QoS Provision with Path Protection for Next Generation SONET*

Nirwan Ansari[†], Gang Cheng[†], Stephen Israel[±], Yuanqiu Luo[†], Jonathan Ma[±] and Li Zhu[†]

[†] Advanced Networking Laboratory
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
University Heights, Newark, NJ 07102

[±] OpenCon Communication Systems, Inc.
377 Hose Lane
Piscataway, NJ 08854

***Abstract*-We present features of next generation SONET, focusing particularly on path management. Factors such as QoS metrics and path protection are keys to realize automatic and dynamic path management. Two algorithms (the sequential algorithm and the parallel algorithm) to provide QoS and protection path in SONET are proposed and discussed in details. They balance the network load by circumventing heavily loaded links, and reduce the network resource consumption by selecting the shortest paths. Simulation results demonstrate that they are practical solutions for path management of next generation SONET.**

I. Introduction

SONET (Synchronous Optical NETwork) is a standard formulated by American National Standards Institute (ANSI). It defines optical carrier (OC) levels and electrically equivalent synchronous transport signals (STSs) for the fiber-optic-based transmission hierarchy [11]. SONET has been deployed in almost every carrier's network for about 20 years as a layer one transport mechanism to deliver voice traffic, and has proven to be very reliable in delivering voice service.

In response to the Internet boom, a turning point occurred over the past years: data surpassed voice traffic. The Internet boom changes the characteristics of service on SONET from voice traffic into data traffic. Voice traffic is delay sensitive, and therefore fixed bandwidth assignment is desired regardless of the actual usage. Data traffic is bursty, and therefore fixed bandwidth assignment creates enormous wastage of bandwidth. Static path assignment does not satisfy the real-time bursty data traffic requirement, thus leading to inefficient network resource utilization and bad service for customers. In this paper, we provide solutions for path management in SONET that contribute to the main feature of next generation SONET. In Section II, some limitations of today's SONET network are discussed. Solutions for these drawbacks comprising the basic functions of next generation SONET are presented in Section III. In Section IV, we focus on path management for next generation SONET. The purpose is to find the QoS guaranteed path with protection. Two practical algorithms to find such paths are proposed, and their effectiveness is demonstrated by simulation results.

II. Limitation of Today's Sonet

*** This work has been supported in part by OpenCon Communication Systems, Inc., the New Jersey Commission on Higher Education via the NJI-TOWER project, and the New Jersey Commission on Science and Technology via the NT Center for Wireless Telecommunications.**

As indicated above, one of the major problems with today's SONET is the inefficiency associated with transporting data traffic. The major limitations that SONET is unfit for Internet-based communications are:

- **Static provision:** In today's SONET networks, bandwidth is allocated between two end points. Bandwith allocated to a path cannot be reused for other purpose until the path is released, regardless of its usage. This static provisioning results in enormous wastage of bandwidth.
- **Low bandwidth utilization:** Today's SONET bandwidth allocation requires manual intervention. Network engineers usually over-engineer the network to leave a margin even during peak traffic periods. The average circuit utilization rate is very low, 5 to 10 percent in the access network, and 20 to 30 percent in the core. During a busy period, the peak utilization rate might reach 50 to 70 percent.
- **Long time to provision:** It takes days or weeks to provision a path from one end point to another end point in today's SONET network. The process is manual, error prone and very costly.

III. Next Generation Sonet

Growing popularity of new applications such as live videoconference require next generation SONET to find innovative solutions to meet the increasing demand for more bandwidth and provide the necessary quality of service on a per application basis. Next generation SONET must be a smart one with such basic smart functions as the followings:

- **QoS path provision:** Path provision in next generation SONET will be automated. SONET network will support protocols that enable it to discover the network topology and network resources of the incident links. In addition to automatic resources discovery, next generation SONET will have to support signaling protocols such as RSVP-TE (Resource reSerVation Protocol with Traffic Engineering extensions) [3] and CR-LDP (Constraint-based Routed Label Distribution Protocol) [1] to automate path set up and tear down. Such path is quality guaranteed with at least one QoS metric. In other words, the dynamic path can transmit diverse data traffic in response to all kinds of real-time requirements.
- **Efficient path protection:** Service protection in SONET transmission is largely concerned with providing redundancy to increase the overall availability of end-to-end paths. This is necessary because the net availability performance obtained by cascading a large number of network elements is incompatible with the expectation of many customers. Path protection contributes to network fault tolerance.

- **High bandwidth utilization:** Since data traffic is essentially bursty, allocating fixed bandwidth with a static path is highly inefficient. In order to reduce under utilization of bandwidth, next generation SONET network must support statistical multiplexing of traffic from data ports to increase bandwidth utilization through over subscription.
- **Migration strategy:** The network will evolve from SONET based network to full photonic network based on WDM (Wavelength Division Multiplexing). Next generation SONET network must be positioned to facilitate migration to full photonic network. It should provide such optical modules, mapping the traffic onto one of the wavelengths supported by WDM or DWDM (Dense WDM) [4].

IV. Path Management in Next Generation Sonet

Path management in next generation SONET is composed of three major components: link state information, path computation, and path protection. Link state information component collects information about SONET network topology, bandwidth availability, and other network resources availability. Extensions to network layer routing protocols such as IS-IS (Intermediate System to Intermediate System) or OSPF (Open Shortest Path First) [9] can be used for this purpose. In this paper, we focus on the latter two functions, with the purpose of providing QoS constrained path and path protection for customers. Providing path with QoS guarantee and with protection is a traffic engineering mechanism, which focuses on minimizing the network resource consumption and balancing the network load. QoS path and protection path provide major performance improvement for path management of next generation SONET, and are the basis to other smart functions such as dynamic bandwidth allocation and high bandwidth utilization.

A. QoS Path Provision and Path Protection

Packets entering into SONET by the edge nodes are routed to their destination based on the destination address. From routing point of view, the egress ports are STS-N level paths. To realize quality guarantee, each packet must meet its bandwidth, delay, and delay jitter requirements.

Service requirements have to be expressed as measurable QoS metrics such as bandwidth, delay, jitter, and loss rate. Different metrics may have different features. There are three types of metrics: additive, multiplicative, and concave [5]. Examples of additive metrics are delay, jitter, and hop-count; an example of multiplicative metrics is loss rate; bandwidth is an example of concave metrics.

Path computation involves identifying one or more paths through SONET network that satisfy a set of QoS parameters such as loss rate, delay, and delay jitter. Selecting a path that satisfies all the QoS parameters is an NP-complete problem [12]. Therefore, generally the path computation module simplifies the problem by searching for paths that can satisfy just one or two QoS parameters at the most.

Note that SONET is a TDM system with a time slot of $125\mu s$. The delay and delay jitter can be easily bounded by statistics of the data, and thus bandwidth should be the first consideration and has the highest priority among all QoS metrics in SONET.

Another characteristics of SONET QoS metrics is: the allocated bandwidth in SONET network is discrete with the basic granularity of VT1.5, STS-1 (OC-1), STS-3 (OC-3), STS-12 (OC-12), STS-24 (OC-24), STS-48 (OC-48), and STS-192 (OC-192). To overcome the drawback of today's SONET, bandwidth must be dynamically allocated according to this granularity. Customer's request imposes a minimum bandwidth requirement on the path that will ensure QoS guarantee. When data enter SONET, they carry the bandwidth requirement information, and report it to the edge nodes.

Since SONET network path is mainly in the core or backbone of the whole network, the protection path is necessary to minimize or eliminate data loss even in the worst case of working path failure. If the working path fails in the situation such as node dropping and link failing, the data traffic must be switched to the protection path to continue the data flow. Also, the two paths should be disjoint in order to guarantee that the protection path works well in the worst case if all links or nodes along the working path fail for some reasons. This capability of path protection is the main feature of next generation SONET. The path protection is to provision fault-tolerance. This protection is employed effectively on a network-wide basis with the goal of providing a high level of path availability as perceived by the customers.

B. QoS Provision with Path Protection Algorithms

Finding a path with protection is one application of DPP (Disjoint Paths Problem). A number of research results on DPP have been reported in the literature. In [10] the authors developed a distributed asynchronous algorithm of shortest disjoint paths. It minimizes the total length of the two paths between the source and the destination. Reference [6] suggests the idea of spanning tree-based distributed algorithm. Torrieri [14] established a set of short disjoint paths in a communications network by using adjacency matrix to construct the optimal paths set. Taft-Plotkin *et al.* [13] proposed a QoS-based maximally disjoint paths algorithm for precomputing paths.

All these reported algorithms address important issues of DPP. However, none of them present algorithms that are to find two disjoint paths between a pair of nodes on-demand such that both of the paths are guaranteed to satisfy the QoS requirement. In this section, we propose two algorithms to provide on-demand QoS path and protection path in SONET. Note that any two or more of hop-count, delay, delay jitter and loss rate in any combination as QoS metrics are NP-complete. The only feasible combinations are bandwidth and one of the other four (hop-count, delay, delay jitter, and loss rate) [15]. Although these four are all very useful metrics, we believe that for the majority of applications in SONET, delay is comparatively more important and practical than the others.

The bandwidth we are interested in here is the residual bandwidth that is available for new traffic. The bandwidth of a path is defined as the minimum of the residual bandwidth of all links on the path, or the maximal reservable bandwidth on the path. A path is feasible if the available or unreserved bandwidth of all links on the path is equal to or larger than the customer's requested bandwidth.

From the point of traffic engineering, a routing algorithm for SONET must balance the network load and limit the resource consumption [8]. The network load can be balanced by selecting the path with light traffic, while resource consumption can be reduced by restricting the length of the selected path. Our algorithms limit resource consumption by choosing the shortest path, and balance network load by pruning heavily loaded links.

Note that SONET is a TDM system with $125\mu s$ time slot, and the delay of a SONET path is proportional to the path length. Thus, the path with the shortest length achieves the least delay. In other words, our algorithms aim to find the shortest path with one protection path and with bandwidth guarantee. The following notations are adopted:

$W(N,L)$ ---- SONET network with node set N and link set L;
(s, t) ---- a node pair with source s and target t;
$P(s,\ldots,t)$ ---- a path with source node s and target node t;
b_i ---- available bandwidth of link i, $\forall i \in L$;
d ---- length of a path;
b ---- bandwidth of a path, $b=min\{ b_i \mid i \in P\}$;
b_r ---- bandwidth requirement for setting up a path.

1) *The Sequential Algorithm*

```
Sequential Algorithm

Input:  W(N,L), source s, target t, bandwidth requirement $b_r$,
        redo times $n_w$ and $n_p$
Output: 1, if two disjoint paths are found: working path $P_w(s,...,t)$,
              protection path $P_p(s,...,t)$
        0, if fail to get $P_w(s,...,t)$ and $P_p(s,...,t)$
begin
    for k = 0 to $n_w$
        for all link i in L
            if ( $b_i < b_r$ )
              remove link i from W(N,L);
            end if
        end for                                          // prune links
        $P_w(s,...,t)$ = Bellman-Ford(W(N,L),s,t);       // working path
        if (Reserve($P_w(s,...,t)$))                     // reserve $b_r$
            for all intermediate nodes j in $P_w(s,...,t)$
                remove node j and its connection links from W(N,L);
            end for
            for m = 0 to $n_p$
                $P_p(s,...,t)$= Bellman-Ford(W(N,L),s,t); // protection path
                if (Reserve($P_p(s,...,t)$))     // reserve $b_r$
                    return 1;
                else
                    updateinfor( );              // update information
                end if
            end for
            return 0;
        else
            updateinfor( );                      // update information
        end if
    end for
    return 0;
end

function: Bellman-Ford(W(N,L),s,t)
returns:  the shortest path P(s,...,t)
function: Reserve(P(s,...,t))
returns:  1, if successfully reserve $b_r$ along path P
          0, if fail to reserve $b_r$ along path P
function: updateinfor( )
returns:  update link state information
```

Fig. 1. Pseudo-code of the Sequential Algorithm

The sequential algorithm finds two disjoint paths from source node s to destination node t sequentially. It can be simply described as: handle the concave constraint (i.e., bandwidth) by first pruning out all links that do not satisfy the constraint, and then find the shortest path as the working path $P_w(s,\ldots,t)$; prune the links and intermediate nodes of the working path from the network topology and find the shortest path of the reduced network as the protection path $P_p(s,\ldots,t)$.

The sequential algorithm consists of the following major phases, and its pseudo code is shown in Fig. 1.

- *Pruning links:* before finding the working path, prune all links with available bandwidth less than b_r (customer's bandwidth requirement) from the network topology. Before finding the protection path, prune all intermediate links and nodes on the working path to ensure that the protection path is disjoint from the working path.
- *Finding the shortest path:* find the shortest path from s to t in the pruned network topology by Bellman-Ford algorithm. The selected shortest path can guarantee the shortest delay.
- *Reserving path:* reserve customer's requested bandwidth b_r along all links of the selected path (working path or protection path). Owning to the stale network information, the seemingly feasible path in Bellman-Ford algorithm may not afford b_r on all its links. This path cannot be reserved. In such cases, we must update the link state information and reconfigure the path using the new link state information.
- *Updating information:* update link state information, especially the information of available bandwidth, so that the next calculation is based on the new link state information.

The sequential algorithm terminates when it finds the two paths (working path and protection path) and successfully reserves b_r along them. If it cannot reserve b_r for the stale network information, this algorithm allows recomputing the working path at most n_w times and the protection path at most n_p times. The complexity of the Bellman-Ford algorithm [2] is $O(ne)$, where n is the number of nodes and e is the number of links in the network. The worst case computational complexity of the sequential algorithm is $O(ne(n_w + n_p))$, where n_w is the largest allowable "redo" trials to find the working path, and n_p is the largest allowable "redo" trials to find the protection path.

2) *The Parallel Algorithm*

Instead of finding and reserving the paths one by one, we can use the parallel algorithm to compute both paths simultaneously. The parallel algorithm can be described as follows: start from the source node s (i.e., path $P(s,\ldots,s)$), expand it to its incident links, then, all the paths from s to t can be reached; however, by pruning insufficient links, only the paths within which each link has equal or more available bandwidth than b_r are expanded. The candidate paths, *CandidateList* (all paths expanded from source s not reaching t), are sorted properly, such that the candidate path with the shortest length is selected and expanded first. Form the feasible paths, *PathList* (all paths from s to t which have sufficient bandwidth), choose the shortest one as the working path $P_w(s,\ldots,t)$, choose another one disconnected from the working path as the protection path $P_p(s,\ldots,t)$.

The parallel algorithm grows a path list (*PathList*), which contains feasible paths from s to t. The path list is ordered such that the shortest one is at the head. Another path list (*CandidateList*) includes the extended paths from s that do not reach t. All the extended paths that cannot guarantee the bandwidth requirement are pruned from the *CandidateList* before further expansion. The *CandidateList* is sorted such that the shortest extended path is at the head. When the extended path reaches t, remove it from the *CandidateList*, and move it into the *PathList*. All feasible paths are in the *PathList* when the *CandidateList* is empty.

The pseudo-code of the parallel algorithm is shown in Fig. 2, and includes the following major steps:

```
Parallel Algorithm

Input:  W(N,L), source s, target t, bandwidth requirement $b_r$, redo times $n_r$
Output: 1, if two disjoint paths are found: working path $P_w(s,...,t)$,
           protection path $P_p(s,...,t)$
        0, if fail to get $P_w(s,...,t)$ and $P_p(s,...,t)$
begin
    for $k = 0$ to $n_r$
      CandidatList←P(s,...,s);          // initialize CandidateList
      while(CandidateList is not empty)
          sort (CandidateList);
          get and remove first path $P_f$ from CandidateList;
          if ($P_f$ reaches t )
             insert $P_f$ into PathList;
             continue;
          end if                         // if reaches t, insert it into PathList
          while (incident links of $P_f$ not empty)
                for all incident links i
                   if ( $b_i < b_r$ )
                      continue;
                   end if
                   insert the extension path into CandidateList;
                end for                 // expand candidate path
          end while
      end while
      sort (PathList) ;
      $P_w(s,...,t)$ = first path in PathList;                 // working path
      $P_p(s,...,t)$ = finddisjoint ($P_w(s,...,t)$, PathList); // protection path
      if (Reserve($P_w(s,...,t)$ && $P_p(s,...,t)$))           // reserve $b_r$
         return 1;
      else
         updateinfor();      // update information
      end if
    end for
    return 0;
end

function: sort(PathList)
returns:   a path list such that the shortest path is at the head of the list
function: finddisjoint (P(s,...,t), PathList)
returns:   the shortest path in PathList which is disjoint from P(s,...,t)
function: updateinfor( )
returns:   update link state information
```

Fig. 2. Pseudo-code of the Parallel Algorithm

- *Expanding path:* remove and expand the first path in the *CandidateList* one step further to get all possible extended paths. Initially, the only path in the *CandidateList* is *P(s,...,s).*
- *Inserting path:* if the expanded path reaches *t*, insert it into the *PathList*, otherwise insert it into the *CandidateList*.
- *Sorting paths:* sort all paths in the *CandidateList*/*PathList* so that the first one is the shortest path. The first path in the *PathList* will be selected as the working path. The first one in the *CandidateList* will be expanded first.
- *Finding two disjoint paths:* in the *PathList*, let the first shortest one be the working path $P_w(s,...,t)$, and choose another one disjoint from the working path with the shortest length as the protection path $P_p(s,...,t)$.
- *Reserving:* reserve b_r along the two chosen paths simultaneously. When the link state information is stale, the working path or the protection path may not be really feasible, for one or more links on the paths may have available bandwidth less than b_r. In such cases, link state information must be updated.
- *Updating information:* update link state information, especially the information of available bandwidth, so that the next calculation is based on the new link state information.

This algorithm is similar to the A* search strategy widely used in Artificial Intelligence, but with some pruning constraints [7]. Here, links with insufficient available bandwidth are pruned before further expansion. The algorithm terminates when two disjoint paths with sufficient bandwidth are found and reserved simultaneously. Otherwise, it recomputes the *PathList* at most n_r times. The complexity is $O(n_r g^2 kh(h+1+log(kgh)))$, where *g* is the average number of connectivity of a node in the network*, k* is the number of paths in the *PathList, h* is the length of the protection path, and n_r is the largest allowable "redo" trials to find the paths if the network information is stale.

C. Simulation Results

The performance of the above two algorithms is evaluated in the SONET network with 200 randomly distributed nodes. It is a mesh architecture. Each link's total capacity (i.e., the reserved bandwidth and the unreserved bandwidth) is scaled into 1000 units. For example, if the link is OC-196, which is 9953.280M, one unit equals 10M. The initial network load is: 20%, 30% and 50% of the network resources (i.e., bandwidth) are used up, respectively. In other words, when the customer requests to set up a QoS path with protection, the SONET network has already used up 20%, 30% or 50% of the network bandwidth resources, respectively.

The requested bandwidth is uniformly distributed from 50 units to 500 units. The traffic loads are evenly distributed; this means that a new request selects with equal probability any pair of nodes as its source and destination. A request can be rejected either because such paths (one working path and one protection path) do not exist, or because the customer's bandwidth requirement cannot be reserved along the elected paths. The latter one can be caused by stale information.

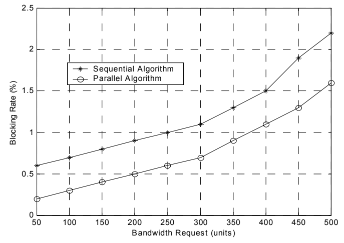
Simulation results with three scenarios are presented in Fig. 3. Case with 20% bandwidth used in the initial network load is the representative of an access network; 30% bandwidth is normally used up in a core network; case with 50% bandwidth occupied represents the busy period.

In the simulation, as the bandwidth requirement increases, the blocking rate in both algorithms increases gradually. With the same initial state and same bandwidth requirement, the blocking rate of the parallel algorithm is less than that of the sequential algorithm. This is attributed by the fact that that the parallel algorithm reserves the selected two paths simultaneously whereas the sequential algorithm needs to reserve the paths one by one. Also, the link state information changes more often in the sequential algorithm since it takes more time. The trade-off for decreased blocking rate in the parallel algorithm is the more space requirement for storing the parallel computation results of feasible paths.

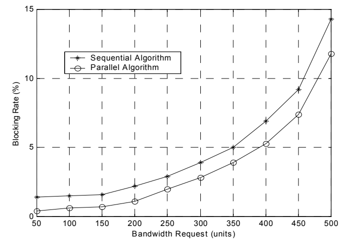
Overall, the blocking rate is proportional to the initial network load. When the initial network load is heavy, our algorithms perform poorly because they tend to allocate long expensive paths to guarantee the bandwidth requirement, thus penalizing late arrival requests.

Finding paths with multiple constraints is an NP-complete problem. Our algorithms first reduce the NP-complete problem to a simpler one by pruning, and then solve the new problem by an extended Bellman-Ford algorithm (the sequential algorithm) or a modified A* search algorithm (the parallel algorithm) to find the shortest paths. Both the proposed algorithms result in polynomial time computation.

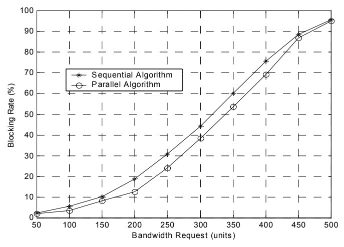
Simulation results show that both of them can find the QoS path with the necessary protection in normal cases as well as in very busy network condition. The trade-off between the two algorithms is the storage space and blocking rate: the sequential algorithm needs less storage space (to store the two selected paths) with a higher blocking rate while the parallel algorithm needs more storage space (to store all the feasible paths in the *PathList*) with a lower blocking rate.



(a) Initial network load 20% (access network)



(b) Initial network load 30% (core network)



(c) Initial network load 50% (busy period)

Fig. 3. Simulation Results: (a) Initial network load 20% (access network), (b) Initial network load 30% (core network), (c) Initial network load 50% (busy period)

V. CONCLUSIONS

In this paper, necessary features of the next generation SONET that will overcome today's limitation are discussed. Effective path management in the next generation SONET is the key point to realize traffic engineering and to incorporate smart functions such as dynamic bandwidth allocation and high bandwidth utilization. To guarantee QoS as well as to balance network resource utilization, our scheme is to provide bandwidth guaranteed path and one disjoint protection path. Two practical algorithms have been proposed. They minimize the resource consumption by choosing the shortest path, and balance the network load by selecting the lightly loaded links. The sequential algorithm modifies Bellman-Ford to find two disjoint shortest paths one by one; the parallel algorithm finds two disjoint paths by expanding candidate paths to the incident links with sufficient available bandwidth. Simulation results show that they work well in normal cases (normal access network and core network) as well as in busy period. The trade-off between the two algorithms is the storage space and blocking rate. To the best of our knowledge, our study is the first to evaluate the dynamic QoS path computation with path protection for next generation SONET.

REFERENCES

[1] J. Ash, M. Girish, E. Gray, B. Jamoussi, and G. Wright, "Applicability statement for CR-LDP,"*IETF Internet Draft <draft-ietf-mpls-crldp-applic-01.txt>,* July 2000.

[2] R. E. Bellman, "On a routing problem," *Quality of Applied Mathematics*, 1958, 16, pp. 87-90.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)— version 1 functional specification," *IETF RFC 2205*, Sep. 1997.

[4] D. Cavendish, "Evolution of optical transport technologies: from SONET/SDH to WDM," *IEEE Communications Magazine*, vol. 38, no. 6, pp. 164-172, June 2000.

[5] A. Fei and M. Gerla, "Smart forwarding technique for routing with multiple QoS constrains," in *Proceedings of IEEE GLOBECOM '00*, vol. 1, 2000, pp. 599 –604.

[6] K. Ishida, Y. Kakuda, and T. Kikuno, "A routing protocol for finding two node-disjoint paths in computer networks," in *Proceedings of IEEE International Conference on Network Protocols*, 1995, pp. 340-347.

[7] G. Liu and K. G. Ramakrishnan, "A* prune: an algorithm for finding k shortest paths subject to multiple constraints," in *Proceedings of IEEE INFOCOM '01*, vol. 2, 2001, pp. 743-749.

[8] Q. Ma and P. Steenkiste, "On path selection for traffic with bandwidth guarantees," in *Proceedings of 1997 International Conference on Network Protocols*, 1997,

[9] J. Moy, "OSPF version 2," *IETF RFC 2178*, July 1997.

[10] R. Ogier and N. Shacham, "A distributed algorithm for finding shortest pairs of disjoint paths," in *Proceedings of IEEE INFOCOM '89*, vol. 1, 1989, pp. 173-182.

[11] M. Sexon and A. Reid, *Transmission Networking: SONET and the Synchronous Digital Hierarchy*. Artech House, Norwood, MA, 1992.

[12] J. Song, H. K. Pung, and L. Jacob, "A multi-constrained distributed QoS routing algorithm," in *Proceedings of IEEE International Conference on Networks,* 2000, pp. 165-171.

[13] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-service routing using maximally disjoint paths," in *Proceedings of IEEE IWQoS '99*, 1999, pp. 119-128.

[14] D. Torrieri, "Algorithms for finding an optimal set of short disjoint paths in a communication network," *IEEE Transactions on Communications*, vol. 40, no. 11, pp. 1698-1702, Nov. 1992.

[15] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, Sep. 1996.