A Local Coefficient Based Load Sensitive Routing Protocol for Providing QoS

Anunay Tiwari and Anirudha Sahoo
Kanwal Rekhi School of Information Technology
Indian Institute of Technology Bombay, Powai, Mumbai - 400076, India
Email: {anunay, sahoo}@it.iitb.ac.in

Abstract

In an Open Shortest Path First (OSPF) based best effort network, the OSPF shortest path can become the bottleneck when congestion occurs. OSPF cannot forward packets though less congested alternate paths. Hence, OSPF cannot be used to provide Quality of Service. Earlier, we reported a Load Sensitive Routing (LSR) algorithm which finds alternate path based on OSPF property. In the earlier work, the LSR used global *coefficients i.e. all the nodes in the network use the same coefficient for a given destination. But assigning network-wide global coefficient may lead to uneven distribution of alternate paths. That is, some nodes may have many alternate paths whereas others may have few or none. The use of global coefficient was thought to be necessary to make the protocol loop free. In this study, we allow nodes to choose LSR coefficients locally (we call the coefficient L-LSR coefficient) while still retaining the loop-free property. This leads to nodes having more number of alternate paths than the case where they had to use global coefficient. But allowing local coefficients makes the process of calculating the local coefficients complex. Since our protocol has to be loop free, the local coefficients have to be calculated such that the loop free OSPF property is still satisfied. This paper presents detailed algorithm for calculating L-LSR coefficients. Using simulation, we show that L-LSR algorithm not only performs better than OSPF, but also has very significant performance improvement over the other LSR family of algorithms.*

1. Introduction

Shortest path routing (SPF) protocols have been studied quite extensively [7]. But all of them suffer from QoS related shortcomings. Specifically, it is well known that when network load increases, shortest path between source and destination becomes congested. Although there may be some less congested alternate paths available, the routing protocol cannot reroute packets via the alternate paths. This is obviously not conducive to real time applications like Voice over IP, video streaming etc. Hence, there is a need for providing QoS support in the routing protocol.

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) [12]. 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. 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. In our earlier study, we proposed an algorithm which finds alternate paths based on *global* coefficients, called *LSR coefficient* [14]. That is, 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 necessary to provide loop free alternate paths. In this study, we allow nodes to choose the coefficients (we call as Local LSR coefficient or L-LSR coefficient) locally. This gives much more freedom to individual nodes. Hence this should potentially give rise to more alternate paths for a node. In this paper, we provide a detailed algorithm to come up with L-LSR coefficients for a node and provide a formal proof that the loop free property still holds. The following are the advantages of L-LSR protocol.

- *Better performance*: As mentioned earlier L-LSR potentially results in more alternate paths because each node chooses its L-LSR coefficient locally. This provides much better performance in terms of average delay and percentage packet drop than OSPF and other protocols in LSR family.
- *Less overhead and scalability*: Our L-LSR protocol does not use flooding to advertise congestion, rather the notification is contained only to the neighbors of the congested node. Hence it has less overhead and can scale easily to large networks.
- *Coexistence with OSPF router*: L-LSR protocol does not require that all the nodes in the networks be running L-LSR. The network can be a mixture of nodes running OSPF and L-LSR. This allows a service provider to deploy our protocol in phases.

- *Loop free property*: Since L-LSR protocol is based on loop-free property of OSPF, packets forwarded using alternate next hop cannot loop. Thus, there is no need for loop detection. This allows the packet forwarding logic to remain the same as OSPF.

QoS routing has been studied quite extensively. A cheapest path algorithm from one source to all destinations when links have two weights (cost and delay) is studied in [9]. The cheapest path is chosen such that the delay along the path is not more than a certain threshold. In [16], 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 [13, 3]. Some other solutions in the literature use source routing along with shortest path routing to achieve the goal [15]. But security is a major concern in source routing. Routing on alternate paths based on shortest path first has been studied in [19]. 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 [15, 5]. Traffic Engineering extension to OSPF has been proposed in [8] 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.

2. The Local Load Sensitive Routing (L-LSR) Protocol

Before we present the L-LSR protocol, we present the system model used by our protocol.

2.1. Network

We model a network consisting of N nodes. A node P is identified by $Node(P)$, $0 \leq P < N$. 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 $OSPF\ cost$ associated with it and is denoted by $OC(P, Q)$. $OSPF\ cost$ 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 $HC(P, Q)$.

2.2. Routing Table

Each node builds a routing table from the network topology. Given a network topology, a node runs Dijkstra's shortest path algorithm with itself as the source. Each entry in the routing table is a quadruple consisting of $destination$ $node$, $nexthop$, $OSPF\ cost$, $HopCount$. 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 nexthop when LSR based alternate path is used due to congestion. Thus, the use of alternate path is transparent to the packet forwarding engine.

2.3. Messages

There are two control messages used by L-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 $Link(P, Q)$.

2.4. Overview of L-LSR Protocol

The L-LSR protocol is very similar to the LSR [14] and E-LSR protocols [18] in the sense that the forwarding and processing of the control messages happen in the exact same manner. The main difference is the method of choosing the coefficients (we discuss about coefficients in subsequent sections) which are used while finding alternate paths. In LSR and E-LSR, nodes get global coefficients i.e. for a given destination, all the nodes in the network use the same coefficient. The global coefficient is calcualted based on some optimization function. For example, in LSR the global coefficient is calcualted such that the total number of alternate paths (AP) is maximized. But this optimization may lead to uneven distribution of APs. For example, some nodes may get many alternate paths whereas some other nodes may have none or few. To address this problem, in E-LSR, the optimization function maximizes total number of APs subject to the constraint that number of nodes having at least one AP is maximized. In contrast to LSR and E-LSR, L-LSR assigns local coefficients to nodes which potentially can lead to more alternate paths. Note that the coefficients are calcualted in the control plane of the routing protocol and they are recalculated only when the topology of the network changes. But letting nodes choose local coefficient makes the coefficient calculation more complex. This is because the local coefficients at different nodes has to be chosen such that there will not be any loops in the packet forwarding path. In the next section, we show a graph theoretic method by which local coefficients are calcualted in L-LSR.

Now we present forwarding and processing of control message by L-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 $Node(Q)$.
- When $Node(R)$, a neighbor of $Node(P)$, receives $CongestionNotification$ 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 L-LSR next hops to forward packets. The

method for calculating L-LSR alternate next hop is described in the next section. If there are more than one alternate L-LSR next hops, then the one with the *least cost* to the destination is chosen. This new L-LSR next hop is put into $nexthop$ entry of routing table so that packets are routed transparently by the packet forwarding engine through L-LSR alternate path. $Node(P)$ also follows the same procedure for finding L-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.

2.5. Properties of Alternate Path

As mentioned earlier, in L-LSR, nodes have local coefficients. Thus, nodes now have more flexibility in choosing the coefficients such that the number of alternate paths for a node is maximized. But L-LSR still needs to provide loop free alternate paths. This makes calculation of local coefficients quite complex. This paper provides the detailed method of calculating local coefficients.

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[1]. We denote such paths as $QoSPath(S, D)$ from source $Node(S)$ to destination $Node(D)$.

Alternate paths in L-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, $a \geq 0 \quad b \geq 0$ and $(a, b) \neq 0$. The notation $(a, b) \neq 0$ means that a and b cannot be zero simultaneously.

[1]The same method can be applied if QoS needs to be provided to OSPF paths to a different destination.

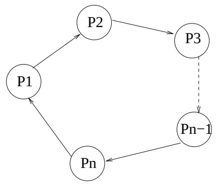

Figure 1. A loop Formation in Packet Forwarding

Without loss of generality a can be substituted as 1 and we get

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

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. In LSR algorithm reported earlier, for a particular destination, all the nodes in the entire network use the *same* LSR coefficient b. Network-wide single value of b may cause some nodes to have many alternate paths whereas some other nodes may have very few alternate paths or none at all. So in L-LSR, instead of having a global coefficient, there is one coefficient, termed as *L-LSR coefficient* , for each node for a given destination. *L-LSR coefficient* of any $Node(P)$ is denoted by $b(P, D)$ for destination D. Now if $Node(P)$ forwards packet to $Node(Q)$ for destination $Node(D)$ then the following constraint should be satisfied.

$$HC(Q, D)+b(Q, D)*OC(Q, D) < HC(P, D)+b(P, D)*OC(P, D) \tag{5}$$

This is called the L-LSR constraint that has to be satisfied when *any* node forwards packets to its neighbor using L-LSR algorithm.

The L-LSR coefficients have to be assigned such that L-LSR constraint in equation (5) is satisfied. This constraint should be satisfied for both OSPF next hop as well as for L-LSR next hops. Thus, calculating local L-LSR coefficients correctly is the main step in L-LSR protocol. Note that for every node, there will be a local L-LSR coefficient for each destination in the network. For a given destination, once the L-LSR coefficient is calculated, then the *eligible* alternate next hops are the ones that satisfy the L-LSR constraint in equation (5). If there are more than one alternate next hop, then L-LSR chooses the one with least cost to the destination.

2.6. Loop Free Property

In this section we provide a formal proof that our L-LSR protocol provides loop free packet forwarding.

Theorem 1 *If local L-LSR coefficients are chosen such that both OSPF and L-LSR forwarding satisfy the L-LSR constraint given in equation(5), 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. Figure 1 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).

$$HC(P_2, D) + b(P_2, D) * OC(P_2, D) < HC(P_1, D) + b(P_1, D) * OC(P_1, D) \tag{6}$$

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.

$$HC(P_3, D) + b(P_3, D) * OC(P_3, D) < HC(P_2, D) + b(P_2, D) * OC(P_2, D) \tag{7}$$

$$\vdots$$

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

Combining above $n - 1$ inequalities we get

$$HC(P_n, D) + b(P_n, D) * OC(P_n, D) < HC(P_1, D) + b(P_1, D) * OC(P_1, D) \tag{8}$$

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

$$HC(P_1, D)+b(P_1, D)*OC(P_1, D) < HC(P_n, D)+b(P_n, D)*OC(P_n, D) \tag{9}$$

Since (9) contradicts (8), such a loop is not possible. □

3. L-LSR Coefficient Calculation

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.

1. $b(X, D)$: L-LSR coefficient of $Node(X)$ for destination $Node(D)$.
2. $Neighbor(X)$: Neighbor of $Node(X)$.
3. $No\_of\_neighbors(X)$: Number of neighbors of $Node(X)$.
4. $QoSPath(S, D)$: 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.
5. $G_Q(V, E_Q, D)$: 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.
6. $DirectedEdge(X, Y)$: A directed edge from $Node(X)$ to $Node(Y)$.
7. $T(V, E_T, D)$: Sink tree[2] rooted at destination $Node(D)$ [4].
8. $CE(i, j)$: 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 $T(V, E_T, D)$.
9. $ME(i, j)$: 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[3]. 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 $infinity$.
10. $weight(X, Y)$: Weight of the edge from $Node(X)$ to $Node(Y)$.

Now, we explain some of the above notations, using an example topology shown in Figure 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 Figure 3. The sink tree is built from the original graph and consists of all OSPF paths 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 Figure 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 Figure 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 and the neighbor for which it is the OSPF next hop[4]. 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[5] than the out degree of the node from which edge originates. 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.

[2]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.

[3]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.

[4]These edges would already be present in $T(V, E_T, D)$.

[5]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.

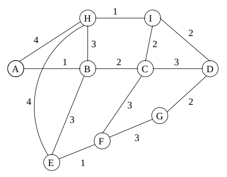

Figure 2. Example Topology to Explain the Notations Used for L-LSR Coefficient Calculation

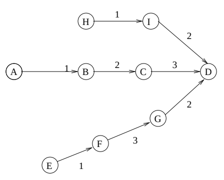

Figure 3. Sink Tree of the Example Topology Rooted at Destination D

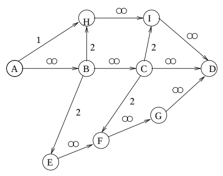

Figure 4. QoS Graph of the Example Topology

Algorithm 1 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 $Node(Y)$ in $G_Q(V, E_Q, D)$ do
3:   weight(X, Y) = $\infty$
4: end for
5: for all $QoS\_path(S_i, D)$ in $Set\_of\_QoS\_paths(D)$ do
6:   for all $Node(X)$ present along $QoS\_path(S_i, D)$ do
7:     $no\_of\_neighbors = 0$ ;
8:     for all $Neighbor(X)$ do
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 [6]. 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 [11]. A polynomial time approximate algorithm $FAS(.)$ for minimum feedback arc set problem is reported in [6]. 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 $G_{AQ}(V, E_{AQ}, D)$.

The $create\_acyclic\_graph(G_Q(V, E_Q, D))$ uses $FAS(.)$ given in [6]. 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

Algorithm 2 $create\_acyclic\_graph(G_Q(V, E_Q, D))$

```
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 $G_{AQ}(V, E_{AQ}, D))$
```

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.

$$\begin{aligned} & HC(v_j, D) + b(v_j, D) * OC(v_j, D) \\ & \quad < HC(v_i, D) + b(v_i, D) * OC(v_i, D) \qquad (10) \\ & i.e.\ b(v_j, D) * OC(v_j, D) - b(v_i, D) * OC(v_i, D) \\ & \quad < HC(v_i, D) - HC(v_j, D) \qquad (11) \\ & i.e.\ b'(v_j, D) - b'(v_i, D) < weight(v_j, v_i) \qquad (12) \end{aligned}$$

where

$$\begin{aligned} weight(v_j, v_i) &= HC(v_i, D) - HC(v_j, D) \qquad (13) \\ b'(v_i, D) &= b(v_i, D) * OC(v_i, D) \qquad (14) \\ b'(v_j, D) &= b(v_j, D) * OC(v_j, D) \qquad (15) \end{aligned}$$

Thus, as per inequality (12), $G_{AQ}(V, E_{AQ}, D))$ can be converted to a constraint graph $GC(V, E_{GC}, D)$ [17] 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

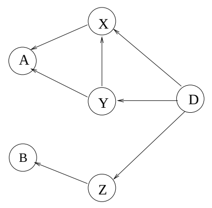

Figure 5. An Example for *getNextNodeList()*

$GC(V, E_{GC}, D)$ can be obtained from $G_{AQ}(V, E_{AQ}, D))$ by reversing the direction of edges and assigning weights according to (13).

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 (12)) get satisfied.

$$b'(node\_visit, D) - b'(Neighbor(node\_visit), D) < weight(node\_visit, Neighbor(node\_visit)) \quad (16)$$

The step 11 ensures that $b'(X, D)$ for any $Node(X)$ is assigned such that it satisfies L-LSR constraints (16) along all its incoming edges and also ensures that $b'(X, D)$ is always a positive number. We define $getNextNodeList(X)$ 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) [17] 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 Figure 5 where D is the source node. $getNextNodeList(D)$ will return Y and Z. X is excluded from the list since YX incoming 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 (5). If the inequality is satisfied, then the neighbor can be an alternate next hop. If there are multiple neighbors for which (5) is satisfied, then the node should choose the neighbor which has the least cost to the destination.

4. Simulation Experiment

In this section, we present our simulation set up and performance comparison of L-LSR algorithm with E-LSR, LSR and OSPF algorithms. Our simulation was done using NS2 simulator [1].

Algorithm 3 calculate_coefficient($GC(V, E_{GC}, D)$)

```
1:  for all Node(X) in $V$ do
2:    $b'(X, D)$ = 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
        (16) 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
```

4.1. Simulation Topology

The topology used in our simulation is shown in Figure 6. 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 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 [2] as follows

$$link\_cost = \lceil 1000000/link\,bandwidth\;in\;bps \rceil \quad (17)$$

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.

1. *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 64kbps (packet size : 160bytes and interval : 0.02 sec)[6]. 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.
2. *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[7], mean ON period : 50 msec, mean OFF period : 50 msec, 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.

[6]This simulates G.711 voice codec.

[7]This is the path MTU recommended in [10].

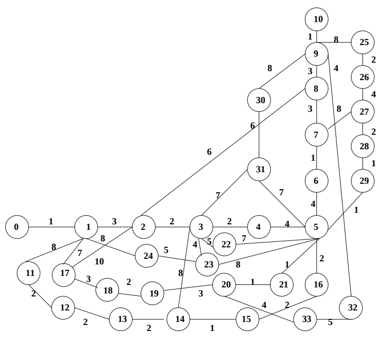
Figure 6. Topology Used for Simulation

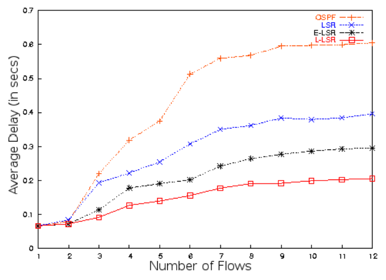
Figure 7. Average Delay Vs Number of Flows (Scenario A) along QoSPath(10,5)

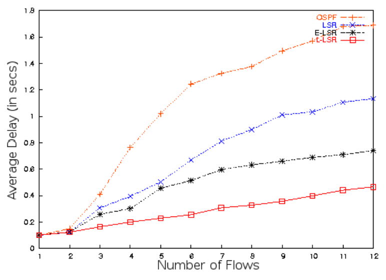
Figure 8. Average Delay Vs Number of Flows (Scenario B) along QoSPath(10,5)

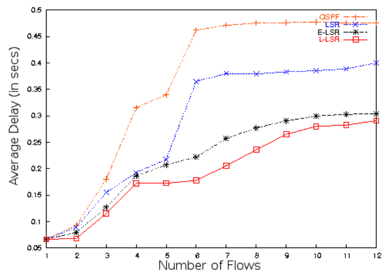
Figure 9. Average Delay Vs Number of Flows (Scenario A) along QoSPath(0,5)

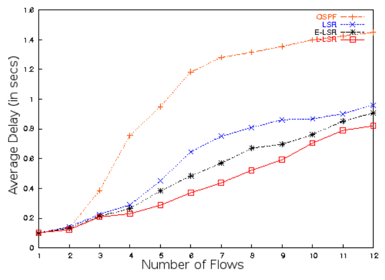
Figure 10. Average Delay Vs Number of Flows (Scenario B) along QoSPath(0,5)

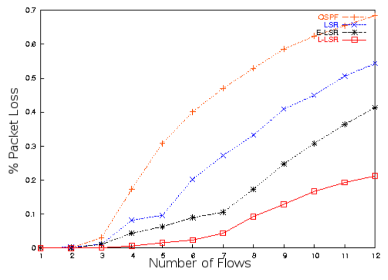
Figure 11. Percentage Packet Drop Vs Number of Flows (Scenario A) QoSPath(10,5)

4.2. 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)[8] as performance metrics. 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. Similarly, in *Scenario B* the number of data flows is increased and the corresponding average delay and PPD of the data flows are measured.

Figure 7 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 9 flows), since queues are almost full, delay plateaus around 0.59 secs and PPD is quite high at that load. In the event of congestion, 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 potenitally 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 Figure 8 across the three protocols in *Scenario B*. Also, Figures 9 and 10 show the similar performance for number of voice and data flows along QoSPath(0,5) respectively.

Figure 11 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 Figure 12 across the three protocols in *Scenario B*. Also, Figures 13 and 14 show similar rela-

[8]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.

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

Table 1. Maximum Percentage Reduction in Average Delay and PPD of L-LSR over Other Protocols in Scenario A

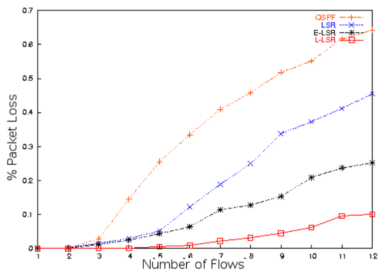

Figure 12. Percentage Packet Drop Vs Number of Flows (Scenario B) along QoSPath(10,5)

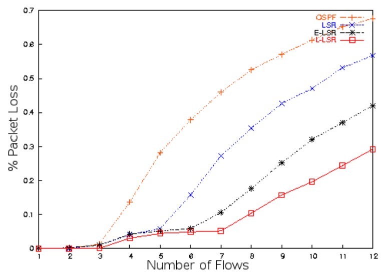

Figure 13. Percentage Packet Drop Vs Number of Flows (Scenario A) along QoSPath(0,5)

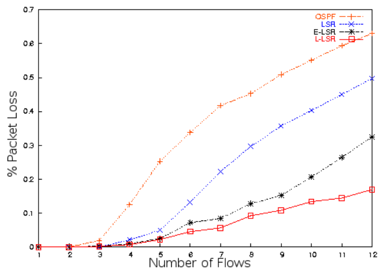

Figure 14. Percentage Packet Drop Vs Number of Flows (Scenario B) along QoSPath(0,5)

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

Table 2. Maximum Percentage Reduction in Average Delay and PPD of L-LSR over Other Protocols in Scenario B

tive performance for number of voice and data flows along QoSPath(0,5) respectively.

Table 1 and Table 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.

5. Conclusion and Future Work

We have presented a local coefficient based load sensitive QoS routing protocol L-LSR which provides loop free routing via alternate paths in the event of congestion. In this protocol, each node can have its own local operating parameter (called L-LSR coefficient). Thus, it leads to more nodes having alternate paths than other protocols (LSR and E-LSR) of the same family we had reported earlier. Also, in L-LSR, a node may have more alternate paths than in LSR and E-LSR. We have shown, through simulation, that performance of L-LSR, in terms of delay and PPD, is far better than OSPF. L-LSR also achieves very significant performance improvement over other protocols of LSR family. Hence, it can be very effective in providing QoS at the routing layer.

We intend to look at the effect of route flapping in the performance of L-LSR and propose an effective route flapping mechanism for it. We would like to study how traffic can be split between the OSPF and the alternate path to improve the performance. We would look at different schemes of splitting the traffic between the OSPF and the alternate paths: equal split, splitting based on the relative cost of the paths, splitting based on the current load along the paths. Impact of splitting traffic along different alternate paths on the transport layer protocol needs to be studied.

References

[1] NS2 simulator. `http://www.isi.edu/nsnam/ns/`.

[2] OSPF Design Guide. `http://www.cisco.com/warp/public/104/2.html`.

[3] A. Shaikh and J. Rexford and K. Shin. Efficient precomputation of quality-of-service routes. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, July 1998.

[4] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall India, Fourth edition, 2003.

[5] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, A. Przygienda, and D.Williams. QoS routing mechanisms and OSPF extensions. *RFC 2676*, April 1999.

[6] C. Demetrescu, I. Finocchi. Combinatorial Algorithms for Feedback Problems in Directed Graphs. *ACM Information Processing Letters*, 3(86):129–136, May 2003.

[7] C. Huitema. *Routing in the Internet*. Prentice-Hall PTR, 1995.

[8] D. Katz, K. Kompella, D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2. *RFC 2370*, September 2003.

[9] A. Goel, K. G. Ramakrishnan, D. Katatria, and D. Logothetis. Efficient Computation of Delay-sensitive Routes from One Source to All Destinations. In *Proceedings of IEEE Infocom*, 2001.

[10] D. J. Mogul and S. Deering. Path MTU discovery. *RFC 1191*, November 1990.

[11] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[12] J. Moy. OSPF version 2. *RFC 1583*, March 1994.

[13] Q. Ma and P. Steenkiste. On Path Selection for Traffic with Bandwidth Guarantees. In *IEEE International Conference on Network Protocols*, October 1997.

[14] A. Sahoo. An OSPF Based Load-Sensitive QoS Routing Algorithm using Alternate Paths. In *IEEE International Conference on Computer Communication Networks*, October 2002.

[15] A. Segall, P. Bhagwat, and A. Krishna. QoS Routing Using Alternate Paths. *Journal of High Speed Networks*, 7(2):141–158, 1998.

[16] 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):541–550, 2002.

[17] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clefford Stein. *Introduction To Algorithms*. Prentice Hall, 2005.

[18] A. Tiwari and A. Sahoo. Providing QoS Support in OSPF Based Best Effort Network. In *IEEE International Conference on Networks*, November 2005.

[19] Z. Wang and J. Crowcroft. Shortest path first with emergency exits. *ACM SIGCOMM 90*, pages 166–176, Sept 1990.