Link State Routing Extensions for the Enhancement of QoS Signaling Protocol

Ting-Chao Hou, Yun-Cheng Lin and Wei-Chu Lai
Dept. of Electrical Engineering
National Chung Cheng University
Chia-Yi 62100, Taiwan
05-2720411 33201
tch@ee.ccu.edu.tw, enq@alumni.ccu.edu.tw and opq@alumni.ccu.edu.tw

Abstraction

The popularity and usefulness of the Internet have been growing exponentially in the past several years. Provisioning an Internet Service Provider (ISP) backbone network for intra-domain IP traffic is a big challenge. At the same time, it is urgent for ISPs to provide Quality of Service (QoS) to customers for guaranteeing the loss, delay or bandwidth requirement of data transmission. In this paper, we investigate the extensions to the link-state routing protocol for QoS routing (constraint-based link-state routing), the concept of traffic engineering, and the interaction between QoS signaling protocols and constraint-based link-state routing. We adopt a practical viewpoint in a way that mimics the actual structure in a router to propose the constraint-based link-state routing architecture and discuss the general operations within this architecture. By incorporating a simple signaling protocol, the complete actions regarding QoS routing in the traffic-engineering domain are better understood.

Keywords

Routing Protocol, OSPF, QoS Routing, Traffic Engineering

1. Introduction

The popularity and usefulness of the Internet have been growing exponentially in the past several years. Provisioning an Internet Service Provider (ISP) backbone network for intra-domain IP traffic is a big challenge, particularly due to rapid growth of the network and user demands. The capability and capacity of current network may be considered insufficient to meet the current demands. At the same time, it is urgent for ISPs to provide Quality

of Service (QoS) to customers for guaranteeing the loss, delay or bandwidth requirement of data transmission. All of these issues point to the importance of Multi-Protocol Label Switch (MPLS) and traffic engineering, which make more efficient use of existing network resources.

In an MPLS network, packets are transported through the Label Switched Path (LSP) pre-established by the signaling protocol, which co-works with the routing protocol. MPLS increases the performance of packet forwarding through a network by using information contained in labels inserted between the Layer 3 header and the Layer 2 header of IP packets. On the other hand, traffic engineering addresses the issues of bandwidth provisioning and performance optimization in Internet backbones. Combined with traffic engineering, MPLS (or MPLS-TE) can help network providers to make the best use of available resources and reduce costs. Extending the conventional Open Shortest Path First (OSPF) routing protocol to support QoS routing is recommended as an appropriate method for realizing MPLS-TE signaling protocol [1].

OSPF is a widely used link-state routing protocol [2] [3] [4]. Each OSPF router distributes link-state routing information with all other routers belonging to a single Autonomous System (AS). Thus each participating OSPF router maintains an identical link-state database describing the AS's topology. Each individual piece of this database is a particular router's local state. In conventional OSPF routing protocol, the metric of a path is often just set as a constant without taking any dynamic information for QoS into account. Therefore, these metrics from source to destination are useless for supporting QoS routing. The QoS signaling protocol such as Resource Reservation Protocol with traffic engineering (RSVP-TE) [5] currently used in the MPLS environment merely make use of the next-hop information gathered by the Layer 3 routing protocols. As such, they have to check the information of available resource one by one with the resource manager of each router along the entire path typically.

OSPF Extensions for QoS routing mechanisms (or Constraint-based SPF, CSPF) [6] is a set of additions to the conventional OSPF routing protocol to support QoS in IP network. The additions include the metrics required to support QoS, the extension to the OSPF link-state advertisement mechanism to propagate updates of QoS metrics, and the modifications to the path selection to accommodate QoS requests. Each router running CSPF can obtain the information needed to compute QoS paths and select a path capable of meeting the QoS requirements. The OSPF with traffic engineering (or OSPF-TE) [7] is based on CSPF, added with some modifications to support traffic engineering over the MPLS environment.

In this paper, we investigate issues regarding realizing the QoS signaling protocol with the CSPF routing mechanism and the performance improvement for QoS flows. We discuss

the structure of a typical link-state routing protocol implementation on a widely used network simulation tool - Network Simulator 2 (*ns-2*). Then we propose some modifications to this architecture to support Constraint-based link state routing for QoS flows.

The organization of this paper is as follows: In Section 2, we provide the basic concept of the conventional link state routing protocol. In Section 3, we first describe the significant issues of QoS routing and traffic engineering. Then we discuss CSPF, which is used to increase the network utilization, and performance for QoS network and the QoS signaling protocol is described briefly to show how these protocols cooperate with traffic engineering routing mechanisms. In Section 4, we investigate a link-state routing protocol that had been implemented on *ns-2*. Section 5 presents the simulation environment of the simulation, the topology, the traffic models and the benefits from extensions to the conventional link-state routing protocol. The conclusions of this paper are stated in chapter 6.

2. Conventional OSPF

The OSPF routing protocol was developed by the OSPF working group of the Internet Engineering Task Force. It has been designed explicitly for the TCP/IP Internet environment. OSPF also provides for the authentication of routing updates, and utilizes the functions of IP when sending or receiving the updates. In addition, much work has been done to produce a protocol that responds quickly to topology changes, yet involves small amounts of routing protocol traffic.

2.1 The basic concept of OSPF

OSPF is a link-state routing protocol as opposed to Routing Information Protocol (RIP), which is a distance-vector routing protocol. A link is just another word for router interface, so OSPF could be called an interface-state routing protocol. The term state refers to the "UP" or "DOWN" of the interface, the address of the interface and the associated metric for an active link.

The metric of link-state indicates the cost of transmitting packets on the various links. OSPF doesn't even define the units of the link cost. That is configurable by the network administrator.

OSPF exchanges link state information with other OSPF routers. Instead of informing other routers what node they can reach and what the distance is as the RIP does, OSPF routers inform others of the state of their network interfaces, the networks these interfaces are attached to, and the cost of using the interfaces.

Obviously every router has a different link state from every other router at the initial

stage. Each router's link state can be referred to as a local link state. The router distributes its local state throughout the Autonomous System by flooding. These local link states are propagated throughout the OSPF network until every OSPF router has a complete and identical link-state database.

In a link-state routing protocol, each router maintains a database describing the Autonomous System's topology. This database is referred to as the link-state database. Each participating router has an identical database. Each individual piece of this database is a particular router's local state.

Once every router has received all the local link states, each router will use the Dijkstra algorithm with the information from the subset of the link state database. Each router can build a tree with itself as the root and the branches representing the shortest routes to all the subnets belong to one AS. Each OSPF router will use the shortest path tree to build its routing table.

The details of OSPF can be found in [2] and [3].

3. QoS Routing and OSPF Extensions

The need for QoS [8] capabilities in the Internet comes from the fact that best-effort service does not meet the needs of many new applications, which require some degree of resource assurance in order to operate effectively. Diverse customer requirements also create a need for service providers to offer different levels of services in the Internet.

Moreover, in this competitive business, service providers must balance two conflicting goals. First, they must meet the customer's expectation of guaranteed Service Level Agreements (SLA) for QoS. Second, they must manage network resources well to reduce the cost of provisioning the services.

3.1 Traffic Engineering and QoS

Traffic engineering is concerned with the performance optimization of operational networks. Its main objective is to reduce congestion and improve resource utilization across the network through carefully managing the traffic distribution inside a network. Over the past few years, traffic engineering has become an indispensable tool in the performance management of large Internet backbones. It aims to improve network performance through optimization of resource utilization in the network. One important issue that we need to address is the objective of the optimization.

Typically, the optimal operating point is reached when traffic is evenly distributed across the network. With balanced traffic distribution, both queuing delay and loss rates are at their lowest points.

Obviously, these objectives cannot be achieved through destination-based IP routing to make possible such optimization. In traffic engineering, the constraint-based routing mechanism is a key component. Advanced route selection technique is used to calculate traffic paths based on the optimization objectives. To perform such optimization, the constraint-based routing often needs extensive information on topology and traffic demands.

3.2 The capabilities of QoS Routing

In a constraint-based routing algorithm, any changes of link state in the network topology must be monitored. A constraint-based routing algorithm relies on reasonably accurate information about traffic demands of users. In some scenarios, the demands may have to be estimated based on traffic measurement. Many types of statistics are derived directly from the routers itself, such as traffic loads or latencies on links. For improving asset utilization as example, it is necessary to know the traffic load (or available bandwidth) of each link.

Central to a traffic-engineering system is the constraint-based route computation mechanism in a router, which calculates routes based on QoS traffic demands. In current IP routing, the shortest-path computation is used to select routes based on the weights assigned to links. For constraint-based route computation, however, the route selection must be subject to multiple complex constraints; for example, resource optimization objectives or latency constraints. For this reason, routing for traffic engineering is often referred to as constraint-based routing. We describe these additions to the conventional OSPF routing protocol for traffic engineering in the following section.

3.3 OSPF extensions to support QoS routing mechanism

The process of selecting a path that can satisfy the QoS requirements of a new flow relies on both the knowledge of the flow's characteristics and information about the available resources in the network.

3.3.1 Metrics

In general, the network prefers to select the "cheapest" path among all paths suitable for a new flow. The quality of OSPF routing depends highly on the choice of weights. Nevertheless, as recommended by Cisco [9], these are often just set inversely proportional to the capacities of the links, without considering any QoS information.

On the other hand, for the QoS variant requirements, the path selection process involves several kinds of metrics [6]. We assume that most QoS requirements are the capacities of links, i.e., the guaranteed bandwidth. Since for a link to be capable of accepting a new flow with the given bandwidth requirements, at least that much bandwidth must be still available on the link. Thus, the variation of one link’s available bandwidth needs to be ad-

vertised as part of extended LSAs exchange, so that accurate information for QoS routing is available to the path selection algorithm. Of course, when selecting a path for a delay sensitive request from applications such as Voice over IP (VoIP) or Video on demand (VoD), link propagation delay can be used.

Hop-count is also a measure of the path cost to the network. A path with a smaller number of hops is typically preferable, since it consumes fewer network resources. As a result, the path selection algorithm will attempt to find the minimum hop path capable of satisfying the requirements of a given request. Hop-count is a metric that does not cause changes to existing LSAs, and it is only used implicitly as part of the path selection algorithm.

3.3.2 LSAs Distribution

When the link-state information of one QoS router needs to be distributed to the rest of the network nodes is another important issue. Ideally, routers should have the most current view of the bandwidth available on all links in the network, so that they can make the most accurate decision on which path to select. Unfortunately, this will cause very frequent updates, which is neither scalable nor practical. In general, there is a trade-off between the protocol overhead and the accuracy of the link state information that the QoS path selection algorithm depends on.

3.3.3 Constraint-Based Routing Calculation

The routing table calculation for constraint-based routing can be performed in two different modes: pre-computation or on-demand. In the pre-computation mode, route computation is performed for all routes periodically with current information. Moreover, in the on-demand mode, a route computation is triggered for each new request.

There are three calculation methods proposed in [6]: Bellman-Ford (BF) based pre-computation Algorithm, on-demand Dijkstra Algorithm, and pre-computation using Dijkstra Algorithm. The BF shortest path algorithm is adapted to compute paths of maximum available bandwidth for all hop counts. It is a property of the BF algorithm that after *h*-th iterations, it identifies the optimal path between the source and each destination, among paths of at most h hops. In other words, the cost of a path is a function of the smallest available bandwidth among all links of the path. However, because the BF algorithm progresses by increasing hop count, it essentially provides the hop-count of a path as a second optimization criterion.

The pre-computation and on-demand Dijkstra algorithm for traffic engineering is a modified version of the Dijkstra shortest path algorithm. The benefit of using Dijkstra algorithm for all destinations and bandwidth values is a greater synergy with existing OSPF im-

plementations. However, in the practical networking scenarios, the BF algorithm offers an efficient solution to the shortest path problem, one that often outperforms the Dijkstra algorithm. The discussion regarding the computational complexity between the two algorithms can be found in [11]. Because the BF-based algorithm is similar to the Dijkstra algorithm and the details of BF-based algorithm can be found in [6], we only describe the Dijkstra algorithm below.

The Dijkstra algorithm to compute all "best" paths is to consecutively compute shortest path spanning trees starting from a complete graph and removing links with bandwidth less than the threshold used in the previous computation.

In the on-demand mode, the Dijkstra algorithm generates the shortest path tree similar to the pre-computation mode, but only at the level of the request bandwidth and finished as the destination node is added in the tree. Once shortest path tree is formed, the signaling protocol can look up the destination node in the tree to get the explicit node list by using the "previous" pointer to trace backward to the root.

The following section will present a general signal protocols used for MPLS: RSVP-TE and how these protocols work with OSPF-TE.

3.4 MPLS and RSVP-TE

The technique that MPLS [8] uses is known as label switching. A short, fixed-length label is encoded into the packet header and used for packet forwarding. When a label switch router (LSR) receives a labeled packet, it uses the incoming label in the packet. With label switching, the path that a packet traverses through, called the label switched path (LSP), has to be set up before it can be used. The LSP is pre-established by the signaling protocol that utilizes the routing protocol. We describe the signaling protocols next.

The RSVP [10] protocol is used by hosts to communicate service requirements to the network and by routers in the network to establish a reservation state along a path. The RSVP protocol was designed to be an add-on protocol to the existing IP protocol suite and it is used to establish a resource reservation between a sender and a receiver. The decision to select a path for a flow is done separately by routing; the RSVP process simply consults the forwarding table and sends the RSVP messages accordingly.

The RSVP-TE [5] protocol extends the original RSVP protocol to perform label distribution for MPLS and support explicit routing. The explicit routing object encapsulates a concatenation of hops that constitutes the explicitly routed path; the procedures are shown in Fig. 3-1. One goal of the explicit routing is to optimize the utilization of network resources and enhance traffic-oriented performance characteristics. As the scale of network increases and the number of request grows, the explicit routing of RSVP-TE without Constraint-based SPF will be too complex to specify administratively. If the explicit routing mechanism of

RSVP-TE can derive the information from CSPF where the link-state database is based on QoS and policy requirement, the node list needed by explicit routing will automatically be generated by a suitable entity.

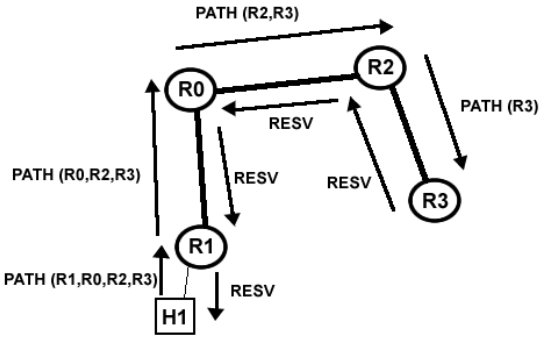


Figure 3-1: Explicit routing operation between nodes.

When the OSPF-TE is supported in the router, the LSAs for QoS are exchanged by OSPF-TE; so, the routing table carries the resources information of all links. For these reasons, the RSVP-TE agent will know whether it has enough resources to the destination derived from routing table with QoS parameter. The system model with CSPF on a RSVP-TE capable router is shown in Fig. 3-2.

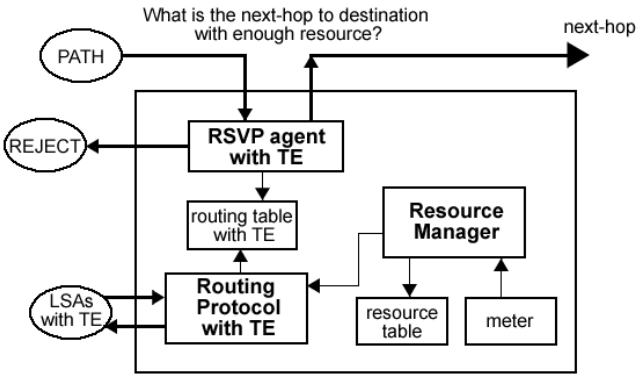


Figure 3-2: Basic RSVP-TE operations on one router running OSPF-TE.

3.4 The Overall Model in Traffic Engineering

For the edge router in an MPLS domain, when the MPLS controller needs to create an LSP before transmission, it triggers the signaling protocol such as RSVP-TE or CR-LDP, then the subsequent operations are the same as the descriptions mentioned previously. The overall model and the procedures are shown in Fig. 3-3, which summarize the architecture and procedures in realizing traffic engineering in a router.

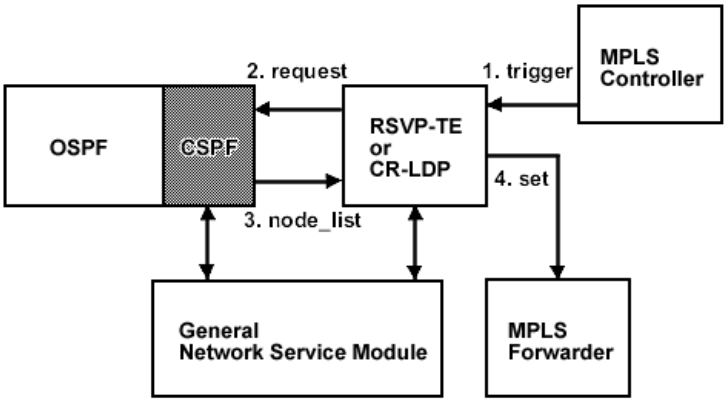


Figure 3-3: The relationship of protocols in an MPLS-TE system.

4. The Link-State Routing Protocol in ns-2

The ns-2[1] is a publicly available discrete event simulator targeted at networking research. The *ns-2* provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. It is an object-oriented simulator, written in C++, with an Object-Tcl (OTcl) interpreter as a front-end. The official user guide of *ns-2* can be found in [16].

4.1 The Module of Link-State Routing Protocol

The ns-2 is an object-oriented and module-based Network Simulator tool. This conforms to the actual system of routers in a practical network. We show the essential correlation of modules in Fig. 4-1

- Node Module: It is the infrastructure such as receive and transmit capability of one node.
- Link Module: It manages the information and status of all links to others.
- Queue Module: It represents the operation of each link's queues.
- rtObject Module: This is the general routing component such as modification to the routing table and the interface of some routing protocols to this node.
- LS Module, Static Module and DV Module: These stand for Link-State Routing Module, Static Routing Module and Distance Vector Module respectively. All these are different routing mechanisms, and we describe the details of LS module in the following sections.
- rtModel Module: It can control the link status (e.g. the "up" or "down", cost and

[1] The source code of ns-2 is available at http://www.isi.edu/nsnam/ns/

what kind of queue the link attaches to).

- RouteLogic Module: It gets routing table from rtObejct or sets specific routing mechanism to rtObject.

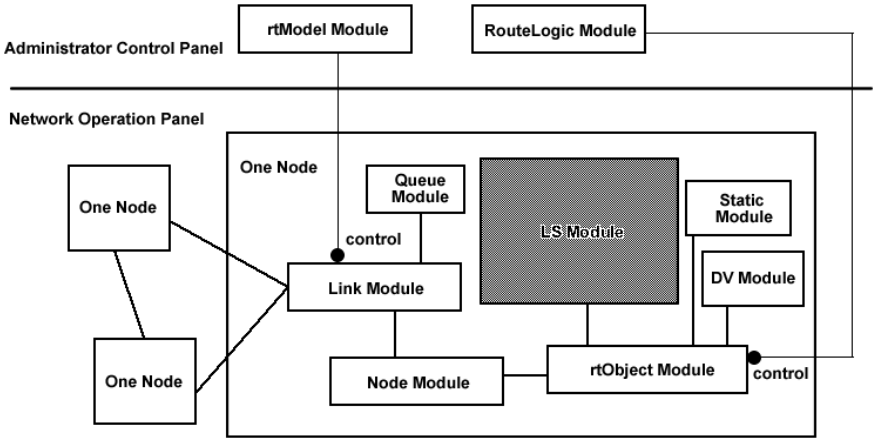


Figure 4-1: Basic correlation of modules in NS2.

The core components of the Link-State Routing Protocol are shown in Fig. 4-2. These are all the chief components of a typical link-state routing protocol, so we can understand the practical system architecture by referring to it. We describe each component as following:

- MyNode: It points to the Node Module used for the transmission and receiving of packets.
- PeerIDList: This records the information of neighbors running the identical link-state routing protocol. When the flooding mechanism is invoked, the link-state routing protocol must know what nodes need to send the update packets to. In addition, it must keep these neighbors in touch. So if any neighbor of this node crashes or is activated, the PeerIDList must be refreshed and this information needs to broadcast to others.
- LinkStateList: A link list that records all the LSAs received from other nodes or generated by this node.
- LinkStateDatabase: When the shortest path tree calculation is triggered, the shortest path tree information is recorded in this component.
- Routing table: Upon one node is inserted into the shortest path tree, this node information will be added into the routing table component.
- messageCenter: It can classify the type of received packets and generate the

Link-State packets to be transmitted out to other nodes.

- ackManager: If a packet needs to be sent out, it must register this record at the ackManager in expectation that the corresponding acknowledgement packet will return later.
- lsaHistory and tmpHistory: These are used to check the duplication of received packets. If the packet is a duplicate, it will be dropped and will not be flooded out.

The details regarding the Dijkstra's shortest path tree algorithm can be found in most network-related books, including [3].

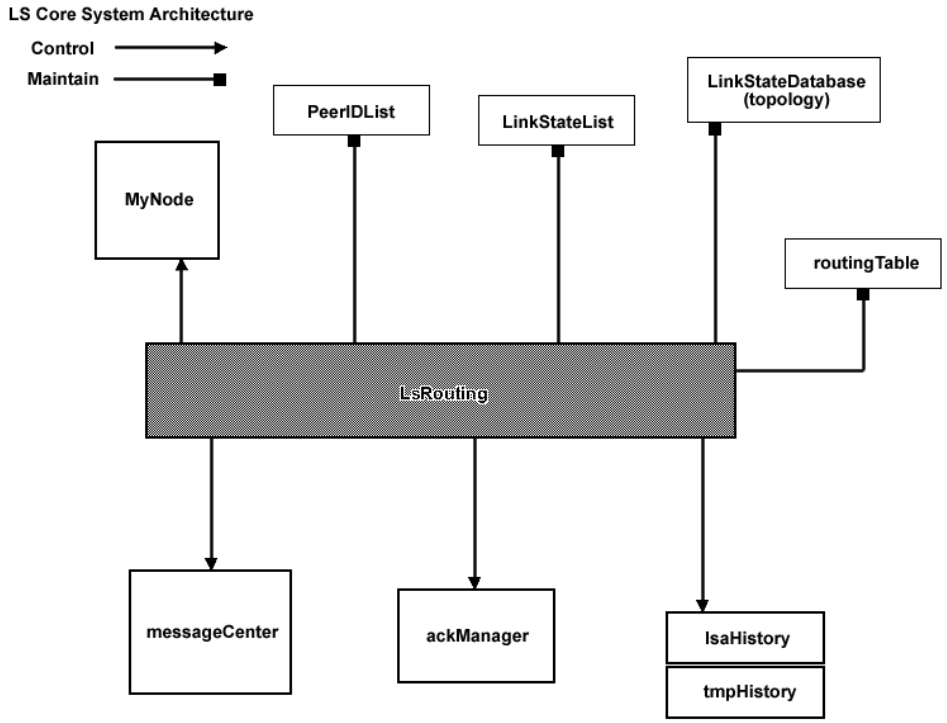


Figure 4-2: The link-state routing's architecture.

4.2 Extensions to the Link-State Routing Protocol

The flow chart of Dijkstra based QoS routing is shown in Fig. 4-10.

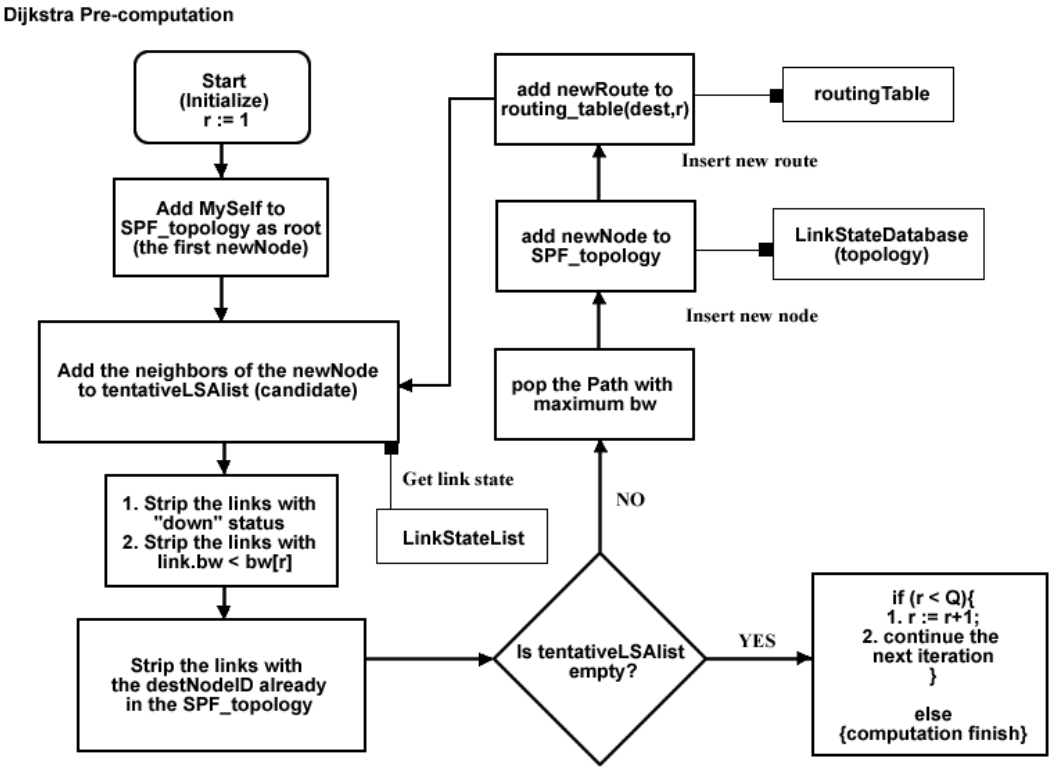


Figure 4-3: The flow chart of Dijkstra based QoS routing.

We create a simple signaling mechanism to test our QoS link-state routing protocol. First, the source node uses the destination and required bandwidth as parameters to get the explicit node list from the QoS link state routing protocol. Second, it passes the flow_id, next_hop and bandwidth to the resource manager, which uses them to update the forwarding table. Third, it sends the node list to the next_hop specified at the top of the node list. The second node then pops the first hop from the node list, passes the information to the resource manager, and then sends the node list to the next_hop node. The process continues until the node list is empty. Finally, the source node sends the data with the flow_id set previously. The diagram of this signaling mechanism is shown in Fig. 4-4. A node's viewpoint of this mechanism is shown in Fig. 4-5. This mechanism corresponds to the on-demand mode explained in section 3.3.3.

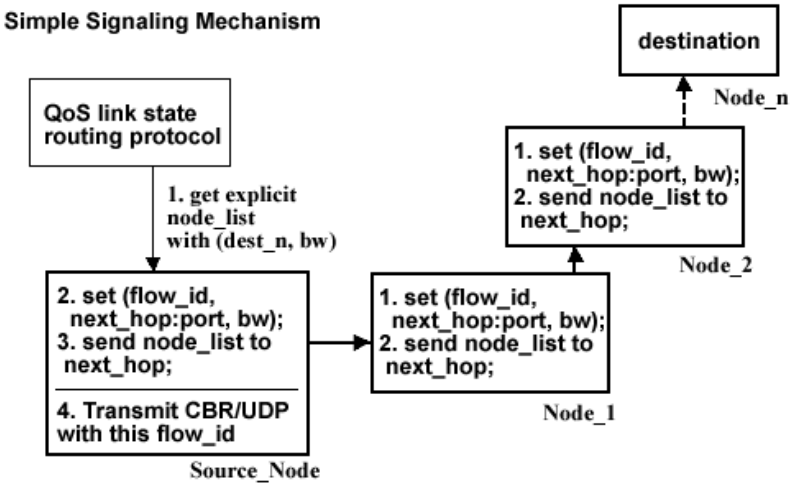


Figure 4-4: A simple signaling mechanism.

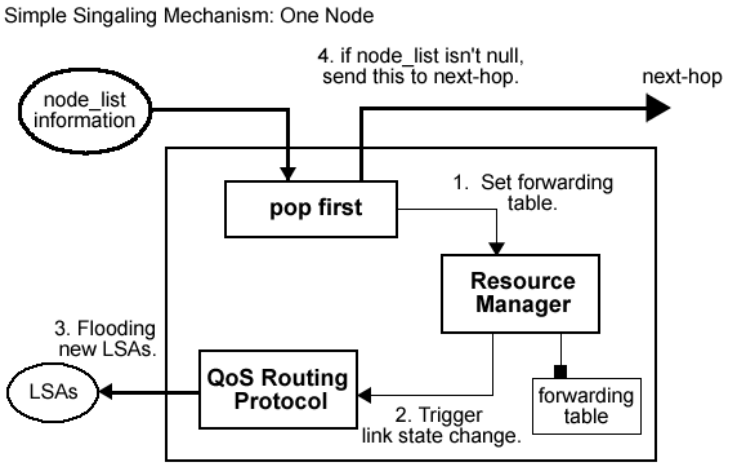


Figure 4-5: A node's viewpoint of the simple signaling mechanism.

5. Comparisons of Link Utilization

This section evaluates the network link utilization when the constraint-based link-state routing is used in comparison with the conventional link-state routing. We mainly use two different topologies to evaluate the performance. The first topology is the National Science Foundation Network (NSFNET) topology shown in Fig. 5-1. The second topology is generated by the Georgia Tech Internetwork Topology Models (GT-ITM) graph generation package[2] that supports the random generation of network topology. This topology comprises of 36 nodes and 62 links. We assume that the capacity of each link is 10Mbp.

We adopt the average utilization of all links as the major performance metric. The av-

[2] The source code of GT-ITM is available at http://www.cc.gatech.edu/projects/gtitm/

erage utilization is defined as $\sum \frac{r_i}{BW_i}$, where the BW_i is the capacity of link i, and the r_i is the bandwidth used currently at link i. If the loading of any link exceeds the maximum bandwidth, the utilization of this link will be marked as 1 (i.e., $0 \leq r_i \leq BW_i$).

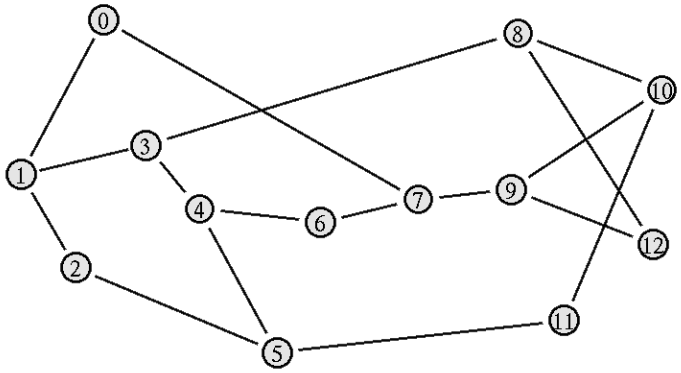


Figure 5-1:The NSFNET topology.

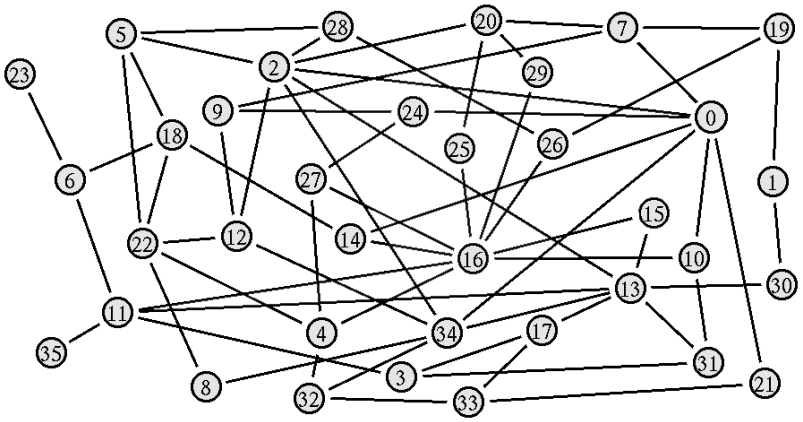


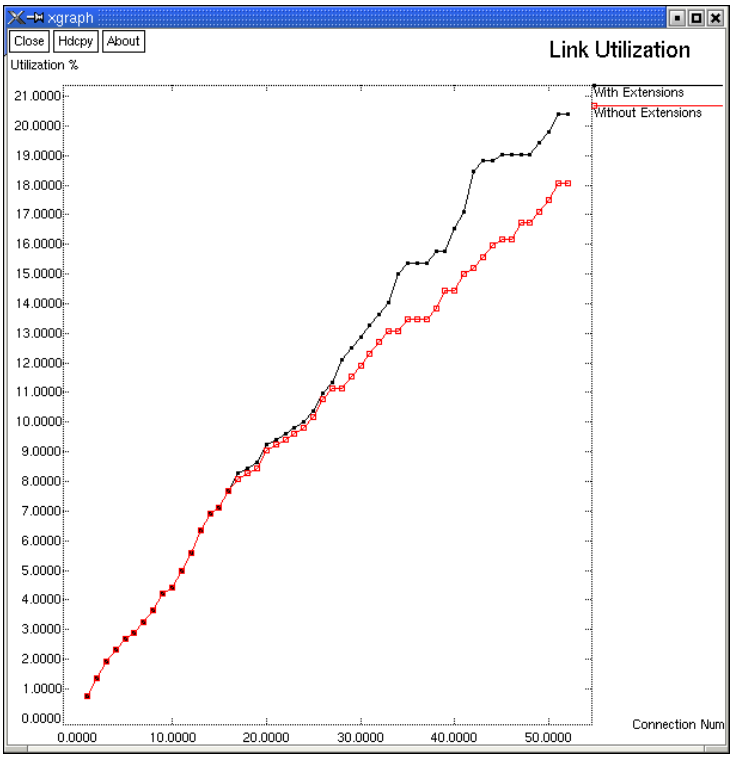
Figure 5-2: A random topology generated by GT-ITM: it includes 36 nodes and 62 links.

The traffic is generated randomly as follows:

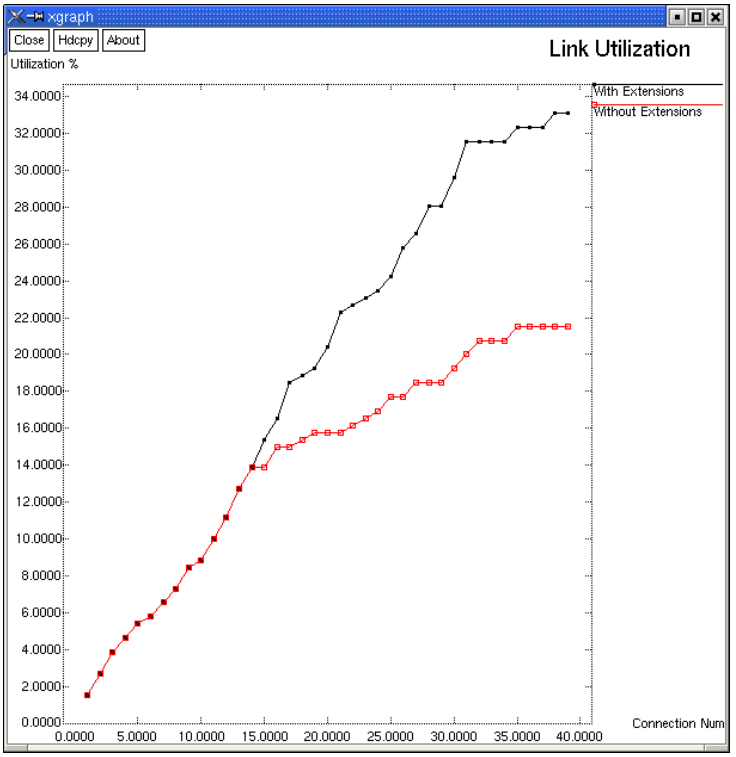
```
for i:=1 to iteration do
 begin;
   for  src :=1 to N do  /* N := the number of nodes */
    begin
          bw := required_bandwidth;
          dest := random [1:N, exclude src];
          create_connection from src to dest with bw;
    end;
 end;
```

The "*iteration*" depends on when the link utilization starts to level off; the source nodes (src) are distributed evenly among all nodes; and the destination node (dest) is generated randomly. Notice that the pair of source node and destination node used by the conven-

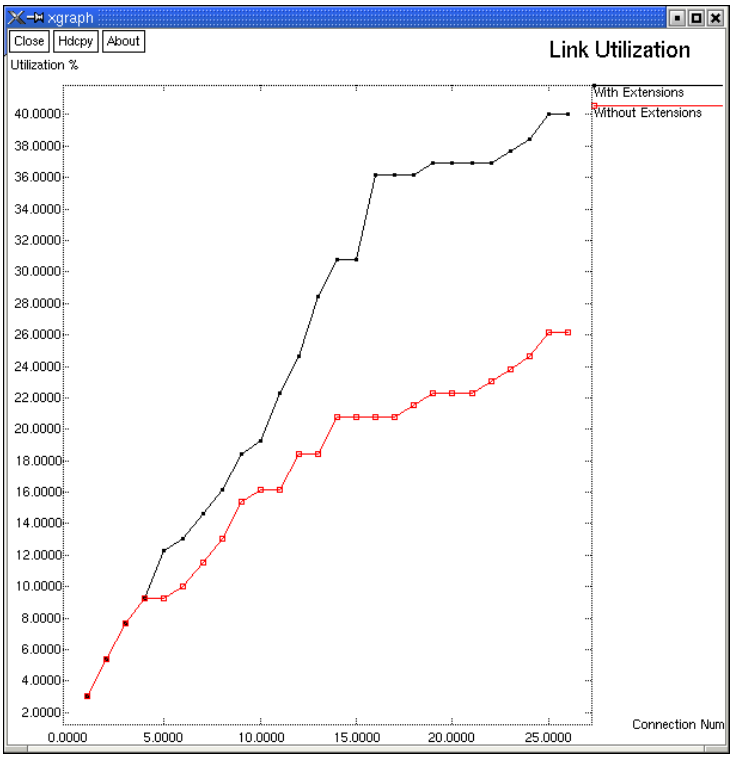
tional link-state routing protocol is the same as those used by the constraint-based link-state routing protocol in order. The results of the average utilization in both topologies are shown in Fig. 5-3 and Fig. 5-4, respectively. The unit of X-axis is the connection number and the unit of Y-axis is the average link utilization (%). The upper line indicates the constraint-based routing and the lower one is the traditional link-state routing protocol. In general, as the number of connections increases, the constraint-based link-state routing provides higher link utilization than the conventional link state routing. In addition, as the bandwidth requirement per connection become larger, the difference becomes wider.



(A): Total 52 connections, 2Mbps per connection.

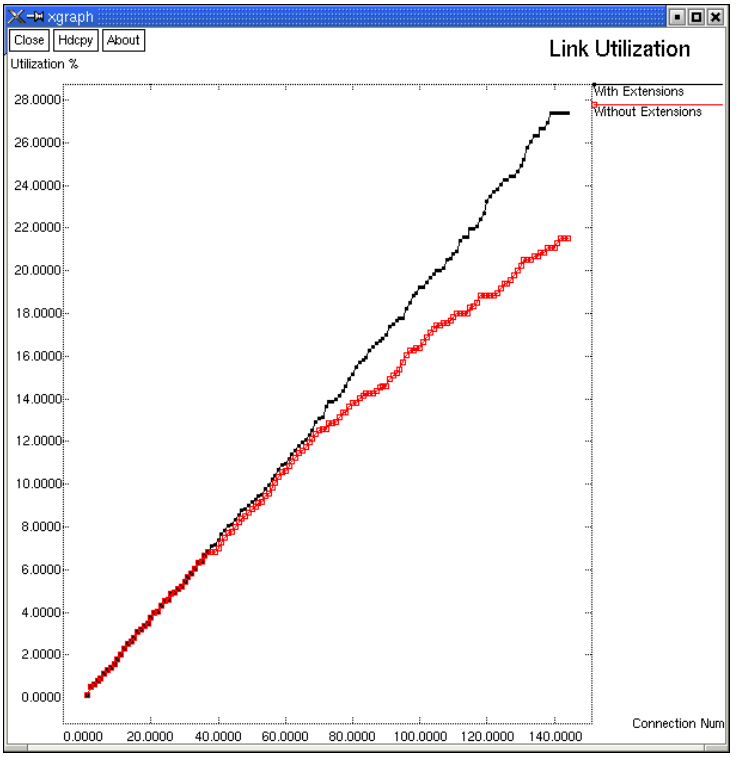


(B): Total 39 connections, 3Mbps per connection.

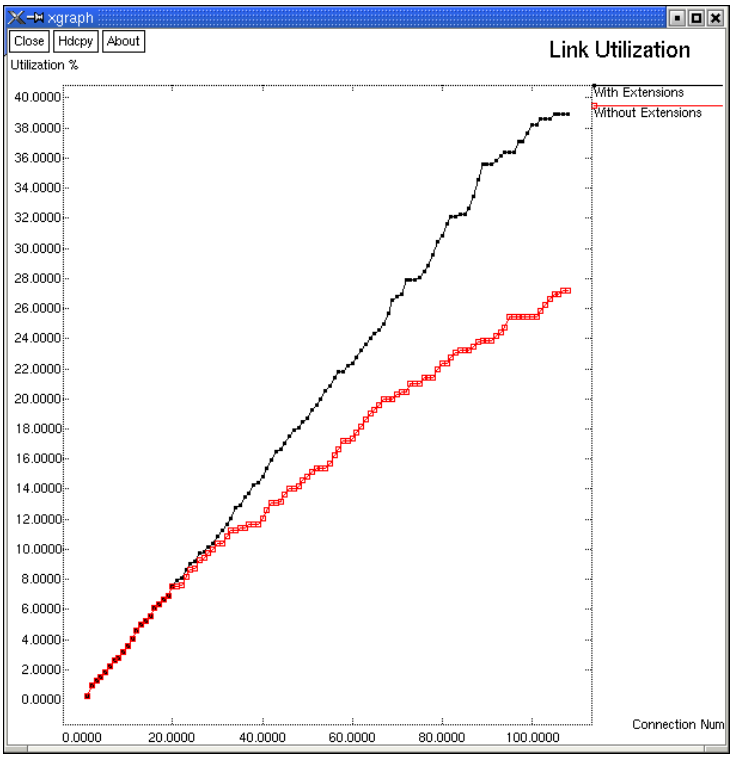


(C): Total 26 connections, 4Mbps per connection.

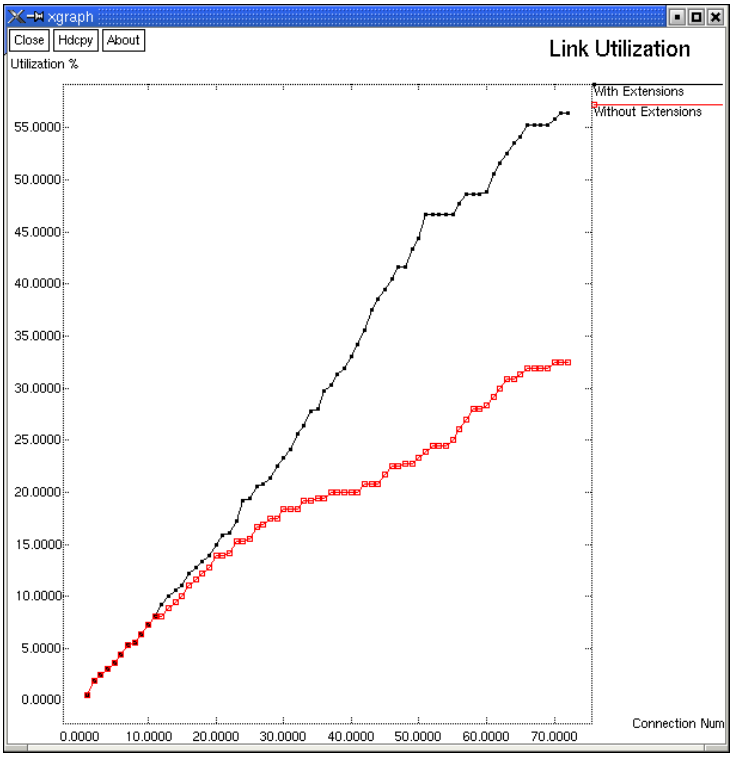
Figure 5-3: Comparisons of the average link utilization with the NSFNET topology.



(A): Total 144 connections, 2Mbps per connection.



(B): Total 108 connections, 3Mbps per connection.



(C): Total 72 connections, 4Mbps per connection.

Figure 5-4: Comparisons of average link utilization with a random topology.

The advantage of the constant-based link-state routing is obvious from these figures. The average link utilization of constraint-based link-state routing is always higher than the conventional one's. It achieves the goals of traffic engineering - distributing the loading more evenly, reducing the congestion, and guaranteeing the QoS.

We next compare the blocking probability.

The blocking probability is defined as $\frac{rejected\_connection\_number}{total\_connection\_number}$. We show the blocking probability of two topologies in Table 5-1 and Table 5-2, respectively. The blocking probability of the conventional link-state routing is always higher. We observe that the more bandwidth per connection, the higher the network performance. When the degree of the network-mesh is higher, the advantage of the constraint-based link-state routing is more obvious.

	With Extensions	**Without extensions**
2Mbps	15%	15%
3Mbps	18%	33%
4Mbps	27%	35%

Table 5-1: Comparisons of blocking probability with NSFNET topology.

	With Extensions	Without extensions
2Mbps	10%	20%
3Mbps	15%	29%
4Mbps	15%	35%

Table 5-2: Comparisons of blocking probability with a random topology.

There are, however, some prices to pay in gaining this better utilization. First, there is the need to regenerate the shortest path tree, i.e., there are more shortest path tree computations. In the conventional link-state routing protocol, the shortest path tree computation is done once in the initial stage. For our constraint-based link-state routing protocol, with the on-demand mode for example, the number of computations is proportional to the number of signalings originated from this node.

Second, the numbers of LSAs' broadcasts are different too. In the conventional link-state routing, it only needs to broadcast once in the initial stage, if there is no link-state change later. However, in the constraint-based link-state routing, each node needs to broadcast the LSAs when the signaling message visits this node because this node is on the explicit node list.

6 Conclusion

In this paper, we have presented extensions to the link-state routing protocol for QoS routing. We first reviewed the OSPF, which is a widely used link-state routing protocol, the concept of traffic engineering, and the interaction between QoS signaling protocols and CSPF. Then we proposed the constraint-based link-state routing architecture and discussed the general operations within this architecture. In our approach, we adopted a practical viewpoint and constructed this constraint-based link-state routing architecture in a way that mimics the actual structure in a router. By incorporating a simple signaling protocol, the complete actions regarding QoS routing in the traffic-engineering domain are better understood. Lastly, our simulation results showed that the average link utilization of the entire network is increased, verifying that the loadings of all flows are distributed more evenly.

Reference

[1] D. Awduche, J.Malcolm, J. Agogbua, M.D'dell abd J. MacManus, "Requirements for Traffic Engineering over MPLS," RFC 2702, September 1999, Available http://www.ietf.org/rfc/rfc2702.txt.

[2] J. Moy, "OSPF – Anatomy of an Internet Routing Protocol," *Addison-Wesley*, 1998.

[3] J. Moy, "OSPF Version 2", RFC 2328, April 1998, Available http://www.ietf.org/rfc/rfc2328.txt

[4] J. Moy, "OSPF Standardization Report", RFC 2329, April 1998, Available http://www.ietf.org/rfc/rfc2329.txt.

[5] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001, Available http://www.ietf.org/rfc/rfc3209.txt.

[6] G. Apostolopoulos, S. Karmat, R. A. Guerin, A. Orda and T. Przygienda, "QoS Routing Mechanisms and OSPF Extensions," RFC 2676, August 1999, Available http://www.ietf.org/rfc/rfc2676.txt.

[7] D. Katz, D. Yeung and K. Kompella, " Traffic Engineering Extensions to OSPF," Internet Draft, Available http://www.ietf.org/internet-drafts/draft-katz-yeung-ospf-traffic-06.txt.

[8] Z. Wang "Internet QoS – Architectures and Mechanisms for Quality of Service", *Morgan Kaufmann*, pp 11-13, 2001.

[9] William R. Parkhusrt, "Cisco Router OSPF," *McGraw Hill*, pp 67-92, 1998.

[10] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC 2205, September 1997, Available http://www.ietf.org/rfc/rfc2205.txt.

[11] R. A. Guerin, A. Orda and D. Williams "QoS Routing Mechanisms and OSPF Extensions." IEEE GLOBECOM 97, Vol. 3 pp 1003 – 1908.

[12] A. Ariza, E. Casilari and F. Sandoval, "Strategies for updating link states in QoS routers, " Electronics Letters, Vol. 36 No. 20 pp 1749-1750, 28th September 2000.

[13] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," INFOCOM, 2000.

[14] T. Ye, H. T. Kaur, S. Kalyanaraman, K. S. Vastola and S. Yadav, "Dynamic Optimization of OSPF Weights using Online Simulation," INFOCOM 2002.

[15] G. Apostolopoulos, R Guerin and S. Kamat, "Implemeatation and Performance Measurements of QoS Routing Extensions to OSPF," INFOCOM, March 1999.

[16] K. Fall and K. Varadhan, "The NS manual," April 2002. Available http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf.