing in further resources.

Service providers require faster provisioning of services to satisfy more customer requests in less time. Typically, services are provisioned manually by the service providers by logging into management systems and doing provisioning related activities using them. They have to take into account the requested demand, required QoS, available network resources and the ability to accommodate future requests before starting the work of provisioning a requested service. This implies manual path computation and maintenance of huge inventory details. Then the configuration on the network elements involved has to be done. These processes being manual, are tedious and more error prone. It might not result in optimal usage of network resources. The time required to provision a service will be more. But minimizing service turn-on times is the key to accommodate more service requests in less time which results in maximizing the revenue for service providers.

The solution to this problem is to have automatic provisioning systems. Operators should be able to give the end-points of the service and the bandwidth required and the system should be able to provision the bandwidth automatically by calculating the appropriate path such that the network resources are optimally utilized. Network Management Systems (NMS) or Provisioning Management Systems (PMS) which have the complete network topology and the inventory details should be able to perform the provisioning operation. Firstly the NMS/PMS should be able to get the inventory details from the database or the Element Management Systems (EMS) or the Network Elements (NE) directly and it should be able to issue configuration commands to them to provision the requested capacity in the network elements involved. The other task is to find the best path to be used to satisfy the requested service taking into account the capacity that is already in use, free capacity available such that more future requests can be accommodated and the network resources are utilized optimally. This second aspect is the focus of this work.

The remainder of this paper is organized as follows. Section II gives details regarding related work. Section III describes the path computation problem in SDH networks by describing the unique aspects that need to be considered. Section IV describes the details of the various algorithms proposed in this paper and their limitations if any. Also, it describes how the algorithms can be extended for computing dedicated protection paths in

1-4244-0799-0/07/$25.00 ©2007 IEEE

addition to the computing a working path. The performance of the algorithms are analyzed in Section V. Finally, Section VI concludes this paper.

## II. Related Work

Various earlier works have addressed the path computation problem in different networks. One of them is the all-optical network lightpath provisioning problem. The issue here is to provide routes to the lightpath requests and to assign wavelengths on each of the links along this route among the possible choices so as to optimize a certain performance metric. This is the commonly known Routing and Wavelength Assignment (RWA) problem[2], [3], [4]. This has been considered with both static and dynamic traffic demands. Also, it has been studied with no wavelength conversion, partial wavelength conversion and full wavelength conversion. In work to date, it has been formulated as a difficult integer linear programming (ILP) problem that does not lend itself to efficient solution. Different heuristics have been proposed and the problem solved based on different assumptions[5]. An integrated topology independent procedure to solve the RWA problem to minimize the number of SONET ADMs is proposed in [6].

Distributed path computation and provisioning is another area studied extensively. Many centralized schemes have been proposed which take the entire network topology and compute paths for the requested bandwidth demands. Since centralized schemes are not scalable for large network sizes, distributed route computation and provisioning has been proposed[7]. While distributed control enhances scalability and flexibility, it might result in suboptimal path computation as a result of summarization of resource information. Summarization is needed since each node cannot maintain the complete network topology and resource information. The tradeoff is between the path computation efficiency and the amount of information to be disseminated.

In this regard, generalized MPLS (GMPLS) by IETF is being introduced as the most suitable control plane solution for next-generation optical infrastructure. This provides a framework in which the well-known and proven MPLS paradigm is being extended to be a control plane not only for routers etc. but also for optical and other transport network elements. This is achieved by means of Traffic Engineering (TE) extensions to a suite of protocols like OSPF, IS-IS, RSVP, LDP etc. Apart from IETF, ITU has defined a set of distributed control plane specifications for the Automatic Switched Optical Network (ASON). OIF is also working on user-to-network interface (UNI) and network-to-network interface (NNI) for optical networks. Path computation in SONET rings using GMPLS by creating Label Switched Paths (LSP) by making small changes in OSPF-TE LSA processing rules and RSVP-TE signaling mechanism is proposed in [8].

The problem of efficiently managing and configuring a transport network to both route and protect services in the face of traffic uncertainty in time and space has been framed within what is called the Shared Backup Path Protection (SBPP)[9] approach to dynamic survivable service routing. SBPP can be thought of as arranging a set of diverse protection paths but keeping the spare links on the backup paths unconnected and shared with other setups, with connection occurring as needed upon failure. The aim is to increase capacity efficiency by allowing sharing of protection channels between primary working paths that do not have any single-failure modalities in common. Technically, such primary paths are said to have no Shared Risk Link Groups (SRLG) in common. As a result, fewer spare capacity is needed to protect more working capacity with the assumption that not all the working paths fail at the same time.

The Protected Working Capacity Envelope (PWCE)[9] concept involves adaptation of static design models to create not an exact solution for one single demand matrix, but an envelope of protected working channels, well suited for a large family of random demand instances that may be somehow related to a single representative demand pattern. An important property is that actions of any type related to ensuring protection occur only on the time-scale of the statistical evolution of the network load pattern itself, not on the time-scale of individual connections. Protection cycles (p-cycles) with support for the protection of spans on the ring is chosen for implementing PWCE.

The problem of dynamically routing QoS guaranteed paths with restoration in MPLS networks is studied in [10]. Restorability implies that to successfully route a path set-up request both an active path and an alternate link (node) disjoint backup path have to be routed at the same time. The backup paths can be shared amongst certain active paths while still maintaining restorability. ILP formulation of this problem is given and it is shown that a partial information scenario which uses only aggregated information instead of complete per-path information for routing performs well and the number of requests rejected is very close to the full information bound.

Two approaches for provisioning of survivable Data over SONET/SDH (DoS) connections utilizing the inverse-multiplexing capability of virtual concatenation have been proposed in [11]. Virtual concatenation is a technique which groups an arbitrary number of SONET/SDH containers, which may not be contiguous, to create a bigger container. A new scheme for real-time bandwidth allocation and path restoration in SONET mesh networks, in response to demand and load dynamics is presented in [12]. Various options for routing in SDH networks for ring and mesh topologies are described in [13]. A sequential algorithm and a parallel algorithm for computing disjoint paths for protection and QoS for next generation SONET networks have been proposed in [14] where the shortest path is computed based on the delay in the links and the bandwidth requested. A generic approach to service-differentiated connection accommodation in wavelength-routed networks is proposed in [15].

## III. Path Computation Problem in SDH Networks

All these works have one thing in common. They consider a graph of the network topology containing nodes representing

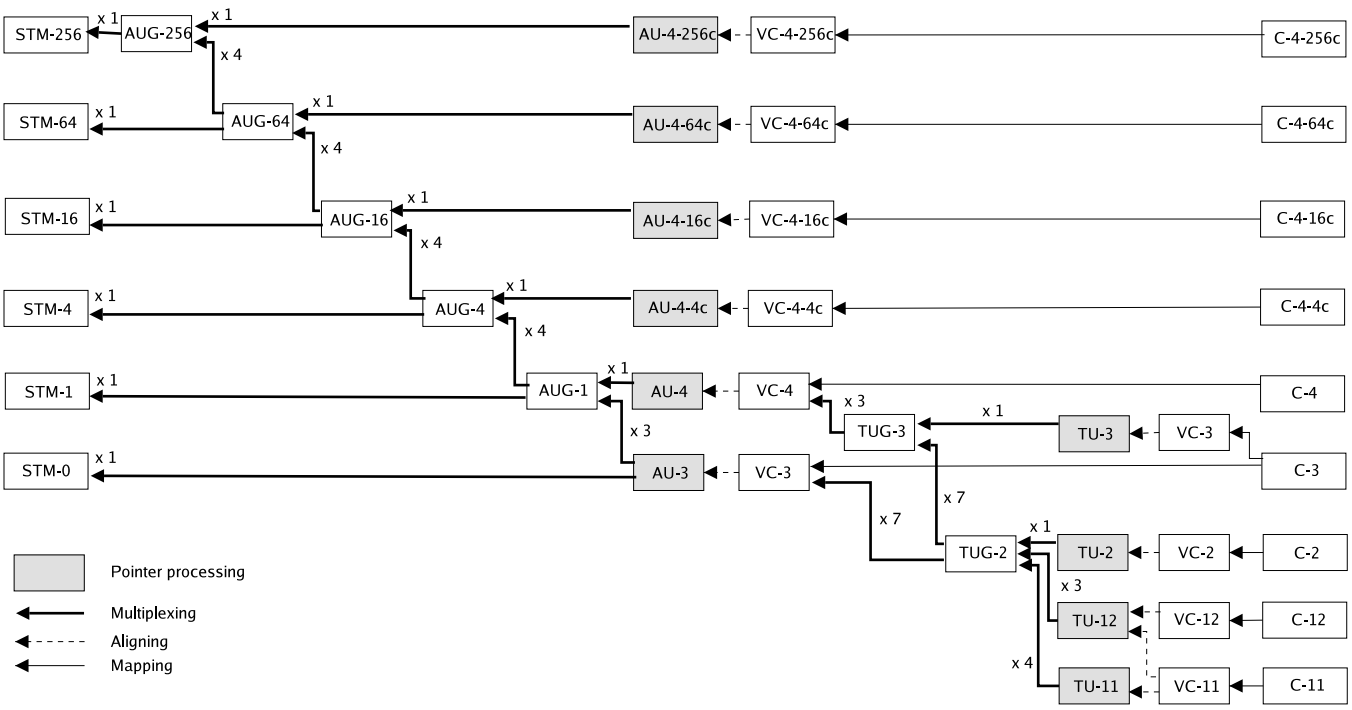


Fig. 1. SDH Multiplexing Structure

the network elements present and links inter-connecting them with integer units of capacity. This allows to them to formulate the problem mathematically and numerical solutions are thus possible. In some cases, weights are associated with the links taking into account different constraints and traffic engineering parameters. Those algorithms then attempt to compute paths such that they take the shortest route or a route that minimizes the network resources consumed. For protection paths, the aim is to use the existing protection routes as much as possible by sharing in such a way their corresponding working paths are not affected by a single failure.

But when it comes to provisioning in transport networks particularly SONET/SDH networks, links cannot be simply considered to have certain integer units of capacity as the multiplexing structure defined by those technologies has to be followed while provisioning the requested services. The multiplexing structure defined by SDH[16] is shown in Fig. 1.

The common G.703[17] signals such as E1, E4, DS1, DS3 etc. are mapped into their corresponding virtual containers VC-12, VC-4, VC-11, VC-3. The lower order virtual containers like VC-11, VC-12, VC-2 are pointed to by their tributary unit (TU) pointers. These are then multiplexed into tributary unit groups (TUG) which are then multiplexed into higher order virtual containers like VC-3 and VC-4. They are pointed to by their administrative unit (AU) pointers and then multiplexed into administrative unit groups (AUG) which are in turn multiplexed into one of the possible Synchronous Transport Modules (STM) which are the units of transmission in SDH. Similar structure exists for SONET as well.

Because of this multiplexing structure of SDH, each link cannot be assumed to have simply some integer units of capacity and the free capacity cannot be obtained simply by subtracting the alloted capacity from the maximum capacity and allocations cannot be made until the free capacity becomes zero. In SDH, higher order containers like VC-4 have to be used to create trails between the source and destination nodes before provisioning any bandwidth between two points. This means that even to have a E1 bandwidth which maps to VC-12 in SDH between two points, a single VC-4 trail or a sequence of VC-4 trails have to be available already or have to be established newly.

When a VC-12 (approximately 2 Mbps) is provisioned in a newly created VC-4 trail, then that trail cannot support a future VC-4 request since the full capacity is not available any more. Similarly, it can accommodate only two future VC-3 requests since out of a maximum of 3 VC-3s that can be multiplexed to form a VC-4, the first one is broken to accommodate the requested VC-12 service. If 3 VC-12s are provisioned in the 3 different VC-3s, then that VC-4 trail cannot accommodate any further VC-3 requests even though only 3 * 2 = 6 Mbps out of the potential 140 Mbps for a VC-4 is currently alloted since all VC-3s are broken. Similar restrictions apply for other rates like VC-2 and VC-11. Also in SDH, any arbitrary bandwidth cannot be provisioned in a trail. Only those standard rates defined in SDH are allowed. So the requests cannot be in numerical units of bandwidth, but in one of those standard rates.

In summary, the existing path computation algorithms cannot be used as such for SDH networks for the following two reasons:

1) In SDH systems, trails have to be present or created newly before provisioning any service.
2) SDH defines a strict multiplexing structure which has to be followed while provisioning a service. All service requests must fit into one of the standard rates supported by SDH.

This paper deals with path computation algorithms for SDH taking into account the above two considerations.

## IV. Path Computation Algorithms

A path computation algorithm finds minimum cost path between the source and destination nodes for a requested service with some requested capacity. The algorithms have to be dynamic since the requests come online and they have to be satisfied as soon as possible. Since the algorithms have to be dynamic they will have no idea on the future requests that are possible. The algorithms should have performed such that they have used the network resources efficiently until that instant. The algorithms presented in this paper are online without taking into account the possibility of reconfiguration of the services later for better network resource utilization.

Service providers like to achieve the twin goals of maximizing the number of requests serviced and minimizing the total bandwidth consumed. Also, the number of trails created should be minimum so that the bandwidth is not fragmented. Less fragmentation means more number of high bandwidth requests getting accepted in the future. This implies the network resources have to be optimally utilized so that more number of future requests can be accommodated resulting in maximization of the revenue. Therefore, the path computation algorithms should be able to function such that the following three goals are met in the best possible way:

1) Minimizing the number of requests rejected.
2) Minimizing the total amount of bandwidth consumed to satisfy the requested services.
3) Minimizing the number of trails created.

The topology of the network is represented as a graph $G(V, E)$, where $V$ is the set of network elements or nodes present in the network and $E$ is the set of edges in the network. The set of edges include both the set of the links and the set of trails present in the network. Since the services can be provisioned only over VC-4 trails because of the SDH multiplexing structure, the set $E$ includes both the physical links connecting the network elements and the logical trails created between them for satisfying service requests. Each physical link is associated with a weight which can be chosen based on any criteria such as distance etc. If there is no need to distinguish the physical links based on any criteria then all their weights can be made equal.

A sample network is shown in Fig. 2. The solid lines represent physical links and the dashed lines represent trails which are logical links. If the capacity of the link is STM-N, then N trails can use that link. A path is a sequence of links and trails. If the path contains links, then trails have to be created on those links for carrying traffic.

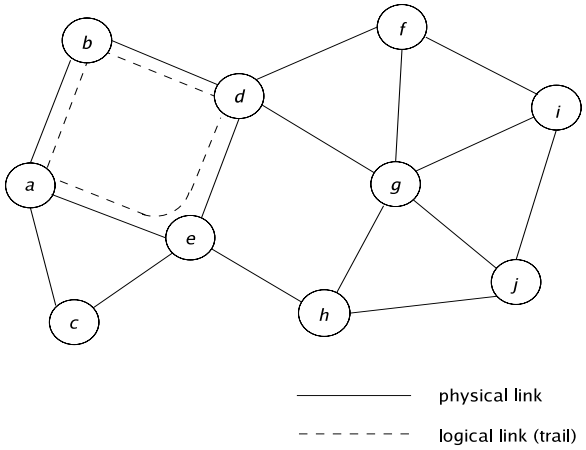


Fig. 2. A Sample Network

When a service request comes and there exists a sequence of trails, it makes sense to use those trails rather than create new trails. So, trails have to be given higher preference than physical links which in turn means that the weight of a trail has to be less than the sum of the weights of the links. This is achieved for a trail $t$ as:

$$w(t) = \alpha * \sum_{l \in L} w(l) \tag{1}$$

where $w(t)$ is the weight associated with the trail $t$, $w(l)$ is the weight associated with the link $l$, $L$ is set of links on the path through which the trail is established, $\alpha$ is some value between 0 and 1 and * denotes multiplication. Thus this equation assigns the weight for a trail as a fraction of the sum of the weights of the links traversed by that trail. As a result, the weight of the trail will be less than or equal to the sum of the weights of the links traversed. So this trail will be chosen instead of establishing new trail(s) along that path. Lower weights are given higher preference since the path chosen will be shorter if those links or trails are used during path computation. The weight of the physical link can be thought of as the cost of creating a new trail on that link.

When a new trail needs to be created, it can be created as an end-to-end trail between the source and destination nodes or individual trails can be created on the links that are part of the path that is found. The latter option is used in all the algorithms since the creation of an end-to-end trail may not result in efficient network resource utilization as it can be used only by those requests that need to go through both the nodes between which the trail was created. In Fig. 2, the trail *aed* is an example of an end-to-end trail. Though this trail passes through *e*, it can carry traffic only between *a* and *d* as the VC-4 used is only cross-connected and not terminated at *e*. To have better resource utilization of network resources, it makes sense to create two separate trails, one between *a* and *e* and the other between *e* and *d* so that there is flexibility to carry traffic between any two of those three nodes.

The algorithms described use the weights as calculated above in finding the paths. Dijkstra's shortest path algorithm is used to find the weighted shortest path in the graph of the network. Shortest path here refers to the path such that the sum of the weights of the links and trails constituting it is the minimum. A simple algorithm is described initially. Then two more algorithms which address the limitations of the earlier algorithms are described to get better results. The path computation algorithms described below find a working path for the requested service. Extension to these algorithms to find dedicated protection paths is provided finally.

*Algorithm 1*

1) Find the shortest path in the graph containing only logical trails
2) If no path exists, go to Step 4
3) If capacity is available in all the trails along the path found
   then
      provision the requested capacity and adjust free capacity in the trails along the path found
      stop
   else
      eliminate those trails where capacity is not available and go to Step 1
4) Find the shortest path in the graph containing only physical links
5) If no path exists, then report failure and stop
6) If capacity is available in all the links along the path found
   then
      create trails wherever required or use existing trails and provision the requested capacity in them and adjust free capacity
   else
      eliminate those links and trails where capacity is not available and go to Step 4

This algorithm follows a greedy approach. It tries to find the shortest path in the graph containing only trails so that no new trails need to be created. If no path exists in that graph, then it finds the shortest path in the graph containing only physical links. If there are trails present along the links constituting that path, then they are used else new trails are created. The requested capacity is provisioned on the trails found or created and the free capacity is adjusted accordingly. If capacity is not available in any of the trails or links found, then they are eliminated and the process is repeated until either a path that can accommodate the requested capacity is found or no more paths can be found in which case the service request is rejected.

*Limitations of Algorithm 1*

- It tries to find a path in the graph containing only trails even though it may be circuitous thus consuming more bandwidth
- If no path can be found in the graph containing only trails, it finds a path in the graph containing only links without taking account all the trails that are already present

*Algorithm 2*

The main reason for the limitations of Algorithm 1 is that it performs the same sequences of steps twice once on the graph containing only trails and again on the graph containing only links. This can be overcome by a forming a single graph containing both the trails and links that are present. There must be differentiation between trails and links so that trails are preferred to links wherever possible. This is achieved by assigning weights for trails as described earlier as a fraction of the sum of the weights of the links traversed by it. The modified algorithm is given below:

1) Find the shortest path in the graph containing both links and trails
2) If no path exists, then report failure and stop
3) If capacity is available in all the links and trails along the path
   then
      create trails wherever required or use existing trails and provision the requested capacity in them and adjust free capacity
   else
      eliminate those links and trails where capacity is not available and go to Step 1

The shortest path computed by this algorithm will have a combination of both trails and links. New trails will be created whenever links are found and the trails will be used as such whenever they are found. This will result in an optimal path containing already created trails that can be used and new trails that are created when they need to be done so. The value of $\alpha$ will affect the behavior of the algorithm in terms of the number of requests accepted and the total bandwidth consumed.

*Limitations of Algorithm 2*

- When more than one shortest path exists between two nodes, one of them is chosen for this request and all the future requests which require those nodes in the path. In this way, it overloads one path instead of distributing the load
- It does not take into account the free capacity available in the links
- It does not give weightage to a trail where a low rate like VC-12 can be accommodated as such when compared to a trail in which a VC-3 or a VC-2 needs to be broken to accommodate the requested rate. This may result in creation of new trails later when VC-3 or VC-2 requests come in the future. In other words, this might result in fragmentation of the capacity in the trails present

*Algorithm 3*

The main reason for the limitations of Algorithm 2 is that the weights are constant for the links and trails. In other words, the weights once assigned are not changed. As a

result, the algorithm is not adaptive to the changing network characteristics. The weights have to be dynamically changed for the algorithm to be truly reflective of the current state of the network. This will lead to better results in terms of the optimal utilization of the network resources.

For links, the dynamic weight adjustment has to be based on the free capacity available in a link. Link weight has to be increased and not decreased so that it doesn't become less than the weight of a trail created on that link. One form of dynamic weight adjustment for a link $l$ is:

$$w(l) = w_i(l) + \frac{c(l)}{c_{max}(l)} * w_i(l) \quad (2)$$

where $w_i(l)$ is the initial weight associated with the link $l$, $c(l)$ is the capacity used in the link $l$ and $c_{max}(l)$ is the maximum possible capacity of the link $l$. The weight of the link now reflects the capacity used and the capacity available in the link. The link weight increases whenever new trails are created in that link. As a result, this link will not be chosen when another link with a lesser capacity utilization compared to this is available. This will result in the balancing of the load across many links that are possible candidates for the shortest path.

However for trails, free capacity is not the most important criteria for dynamically adjusting weights. Instead the weight of the trail can be reduced when there is no need for breaking higher rate containers like VC-3 and VC-2. This will avoid fragmentation of the available bandwidth in trails and new trails need not be created whenever service requests for higher rates are made in the future. This is achieved for a trail $t$ as:

$$w(t) = \beta * \alpha * \sum_{l \in L} w(l) \quad (3)$$

where $\beta$ is a number between 0 and 1 such that the weight of the trail is reduced when there is no need to break high rate containers. Since different rates of bandwidth follow different multiplexing route in SDH, maintaining a single weight for all rates will not reflect the correct status on breaking for each rate. So different dynamic weights for different rates are maintained for each trail and the weight corresponding to the requested rate is used in the path computation step.

According to the SDH multiplexing structure, for a VC-12 and VC-11 to be created it must be multiplexed into a VC-2, then into a VC-3 and finally into a VC-4. The dynamic weight adjustment for VC-12 and VC-11 rates in a trail is:

- If both VC-3 and VC-2 have to be broken, then $\beta$ is set to 1 since all possible containers have to be broken
- If VC-3 need not be broken but a VC-2 has to be broken, then $\beta$ is set to a value less than 1 (referred as $\beta_2$)
- If both VC-3 and VC-2 need not be broken, then this is the most desirable case and $\beta$ is set to a value less than that chosen in the previous case (referred as $\beta_1$)

For a VC-2 to be created it must be multiplexed into a VC-3 and then into a VC-4. The dynamic weight adjustment for VC-2 rate in a trail is:

- If a VC-3 has to be broken, then $\beta$ is set to 1 since all possible containers have to be broken
- If a VC-3 need not be broken, then this is the most desirable case and $\beta$ is set to a value less than 1 (referred as $\beta_3$)

For a VC-3 to be created it must be multiplexed into a VC-4. So a VC-4 always have to be broken to service a VC-3 request. So $\beta$ is always set to 1 in this case. But the use of an existing trail and the creation of a trail for accommodating this VC-3 request has to be differentiated. This is inherently done since the weight of the trail will be used in the earlier case which will be less than the weight of the link which will be used in the latter case.

For a VC-4 request, a new trail has to be created always whose cost will be covered as the weight of the links involved in its creation.

*Provisioning of capacity in trails*

Once the path is computed then the requested capacity has to be provisioned in the trails that belong to that path. To achieve this, the next free slot for each rate in the trail is maintained. For VC-4 rate, the only value that is possible is 1. For VC-3 it is from 1 to 3, for VC-2 it is from 1 to 21, for VC-12 it is from 1 to 63 and for VC-11 it is from 1 to 84.

Whenever a trail is created the free slot for all the rates are set to 1. Subsequently, the slots are alloted sequentially. When high rate containers have to be broken to accommodate a low rate request, then the free slot for those high rates according to the multiplexing structure are increased by 1. For example, if a VC-12 request is made and a new trail created for it, then the free slots for VC-3 and VC-2 are increased by 1. Then the free slot for VC-12 is also increased by 1. Now if a VC-3 request needs to be accommodated in this trail, the second VC-3 will be alloted to this request and the free VC-3 slot value is increased to 3. Now if successive VC-12 requests are made that have to be accommodated in this trail, they will be alloted values from 2 to 21 which is the maximum value that is possible as a result of breaking the first VC-3. If another VC-12 request is made, it will be alloted the slot 43 since the second VC-3 is already alloted for a VC-3 request. Similar mechanism is used when provisioning requests for other rates are made.

*Provisioning of capacity in links*

For links, provisioning involves creation of VC-4 trails along that link. In SDH, the standard transport rates defined are STM-1, STM-4, STM-16, STM-64 and STM-256. The number after STM indicates the number of VC-4 trails that can be created on that link. Whenever new trails are created on the link, the free capacity is reduced accordingly. New trails can be created until the free capacity becomes 0.

*Extension to support dedicated protection paths*

The algorithms described above compute a path for the given service request between two nodes for the requested capacity. This is typically the working path for the given

service request. But customers typically want high availability for the services they take from the service providers. This is achieved by protecting the service through some other route so that even when the working path fails, the service can be continued in a protection path. There are two types of protection: dedicated and shared. A dedicated protection path is exclusively reserved for one working path and protects only that path. This is commonly referred to as 1+1 protection. Shared protection refers to the case where one protection path is shared among many working paths such that any one working path failure is overcome by this protection path. But this can protect only one working path at the maximum. This is commonly referred to as 1+N protection. This is employed when network resources are not at a premium and when it can be assumed that not all working paths will fail at the same time.

To find a dedicated protection path for a given service request in addition to finding a working path, the above algorithms can be used as follows:

1) Use one of the algorithms to find a working path.
2) Eliminate the edges which are part of the working path (if the protection path has to be edge disjoint) and the nodes which are path of the working path (if the protection path has to be node disjoint) from the graph.
3) Use the same algorithm used earlier to get a path which can be used as the protection path.

The above algorithm is similar to the sequential algorithm used in [14].

## V. Performance Results

The performance of the described algorithms are evaluated and the results obtained are provided in this section. The experiments were performed on the three following networks:

- *Network 1:* 70 nodes, 103 links
- *Network 2:* 14 nodes, 21 links (National Science Foundation (NSF) network)
- *Network 3:* 47 nodes, 82 links (a national service provider (VSNL) network)

The service requests are randomly generated and the performance of the algorithms are evaluated by running 10 iterations, each with different sets of service requests for all the three networks, which are simulated. In each iteration, service requests are generated as follows:

1) Per iteration some number of requests are generated such that a small fraction of them are rejected.
2) Source node and destination node are chosen randomly for each service request.
3) The bandwidth requirement is generated in the following proportion: VC4 - 4%, VC3 - 10%, VC2 - 6%, VC12 - 80%. This is roughly the distribution of bandwidth requirements for services received by a well-known national service provider (VSNL).

The number of requests rejected is an important parameter that has to be evaluated. However, since different service requests are for different bandwidth rates, they cannot be treated equally. For example, a VC4 is equivalent to 63 VC12s. So, an execution that rejects a VC4 request due to lack of capacity and accepts less than 63 future VC12 requests will show better performance when compared to an execution that accepts the VC4 request and rejects the future VC12 requests. To overcome this problem, the requests that are accepted have to be weighted according to the relative bandwidth to get a better indication of the performance. This is achieved by keeping VC12 as the base and multiplying each VC4 request by 63, VC3 request by 21 and VC2 request by 3.

In each experiment the following parameters are considered:

1) Weighted number of service requests accepted.
2) Number of service requests rejected.
3) Number of trails created.
4) Percentage of the total bandwidth consumed to satisfy the accepted requests.

The experiments are run with the above configuration for evaluating Algorithm 1. For evaluating Algorithm 2, the value of $\alpha$ in (1) is varied from 0.1 to 1.0 with increments of 0.1. For evaluating Algorithm 3, the values for $\beta_1$, $\beta_2$ and $\beta_3$ are varied from 0.3 to 0.7, 0.4 to 0.8 and 0.5 to 0.9 respectively. The experiments are run for the three networks 10 times each with different sets of random service requests. The results obtained in one of the 10 iterations for the four parameters mentioned above are shown in Figs. 3-6 for Network 1, Figs. 7-10 for Network 2 and Figs. 11-14 for Network 3. Similar results were obtained in the other 9 iterations as well.

From the results obtained, it is found that the Algorithm 2 gives better results by minimizing the number of requests rejected, trails created and total bandwidth utilized than Algorithm 1 for almost all the values of $\alpha$ between 0.1 and 1.0. This shows that the greedy approach of Algorithm 1 is not able to perform as well when compared to Algorithm 2 where the problem is solved by considering a single graph containing both trails and links and the best path as a whole is found. Further, the Algorithm 3 with dynamic adjustment of weights is able to perform better when compared to Algorithm 2 which uses constant weights. This is due to the balancing of load achieved by means of dynamic adjustment of weights which results in the optimal usage of network resources. But, in certain cases it can be observed that Algorithm 1 is giving better results when compared to Algorithm 2 and Algorithm 3 for lower values of $\alpha$. So very low values for $\alpha$ are not desirable.

Also it is found that the values of 0.6 to 0.9 for $\alpha$ results in the minimum number of requests being rejected while the weighted number of requests accepted is maximum. At the same time, the number of trails created is also lesser for these values of $\alpha$. For some values of $\alpha$ other than 0.6 to 0.9, the number of trails created is less when compared with those for $\alpha$ between 0.6 and 0.9. In these cases, the number of requests accepted is less and so the number of trails created is also less. But when the number of trails created proportional to the number of requests accepted is considered, the values of $\alpha$ between 0.6 and 0.9 is better when compared to other values of $\alpha$. The same holds for the percentage of total bandwidth

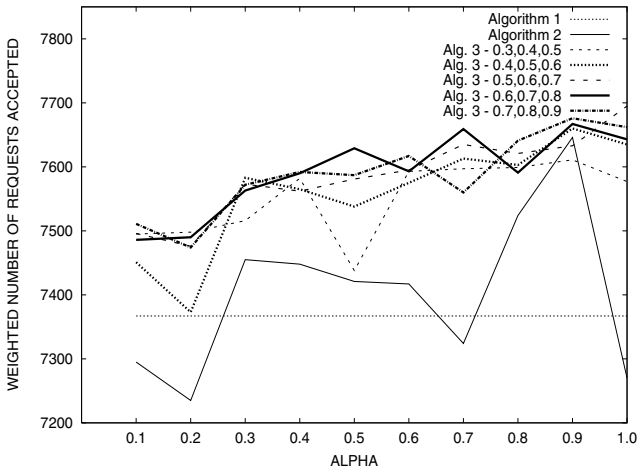


Fig. 3. Weighted number of requests accepted in Network 1

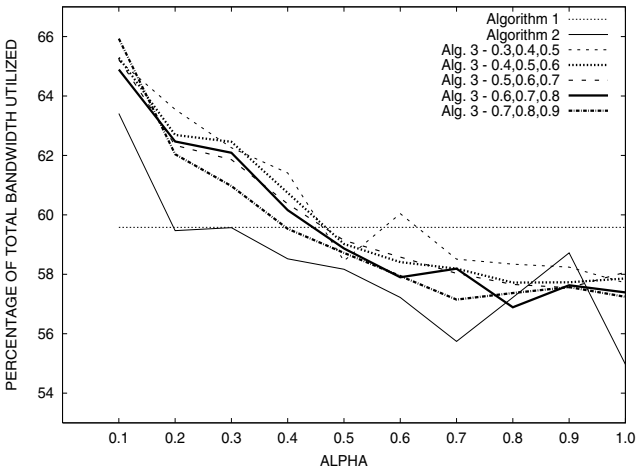


Fig. 6. Bandwidth utilized in Network 1

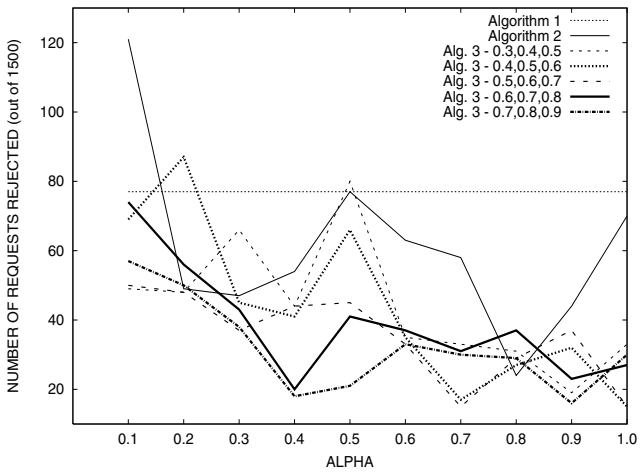


Fig. 4. Number of requests rejected in Network 1

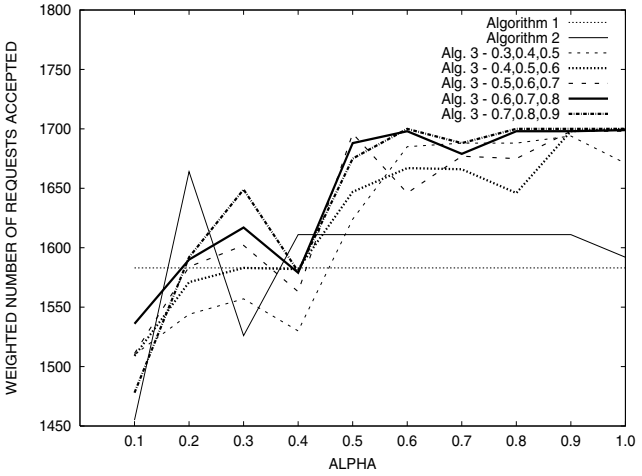


Fig. 7. Weighted number of requests accepted in Network 2

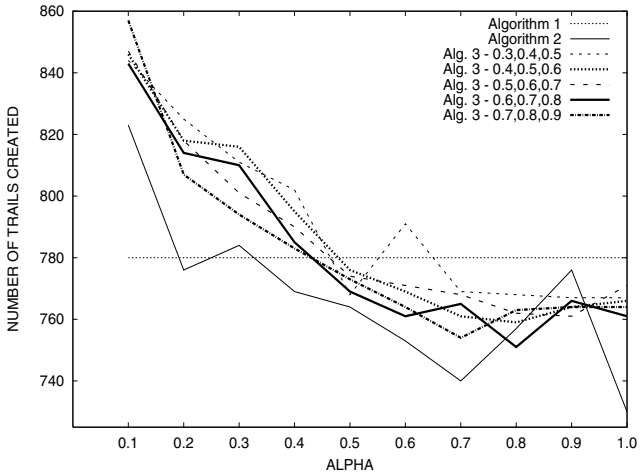


Fig. 5. Number of trails created in Network 1

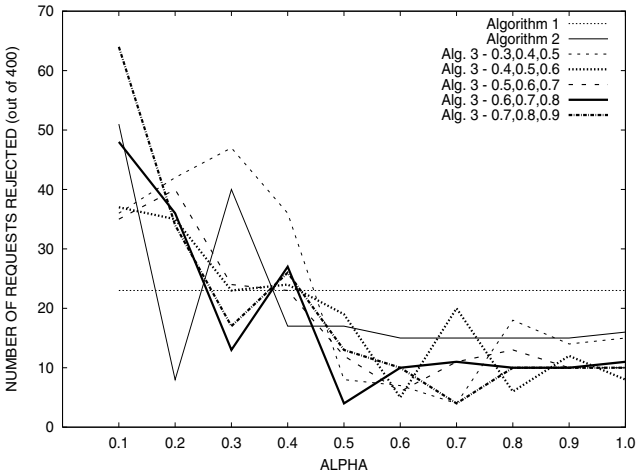


Fig. 8. Number of requests rejected in Network 2

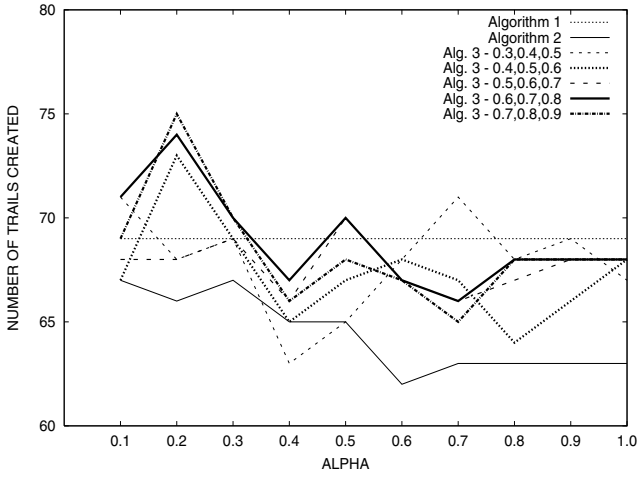


Fig. 9. Number of trails created in Network 2

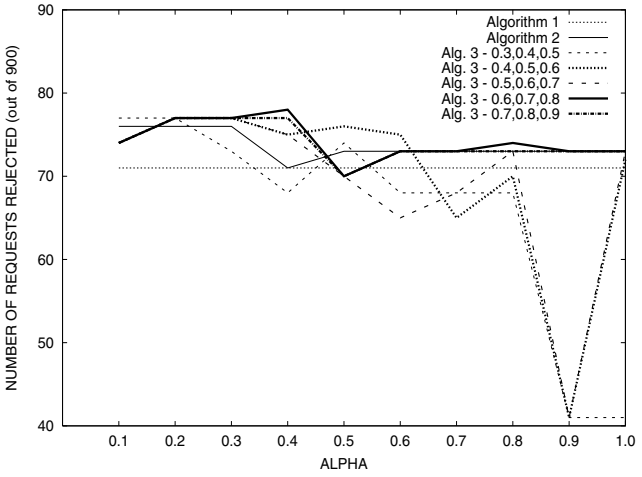


Fig. 12. Number of requests rejected in Network 3

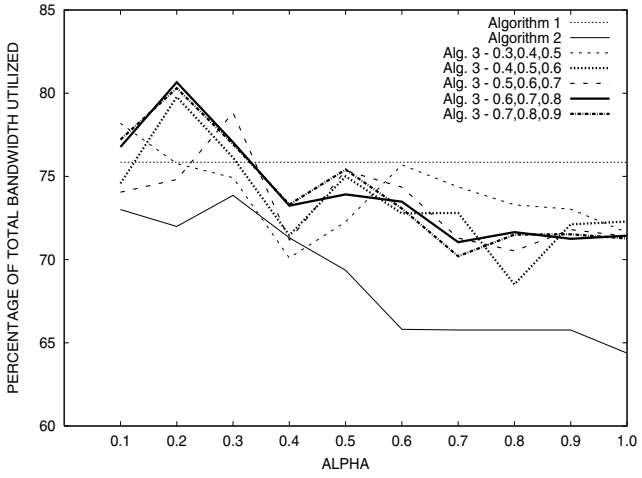


Fig. 10. Bandwidth utilized in Network 2

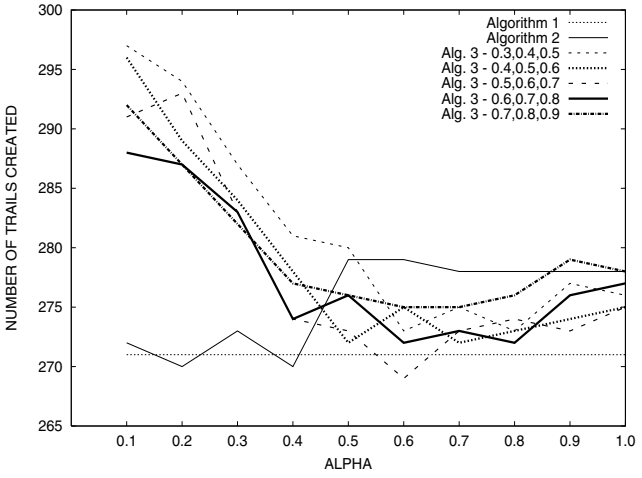


Fig. 13. Number of trails created in Network 3

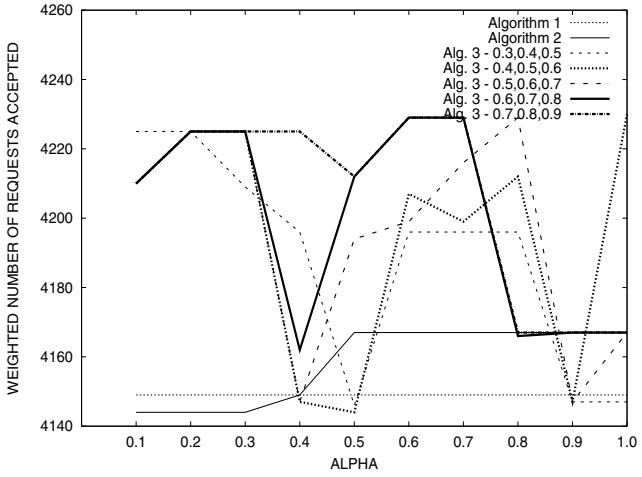


Fig. 11. Weighted number of requests accepted in Network 3

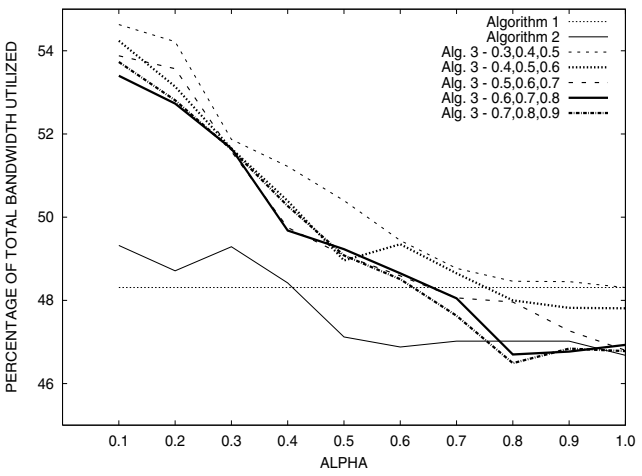


Fig. 14. Bandwidth utilized in Network 3

used. This is consistent for both Algorithm 2 and Algorithm 3. Though, there is a trend showing better performance for increasing values of $\alpha$, it is not uniform. We observed that this is a random behavior which depends on the network used and the service requests given in those cases.

Among the different combination of values for $\beta_1$, $\beta_2$ and $\beta_3$ in Algorithm 3, the combinations 0.5, 0.6, 0.7 and 0.6, 0.7, 0.8 and 0.7, 0.8, 0.9 are found to give better results for the four parameters mentioned above when compared to the other two combinations. But the difference among the different combinations is not very prominent.

From these results, it can be inferred that links and trails must be differentiated. Trails must be given higher preference than links so that already created trails are used as much as possible. This is achieved by means of assigning lesser weights to trails when compared to links. If the fraction $\alpha$ is too less, then it tries to use the existing trails always and this results in consuming more bandwidth and congesting some of the links. If the fraction $\alpha$ is very large, then new trails are created very frequently and this results in fragmentation of bandwidth. The values of 0.6 to 0.9 for $\alpha$ are found to give the optimal results for all the three networks. For the algorithms to be more reactive to the service requests made and the bandwidth available, the weights have to be dynamically varied. For trails, the criteria used is the necessity for breaking higher capacity containers in the SDH multiplexing structure. For links, the criteria used is the number of trails created in the links involved. Dynamically changing weights in this manner results in better usage of network resources and also more requests are accepted. The values of 0.6 to 0.9 for $\beta_1$, $\beta_2$ and $\beta_3$ are found to give better results when compared to other values. This is again due to the fact that lesser values result in more bandwidth utilization whereas very large values result in fragmentation of bandwidth.

Since these algorithms are centralized, the issue of scalability is a critical one. The time taken to compute a path in the 70 nodes, 103 edges network is always less than 1 second on a Pentium 4 (3 GHz) PC. Since this network is fairly large and time taken is very low, these algorithms can be used in provisioning systems. The shortest path computation is done using Dijkstra's shortest path algorithm whose running time is $O((E+V)\log V)$ when implemented using binary heap. Since it does not increase exponentially with increasing network size, these algorithms will be able to compute a shortest path for much larger networks in acceptable times.

## VI. Conclusion

In this paper, new path computation algorithms are presented for service provisioning in SDH networks. These algorithms differ from the conventional path computation algorithms in that they take into account the multiplexing structure defined by SDH which imposes restrictions on the allocation of bandwidth and the fact that higher order trails (logical connections) have to be established to support any bandwidth requirement. The assignment of weights to links and trails to find an optimal path is described.

The algorithms are evaluated for different sets of weights and their performance is presented. It is found that when the weights are dynamically varied according the state of the network and the resources available, the performance is better. Then a simple extension is proposed for finding dedicated protection paths for those working paths which need to be protected. But not every working path will require a dedicated protection path. Sharing of protection paths will result in the optimal usage of network resources.

The extension of these algorithms for supporting shared protection paths will be studied as a future extension. It should be able to take into account the inherent protection capabilities provided in SDH protection mechanisms like Multiplex Section Protection (MSP)[18], Multiplex Section - Shared Protection Ring (MS-SPRing)[18] and Subnetwork Connection Protection (SNCP)[18].

## References


[1] W. Fawaz et al., "Service Level Agreement and Provisioning in Optical Networks," *IEEE Communications Magazine*, Jan. 2004, pp. 36-43

[2] A. E. Ozdaglar and D. P. Bertsekas, "Routing and Wavelength Assignment in Optical Networks," *IEEE/ACM Trans. Networking*, vol. 11, pp. 259- 272, Apr. 2003.

[3] H. Zang, C. Ou and B. Mukherjee, "Path-Protection Routing and Wavelength Assignment (RWA) in WDM Mesh Networks Under Duct-Layer Constraints," *IEEE/ACM Trans. Networking*, vol. 11, pp. 248-258, Apr. 2003.

[4] X. Chu, B. Li and Z. Zhang, "A Dynamic RWA Algorithm in a Wavelength-Routed All-Optical Network with Wavelength Converters," *Proc. IEEE INFOCOM*, 2003, pp. 1795-1804.

[5] M. Alanyali and E. Ayanoglu, "Provisioning Algorithms for WDM Optical Networks," *IEEE/ACM Trans. Networking*, vol. 7, pp. 767-778, Oct. 1999.

[6] S. Janardhanan et al., "A Routing and Wavelength Assignment (RWA) Technique to Minimize the Number of SONET ADMs in WDM Rings," *Proc. HICSS*, vol. 2, pp. 1-10, Jan. 2006.

[7] H. Liu et al., "Distributed Route Computation and Provisioning in Shared Mesh Optical Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1626-1639, Nov. 2004.

[8] S. Das, "Topology Discovery and Path Provisioning in SONET Rings Using GMPLS," *Proc. WOCN*, 2006.

[9] G. Shen and W. D. Grover, "Performance of Protected Working Capacity Envelopes based on p-cycles: Fast, Simple, and Scalable Dynamic Service Provisioning of Survivable Services," *Proc. SPIE*, vol. 5626, pp. 519-533, Nov. 2004.

[10] M. Kodialam and T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," *Proc. IEEE INFOCOM*, 2000, pp. 902-911.

[11] C. Ou et al., "Survivable Virtual Concatenation for Data Over SONET/SDH in Optical Transport Networks," *IEEE/ACM Trans. Networking*, vol. 14, pp. 218-231, Feb. 2006.

[12] A. Gersht, S. Kheradpir and A. Shulman, "Dynamic Bandwidth-Allocation and Path-Restoration in SONET Self-Healing Networks," *IEEE Trans. Reliability*, vol. 45, pp. 321-331, Jun. 1996.

[13] H. Katz, G. F. Sawyers and J. L. Ginger, "SDH Management Network: Architecture, Routing and Addressing," *Proc. IEEE GLOBECOM*, 1993, pp. 223-228.

[14] N. Ansari et al., "QoS Provision with Path Protection for Next Generation SONET," *Proc. IEEE ICC*, 2002, pp. 2152-2156.

[15] A. Jukan and H. R. van As, "Service-Specific Resource Allocation in WDM Networks with Quality Constraints," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2051-2061, Oct. 2000.

[16] ITU-T Recommendation G.707, Network node interface for the synchronous digital hierarchy (SDH), G.707.

[17] ITU-T Recommendation G.703, Physical/electrical characteristics of hierarchical digital interfaces, G.703.

[18] ITU-T Recommendation G.841, Types and characteristics of SDH network protection architectures, G.841.