Available online at www.sciencedirect.com
ScienceDirect
Simulation Modelling Practice and Theory 15 (2007) 426–448
SIMULATION MODELLING PRACTICE AND THEORY
www.elsevier.com/locate/simpat

Providing QoS in OSPF based best effort network using load sensitive routing ☆

Anunay Tiwari, Anirudha Sahoo *

Kanwal Rekhi School of Information Technology, Indian Institute of Technology Bombay, Powai, Mumbai 400076, India

Received 24 February 2006; received in revised form 31 July 2006; accepted 24 November 2006
Available online 16 December 2006

Abstract

In an open shortest path first (OSPF) based best effort network, when a packet experiences congestion, the routing subsystem cannot send it through an alternate path. Thus, it fails to provide desired quality of service (QoS) during congestion. In order to provide QoS we have reported three different load sensitive routing (LSR) protocols in [A. Sahoo, An OSPF based load-sensitive QoS routing algorithm using alternate paths, in: IEEE International Conference on Computer Communication Networks, October 2002; A. Tiwari, A. Sahoo, Providing QoS support in OSPF based best effort network, in: IEEE International Conference on Networks, November 2005; A. Tiwari, A. Sahoo, A local coefficient based load sensitive routing protocol for providing QoS, in: IEEE International Conference on Parallel and Distributed Systems, July 2006]. The LSR protocol forwards packets through alternate paths in case of congestion. The number of alternate paths at any node depends on the value of operating parameter or coefficient used for alternate path calculation. Though the basic protocol in these cases was the same, the methods of choosing operating parameter were different. We referred to these three methods as LSR [A. Sahoo, An OSPF based load-sensitive QoS routing algorithm using alternate paths, in: IEEE International Conference on Computer Communication Networks, October 2002], E-LSR [A. Tiwari, A. Sahoo, Providing QoS support in OSPF based best effort network, in: IEEE International Conference on Networks, November 2005] and L-LSR [A. Tiwari, A. Sahoo, A local coefficient based load sensitive routing protocol for providing QoS, in: IEEE International Conference on Parallel and Distributed Systems, July 2006] coefficient methods. In this paper, we present the LSR protocol along with the three coefficient calculation methods pointing out the reason for going from one method to the next. The main strength of our LSR protocol is that it provides loop free alternate paths in the event of congestion and can interwork with routers running vanilla OSPF protocol. We show through simulation that the LSR protocol based on any of the three different coefficient calculation methods performs much better than OSPF and that out of the three methods proposed by us, L-LSR performs the best.

Keywords: QoS routing; OSPF; Best effort network; Load sensitive routing; Loop free

☆ This research was partially supported by Industrial Research and Consultancy Centre, IIT Bombay under Grant Number 03IR059.
* Corresponding author.
E-mail address: sahoo@it.iitb.ac.in (A. Sahoo).

1569-190X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2006.11.008

1. Introduction

There has been an upsurge in real time applications like Voice over IP, video streaming on the Internet. These applications require quality of service (QoS) to perform satisfactorily. But the current Internet is built on *best effort* infrastructure. Hence there is need for providing QoS on top of best effort network. Usually it is relatively easy to provide QoS to real time application when the application starts. But it is quite difficult to repair the QoS when QoS deteriorates in the middle of running of the application. For example, there are few mechanisms available to provide QoS to VOIP calls when a request for call arrives. Cisco VOIP gateways have Call Admission Control mechanisms in place to admit calls with an accepted level of QoS at the time of call arrival [4]. But when a VOIP call is already connected and the two parties are in conversation, if the QoS of the call deteriorates, then *mid call routing* should be used to reroute the call in a different path to repair the QoS. This should happen transparently without affecting the call. But there is no satisfactory method for providing mid call routing to VOIP or video applications. One effective way would be to provide mid call routing support at the routing layer.

Typically, routing sub-system uses shortest path algorithm [5] like OSPF to route packets. But the routing decision, in this case, is solely based on the destination address of the packets. Hence, packets for a particular destination follow the same path, even though there may be better alternate paths available. Thus, QoS demand of the packets are not considered while routing the packets. If routing protocol can provide support for routing packets along alternate paths, then real time applications like VOIP can perform satisfactorily when the shortest path gets congested. Obviously, this can be exploited for mid call routing.

But routing the packets through better alternate paths is not as straight forward as it may look. One of the challenges is to make the alternate path loop free. If the alternate path protocol is not loop free, then a separate loop detection mechanism has to be put in place. This approach may not be attractive to implementors, since that would mean changing the packet forwarding engine.

In this study, we propose a routing protocol that uses alternate paths to provide QoS along OSPF paths. The network is assumed to be running a link state routing protocol like Open Shortest Path First (OSPF) [6]. Given an OSPF path from a source node to a destination node, the protocol tries to find alternate paths for nodes along the OSPF path. When a node experiences congestion on an outgoing link, it sends congestion notification to all its neighbors except the one connected to it over the congested link. The congestion notification is not flooded, rather it is restricted to only one hop neighbor of the congested node. This node as well as the neighboring nodes then forward packets through alternate paths. The alternate paths are chosen in such a way that the packets do not end up in a loop. Once congestion is over, then the nodes involved in alternate path routing revert back to OSPF routing. Thus, the protocol proposed is very simple, yet is quite effective in providing QoS. But the performance of the protocol depends on being able to find alternate paths for nodes. However, a node cannot arbitrarily choose any neighbor as alternate next hop, rather it has to do so such that the alternate path does not form a loop. The loop free property makes the implementation of the protocol simple, because it does not require a separate loop detection mechanism in the packet forwarding engine. The loop free property of this routing protocol is achieved by adhering to some packet forwarding properties of OSPF protocol, which is loop free. More the alternate paths the better will be the performance of the protocol. The number of alternate paths depends on how the parameter (or coefficient) for finding alternate path is fixed. We present three different methods of finding the parameter. While the basic protocol remains the same the number of alternate paths and the distribution of alternate paths among the nodes change based on the method used. We refer to the protocol as LSR protocol, but choice of alternate path during congestion will change depending on how the operating parameter or coefficient is chosen. Accordingly, we refer to the coefficients as *LSR coefficient* [1], Efficient LSR (*E-LSR coefficient*) [2] and Local LSR (*L-LSR coefficient*) [3]. Note that the operating parameter or coefficient is only calculated one time after OSPF has converged. The same coefficient is used until the topology of the network changes (say due to a link failure), at which time the coefficients are recalculated. Thus, the overhead of calculation of coefficients is quite minimal. In this paper, we discuss how the different coefficients are calculated and present the performance of the protocol when run with different coefficients and compare them with OSPF.

The following are the advantages of LSR protocol over OSPF:

- Better average performance: The LSR protocol tries to find alternate path to route packets when there is congestion in the OSPF path. Hence, packets get better QoS.
- Less overhead and scalability: Our protocol does not use flooding mechanism to communicate congestion of a link. Rather the congestion notification is only contained to the neighbors of the node. Thus, it has less overhead and it can scale easily to large networks.
- Coexistence with OSPF router: Our protocol can be implemented easily with an extension to the framework of OSPF standard [7] by creating a new LSA type. Routers running our algorithm can coexist with routers running vanilla OSPF (without our algorithm). When vanilla OSPF routers get this new LSA types, they will simply drop the LSA. Thus, our LSR protocol can be implemented in the Internet in phases.
- Loop free property: In LSR protocol, alternate next hop is chosen based on next hop property of OSPF routing, which is a loop free protocol. Hence LSR protocol provides loop-free alternate path routing. So there is no need for separate loop detection mechanism in the protocol.

The rest of the paper is organized as follows. In Section 2 we discuss the related work in this area. Section 3 explains the system model used in this paper, Section 4 presents the theory and notations used for LSR and E-LSR coefficient calculation. Section 5 provides the details of LSR coefficient calculation whereas Section 6 does the same for E-LSR coefficient calculation. Section 7 is devoted on the details of L-LSR coefficient calculation. We provide the formal proof of loop free property of LSR protocol in Section 8. We present our simulation results in Section 9 and finally conclude the paper in Section 10.

2. Related work

QoS routing has been studied quite extensively [7–11]. A cheapest path algorithm from one source to all destinations when links have two weights (cost and delay) is studied in [12]. The cheapest path is chosen such that the delay along the path is not more than a certain threshold. In [13], the properties of path weight functions are investigated so that hop-by-hop routing is possible and optimal paths can be computed with the generalized Dijkstra's algorithm. Few studies have analyzed the costs associated with QoS routing [14,15]. Some other solutions in the literature use source routing along with shortest path routing to achieve the goal [16]. But security is a major concern in source routing. Routing on alternate paths based on shortest path first has been studied in [17]. But the disadvantage of this method is that the alternate paths may have loops. Hence a loop detection module is needed in the system. There are few solutions proposed that use flooding to advertise QoS parameters [16,9]. Traffic Engineering extension to OSPF has been proposed in [18] to provide QoS support in OSPF based network. This also uses flooding to advertise QoS related parameters such as *maximum bandwidth*, *unreserved bandwidth*, *traffic engineering metric* etc. But overhead and protocol convergence are main concerns in these approaches. The routing protocol proposed in this paper, does not use flooding to update QoS parameters, rather the change in routing information is confined to the *region* where the QoS has deteriorated. Thus, it has low protocol overhead, low convergence time and does not need a separate loop detection mechanism. QoS can be provided in an IP network by deploying RSVP [19], DiffServ [20] or MPLS [21] in the network. In an MPLS based network traffic engineering has been proposed to provide QoS to traffic flows [22].

3. System model

3.1. Network

We model a network consisting of N nodes. A node P is identified by $Node(P)$, $0 \leqslant P < N$. Nodes in a network are connected by physical links. Physical link from $Node(P)$ to $Node(Q)$ is denoted by $Link(P,Q)$. $Node(P)$ and $Node(Q)$ are said to be neighbors if they are connected by $Link(P,Q)$. Every link $Link(P,Q)$ has a cost $Cost(P,Q) > 0$ associated with it. The OSPF path from $Node(P)$ to $Node(Q)$ has a *OSPFcost* associated with it and is denoted by $OC(P,Q)$. *OSPFcost* is the sum of the cost of each link along the OSPF path. The Number of hops from $Node(P)$ to $Node(Q)$ along the OSPF path is denoted as $HC(P,Q)$.

3.2. Routing table

Each node builds a routing table from the network topology. Given a network topology, a node runs OSPF protocol, i.e., it runs Dijkstra's shortest path algorithm with itself as the source. Each entry in the routing table is a quadruple consisting of *destinationnode*, *nexthop*, *OSPFcost*, *Hop Count*. The *nexthop* will contain the OSPF next hop of the destination when the node uses OSPF for routing. But the *nexthop* will be the LSR next-hop when LSR based alternate path is used due to congestion. Thus, the use of alternate path is transparent to the packet forwarding engine.

3.3. Messages

There are two control messages used by LSR protocol:

- *Congestion Notification:* This message is sent by a node to all its neighbors (except the one connected to it over the congested link) when it detects congestion on that outgoing link. We denote this message by *Congestion*(P, Q) which signifies that a congestion is experienced on the *Link*(P, Q) by *Node*(P) and that *Node*(P) sends this message to all its neighbors except *Node*(Q).
- *Congestion Over:* When a link, which was congested earlier, is no longer congested, this message is sent out to all the neighbors (except the one connected to it over the congested link). We denote this message by *CongestionOver*(P, Q) which is sent by *Node*(P) to all its neighbors except neighbor *Node*(Q) when congestion gets over on *Link*(P, Q).

3.4. Overview LSR protocol

In this section, we present the overview of LSR protocol. Note that this is the protocol followed by the nodes for alternate path routing. But the set of alternate paths for a node will be dependent on how the operating parameter (or coefficient) of the network is chosen.

Now we present forwarding and processing of control message by the LSR protocol:

- When *Node*(P) detects congestion on the link *Link*(P, Q), it sends congestion notification message *Congestion*(P, Q) to all its neighbors except *Node*(Q). The congestion notification is not flooded in the network. It is restricted to only one hop neighbor of the congested node.
- When *Node*(R), a neighbor of *Node*(P), receives *Congestion Notification* message *Congestion*(P, Q), it first gets the set of all destinations for which packets forwarded from *Node*(R) to *Node*(P) would go out on congested link *Link*(P, Q). For each of these destinations, it finds the alternate LSR next hops to forward packets. The method for calculating LSR alternate next hop is dependent on the operating coefficient used and is described later in the paper. If there are more than one alternate LSR next hops, then the one with the *least cost* to the destination is chosen. This new LSR next hop is put into *nexthop* entry of routing table so that packets are routed transparently by the packet forwarding engine through LSR alternate path. *Node*(P) also follows the same procedure for finding LSR alternate next hop.
- When *Node*(P) detects that the congestion is over on link *Link*(P, Q), then it sends congestion over message *CongestionOver*(P, Q) to all its neighbors except *Node*(Q).
- When *Node*(R) receives the *CongestionOver*(P, Q) message it checks the set of all destinations for which packets forwarded from *Node*(R) to *Node*(P) would go out on congested link *Link*(P, Q). For each destination in this set, it resets the next hop entry in the routing table to the OSPF next hop. This makes the packet forwarding engine to transparently revert back to OSPF path. *Node*(P) also reverts back to OSPF next hop in a similar manner.

3.5. Properties of alternate path

For finding alternate paths, we have assumed that QoS should be provided along a few OSPF paths to a particular destination i.e. OSPF paths between few source nodes to a particular destination are chosen as QoS

paths (the same method can be applied if QoS needs to be provided to OSPF paths to a different destination). We denote such paths as $QoSPath(S, D)$ from source $Node(S)$ to destination $Node(D)$.

Alternate paths in LSR are determined based on the following two OSPF properties.

Property 1. *The number of hops from OSPF next hop to a given destination along the OSPF path is less than the number of hops from the current node to the same destination.*

Property 2. *For a given destination, OSPF cost from OSPF next hop is less than the OSPF cost from the current node.*

If $Node(Q)$ is the OSPF next hop of $Node(P)$ for destination $Node(D)$ then from Property 1 we have

$$HC(Q,D) < HC(P,D) \tag{1}$$

And from Property 2, we have

$$OC(Q,D) < OC(P,D) \tag{2}$$

Multiplying both the sides of (1) and (2) by a and b respectively and then combining the two inequalities we have

$$a * HC(Q,D) + b * OC(Q,D) < a * HC(P,D) + b * OC(P,D) \tag{3}$$

where $a \geqslant 0$, $b \geqslant 0$ and $(a,b) \neq 0$. The notation $(a,b) \neq 0$ means that a and b cannot be zero simultaneously. Without loss of generality a can be substituted as 1 and we get

$$HC(Q,D) + b * OC(Q,D) < HC(P,D) + b * OC(P,D)(b \geqslant 0) \tag{4}$$

For a particular node P, a neighbor Q is considered an *eligible* alternate next hop if inequality (4) holds and the neighbor Q is not the OSPF next hop. This ensures that when alternate next hops are chosen, they still conform to OSPF property. This is important for providing loop free alternate paths.

4. Theory and notations used in LSR and E-LSR coefficient calculation

We start with theory and notations used for calculating LSR and E-LSR coefficients. We begin with a theorem which gives the possible cases of finding alternate paths. We provide the theorem here for ease of reference.

Theorem 1. *Let $OC(P,D)$ and $HC(P,D)$ be the OSPF cost and OSPF hop count from $Node(P)$ to destination $Node(D)$ respectively. Let $OC(Q,D)$ and $HC(Q,D)$ be the OSPF cost and OSPF hop count from $Node(Q)$ to destination $Node(D)$ respectively. If $Node(Q)$ is a neighbor of $Node(P)$ and not the OSPF next hop for destination $Node(D)$, $Node(Q)$ will qualify as alternate next hop for $Node(P)$ in the following cases*:

Case 1. *If $HC(Q,D) < HC(P,D)$ and $OC(Q,D) \leqslant OC(P,D)$ then $Node(Q)$ can be accepted as alternate next hop if*

$$b \geqslant 0 \tag{5}$$

Case 2. *If $HC(Q,D) < HC(P,D)$ and $OC(Q,D) > OC(P,D)$, then $Node(Q)$ can be accepted as alternate next hop if*

$$b < x \tag{6}$$

where $x = (HC(P,D) - HC(Q,D))/(OC(Q,D) - OC(P,D))$.

Case 3. *If $HC(Q,D) \geqslant HC(P,D)$ and $OC(Q,D) < OC(P,D)$, then $Node(Q)$ can be accepted as alternate next hop if*

$$b > y \tag{7}$$

where $y = (HC(Q,D) - HC(P,D))/(OC(P,D) - OC(Q,D))$.

Proof

Case 1. From Eq. (4), if $Node(Q)$ should be LSR eligible next hop, then

$$HC(Q,D) + b * OC(Q,D) < HC(P,D) + b * OC(P,D) \tag{8}$$

In this case $HC(Q,D) < HC(P,D)$ and $OC(Q,D) \leqslant OC(P,D)$. Clearly, Eq. (4) will be satisfied for $b \geqslant 0$.

Case 2. In this case, $HC(Q,D) < HC(P,D)$ and $OC(Q,D) > OC(P,D)$. From Eq. (4)

$$b * (OC(Q,D) - OC(P,D)) < HC(P,D) - HC(Q,D)$$
$$\text{i.e.,} \quad b < x \tag{9}$$

where $x = (HC(P,D) - HC(Q,D))/(OC(Q,D) - OC(P,D))$. Note that x is a positive term due to the conditions of this case.

Case 3. In this case, $HC(Q,D) \geqslant HC(P,D)$ and $OC(Q,D) < OC(P,D)$, then $Node(D)$. From Eq. (8)

$$(HC(Q,D) - HC(P,D)) < b * (OC(P,D) - OC(Q,D))$$
$$\text{i.e.,} \quad b > y \tag{10}$$

where $y = (HC(Q,D) - HC(P,D))/(OC(P,D) - OC(Q,D))$. Note that y is a positive term due to the conditions of this case. □

Now the task is to determine value of coefficient b. For this purpose, we define two notations $GT(P,p,D)$ and $LT(P,p,D)$ representing the constraints on value of b.

- In cases 1 and 3, the value of b should be greater than 0 and y respectively. Since y is a positive quantity, the two constraints can be combined to one constraint that b should be greater than y. We refer to it as *greater than* (GT for short) constraint. The pth greater than constraint of $Node(P)$ for destination $Node(D)$ is denoted by $b > GT(P,p,D)$. Thus, for a destination $Node(D)$ if the pth greater than constraint is due to neighbor $Node(Q)$ then $GT(P,p,D)$ is equal to y given in Eq. (7).
- In the case 2, the value of b should be less than x. Similarly, we denote the pth *less than* (LT for short) constraint of $Node(P)$ for destination $Node(D)$ by $LT(P,p,D)$. Thus, for destination $Node(D)$, if the pth less than constraint is due to neighbor $Node(Q)$ then $LT(P,p,D)$ is equal to x given in Eq. (6).

Example. In Fig. 7, a service provider wants to use LSR protocol to provide QoS along the OSPF path between ingress node 0 and egress node 5. The OSPF path between these two nodes is 0, 1, 2, 3, 4, 5. Nodes 31 is the neighbor which is not in the OSPF path of node 3 and hence can potentially become eligible alternate next hop for 3. The OSPF cost from 3 to 5 is 6 and the corresponding hop count is 2. The OSPF cost from 31 to 5 is 7 and the corresponding hop count is 1. We use these values in Eq. (4) to get the LSR constraint as follows:

$$1 + 7b < 2 + 6b, \quad \text{i.e., } b < 1$$

Thus, neighbor 31 can become eligible alternate next hop of node 3 if $b < 1$. This is a LT constraint.

Thus, for a particular destination, if a node satisfies m number of constraints (LT and GT), then *potentially* it has m alternate paths for that destination. But only n ($0 \leqslant n \leqslant m$) out of the m potential alternate paths will actually be used for alternate path routing, depending on the operational value of b decided by our algorithm. Thus, fixing the value of *operational* b (denoted as $b_{op}(D)$ for destination $Node(D)$) appropriately is crucial for the efficient operation of LSR and E-LSR algorithms. Remember that we are trying to provide QoS along the OSPF path of an ingress node $Node(S)$ and an egress node $Node(D)$ ($Node(D)$ is the destination). Let this path be denoted by $QoSPath(S,D)$.

In subsequent discussions, our focus will be on the OSPF path $QoSPath(S,D)$ between an ingress node $Node(S)$ and egress node $Node(D)$ with the egress node $Node(D)$ being the destination. We introduce few more notations which are needed to calculate $b_{op}(D)$.

- For destination $Node(D)$, let $SGT_{all}(D)$ contains all the *greater than* constraint parameters $GT(P,p,D)$ of all the nodes along the OSPF path. Let there be m' elements in $SGT_{all}(D)$ denoted by the ordered list $(g'_1, g'_2, \ldots, g'_{m'})$ such that $g'_1 \leqslant g'_2 \cdots \leqslant g'_{m'}$. Now remove duplicate entries from $SGT_{all}(D)$ and sort them in increasing order. Let this sorted list be $SGT(D)$. Let there be m elements in $SGT(D)$ denoted by the ordered list $(g_1, g_2, \ldots, g_m)$ such that $g_1 < g_2 \cdots < g_m$, where g_i, $(1 \leqslant i \leqslant m)$ are the distinct *greater than* constraint parameters.
- Similarly $SLT_{all}(D)$ contains all the *less than* constraint parameters. Let there be n' elements in $SLT_{all}(D)$ denoted by the ordered list $(l'_1, l'_2, \ldots, l'_{n'})$ such that $l'_1 \leqslant l'_2 \cdots \leqslant l'_{n'}$. Now let $SLT(D)$ is the corresponding ordered list for distinct *less than* constraints i.e. $SLT(D)$ is given by $(l_1, l_2, \ldots, l_n)$, such that, $l_1 < l_2 \cdots < l_n$, where l_i $(1 \leqslant i \leqslant n)$ are the distinct *less than* constraint parameters.
- $GT_{min}(S,D)$ represents the minimum value among all the *greater than* constraint parameters of $Node(S)$ for destination $Node(D)$.
- Similarly, $LT_{max}(s,d)$ represents maximum value among all the *less than* constraint parameters of $Node(s)$ for destination $Node(d)$.
- $No\_of\_Constraints\_GT(g_i, D)$ represents the number of GT constraints that will be satisfied if $g_i < b < g_{i+1}$ where g_i and g_{i+1} belong to $SGT(D)$. If C_i^g is the number of *greater than* constraints in $SGT_{all}(D)$. Then we have

$$No\_of\_Constraints\_GT(g_1, D) = C_1^g \tag{11}$$

$$No\_of\_Constraints\_GT(g_i, D) = C_i^g + No\_of\_Constraints\_GT(g_{i-1}, D) \tag{12}$$

  where $1 < i \leqslant m$.
- Similarly $No\_of\_Constraints\_LT(l_i, D)$ represents the number of LT constraints that will be satisfied if $l_{i-1} < b < l_i$ where l_{i-1} and l_i belong to $SLT(d)$. If C_i^l is the number of *less than* constraints in $SLT_{all}(D)$, then

$$No\_of\_Constraints\_LT(l_n, D) = C_n^l \tag{13}$$

$$No\_of\_Constraints\_LT(l_i, D) = C_i^l + No\_of\_Constraints\_LT(l_{i+1}, D) \tag{14}$$

  where $1 \leqslant i < n$.

5. LSR coefficient calculation

In this section, we present the algorithm for calculating LSR coefficient. LSR coefficient is calculated such that the total number of alternate paths in the network (for a given destination) is maximized. *LSR_coefficient_calculation*( ) in Algorithm 1 shows the algorithm used for calculating LSR coefficient. In step 11, it goes through each LT constraint and checks how many alternate paths are possible if the operating parameter is chosen as the LT constraint. It remembers the LT constraint for which the maximum number of alternate paths are obtained. In step 24, it tries each GT constraint and retains the one which gives the maximum number of alternate paths. Finally, in step 36 it sets the operating parameter ($b_{op}(D)$) to either the LT or the GT constraint depending which one resulted in more alternate paths.

Now coming to time complexity of Algorithm 1, the number of LT constraints or GT constraints in $O(N^2)$, where N is the number of nodes in the network. So from step 11 and step 13 it is clear that the time complexity of *LSR_coefficient_calculation*( ) is $O(N^4)$.

Algorithm 1 (LSR_coefficient_calculation($SLT_{all}(D), SGT_{all}(D)$)).

1: $count = 1$, $alt\_path = 0$, $lt\_min = INVALID$, $gt\_max = INVALID$;
2: $alt\_path\_lt = 0$;
   /* there is no greater than constraint */
3: **if** $m' = 0$ **then**

```
    b_op(D) = l'_1 − ε;/* where (l'_1 − ε) > 0 */
else if n' = 0 then
    /* there is no less than constraint */
    b_op(D) = g'_m' + ε; /* where (g'_m' + ε) < Infinity */
else if l'_1 > g'_m' then
    b_op(D) = (g'_m' + l'_1)/2
else
    for all l' = l'_1 to l'_n' do
        alt_path_lt = count;
        for all g' = g'_1 to g'_m' do
            if g' > l' then
                alt_path_lt++;
            end if
        end for
        if alt_path_lt > alt_path then
            alt_path = alt_path_lt; lt_min = l';
        end if
        count++;
    end for
    /* Go through all the greater than constraints in increasing order */
    count = 1, alt_path_gt = 0;
    for all g' = g'_1 to g'_m' do
        alt_path_gt = count;
        for all l' = l'_1 to l'_n' do
            if l' < g' then
                alt_path_gt++
            end if
        end for
        if alt_path_gt > alt_path then
            alt_path = alt_path_gt; gt_max = g';
        end if
        count++;
    end for
    if gt_max != INVALID then
        b_op(D) = gt_max + ε;
    else
        b_op(D) = lt_min − ε;
    end if
end if
```

6. E-LSR coefficient calculation

LSR coefficient was calculated such that the total number of alternate paths in the entire network is maximized. But that may lead to number of alternate paths, that is skewed towards some nodes. That is, some nodes in the network may have too many alternate paths whereas some other nodes may not have any alternate path. Thus, LSR protocol based on LSR coefficient may not able to handle congestion if congestion occurs at one of the nodes which do not have any alternate path. To address this problem we have devised Efficient LSR (E-LSR) coefficient method. The criterion used for choosing the value of $b_{op}(D)$ in E-LSR is that the total number of alternate paths is maximized subject to the constraint that maximum number of nodes in $QoSPath(S, D)$ have at least one alternate path. The rational behind this optimization is that there will be more number of nodes which has alternate paths to avoid congestion along the OSPF path and hence it will lead to better performance.

Towards this goal, we introduce the *objective function( )* that is used for E-LSR coefficient calculation. This function is designed in such a way that the number of alternate paths is maximized with the constraint that maximum number of nodes will have at least one alternate path. The objective function takes four arguments: *low_limit* and *high_limit* specify the range in which the value of b is tested for *optimal* operational E-LSR coefficient. *Path*(i,j) represents the path along which the optimization criteria is applied. *Node*(d) is the destination node.

Procedure 2. Objective_function(*low_limit*, *high_limit*, *Path*(i,j), *Node*(D))

```
1: int n = 0, m = 0;
2: for all Node(P) in Path(i,j)
3:    if (low_limit ⩾ GT_min(P, D)) or (high_limit ⩽ LT_max(P, D))
4:       n++; /* This node has an alternate path */
5:    end if
6: end for
   /* n is number of nodes in Path(i,j) having at least one alternate path */
7: m = No_of_constraints_LT(high_limit, D) + No_of_constraints_GT(low_limit, D);
   /* Now m represents total number of alternate paths if b takes a value between low_limit and high_limit.*/
8: m = m − n;
9: return N * N * n + m; /* N is the total number of nodes */
```

The above objective function defines two parameters, namely, n and m. n represents number of nodes with at least one alternate path and m represents number of alternate paths other than those n alternate paths (if the value of b is chosen between *low_limit* and *high_limit*). The final value returned is $(N * N * n + m)$, where N is the total number of nodes in the network. The following theorem shows that *objective_function( )* will always return a value that would represent maximum alternate paths subject to the constraint that maximum number of nodes have at least one alternate path.

Theorem 2. *When the E-LSR coefficient b is chosen between low_limit and high_limit, let n be the number of nodes with at least one alternate path to a destination Node(D) and m be the total number of alternate paths excluding those n alternate paths in the topology. If N is the total number of nodes in the topology, then* $(N^2 * n + m)$ *represents a value that leads to maximum alternate paths subject to the constraint that maximum number of nodes have at least one alternate path.*

Proof. For a destination *Node*(D), the maximum number of alternate paths possible for a node *Node*(P) in the network is $N-2$. This is because there can never be any alternate path through two nodes: itself and the destination *Node*(D). Let us say that *Node*(P) finds an alternate path through *Node*(Q). Then for *Node*(Q), the maximum number of alternate paths to *Node*(D) would be $N-3$. This is because, in addition to the two nodes (itself and *Node*(D)), *Node*(P) also cannot qualify as alternate next hop for *Node*(Q) (If *Node*(Q) qualifies as the alternate next hop for *Node*(P) from Eq. (4) it is clear that *Node*(P) cannot qualify as alternate next hop for *Node*(Q).). If we continue finding upper bound of alternate paths for each node, then we arrive at the upper bound of total number of alternate paths as given by

$$(N-2)+(N-3)+(N-4)+\cdots+1=(N-1)(N-2)/2$$

The above expression can be taken as an upper bound on m i.e.

$$m<(N-1)(N-2)/2 \tag{15}$$

But $(N-1)(N-2)/2 < N^2$ for $N>0$.

Hence, from Eq. (15),

$$m<N^2 \tag{16}$$

Therefore, if n is given a weight of N^2 (by multiplying it with N^2), then $(N^2 * n)$ will always be greater than m for any value of n and m where $(n,m) \neq 0$. Note that the total number of alternate paths is $(n+m)$. Hence, the objective function value will be maximum for the E-LSR coefficient with the largest value of n. Also, if n is

same for two values of E-LSR coefficient, then the one with larger m will get the larger value from the objective function. Thus, the expression $(N^2 * n + m)$ would lead to maximum number of alternate path subject to the constraint that maximum number of nodes have at least one alternate path. □

The coefficient calculation routine *E_LSR_coefficient_calculation*( ) shown in Algorithm 2 makes use of *objective_function( )* to get the optimal value of b along the OSPF path from *Node*(S) to *Node*(D). This path is denoted as *QoSPath*(S, D). It first checks for trivial cases, where there may be only LT constraints (step 3) or only GT constraints (step 5). Then in steps 13 and 14, note that constraint parameters are not exhaustively tested for optimality. Instead, only intervals between two consecutive GT and LT constraint, where the LT constraint parameter is greater than the GT constraint parameter, are tried. To understand this, refer to Fig. 1. Since GT constraint g_1 and LT constraint l_1 are consecutive and $l_1 > g_1$, the interval is tested for optimality. There is no point in trying the interval (l_1, g_2), since a value of b in that interval will produce at least one less alternate path than a b value chosen in the interval (g_1, l_1) (it will lose alternate path corresponding to l_1). Similarly, interval (g_2, g_3) should not be tried, because it is better to get a value of b in the interval (g_3, l_2) so that at least one more alternate path (corresponding to g_3) can be obtained. Hence, steps 13 and 14 examine only consecutive LT and GT constraints, where the LT constraint parameter is greater than the GT constraint parameter. The algorithm then computes an objective function value for b that belongs to this selected range. Finally, the value of b which results in maximum objective function value is chosen as operating E-LSR coefficient ($b_{\text{op}}(D)$) for a destination *Node*(D). Note that, with little modification, this coefficient calculation algorithm can be used for multiple ingress-egress pairs.

Now let us analyze the time complexity of $E - LSR\_coefficient\_calculation()$. The complexity of *objective_function*( ) is O(N). Since the number of GT constraints is O(N^2), from step 12 it clear that the time complexity of $E - LSR\_coefficient\_calculation()$ is O(N^3). Thus, it is a significant improvement over LSR coefficient calculation.

Algorithm 2 (*E-LSR_coefficient_calculation*(*QoSPath*(S, D))).

1: value_old = 0;
2: *Infinity* = a large number such that it is greater than any constraint parameter (LT or GT);
/* Go through all greater than constraints in increasing order */
/* m is the number of elements in *SGT*(D) */
3: **if** $m = 0$ **then**
4: $b_{\text{op}}(D) = l_1 - \epsilon$, /* where $(l_1 - \epsilon) > 0$ */
/* n is the number of elements in *SLT*(D) */
5: **else if** $n = 0$ **then**
6: $b_{\text{op}}(D) = g_m + \epsilon$; /* where $(g_m + \epsilon) < Infinity$ */
7: **else if** $l_1 > g_m$ **then**
8: $b_{\text{op}}(D) = (g_m + l_1)/2$
9: **else**
10: $SGT'(D) = insert$ 0 to the beginning of *SGT*(D);

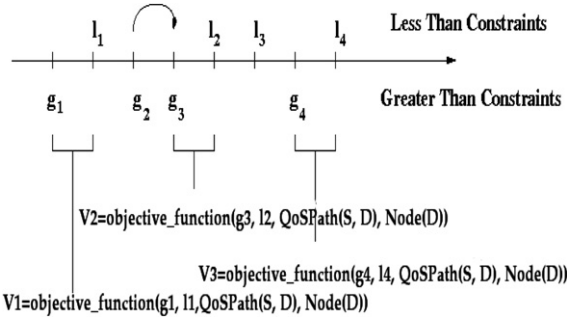

Fig. 1. Coefficient calculation for destination *Node*(d) along *Path*(s, d).

```
11:   SLT'(D) = insert Infinity to the end of SLT(d);
12:   for all constraint g_i in SGT'(D) do
13:     find next high value in SLT'(D), let it be l_j;
14:     if g_{i+1} < lj then
15:       continue;
16:     end if
17:     value = objective_function(g_i, l_j, QoSPath(S, D), Node(D));
18:     if value > value_old then
19:       b_op(D) = (g_i + l_j)/2; value_old = value;
20:     end if
21:   end for
22: end if
```

7. L-LSR coefficient calculation

LSR and E-LSR algorithms find alternate paths based on *global* coefficients, i.e., for a given destination, all the nodes in the entire network use the same coefficient to determine alternate next hop. This is a very limiting factor, which will lead to some nodes *losing* alternate next hops because of the final operational value of global coefficient. This constraint was thought to be necessary to provide loop free alternate paths. Local LSR coefficient or L-LSR coefficient allows nodes to choose the coefficients locally. This gives much more freedom to individual nodes to choose alternate paths. Hence this method should potentially give rise to more alternate paths for a node. In this section, we describe how local coefficients of the nodes in a network are calculated using graph theoretic approach. We need the following notation for this purpose.

(i) $b(X, D)$: L-LSR coefficient of $Node(X)$ for destination $Node(D)$.
(ii) $Neighbor(X)$: Neighbor of $Node(X)$.
(iii) $No\_of\_neighbors(X)$: Number of neighbors of $Node(X)$.
(iv) $QoSPath(S, D)$: It is the OSPF path from source $Node(S)$ to destination $Node(D)$. QoS should be provided when congestion occurs on any of the links along this path. Note that multiple QoS paths can be specified along which QoS would be provided.
(v) $G_Q(V, E_Q, D)$: A directed graph, called QoS graph, where V is the set of vertices and E_Q is set of directed edges between those vertices for destination $Node(D)$. An edge from $Node(v_i)$ to $Node(v_j)$ signifies that $Node(v_j)$ is a *possible* alternate next hop of $Node(v_i)$ for destination $Node(D)$. Later in the section, we show how this graph can be built.
(vi) $DirectedEdge(X, Y)$: A directed edge from $Node(X)$ to $Node(Y)$.
(vii) $T(V, E_T, D)$: Sink tree rooted at destination $Node(D)$ [23]. Note that a sink tree rooted at a node of a graph is the union of the shortest paths from all other nodes to that particular node.
(viii) $CE(i,j)$: It denotes a Cross Edge in $G_Q(V, E_Q, D)$ from any $Node(v_i)$ to $Node(v_j)$ where $Node(v_i)$ and $Node(v_j)$ belong to two different OSPF paths. Hence edge $CE(i,j)$ would not be present in $T(V, E_T, D)$.
(xi) $ME(i,j)$: It denotes a Main Edge in $G_Q(V, E_Q, D)$ from any $Node(v_i)$ to $Node(v_j)$ where $Node(v_i)$ and $Node(v_j)$ belong to the same OSPF path. Note that two nodes are said to be in the same OSPF path (with respect to a destination node) if one of the nodes is along the shortest path (to the same destination) from the other node. Every edge in the sink tree $T(V, E_T, D)$ is a Main Edge. The weights of all the main edges are assigned as *infinity*.
(x) $weight(X, Y)$: Weight of the edge from $Node(X)$ to $Node(Y)$.

Now, we explain some of the above notations, using an example topology shown in Fig. 2. The topology of the network is represented as a graph whose vertices are the nodes of the network and the edges are the links in the network. The cost of the links are labeled along the edges. Sink tree of this topology, $T(V, E_T, D)$, rooted at destination D is shown in Fig. 3. The sink tree is built from the original graph and consists of all OSPF paths

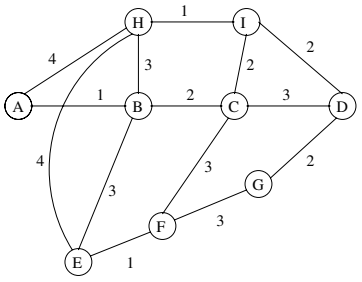

Fig. 2. Example topology to explain the notations used for L-LSR coefficient calculation.

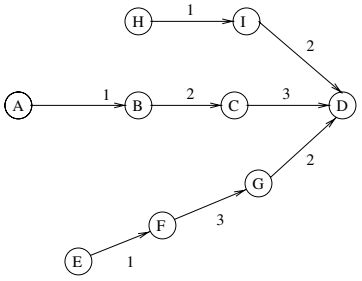

Fig. 3. Sink tree of the example topology rooted at destination D.

from all other nodes to destination node D. Existence of an edge from $Node(v_i)$ to $Node(v_j)$ in the sink tree means that $Node(v_j)$ is OSPF next hop of $Node(v_i)$ for destination $Node(D)$. For example, existence of edge B, C in the sink tree means that C is the OSPF next hop of B for destination D. For this example, we have chosen OSPF path A, B, C, D as the QoS path. Thus, QoS should be provided when any of the links AB, BC or CD is congested. This QoS path would be denoted as $QoSPath(A, D)$. The corresponding QoS graph, $G_Q(V, E_Q, D)$, is shown in Fig. 4. This is built using the algorithm *createQoSGraph* (described later in this Section). Note that edge EH in the original topology does not appear in the QoS graph. This is because neither E nor H is part of the QoS path. In this QoS graph, the edge from node B to node H is a cross edge denoted as $CE(B, H)$. This is because B and H belong to two different OSPF paths. B has four neighbors: A, C, E, H. But B cannot choose C as alternated next hop since C is the OSPF next hop. B also cannot choose A as alternate next hop since it is the OSPF next hop of A. Hence, B can only choose two neighbors, E and H, as potential alternate next hops. Thus, cross edges BE and BH are both assigned a weight of 2. Edge BC, denoted as $ME(B, C)$, is a main edge in the QoS graph, since it is an edge along the OSPF path from A to D. All the main edges have a weight of *infinity* as shown in Fig. 4.

The following algorithm *createQoSGraph* creates $G_Q(V, E_Q, D)$, starting with $T(V, E_T, D)$. For each node along a QoS path, the algorithm adds edges from the node to all its neighbors, except its OSPF next hop

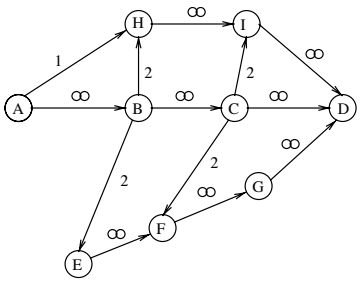

Fig. 4. QoS graph of the example topology.

and the neighbor for which it is the OSPF next hop (these edges would already be present in $T(V, E_T, D)$). Thus, $G_Q(V, E_Q, D)$ represents the neighboring relationship of nodes from the packet forwarding point of view. This algorithm assumes that a node along QoS path can potentially have all its neighbors as alternate next hops. But the node excludes its OSPF next hop and the node for which it is the OSPF next hop from the alternate next hop list. The weight of a cross edge is one less than the out degree of the node from which the edge originates. This is because the edge from the node to its OSPF next hop should be excluded from the out degree, since it does not connect to an alternate next hop. Thus, if a node has many alternate next hops, the weight of outgoing cross edges from that node will be higher than the node with fewer alternate next hops.

Algorithm 3 (createQoSGraph ($Set\_of\_QoS\_paths(D), T(V, E_T, D)$)).

```
1: G_Q(V, E_Q, D) = T(V, E_T, D)
2: for all edges from Node(X) to Node(Y) do in G_Q(V, E_Q, D)
3:     weight (X, Y) = ∞
4: end for
5: for all QoS_path(S_i, D) in Set_of_QoS_paths(D) do
6:    for all Node(X) present do along QoS_path(S_i, D)
7:       no_of_neighbors = 0;
8:       for all Neighbor(X)
9:          if (Neighbor(X) is not OSPF next hop of Node(X)) AND (Node(X) is not OSPF next hop of
   Neighbor(X)) then
10:            no_of_neighbors++;
11:         end if
12:      end for /* no_of_neighbors contains the out degree of Node(X) */
13:      for all Neighbor(X) do
14:         if (Neighbor(X) is not OSPF next hop of Node(X)) AND (Node(X) is not OSPF next hop of
   Neighbor(X)) then
15:            Add an edge from Node(X) to Neighbor(X) in G_Q(V, E_Q, D);
16:            weight(X, Neighbor(X)) = no_of_neighbors;
17:         end if
18:      end for
19:   end for
20: end for
```

But addition of new edges to $T(V, E_T, D)$ may create cycles in $G_Q(V, E_Q, D)$. This means that packets will loop when sent along the alternate path. So some edges have to be removed from $G_Q(V, E_Q, D)$ to make it acyclic. This would ensure that packets do not loop in the alternate path. This problem is termed as *Feedback arc set* problem [24]. A feedback arc set of a (directed) graph is a subset of its arcs whose removal makes the graph acyclic. Similarly, *the minimum feedback arc set* problem consists of finding a minimum weight set of arcs such that after their removal the graph is acyclic. Both problems are NP-complete [25]. A polynomial time approximate algorithm $FAS(\cdot)$ for minimum feedback arc set problem is reported in [24]. We make use of the same algorithm to remove cycles from $G_Q(V, E_Q, D)$.

Create_acyclic_graph($G_Q(V, E_Q, D)$) algorithm, shown below, converts $G_Q(V, E_Q, D)$ into an acyclic graph by removing the edge with maximum weight from a cycle. The reason behind the criteria is that edges having higher weight correspond to nodes having more alternate paths. So the edge which has the maximum weight in the cycle should be removed. Let the resultant acyclic graph be $G_{AQ}(V, E_{AQ}, D)$.

Algorithm 4 (*create_acyclic_graph*($G_Q(V, E_Q, D)$)).

```
1: max_weight = maximum weight out of CE(i,j) for all i, j
2: for all CE(i,j) do
3:    weight(i,j) = max_weight − weight(i,j)
```

4: **end for**
/* Let the new graph be $G'_Q(V,E,D)$ */.
5: $G_{AQ}(V,E_{AQ},D) = FAS(G'_Q(V,E,D))$
6: return acyclic graph $G_{AQ}(V,E_{AQ},D)$

The *create_acyclic_graph*($G_Q(V,E_Q,D)$) uses $FAS(\cdot)$ given in [24]. FAS finds a minimum feedback arc set of G in $O(E_Q.V)$ worst case running time. The step 3 essentially transforms the weight of edges such that the edge with maximum weight will have minimum weight and vice-versa. This enables us to apply FAS(·) algorithm directly. Then in step 5 $FAS(G'_Q(V,E,D))$ removes a set of edges with minimum weight such that all cycles are broken in $G'_Q(V,E,D)$. This implies that a set of edges with maximum weight are removed to break cycles in $G_Q(V,E_Q,D)$. Let this acyclic graph be $G_{AQ}(V,E_{AQ},D)$ in which an directed edge from $Node(v_i)$ to $Node(v_j)$ indicates that $Node(v_i)$ can forward packets to $Node(v_j)$ without forming loops for destination $Node(D)$.

Thus, $G_{AQ}(V,E_{AQ},D)$ is the topology that can be used to forward packets using L-LSR protocol. Every node could just store this graph and use this graph when finding out the alternate next hop. But this would not be efficient in terms of storage, especially since a node has to store one such graph for every destination node. Hence, given this acyclic graph, we find the corresponding L-LSR coefficients such that $G_{AQ}(V,E_{AQ},D)$ is used while finding alternate next hop. Let $b(v_i,D)$ and $b(v_j,D)$ be the L-LSR coefficient of $Node(v_i)$ and $Node(v_j)$ for destination $Node(D)$ respectively. If $Node(v_i)$ can choose $Node(v_j)$ as its alternate next hop then the L-LSR constraint must be satisfied as follows:

$$HC(v_j,D) + b(v_j,D) * OC(v_j,D) < HC(v_i,D) + b(v_i,D) * OC(v_i,D) \tag{17}$$

$$\text{i.e.} \quad b(v_j,D) * OC(v_j,D) - b(v_i,D) * OC(v_i,D) < HC(v_i,D) - HC(v_j,D) \tag{18}$$

$$\text{i.e.} \quad b'(v_j,D) - b'(v_i,D) < weight(v_j,v_i) \tag{19}$$

where

$$weight(v_j,v_i) = HC(v_i,D) - HC(v_j,D) \tag{20}$$

$$b'(v_i,D) = b(v_i,D) * OC(v_i,D) \tag{21}$$

$$b'(v_j,D) = b(v_j,D) * OC(v_j,D) \tag{22}$$

Thus, as per inequality (19), $G_{AQ}(V,E_{AQ},D)$ can be converted to a constraint graph $GC(V,E_{GC},D)$ [26] where there will be a directed edge from $Node(v_j)$ to $Node(v_i)$ having weight $HC(v_i,D) - HC(v_j,D)$. This means, that $GC(V,E_{GC},D)$ can be obtained from $G_{AQ}(V,E_{AQ},D)$ by reversing the direction of edges and assigning weights according to (20).

The *calculate_coefficient* algorithm calculates L-LSR coefficients of all the nodes. It is clear that destination $Node(D)$ will be a source vertex in $GC(V,E_{GC},D)$ since its incoming degree is 0. The algorithm starts with the source vertex of $GC(V,E_{GC},D)$. We assume that $b'(X,D)$ is K where K is any positive real number. Let the currently visited node be *node_visit* and $b'(node\_visit,D)$ is already calculated. Now let $Neighbor(node\_visit)$ be a neighbor of *node_visit* in $GC(V,E_{GC},D)$ then coefficient corresponding to $Neighbor(node\_visit)$ is calculated so that following constraint (applying (19)) get satisfied.

$$b'(node\_visit,D) - b'(Neighbor(node\_visit),D) < weight(node\_visit, Neighbor(node\_visit)) \tag{23}$$

The step 11 ensures that $b'(X,D)$ for any $Node(X)$ is assigned such that it satisfies L-LSR constraints (23) along all its incoming edges and also ensures that $b'(X,D)$ is always a positive number. We define *getNextNode List*(X) function which will return the list of neighbors of $Node(X)$ such that all incoming edges to those neighbors are visited and they have at least one outgoing edge. The step 15 enqueues all the nodes returned by *getNextNodeList*( ) to the *node_visit_list* queue. The *getNextNodeList*(X) is similar to Breadth First Search (BFS) [26] as it is necessary that before calculating the L-LSR coefficient corresponding to any node, the L-LSR constraints corresponding to all its parent must be available. As an example, refer to Fig. 5 where D is the source node. *getNextNodeList*(D) will return Y and Z. X is excluded from the list since YX incoming

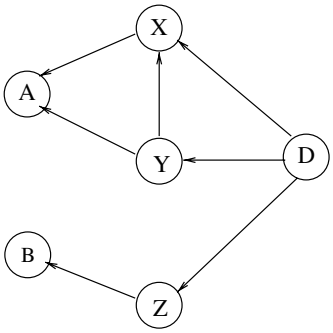

Fig. 5. An example for *getNextNodeList( )*.

edge has not been visited for X. Note that a BFS search at this stage would have returned X, Y and Z for the next round. Later, when Y becomes *node_visit*, it will traverse edge YX and then *getNextNodeList*(Y) would put X in the *node_visit_list*, since now all the incoming edges of X has been traversed.

Once the L-LSR coefficient of all the nodes are calculated, it is easy for a node to find out which neighbor can be an alternate next hop for a given destination. For every neighbor it needs to apply inequality (4). If the inequality is satisfied, then the neighbor can be an alternate next hop. If there are multiple neighbors for which (4) is satisfied, then the node should choose the neighbor which has the least cost to the destination.

8. Loop free property of LSR protocol

The LSR protocol is loop free because the three different coefficient calculation methods are based on OSPF properties which is loop free. Since L-LSR uses local coefficients, we first prove that L-LSR is loop free. Then we explain how LSR and E-LSR can also be proved loop free in a very similar manner.

Algorithm 5 (*calculate_coefficient*($GC(V, E_{GC}, D)$)).

1: **for all** Node(X) in V **do**
2: $b'(X, D) = \mathrm{K}$
3: **end for**
4: edge_list = set of all edges in $GC(V, E_{GC}, D)$.
5: node_visit = $Node(D)$ /* Start with source node */
6: $node\_visit\_list = \{Node(D)\}$./* node_visit_list is queue of nodes to be visited */.
7: $b(node\_visit, D) = b'(node\_visit, D)$
8: **while** edge_list is NOT empty **do**
9: $node\_visit = DEQUE(node\_visit\_list)$ /* Remove the first node from node_visit_list */
10: **for all** Neighbor(node_visit) **do**
11: $b'(Neighbor(node\_visit), D) = \max(b'(Neighbor(node\_visit), D), (b'(node\_visit, D) - weight(node\_visit, Neighbor(X)))) + C_1$ /* this is according to (23) and C_1 is a positive real number
12: $edge\_list = edge\_list - DirectedEdge(node\_visit, Neighbor(node\_visit))$ /* Remove the edge after visiting it */
13: $b(Neighbor(node\_visit), D) = b'(Neighbor(node\_visit), D)/OC(Neighbor(node\_visit), D)$
14: **end for**
15: $ENQUE(node\_visit\_list, getNextNodeList(node\_visit))$
16: **end while**

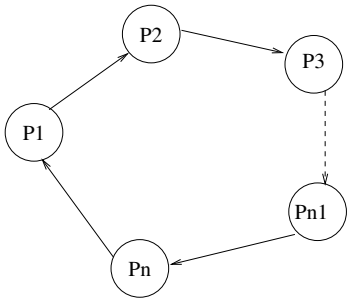

Fig. 6. A loop formation in packet forwarding.

Theorem 3. *If local L-LSR coefficients are chosen such that both OSPF and L-LSR forwarding satisfy the L-LSR constraint given in Eq. (4), then L-LSR protocol is loop free.*

Proof. We prove this theorem by contradiction. Let us assume that L-LSR protocol will have a loop. Fig. 6 shows a case where a loop is formed. Let the loop consist of n nodes (for any $n > 1$) such that $Node(P_1)$ forwards packet destined to $Node(D)$ (not shown in figure) to $Node(P_2)$ which forwards packet to $Node(P_3)$ and so on. The forwarding of packets between any pair of nodes may follow L-LSR or OSPF routing protocol. Now as $Node(P_1)$ forwards packet to $Node(P_2)$ for destination $Node(D)$, the following L-LSR constraint should be satisfied regardless of whether L-LSR or OSPF routing is used (the L-LSR coefficients are chosen such that both OSPF and L-LSR next hops satisfy the L-LSR constraint).

$$HC(P_2, D) + b(P_2, D) * OC(P_2, D) < HC(P_1, D) + b(P_1, D) * OC(P_1, D) \tag{24}$$

Similarly, $Node(P_2)$ forwards packet to $Node(P_3)$ for destination $Node(D)$ and so on. Finally, $Node(P_n)$ forwards packet to $Node(P_1)$ for the same destination. This can happen only if following set of L-LSR constraints are satisfied.

$$HC(P_3, D) + b(P_3, D) * OC(P_3, D) < HC(P_2, D) + b(P_2, D) * OC(P_2, D) \tag{25}$$

$$\vdots$$

$$HC(P_n, D) + b(P_n, D) * OC(P_n, D) < HC(P_{n-1}, D) + b(P_{n-1}, D) * OC(P_{n-1}, D)$$

Combining above $(n - 1)$ inequalities we get

$$HC(P_n, D) + b(P_n, D) * OC(P_n, D) < HC(P_1, D) + b(P_1, D) * OC(P_1, D) \tag{26}$$

Since $Node(P_n)$ forwards packet to $Node(P_1)$, the corresponding L-LSR constraint should be satisfied as shown below.

$$HC(P_1, D) + b(P_1, D) * OC(P_1, D) < HC(P_n, D) + b(P_n, D) * OC(P_n, D) \tag{27}$$

Clearly, (27) contradicts (26). Hence such a loop is not possible. □

To prove that LSR and E-LSR are loop free, it is a simple matter of following the similar steps except that the coefficient used will be global i.e., the coefficients used in the above inequalities will only be denoted as b.

9. Simulation experiment

In this section, we present our simulation setup and performance comparison of L-LSR algorithm with E-LSR, LSR and OSPF algorithms. Our simulation was done using ns2 simulator [27].

9.1. Simulation topology

The topology used in our simulation is shown in Fig. 7. There are 34 nodes in the topology. We have chosen two QoS paths in the topology destined to $Node(5)$: 0, 1, 2, 3, 4, 5 and 10, 9, 8, 7, 6, 5 represented

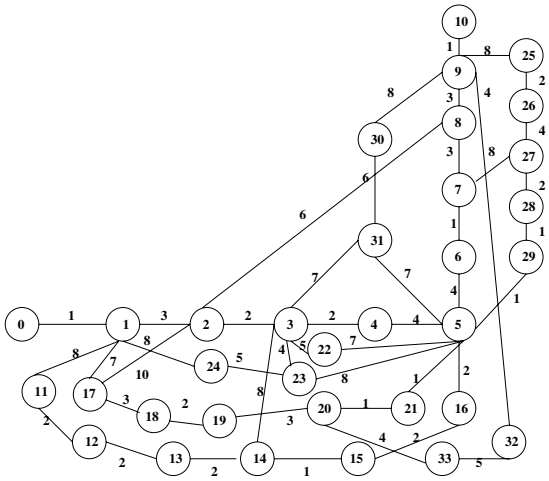

Fig. 7. Topology used for simulation.

by $QoSPath(0,5)$ and $QoSPath(10,5)$. Thus, QoS will be provided along these two paths. OSPF costs of the links are shown in the figure. Cost of links are assigned according to the guideline given in [28] as follows:

$$link\_cost = \lceil 1,000,000/link\ bandwidth\ in\ bps \rceil \tag{28}$$

All the links along the QoS paths are monitored for congestion. The congestion threshold is set to 90% i.e if utilization of a link exceeds 90%, then the link is assumed to be congested.

We have simulated different scenarios as follows:

(i) *Scenario A:* This scenario simulates voice traffic along the QoS paths. We model each voice traffic flow as Constant Bit Rate (CBR) traffic with bandwidth requirement of 64 kbps (packet size: 160 bytes and interval: 0.02 s).[1] A number of such flows destined to node 5 originates from two sources i.e. node 0 and node 10. Thus, it simulates the scenario of voice flows sent along the two QoS paths.

(ii) *Scenario B:* This scenario simulates data traffic along the QoS paths destined to node 5. Each flow is Exponential ON/OFF traffic (packet size: 576 bytes,[2] mean ON period: 50 ms, mean OFF period: 50 ms, average rate: 128 kbps)

We generate cross traffic in other paths in both *Scenario A* and *Scenario B*, to account for the network traffic flowing through other nodes. This cross traffic is generated as follows: source and destination nodes are chosen randomly from among all the nodes in the network. Then each source and destination pair exchange traffic which follows Poisson distribution with an average rate of 32 kbps.

9.2. Results

For performance comparison between L-LSR, E-LSR, LSR and OSPF algorithms, we have used *average delay* of packets from source node to destination node along the designated QoS paths and *percentage packet drop* (PPD) as performance metrics. PPD is defined as the ratio of number of packets not received at the destination to the total number of packets sent from the source. In *Scenario A*, for a given number of voice traffic flows along the QoS paths, we measure the average delay and PPD of those voice flows. The number of voice flows is gradually increased to observe the system performance at various voice traffic load conditions. Sim-

[1] This simulates G.711 voice codec.

[2] This is the path MTU recommended in [29].

ilarly, in *Scenario B* the number of data flows is increased and the corresponding average delay and PPD of the data flows are measured.

Fig. 8 shows the average delay of voice flows in *Scenario A* for different routing protocols, as the number of voice flows (hence load along the path) increases along the QoSPath(10, 5). Clearly, average delay in the case of OSPF algorithm is more than that of LSR algorithm. And average delay in the case of LSR algorithm, in turn, is more than that of E-LSR algorithm for any load. Further, the average delay of L-LSR is the least. In the case of OSPF, when the OSPF path gets congested, OSPF does not reroute packets through any alternate paths, hence delay in this case is the largest. Furthermore, at a high load (more than nine flows), since queues are almost full, delay plateaus around 0.59 s and PPD is quite high at that load. In the event of congestion,

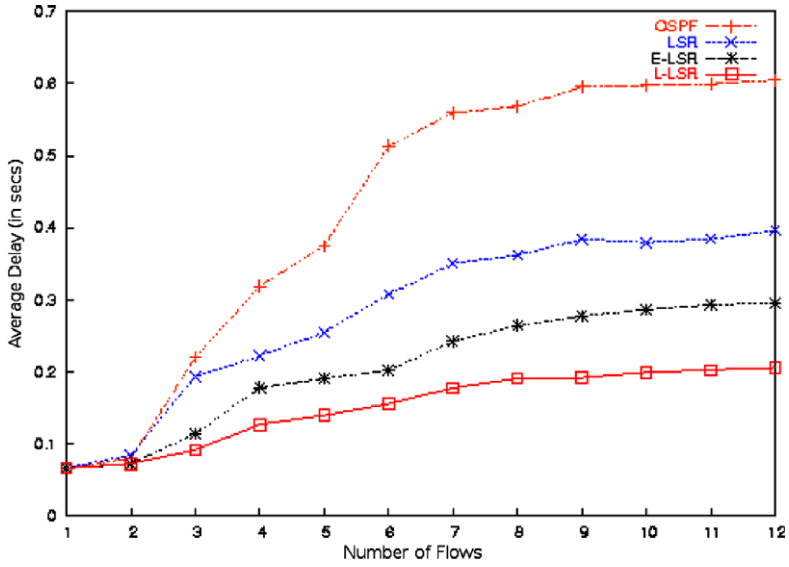

Fig. 8. Average delay vs number of flows (Scenario A) along QoSPath(10, 5).

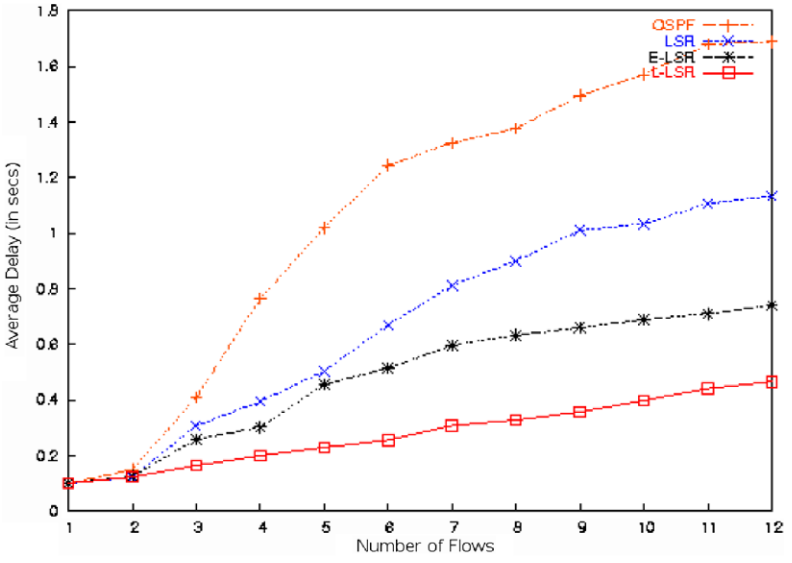

Fig. 9. Average delay vs number of flows (Scenario B) along QoSPath(10, 5).

LSR, E-LSR and L-LSR reroute packets through alternate paths, which leads to lower delay than OSPF. The reason behind the observed relative performance of L-LSR, E-LSR and LSR is explained as follows. In QoSPath(10, 5), only node 9 has an alternate path in LSR. In case of E-LSR both node 9 and node 7 have alternate paths. While in case of L-LSR all three nodes node 7, node 8 and node 9 have alternate paths. Note that in LSR and E-LSR a single value of coefficient is chosen for a given destination for all nodes in the network. This is a restricted requirement, which leads to less alternate paths. But in case of L-LSR each node chooses its own local coefficient. Thus, L-LSR potentially can find alternate paths for more nodes in the network and each of these nodes can have more alternate paths. Similar trend is observed in Fig. 9 across the three protocols in *Scenario B*. Also, Figs. 10 and 11 show the similar performance for number of voice and data flows along QoSPath(0, 5) respectively.

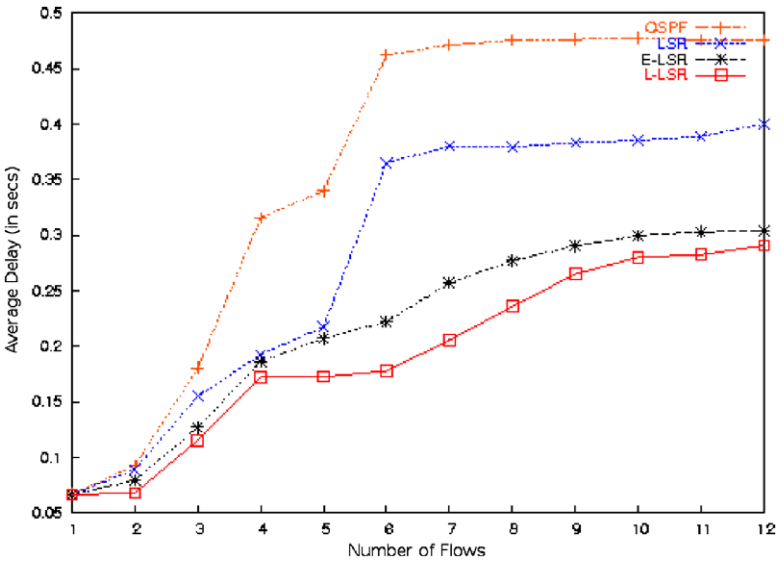

Fig. 10. Average delay vs number of flows (Scenario A) along QoSPath(0, 5).

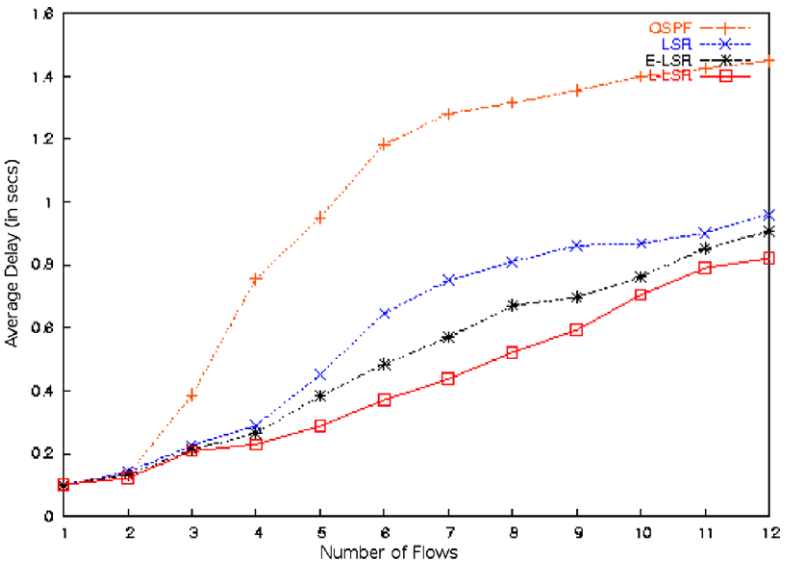

Fig. 11. Average delay vs number of flows (Scenario B) along QoSPath(0, 5).

Fig. 12 shows the corresponding comparison based on PPD in *Scenario A* for flows along QoSPath(10, 5). Here also L-LSR has the least PPD. The PPD of E-LSR is lesser than LSR which is lesser than OSPF. Also PPD increases as the number of flows along QoS paths increases. The similar trend is observed in Fig. 13 across the three protocols in *Scenario B*. Also, Figs. 14 and 15 show similar relative performance for number of voice and data flows along QoSPath(0, 5) respectively.

Tables 1 and 2 list maximum percentage reduction in average delay (MPRAD) and PPD (MPRPPD) of L-LSR protocol over other protocols. For example, in *Scenario A* the average delay is reduced by as much as 66%, 52% and 30% over OSPF, LSR and E-LSR respectively along QoSPath(10, 5). Corresponding numbers for PPD are 69%, 61% and 48% respectively. Thus, we can conclude that L-LSR performs the best in terms of average delay and PPD along both the QoS paths and the performance improvement is quite significant.

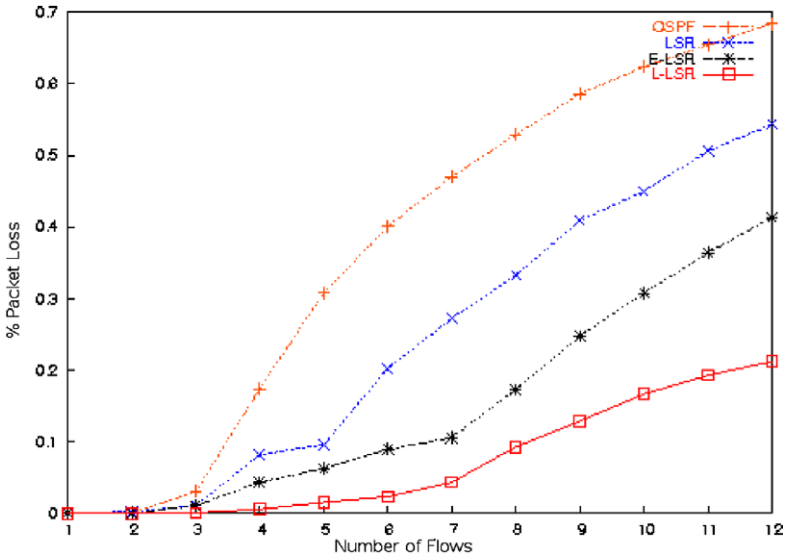

Fig. 12. Percentage packet drop vs number of flows (Scenario A) QoSPath(10, 5).

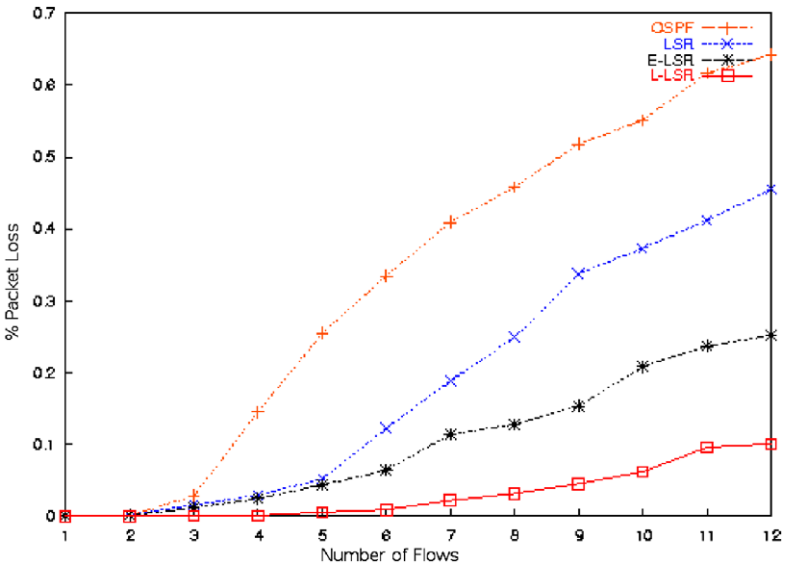

Fig. 13. Percentage packet drop vs number of flows (Scenario B) along QoSPath(10, 5).

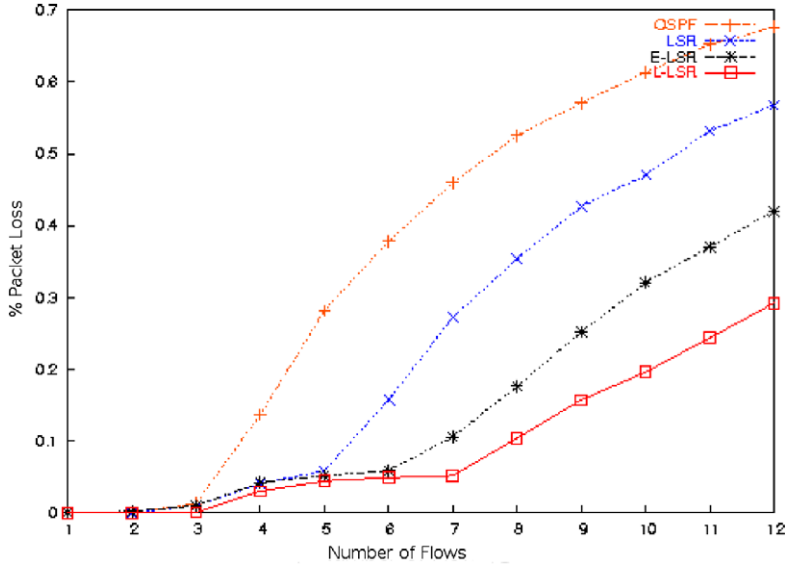

Fig. 14. Percentage packet drop vs number of flows (Scenario A) along QoSPath(0, 5).

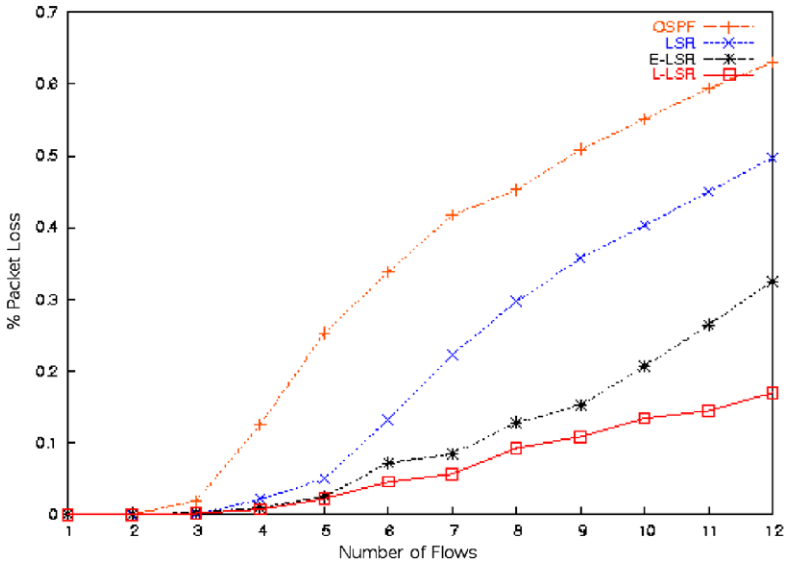

Fig. 15. Percentage packet drop vs number of flows (Scenario B) along QoSPath(0, 5).

Table 1
Maximum percentage reduction in average delay and PPD of L-LSR over other protocols (Scenario A)

	Over OSPF		Over LSR		Over E-LSR	
	QoSPath(10, 5)	QoSPath(0, 5)	QoSPath(10, 5)	QoSPath(0, 5)	QoSPath(10, 5)	QoSPath(0, 5)
MPRAD	66	63	52	51	30	20
MPRPPD	69	56	61	44	48	30

Table 2
Maximum percentage reduction in average delay and PPD of L-LSR over other protocols (Scenario B)

	Over OSPF		Over LSR		Over E-LSR	
	QoSPath(10, 5)	QoSPath(0, 5)	QoSPath(10, 5)	QoSPath(0, 5)	QoSPath(10, 5)	QoSPath(0, 5)
MPRAD	72	63	63	43	48	22
MPRPPD	84	73	78	68	60	47

10. Conclusion

We have presented a load sensitive routing (LSR) protocol for best effort network. We have proposed three different ways of calculating the operating parameter (or coefficient) for LSR protocol. We started off with LSR coefficient, pointed out its limitations which was mitigated by E-LSR coefficient. Finally, we proposed L-LSR coefficient which is the most efficient of all because it allows nodes to choose coefficient locally. We have compared the performance of all the three coefficient-based LSR protocol among each other as well as with OSPF. We have shown through simulation that all the LSR family of protocol perform better than OSPF in term of delay and PPD. Moreover, L-LSR achieves very significant performance improvement over LSR and E-LSR. Thus, L-LSR can be very effective in providing QoS in best effort networks.

References

[1] A. Sahoo, An OSPF based load-sensitive QoS routing algorithm using alternate paths, in: IEEE International Conference on Computer Communication Networks, October 2002.
[2] A. Tiwari, A. Sahoo, Providing QoS support in OSPF based best effort network, in: IEEE International Conference on Networks, November 2005.
[3] A. Tiwari, A. Sahoo, A local coefficient based load sensitive routing protocol for providing QoS, in: IEEE International Conference on Parallel and Distributed Systems, July 2006.
[4] SIP: Measurement-Based Call Admission Control for SIP, http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcacsip.htm.
[5] C. Huitema, Routing in the Internet, Prentice-Hall PTR, 1995.
[6] J. Moy, OSPF version 2, RFC 1583.
[7] G. Apostolopoulos, R. Guerin, S. Kamat, Implementation and performance measurements of QoS routing extensions to OSPF, in: Proceedings of IEEE Infocom, 1999.
[8] W.C. Lee, M.G. Hluchyj, P.A. Humblet, Routing subject to quality of service constraints in integrated communication networks, IEEE Network (1995) 46–55.
[9] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, A. Przygienda, D. Williams, QoS routing mechanisms and OSPF extensions, RFC 2676.
[10] Z. Wang, J. Crowcroft, Quality of service routing for supporting multimedia applications, IEEE Journal of Selected Areas in Communications 14 (7) (1996) 1228–1234.
[11] R. Widyonon, The design and evaluation of routing algorithms for real-time channels, Tech. Rep., June 1994.
[12] A. Goel, K.G. Ramakrishnan, D. Katatria, D. Logothetis, Efficient computation of delay-sensitive routes from one source to all destinations, in: Proceedings of IEEE Infocom, 2001.
[13] J.L. Sobrinho, Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet, IEEE/ACM Transactions on Networking 10 (4) (2002) 541–550.
[14] Q. Ma, P. Steenkiste, On path selection for traffic with bandwidth guarantees, in: IEEE International Conference on Network Protocols, October 1997.
[15] A. Shaikh, J. Rexford, K. Shin, Efficient precomputation of quality-of-service routes, in: Workshop on Network and Operating Systems Support for Digital Audio and Video, July 1998.
[16] A. Segall, P. Bhagwat, A. Krishna, QoS routing using alternate paths, Journal of High Speed Networks 7 (2) (1998) 141–158.
[17] Z. Wang, J. Crowcroft, Shortest path first with emergency exits, ACM SIGCOMM 90 (1990) 166–176.
[18] D. Katz, K. Kompella, D. Yeung, Traffic Engineering (TE) Extensions to OSPF Version 2, RFC 2370.
[19] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource Reservation Protocol (RSVP) Version 1 Functional Specification, RFC 2205.
[20] S. Blake, D. Black, M. Carlson, An architecture for differentiated services, RFC 2475.
[21] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, RFC 3031.
[22] D. Awduche, J. Malcom, J. Agogbua, M. O'Dell, J. McManus, Requirements for traffic engineering over MPLS, RFC 2702.
[23] Andrew S. Tanenbaum, Computer Networks, fourth ed., Prentice-Hall India, 2003.

[24] C. Demetrescu, I. Finocchi, Combinatorial algorithms for feedback problems in directed graphs, ACM Information Processing Letters 3 (86) (2003) 129–136.
[25] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.
[26] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clefford Stein, Introduction to Algorithms, Prentice Hall, 2005.
[27] NS2 simulator, http://www.isi.edu/nsnam/ns/.
[28] OSPF Design Guide, http://www.cisco.com/warp/public/104/2.html.
[29] D.J. Mogul, S. Deering, Path MTU discovery, RFC 1191.