Traffic Engineering Algorithms Using MPLS for Service Differentiation

Richard Rabbat, Massachusetts Institute of Technology, 77 Massachusetts Avenue, # 1-107, Cambridge, MA 02139
Ken Laberteaux, Nirav Modi, John Kenney, Tellabs Research Center, 3740 Edison Lakes Parkway, Mishawaka, IN 46545

***Abstract*-This paper proposes an approach to Traffic Engineering that uses Differentiated Services (diffserv) and Multi-Protocol Label Switching (MPLS) to provide quantitative QoS guarantees over an IP network. An algorithm that determines QoS-constrained routes is proposed and a framework that uses such an algorithm is outlined. This framework removes the responsibility of QoS guarantees from the core nodes, thereby reducing their complexity.**

I. Introduction

This paper proposes a Traffic Engineering methodology that uses diffserv and MPLS to provide quantitative QoS guarantees over an IP network based on the PASTE architecture proposed in [8]. We provide mechanisms and algorithms that will enable a service provider or network operator to make resource reservations. The model uses a network-wide aware approach in making decisions. A Centralized Resource Manager (CRM) keeps track of an Autonomous System's (AS) resources and accepts connection requests by setting up Label Switched Paths (LSP) that will service that request with the necessary resources.

The paper presents an algorithm that deals with the changing resource needs of an already existing LSP. The architecture provides the ability to do traffic restoration due to resource failure by keeping a list of candidate paths at the CRM that can be used in that event.

II. The Need for Traffic Engineering

The Internet Engineering Task Force (IETF) is working to produce protocols to support differentiated Quality of Service (QoS) on IP networks. Currently the Internet treats data in the same manner, making no differentiation based on the source/destination or nature of the data. The goal, however, is to move to IP-based Internet applications that have specific requirements. For example, voice data is intolerant to excessive time delay or jitter. Conversely, the processing of a financial transaction may be tolerant to moderate delays.

A. Differentiated Services

Contributors of the IETF envision a next-generation Internet that can offer choices to customers and applications as to the treatment of their data. Towards this goal, the IETF has proposed the Differentiated Service (*diffserv*) architecture to enable IP networks to support multiple QoS needs [4].

Interior nodes and boundary nodes are grouped into an AS. An AS consists of a group of nodes administered by a single entity. A diffserv domain is defined in [4] as a "contiguous set of DS nodes that operate with a common service provisioning policy and set of PHB groups implemented on each node". For the purposes of this paper, we have assumed a single DS domain within each AS. Source/destination pairs may directly connect to a single AS (Fig. 1), or traverse more than one AS.

Boundary nodes are generally called ingress (for flows entering the network) and egress (for flows exiting the network) nodes. Interior nodes do not keep any per-micro-flow state information. Differentiated Services domains mark each packet of each flow with one of a small number of Differentiated Services Code Points (DSCP). All packets marked with the same DSCP are collectively called a Behavior Aggregated (BA). The term Per-Hop Behavior (PHB) is defined in [4] as "a description of the externally observable forwarding behavior of a DS node applied to a particular DS behavior aggregate."

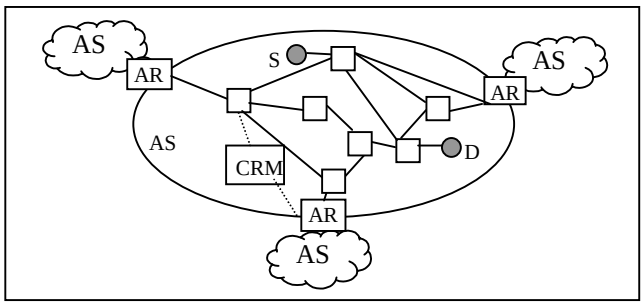

Fig. 1. ISP's AS and relationship to other networks.

The IETF has defined one PHB and a PHB group, namely the Expedited Forwarding (EF) PHB [7] and the Assured Forwarding (AF) PHB Group [6]. Examples of end-to-end service using the EF PHB include Virtual Leased Lines [7]. The "Assured Forwarding (AF) PHB group is a means for a provider DS domain to offer different levels of forwarding assurances for IP packets received from a customer DS domain [6]." Reference [4] outlines two other architectural building blocks, Traffic Classifiers and Conditioners and Network Resource Allocators. Traffic Classifiers and Conditioners protect interior nodes from resource starvation. Generally, the DS domain services flows under a Traffic Conditioning Specification (TCS) decided between the domain and its flow's sources. Violating flows that arrive at an ingress node will be dropped, shaped, or remarked as defined by the TCS.

B. Combining MPLS and Diffserv

An internal diffserv node treats all packets of a particular Behavior Aggregate identically. If a particular customer's flow shares the same DSCP with other flows, it is difficult to characterize the treatment of a customer's packets at an output port without knowing the number of other flows with the same DSCP. A customer's perceived end-to-end service will be a function of the service received at each node along its path, and thus is even harder to characterize.

One protocol which is capable of specifying, or "pinning", a flow's route that provides quantitative guarantees is Multi-protocol Label Switching (MPLS). MPLS is a protocol that

0-7803-6286-1/00/$10.00 (c) 2000 IEEE

can create tunnels between a pair of nodes. An IP packet traversing an LSP is prefixed with an MPLS header. When a router receives a packet with an MPLS header, it uses a separate MPLS forwarding table to determine the next hop.

In summary, MPLS will pin a particular route for a flow determined by a Network Resource Allocation process. MPLS will specify a next hop and diffserv will specify the treatment of a packet waiting to make that next hop.

C. The Centralized Resource Manager

A Centralized Resource Manager (CRM) is proposed to provide Network Resource Allocation. It becomes the primary contact when a customer wishes to initiate a new or expanded TCS. While the characteristics of a flow might change, the CRM acts only when a customer wishes to change its TCS. As an example, a customer may request the DS domain to support traffic between nodes *A* and *B* that will support 30 IP Telephony conversations. The CRM would be responsible for finding a path between A and B. While the flow's characteristics may change over time as calls are instantiated and torn down, the TCS would not change.

The CRM knows the network topology from the sysadmin or from the link state descriptors from each node running OSPF [10]. The CRM also maintains a database containing the unreserved resources at each output port of each node available for flows with quantitative QoS requirements.

As it creates a path for a flow with quantitative QoS requirements, the CRM follows the following steps:

1. When the CRM receives a request for TCS with a QoS requirement, it determines a set of possible routes, and picks a route that meets the QoS requirement.
2. Once a path has been identified, the CRM must assure that the flow follows this path. Appropriate MPLS label-switched label distribution should be used.
3. The CRM updates its database of available resources to reflect the allocation for the new flow.
4. The CRM signals the ingress router with the information needed to mark and police the new flow and informs the customer that it can send data into the network.
5. The CRM will continue to review OSPF link state advertisements to detect a link failure.

At initialization, the CRM will have knowledge of the resources available for quantitative TCS's. This database does not know of all resources available at each node, but only the resources reserved by the network operator for flows requiring quantitative guarantees.

III. QoS Routing: Building Feasible Paths

A. Open Shortest Path First (OSPF) and QoS Extensions to OSPF.

Interior Gateway Protocols (IGP's) are responsible for routing and route update. They route data by selecting the path with the least cost. OSPF is one such IGP. The most frequent implementation of OSPF allocates unit cost to all links, leading the cost function to pick the least number of hops as the shortest path. Problems arise when:

1. multiple streams converge on specific links or nodes, or
2. a traffic stream is routed through a link or node that lacks enough bandwidth to service it [3].

Extensions such as those proposed in [2] and [12] to support QoS routing based on OSPF have been proposed to take into consideration both aspects of the problem. QOSPF [12] is a proposed extension to OSPF to support QoS by flooding the network with the available and used link resources. The proposal makes routing decisions based on topology, link resources available and traffic requirements. The QOSPF framework uses RSVP [5] for signaling, allowing ingress routers to send the QoS requirements for incoming traffic in an RSVP PATH message. If a QoS route can be computed and a path reserved, an RSVP RESV message is sent back, reserving the resource and accepting the request. Another approach that we call QoS-OSPF [2] uses measurements to keep individual nodes' view of the network updated. QOSPF [12] bases its calculation on state information rather than measurement. RSVP is used to communicate QoS requirements to each node. Both QOSPF and QoS-OSPF choose a route by solving a shortest path algorithm using link costs dependent on available resources. Consequently, the time between runs of the Dijkstra (shortest path) algorithm is much smaller than in OSPF, creating a higher computational burden.

In the case of QoS-OSPF, the nodes determine their available resources by direct measurement, and then flood the network with this information. Since the amount of available resources changes rapidly, especially in the context of bursty Internet traffic, QoS-OSPF generates a substantial communication overhead. QoS-OSPF tries to minimize this overhead by using a trigger mechanism. Triggers at a node fire every period T, or when a link resource has changed by a given percentage. Another potential problem occurs when QoS-OSPF measures underutilized but allocated resources. These resources could be reallocated, causing packet drops once the client starts using the full VLL allocation.

Our approach addresses the above stated problems by making a CRM responsible for all resource allocations. The CRM relies on its view of the AS, the available resources and the reservations it has accepted to service QoS requests. Many issues of signaling overhead and delay are avoided.

B. Algorithms: Paths and Alternate Paths Pre-Computation

The CRM first determines the shortest paths between all ingress points, by running Dijkstra algorithms starting at each node. When they have finished running, the CRM knows the shortest path between any two nodes. The CRM then runs a series of Dijkstra algorithms with a modified topology. The number of algorithm runs for each node corresponds to the number of outgoing links from that node. The algorithm effectively tries to find other candidate paths that do not go through the first link of the shortest path for all source-destination pairs. In order to do this, the CRM sets the outgoing link cost to ∞ and runs a modified Dijkstra algorithm. This allows the CRM to find alternative paths starting at the nodes to the shortest path. Some nodes would obviously not have second candidate paths to a path $\boldsymbol{p}$; such is the case of some border routers that might only have one outgoing link. However, one or more nodes further along

that path –such as interior nodes- would find alternate routes for part of the path if they existed. The Dijkstra algorithm will only be run to recalculate shortest paths for the nodes use the link that we consider to be down. Although the worst-case order running time remains the same, in practice the Dijkstra algorithm ends faster. The *CAC* decision is explained in section IV.

IV. Connection Admission Control

A. Meeting Traffic Requirements

The problem of finding a path that satisfies several QoS constraints is NP-complete, but polynomial-time algorithms can be used if one assumes that the network service disciplines are rate proportional [9]. An example of such queuing disciplines is Weighted Fair Queuing [11].

Assume the aggregate traffic source is constrained by its leaky bucket parameters (σ, ρ) where σ is the maximum burst size and ρ is the average token rate. Assume a path ***p*** of *n* hops and link capacities C_i at hop *i*. Let the residual bandwidth on any link *i* be R_i. Let L_{max} be the maximal packet size in the network, $prop_i$ the propagation delay at hop *i* and *r* the amount of bandwidth requested ($\rho \le r \le R_i$) for all $i \in \boldsymbol{p}$. The following bounds based on work in [11] have been found to apply.

The maximum end-to-end delay is given by

$$D(p,r,\sigma) = \frac{\sigma + n \cdot L_{\max}}{r} + \sum_{i=1}^{n}\left(\frac{L_{\max}}{C_i} + prop_i\right) \quad (1)$$

Delay jitter is bounded by

$$J(p,r,\sigma) = \frac{\sigma + n \cdot L_{\max}}{r} \quad (2)$$

Buffer space requirements at hop *i* are given by

$$B(p,\sigma,i) = \sigma + i \cdot L_{\max} \quad (3)$$

B. Selecting the Path

Given those upper bounds, the algorithm needs to verify one or more of the following conditions to meet the respective bounds.

$$\begin{aligned} &D(p,r,\sigma) \le D_{requested}, \\ &J(p,r,\sigma) \le J_{requested}, \\ &B(p,\sigma,i) \le B(i)_{available}. \end{aligned} \quad (4)$$

On the other hand, there are instances where traffic needs to meet a certain delay requirement irrespective of the rate. The maximum bandwidth available on a path *p* is the minimum capacity of all links *l* of *p*. In this case, given the maximum delay requested $D_{requested}$, and $c_{\max}$ on path *p*, traffic rate and delay should meet the following conditions:

$$\begin{aligned} &\rho \le r \le c_{\max} \\ &D(p,r,\sigma) \le D_{requested} \end{aligned} \quad (5)$$

Assume a request for setting up a path between ingress router *ingress1* and egress router *egress1*. That request includes the bandwidth *r* needed, as well as one or more other constraints in terms of delay, delay jitter and buffer space requirement. The CRM maintains a structure ***P*** of the paths it considers for routing the traffic requested. The CRM first looks at the shortest path from *ingress1* to *egress1*. If any link l has $C_l < r$ then it is discarded.

If no link has been pruned, the CRM proceeds to the other constraints to check that they do satisfy the inequalities in (4). If so, the CRM sends a success response to *ingress1* with the chosen path, updates the node resources in its resource database. If a hop has been pruned or if the path does not meet traffic constraints, the CRM adds the available alternate paths between (*ingress1-egress1*) iteratively to ***P*** and performs the same checks. If at this point a feasible path has still not been identified, the CRM may explore other paths based on the new paths added to ***P***.

For each of the paths in ***P***, the CRM iterates over the hops *i* and selects alternate routes between (*ingress1*; *hop i*) on one hand then (hop i; *egress1*), while performing the CAC. For a path *p* identified by its hops: ingress1-node1-node2-node3-egress1, the algorithm tries to perform the CAC on the following paths, while adding them to ***P***:

- `ingress1-alternate route(s)-node1-node2-node3-egress1`
- `ingress1-node1-alternate route(s)-egress1`
- `ingress1-alternate route(s)-node2-node3-egress1`
- `ingress1-node1-node2-alternate route(s)-egress1`
- `ingress1-alternate route(s)-node3-egress1`
- `ingress1-node1-node2-node3-alternate route(s)-egress1`

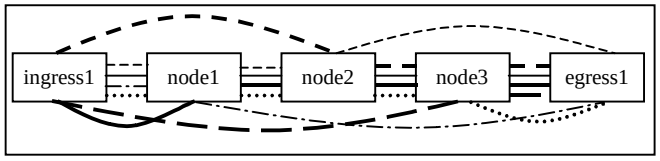

Fig. 2. Alternate Routes considered by the CRM.

The intuition behind this technique is that since the CRM rejects the path, it is because of either a lack of capacity or one of the other QoS constraints. A contributing factor is the lack of capacity at a certain link. By using this technique, the CRM is assured that while considering the alternate routes, it circumvents the overused link. Exploring the different possible routes may become time-consuming. A CRM may decide to stop investigating alternate routes after N different routes have been considered. It would try to make multi-trunk selections instead. The algorithm stops at the first acceptable path.

The CRM does not stop when it has checked the alternate routes; rather it forms possible alternate paths based on the contents of ***P***. It is important that the CRM does not consider a path more than once. To prevent this, it always checks the path *p* under consideration against the list in ***P***. The CRM also makes sure that the alternate paths do not lead to cycles. Therefore, ***P*** never contains walks.

C. The Diamond Problem

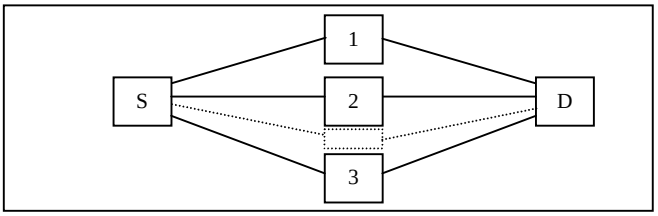

Fig. 3. Diamond Problem.

It is possible that the algorithm fails to discover possible QoS routes. This circumstance is due to a topology where multiple links exist between two nodes or multiple short paths. This leads the algorithm to use up the two links instead of all available links. The logic behind not identifying all possible routes is to limit the processing time required for candidate routes, as well as keep the number of candidate routes small when searching through them. In addition, the Internet backbone topology is a sparsely connected mesh. This kind of topology is more amenable to the solution proposed in this paper, since nodes have few outgoing links.

Consider Fig. 3. By building the list of paths and alternate paths between any two-node pair, using unit cost for all links, the algorithm will identify S-1-D and S-2-D as paths between S and D, but fail to identify S-3-D. The CRM will fail to use S-3-D. This scenario can be solved. The CRM may identify multiple routes at select nodes, by setting several link costs to infinity and running the modified Dijkstra discussed earlier. Those nodes are the routers with a large number of interconnections such as routers A, B and C on the map in Fig. 4. As an application to the diamond problem, after the CRM has identified the primary and alternate route, it will set both c_{s1} and c_{s2} to ∞. This identifies S-3-D as an alternate route. Whenever the algorithm finds a suitable path, it should set up an LSP on that route.

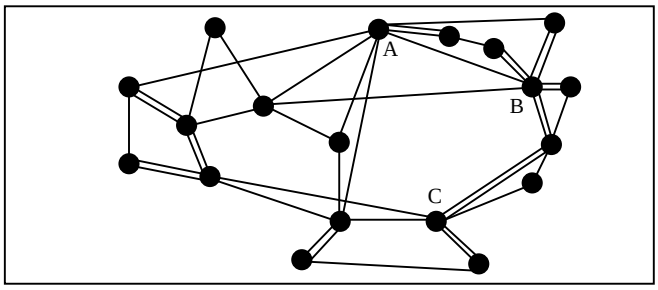

Fig. 4. MCI Internet Backbone Topology [9].

If all candidate routes have been exhausted, the CRM sends back a message to the ingress node, notifying it of its failure to pick a route or tries to make a multi-trunk selection that supports the requested traffic, as discussed in section IV.D.

D. Multi-Trunk Selection

This discussion assumes a network environment where the fine granularity of the micro-flows makes it possible to use multiple paths for routing the same aggregate. It also assumes that the ingress router knows how to route packets unto several trunks. The motivation for that is the need to keep packets that belong to the same (source, destination, port) tuple in-order, therefore on the same Label Switched Path. In that manner, packets of the same tuple need not be reordered.

In order to resolve a "false negative" or the unavailability of any one route to sustain the traffic requirement, the CRM may select multiple paths to route the traffic requested. In order to do so, the CRM should consider the candidate paths in $\boldsymbol{P}$ as explained in section IV.B.

In case we have a bandwidth requirement, the CRM may solve for the maximum bandwidth available from *ingress1* to *egress1*, by running a maximum flow algorithm. A graph *G'* constituted of the nodes and arcs in $\boldsymbol{P}$ will be considered when running the maxflow algorithm from *ingress1* to *egress1*. Though implementations of the maxflow algorithm perform at best at $\mathbf{O}(n^3)$ running time [1], the running involves a smaller number of nodes and arcs than the set of nodes and arcs in the whole AS. Let $C_{ingress1\text{-}egress1}$ be the available capacity from running the maxflow algorithm. If $C_{ingress1\text{-}egress1} < r$, the CRM denies the request; otherwise, it checks whether the other QoS constraints can be met by appropriate distribution of load on select paths considered as follows.

Equation (1) indicates that traffic experiences less delay by increasing rate used on path p, i.e., when $c_{\max}(p)$ is chosen on the path [9]. Therefore, for every path in $\boldsymbol{P}$, the CRM should try to route the maximum allowable capacity and check against the CAC constraints. Alternatively for a delay-constrained traffic, each path should verify (5) to carry a part of the traffic. Label Switched Paths are set up that each correspond to a traffic trunk as mentioned in [8]. $\boldsymbol{P}$ is always reset to an empty set at the end of the algorithm run, irrespective of success or failure.

The approach of pre-computing paths provides a fast solution as opposed to OSPF-based solutions. In addition, communication overhead is reduced using this approach since the CRM keeps track of all resource reservation information.

V. Changing Resource Requirements

An issue arises in dealing with an aggregate of flows, as is the case of *diffserv*, since micro-flows should be able to join or split from this aggregate dynamically.

A. Increasing the Requirement of a Traffic Trunk

This discussion deals with the question of providing a trunk with more resources if need be. Such a scenario happens when a corporate network needs to increase the aggregate so that it accommodates new micro-flows. To do so, it sends a request asking for more resources for the aggregate. It is the responsibility of the CRM to explore whether it can increase the resources assigned to the traffic.

We propose a scheme whereby the CRM tries to service the extra resource needed on any one of the Label-Switched Paths. Since the algorithm presented earlier may instantiate multiple traffic trunks for a particular traffic requirement, the CRM services the extra flow requirements by adding it to an existing traffic trunk. This saves complexity in terms of running the algorithm of section IV.B. In addition, the CRM does not have to publish a new LSP. The CRM adds the

traffic trunks considered to ***P***. If such increase in resource reservation is not possible -when the admission control rejects the flow-, the CRM uses the paths in ***P*** and executes the algorithm discussed to find another route. In case of success, the extra flow will be serviced independently.

B. Decreasing the Requirement of a Traffic Trunk

By decreasing the traffic requirement, the CRM should not try to move the flow *f* from the path *A* it was using to the path *B* that could theoretically support this traffic with the fewer requirements needed. This would lead to out-of-order packet delivery, which is not allowed [6]. Therefore, the CRM will only update its view of the resources available.

VI. RESTORATION: RESPONSE TO LINK FAILURE

In a data network, resource (node or link) failure may happen. It is the responsibility of the network to route the data over different routes in order to keep the flow of information undisrupted. The network may find out about a link failure through the OSPF updates. This section describes a method for dealing with link failure using OSPF as the notification agent for link failure. Other routing protocols may be used if the network administrator provisions a mechanism for making the CRM aware that a link has failed. When a link is no longer available, the OSPF update reflects the new network topology pinpointing the failed link.

At the CRM, this information is crucial for rerouting paths. A CRM receives an OSPF message, updates its view of the topology and knows of link failures. The CRM is responsible for rerouting all LSP's that were using the failed link.

Reference [8] states that different traffic trunks may have different priority. We assume that in the case of a link failure, the CRM selectively reroutes paths starting with the higher priority ones.

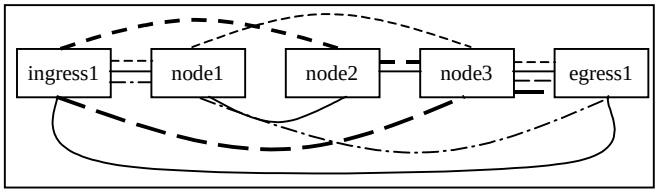

Fig. 5. Paths Investigated in Response to Failure.

Assume the link (node1-node2) is down. For a path ingress1-node1-node2-node3-egress1, the CRM first considers whether the traffic can use the alternate routes between node1 and node2 by considering the path ingress1-node1-alternate route(s)-node2-node3-egress. Furthermore, it adds it to ***P***. If this is possible, then a new LSP is setup. Fig. 5 shows the different paths that the CRM investigates.

- `ingress1-node1-alternate route(s)-node2-node3-egress1`
- `ingress1-alternate route(s)-node2-node3-egress1`
- `ingress1-node1-alternate route(s)-node3-egress1`
- `ingress1-alternate route(s)-node3-egress1`
- `ingress1-node1-alternate routes(s)-egress1`
- `ingress1-alternate route(s)-egress1`

If all these paths fail to sustain the traffic, the CRM would reconsider the paths in P, in a same fashion that sections IV.B and IV.D suggest. The CRM will examine the other trunks that used link (node1-node2) and perform the same rerouting methodology. If the event a path cannot be rerouted, the CRM should send a connection teardown notification at the ingress.

In the event where the CRM is successful at moving traffic from one trunk to another, one may consider the packets that have already reached the interior nodes. Since the previous LSP is no longer in effect, interior nodes may forward the packets in the network using IP routing by stripping the packets of the MPLS header. This reduces the number of TCP flows that go in congestion control.

VII. CONCLUSION

For any end-to-end guarantees to be sustained, controlling the flow of traffic through the network is critical. The use of connection admission, intelligent routing, and protection schemes makes end-to-end QoS a much more feasible prospect. Using the functionality of diffserv, and adding to it the route-pinning functionality of MPLS, we can satisfy quantitative QoS guarantees. The proposed Resource Manager offers a solution that removes complexity at the core, without losing control over network traffic. Knowledge of network status allows allocation of network resources, and thus helps in providing better QoS over IP networks.

REFERENCES

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

[2] A. Apostolopoulos, R. Guerin, S. Kamat, Orda, T. Przygienda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions", Internet Draft, April 1998.

[3] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering over MPLS", Internet Draft, June 1999.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[5] R. Braden, L. Zhang, S. Berson, S., Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC 2205, September 1997.

[6] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

[7] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB, RFC 2598, June 1999.

[8] T. Li and Y. Rekhter, "Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)," RFC 2430, October 1998.

[9] Q. Ma and P. Steenkiste, "Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks," Workshop on Network and Operating Systems Support for Digital Audio and Video, Cambridge, England, July 1998.

[10] J. Moy, *OSPF: anatomy of an Internet routing protocol*, Reading: MA, Addison-Wesley Publishing Company, 1998.

[11] A. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks, Ph.D. dissertation, MIT, Cambridge, MA, February 1992.

[12] Z. Zhang, C. Sanchez, B. Salkewicz and E. Crawley, "Quality of Service Extensions to OSPF or Quality of Service Path First Routing", Internet Draft, September 1997.