Providing QoS Support in OSPF Based Best Effort Network

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, 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. A Load Sensitive Routing algorithm (LSR) has been reported which finds alternate path based on ospf property. The operating parameter (or coefficient) of LSR was chosen such that total number of alternate paths in the network is maximized. In this paper, we argue that a better operating parameter would be one that maximizes total number of alternate paths subject to the constraint that maximum number of nodes have at least one alternate path. Using simulation, we show that this new algorithm, called Efficient LSR (E-LSR) performs better than OSPF and LSR in terms of delay and packet loss. Besides, this paper presents more efficient algorithm for determining the *optimal* operational E-LSR coefficient by pruning the search space and using a simple objective function. In E-LSR algorithm, the number of alternate paths depend on the topology and costs assigned to the links. Hence, this paper provides a *topology guideline* that may be followed by the implementers to make E-LSR more effective.**

I. 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. There are few mechanisms available to provide QoS to VOIP calls when a request for call arrives. For example, 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 [1]. But there is no satisfactory method for providing *mid call routing*[1] 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 [2] 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 implementers, since that would mean changing the packet forwarding engine. [3] reported an alternate path routing algorithm called *LSR* that provides a loop free alternate path routing. The performance of LSR algorithm depends on *LSR coefficient*, which decides how many alternate paths a node will have for a particular destination. In [3], a methodology is provided to calculate LSR coefficients in which 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. We argue that a better way is to maximize the total number of alternate paths such that maximum number of nodes have at least one alternate path. Towards this goal, we have developed an objective function that achieves this optimality. Further, we have devised a more efficient algorithm to calculate the LSR coefficients. Instead of doing an exhaustive search, this algorithm reduces the search space to only intervals where there is a possibility of finding the optimal operational coefficient . We refer to the resulting algorithm as Efficient LSR (E-LSR) algorithm. We show by simulation that E-LSR outperforms LSR and OSPF in terms of delay and packet loss.

The method used by E-LSR algorithm to find alternate path depends on the topology of the network and on the costs assigned to the links. Hence, topology and link costs play a vital role in the performance of E-LSR algorithm. So, in this paper, we provide a topology guideline that may be followed by the implementers so that they can exploit our algorithm and improve the performance of applications running in the network.

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) such that the delay of the path is not more than a certain threshold is studied in [4]. In [5], 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 [6], [7]. Some other solutions in the literature use source routing along with shortest path routing to achieve the goal [8]. But security is a major concern in source routing. Routing on alternate paths based on shortest path first has been studied in [9]. 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 [8], [10]. But

[1]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 is used to reroute the call in a different path to repair the QoS. This should happen transparently without affecting the call.

overhead and protocol convergence are main concerns in these approaches. E-LSR does not use flooding to update QoS parameters, rather the change in routing information is confined to the *region* where the QoS has deteriorated. Further, alternate paths in E-LSR are loop free. Thus, it has low protocol overhead and low convergence time and does not need a separate loop detection mechanism.

II. System Model

We model a network consisting of N nodes. A node i is identified by $Node(i)$, $0 \leq i < N$. Nodes in the network are connected by physical links. $Node(i)$ and $Node(j)$ are said to be neighbors if they are connected by a physical link $Link(i, j)$. A link $Link(i, j)$ has a cost $Cost(i, j) > 0$ associated with it.

The network runs OSPF protocol to build the routing table which is used by the packet forwarding engine. The routing table contains $ospfcost$ and $ospfhopcount$ along with the $nexthop$ for a a particular destination. $ospfcost$ is the cost of the ospf path to the destination. This is the sum of the cost of each link along the ospf path. $ospfhopcount$ is the number of hops along the ospf path. The $nexthop$ is set to ospf next hop to forward packets along the ospf path, whereas it is set to E-LSR next hop to send it along E-LSR alternate path. We denote ospf cost from $Node(p)$ to destination $Node(r)$ as $OC(p, r)$ and the corresponding ospf hop count is denoted as $HC(p, r)$.

There are two control messages used by E-LSR algorithm.*Congestion Notification* 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. When a link, which was congested earlier, is no longer congested,*Congestion Over* message is sent out to all the neighbors (except the one connected to it over the congested link).

III. E-LSR Algorithm

In this section, we start out with the overview of E-LSR algorithm followed by a detailed description of alternate path computation.

A. Overview of E-LSR Algorithm

- When $Node(i)$ detects congestion on the link $Link(i, j)$, it sends $CongestionNotification(i, j)$ message to all its neighbors except $Node(j)$.
- When $Node(k)$, a neighbor of $Node(i)$, receives $CongestionNotification(i, j)$ message it first gets the set of all destinations for which packets forwarded from $Node(k)$ to $Node(i)$ would go out on congested link $Link(i, j)$. For each such destination, it finds the alternate E-LSR next hops to forward packets. The method for calculating E-LSR alternate next hop is described in the next section. If there are more than one alternate E-LSR next hops, then the one with the *least cost* to the destination is chosen (in this case, LSR chooses one of them *randomly*). This new E-LSR next hop is put into next hop entry of active routing table so that packets are routed through E-LSR alternate path. $Node(i)$ also follows the same procedure for finding E-LSR alternate next hop.
- When $Node(i)$ detects that the congestion is over on link $Link(i, j)$, then it sends $CongestionOver(i, j)$ to all its neighbors except $Node(j)$.
- When $Node(k)$ receives the $CongestionOver(i, j)$ it checks the set of all destinations for which packets forwarded from $Node(k)$ to $Node(i)$ would go out on congested link $Link(i, j)$. For each destination in this set, it resets the next hop entry in the active routing table to the ospf next hop.

B. Properties of Alternate Path

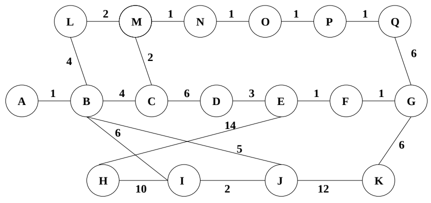

Fig. 1. Topology for our simulation

For finding alternate paths, we have assumed that the OSPF based network is operated by a service provider and the service provider would like to provide QoS along ospf path between an ingress and egress router. That is, alternate paths would be provided to the congested links in the ospf path.

Alternate paths in E-LSR are determined based on the following two ospf properties. Firstly, 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. Secondly, 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 $Node(r)$ then combining the above said two properties we have

$$a * HC(q, r) + b * OC(q, r) < a * HC(p, r) + b * OC(p, r) \quad (1)$$

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. In fact, this constraint is the basis for loop free property of LSR and E-LSR. A formal proof of loop free property can be found in [3].

For a particular node, a neighbor is considered as eligible alternate next hop if inequality(1) holds and if the neighbor is not the OSPF next hop. a and b are called E-LSR coefficient pair. For a particular destination, all the nodes in the entire network use the same E-LSR coefficient pair. This constraint is necessary for loop free alternate path forwarding.

C. Coefficient Calculation

In this section, we provide the detailed method by which E-LSR coefficients are calculated. We begin with a theorem reported in [3] which gives the possible cases of finding alternate paths. We provide the theorem here for ease of reference.

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

Case 1: If $HC(j, d) < HC(i, d)$ and $OC(j, d) \leq OC(i, d)$ then $Node(j)$ can be accepted as alternate next hop if

$a > 0$ and $b \geq 0$.

Case 2: If $HC(j,d) < HC(i,d)$ and $OC(j,d) > OC(i,d)$, then $Node(j)$ can be accepted as alternate next hop if

$$b < x * a \tag{2}$$

where, $x = (HC(i,d) - HC(j,d)/(OC(j,d) - OC(i,d))$
Case 3: If $HC(j,d) \geq HC(i,d)$ and $OC(j,d) < OC(i,d)$, then $Node(j)$ can be accepted as alternate next hop if

$$b > y * a \tag{3}$$

where, $y = (HC(j,d) - HC(i,d))/(OC(i,d) - OC(j,d))$
Proof: Please refer to [11].

In Theorem 1 without loss of generality, we choose $a = 1$ for all destinations. Now the task is to determine value of coefficient b. For this purpose, we define two notations $GT(i,p,d)$ and $LT(i,p,d)$ representing the constraints on value of b.

- In the case 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 p^{th} greater than constraint of $Node(i)$ for destination $Node(d)$ is denoted by $b > GT(i,p,d)$. Thus, for a destination $Node(d)$ if the p^{th} greater than constraint is due to neighbor $Node(j)$ then $GT(i,p,d)$ is equal to y given in equation(3).
- In the case 2, the value of b should be less than x. Similarly, we denote the p^{th} *less than* (LT for short) constraint of $Node(i)$ for destination $Node(d)$ by $LT(i,p,d)$. Thus, for destination $Node(d)$, if the p^{th} less than constraint is due to neighbor $Node(j)$ then $LT(i,p,d)$ is equal to x given in equation(2).

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 ($0 \leq n \leq m$) out of the m potential alternate paths will actually be used for alternate path routing, depending on the *network wide* operational value of b decided by our algorithm. Thus, fixing the network wide value of *operational* b (denoted as $b_{op}(d)$ for destination $Node(d)$) appropriately is crucial for the efficient operation of E-LSR algorithm. 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 $ospfPath(s,d)$. The criterion used for choosing the value of $b_{op}(d)$ is that the total number of alternate paths is maximized subject to the constraint that maximum number of nodes in $ospfPath(s,d)$ 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.

In subsequent discussions, our focus will be on the ospf path $ospfPath(s,d)$ 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 $GT(i,p,d)$ of all the nodes along the ospf path. 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,...,g_m) such that $g_1 < g_2 .. < g_m$, where $g_i, (1 \leq i \leq m)$ are the distinct *greater than* constraint parameters.
- Similarly $SLT_{all}(d)$ contains all the *less than* constraint parameters and $SLT(d)$ is the corresponding ordered list for distinct *less than* constraints i.e. $SLT(d)$ is given by (l_1,l_2,...,l_n), such that, $l_1 < l_2 .. < l_n$, where $l_i, (1 \leq i \leq n)$ 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 $SGT_{all}(d)$.

$$No\_of\_Constraints\_GT(g_1,d) = C_1^g \tag{4}$$

$$\begin{aligned} No\_of\_Constraints\_GT(g_i,d) &= C_i^g \\ +\ No\_of\_Constraints\_GT(g_{i-1},d) \end{aligned} \tag{5}$$

where $1 < i \leq m$.
- 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

$$No\_of\_Constraints\_LT(l_n,d) = C_n^l \tag{6}$$

$$\begin{aligned} No\_of\_Constraints\_LT(l_i,d) &= C_i^l \\ +\ No\_of\_Constraints\_LT(l_{i+1},d) \end{aligned} \tag{7}$$

where $1 \leq i < n$.

Now we introduce the *objective function* that is used for our 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 1 objective_function(low_limit, $high\_limit$, $Path(i,j)$, $Node(d)$)

```
1: int $n = 0,\ m = 0$;
2: for all $node(p)$ in $Path(i,j)$ do
3:   if ($low\_limit \geq GT_{min}(p,d)$) or ($high\_limit \leq LT\_max(p,d)$) then
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 $(N * N * n + m)$. 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.

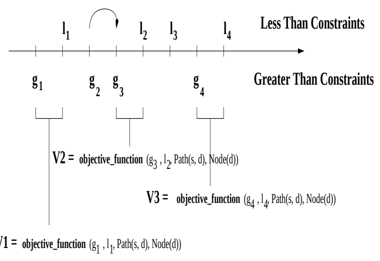
l_1
l_2
l_3
l_4
Less Than Constraints
g_1
g_2
g_3
g_4
Greater Than Constraints
V2 = objective_function (g_3, l_2, Path(s, d), Node(d))
V3 = objective_function (g_4, l_4, Path(s, d), Node(d))
V1 = objective_function (g_1, l_1, Path(s, d), Node(d))

Fig. 2. Coefficient calculation for destination $Node(d)$ along $Path(s, d)$

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: Please refer to [11].

The coefficient calculation routine 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 $ospfPath(s, d)$. It first checks for trivial cases, where there may be only LT constraints (step 3) or only GT constraints (step 5). Then in step 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 Figure 2. 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, step 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 ($b_{op}(d)$) for a destination $Node(d)$. Note that, with little modification, this coefficient calculation algorithm can be used for multiple ingress-egress pairs. It is clear that the time complexity of $coefficient\_calculation()$ is $O(N^3)$. The coefficient calculation algorithm reported in [3] has time complexity of $O(N^4)$. So it is a significant improvement.

IV. Topology Guidelines

Performance of E-LSR depends on nodes having E-LSR alternate paths, which in turn, depends on the topology of the network and the costs assigned to the links. If a node in the ospf path does not have any alternate path to a particular destination node, then the performance of E-LSR algorithm will be affected. To overcome this problem, we propose an algorithm that *may* suggest to establish new links from this node to some other nodes in the topology. But a new link can only be added in topology if the following two constraints are satisfied.

Algorithm 1 coefficient_calculation($ospfPath(s, d)$)

```
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:    $b_{op}(d) = l_1 - \epsilon$, /* where $(l_1 - \epsilon) > 0$ */
      /* $n$ is the number of elements in $SLT(d)$ */
5:  else if $n = 0$ then
6:    $b_{op}(d) = g_m + \epsilon$; /* where $(g_m + \epsilon) < Infinity$ */
7:  else if $l_1 > g_m$ then
8:    $b_{op}(d) = (g_m + l_1)/2$
9:  else
10:   $SGT'(d)$ = insert 0 to the beginning of $SGT(d)$;
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:     $value = objective\_function(g_i, l_j, ospfPath(s, d),$
            $Node(d))$;
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
```

1) **OSPF Constraint:** ospf path should remain unchanged even after making changes to topology. Let $ospfPath(A, D)$ be the ospf path between $Node(A)$ and $Node(D)$. Also, assume that there is no alternate path from $Node(A)$ to destination $Node(D)$ and $Node(B)$ is a neighbor of $Node(A)$ which is not present in $ospfPath(A, D)$. A new link $Link(A, B)$ with link cost $Cost(A, B)$ is put to provide an alternate path for Node(A) to destination $Node(D)$. If ospf path from A to D should remain unchanged then
$$OC(A, D) < OC(B, D) + Cost(A, B) \qquad (8)$$
2) **E-LSR Constraint:** A new link should only be added from $Node(A)$ to $Node(B)$ for destination $Node(D)$ if $Node(B)$ can become eligible alternate next hop of $Node(A)$. $Node(B)$ has to satisfy one of the cases in Theorem 1. Note that the cost of $Link(A, B)$ does not play a role in E-LSR constraint.

In order to explain the algorithm for topology guidelines, we will define these constraints in the following way. $LSR\_constraints(A, B, D, b_{op}(net))$ returns $TRUE$ if $Node(B)$ can become eligible alternate next hop of $Node(A)$ for destination $Node(D)$ when operating E-LSR coefficient is $b_{op}(net)$. Otherwise, it returns $FALSE$. $OSPF\_constraints(A, B, D)$ returns the minimum cost of $Link(A, B)$ for destination $Node(D)$, so that ospf constraint given in equation (8) is satisfied.

Service provider will specify a set of nodes on ospf path for which alternate paths need to be provided. Let this set be $nodes\_to\_be\_protected(D)$ for destination $Node(D)$. The following theorem enables our topology guideline algorithm in making topology decision efficiently.

Theorem 3: Let $Node(i)$ be a node on $ospfPath(s, d)$ from $Node(s)$ to $Node(d)$, for which an alternate next hop is to be found. Let $Node(p)$ be the closest (in terms of cost) neighbor of $Node(d)$ which is not in $ospfPath(s, d)$. For a given operating E-LSR coefficient $b_{op}(d)$, if $Node(p)$ cannot be an eligible alternate next hop for $Node(i)$, then no other node in the network can be an alternate next hop.

Algorithm 2 $Topology\_Guidelines(nodes\_to\_be\_protected[], ospfPath(s,d))$

```
1: Let the *objective function* value that led to existing E-LSR coefficient $b_{op}(d)$ be $OBJ_{b_{op}}$.
2: Find the node in $nodes\_to\_be\_protected(d)$ which has least OSPF cost to destination $Node(d)$. Let this node be denoted as $closest\_node(d)$.
3: Find the closest neighbor of destination $Node(d)$ which is not in $ospfPath(s, d)$. Let this node be denoted as $closest\_neighbor(d)$.
4: if $LSR\_constraints(closest\_node(d), closest\_neighbor(d), Node(d), b_{op}(d))$ then
5:   for all $Node(x)$ $in$ $nodes\_to\_be\_protected(d)$ do
6:     Put $Link(Node(x), closest\_neighbor(d))$ of cost $OSPF\_constraints(Node(x), closest\_neighbor(d), Node(d))$
7:   end for
8: else
9:   for all $Node(x)$ $in$ $nodes\_to\_be\_protected(d)$ do
10:    Assume that $Link(Node(x), closest\_neighbor(d))$ is present in topology.
11:  end for
     /* Due to Theorem 3, we cannot find alternate next hop with $b_{op}(d)$, we need to find a different operating E-LSR coefficient. */
12:  Recalculate E-LSR coefficient $b$ on the new topology for destination $Node(d)$. Let it be $b'_{op}(d)$ and the corresponding objective function value is $OBJ_{b'_{op}}$.
13:  if $OBJ_{b'_{op}} > OBJ_{b_{op}}$ then
14:    $b_{op}(d) = b'_{op}d$.
15:    for all $Node(x)$ $in$ $nodes\_to\_be\_protected(d)$ do
16:      if $LSR\_constraints(Node(x), closest\_neighbor(d), Node(d), b'_{op}(d))$ then
17:        Add $Links(Node(x), closest\_neighbor(d))$ of cost $OSPF\_constraints(Node(x), closest\_neighbor(d), d)$
18:      end if
19:    end for
20:  end if
21: end if
```

Proof: Please refer to [11].

The algorithm for topology guidelines is as shown in Algorithm 2. We first check whether the node from $nodes\_to\_be\_protected(D)$ which has the least OSPF cost to destination $Node(D)$ (say $Node(p)$) can have eligible alternate next hop in topology. If $Node(p)$ can have alternate path then all other nodes from $nodes\_to\_be\_protected(D)$ will have the same eligible alternate next hop for destination $Node(D)$. This is due to the fact that other nodes in $nodes\_to\_be\_protected(D)$ have greater hop count and opsf cost than $Node(p)$. From Theorem 3, if the least cost neighbor of destination $Node(D)$ cannot be eligible alternate next hop for $Node(p)$, then no other node can become eligible alternate next hop for $Node(p)$ (step 8). In that case, we have to recalculate the value of operating E-LSR coefficient. Now if the objective function value corresponding to new value of E-LSR coefficient is less than objective function value corresponding to existing E-LSR coefficient then we continue with the existing value of E-LSR coefficient otherwise we change the value of E-LSR coefficient to the new value. Then, we identify the nodes which can be provided with at least one alternate path with the new value of E-LSR coefficient.

V. Performance Evaluation

A. Simulation Topology

The topology used in our simulation is shown in Figure 1. We have chosen Node A as ingress node and Node G as the egress node for our experiment. The ospf path from node A to node G is A, B, C, D, E, F, G. Thus, QoS will be provided along this path. ospf costs of the links are shown in the figure. Cost of links are assigned according to the guideline given in [12] as follows

$$cost = \lceil 1000000/link\ bandwidth\ in\ bps \rceil \tag{9}$$

All the links along the ospf path 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.

Our simulation is done using NS2 simulator [13]. We have simulated different scenarios as follows.

1) *Scenario A:* This scenario simulates voice traffic along the ospf path. 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). A number of such flows destined to node G originate at node A. Thus, it simulates the scenario of voice flows sent along the ospf path from source node A to destination node G.
2) *Scenario B:* This scenario simulates data traffic along the ospf path from source node A to destination node G. Each flow is Exponential ON/OFF traffic (packet size : 576 bytes [14], mean ON period : 50msec, mean OFF period : 50msec, average rate : 128kbps) and originates from node A and terminates at node G.

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 average rate of 64kbps.

B. Results

For performance comparison between E-LSR, LSR and OSPF algorithms, we have used *average delay* of packets and *percentage packet drop* (PPD) as performance parameters. The number of voice flows and data flows is gradually increased in *Scenario A* and *Scenario B* respectively.

Figure 3 shows the average delay of voice flows in *Scenario A* for different routing algorithms, as the number of voice flows increases. Clearly, average delay in the case E-LSR is lesser than LSR which is lesser than OSPF. The maximum reductions in average delay of E-LSR are 53% and 67% over LSR and OSPF respectively. In the event of congestion, LSR reroutes packets through alternate paths, which leads to lower delay than OSPF. But E-LSR performs even better than LSR in terms of delay. This can be explained as follows. LSR calculates operational coefficient such that the total number of alternate paths in the entire network is maximized. But E-LSR maximizes total number of alternate path subject to the constraint that maximum number of nodes should have at least one alternate path. For our simulation topology, node C does not have any alternate path and node B has three alternate paths when LSR is used, whereas when E-LSR is used, both node B and node C have at least one alternate path. Also, E-LSR always chooses the least cost alternate path among all available alternate paths whereas LSR chooses one randomly. In *Scenario B* the same trend is observed across the three algorithms (Figure 5).

Figure 4 shows the corresponding comparison based on PPD in *Scenario A*. Here also E-LSR has lower PPD than LSR which is lesser than that of OSPF. The maximum reductions in PPD of E-LSR are 71% and 81% over LSR and OSPF respectively. One interesting observation for E-LSR is that the PPD decreases even when the number of flows increases from 2 to 6. This is because, for that range of number of flows, E-LSR started rerouting packets through alternate paths, which are not much congested. The effect of E-LSR alternate path routing is experienced more as the number of flows increases, which leads to decrease in PPD. But when number of flows increases beyond 6, the alternate paths also become congested. Hence, PPD increase beyond this point. The same behavior is

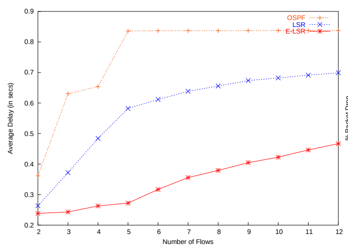

Fig. 3. Average Delay vs Number of Flows (Scenario A)

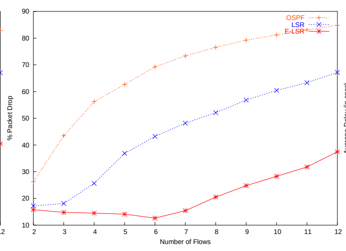

Fig. 4. Percentage Packet Drop vs Number of Flows (Scenario A)

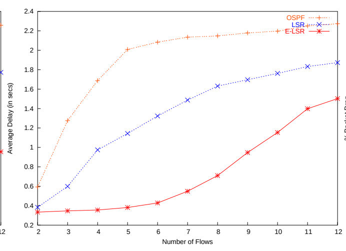

Fig. 5. Average Delay vs Number of Flows (Scenario B)

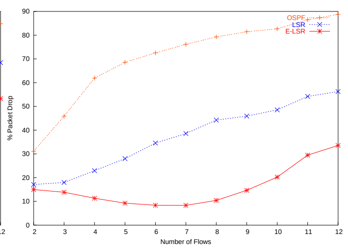

Fig. 6. Percentage Packet Drop vs Number of Flows (Scenario B)

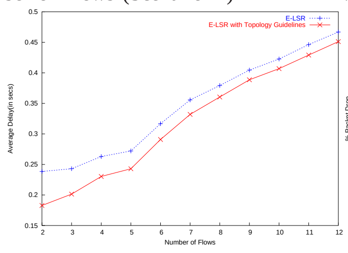

Fig. 7. Average Delay vs Number of Flows (Scenario A with topology guideline)

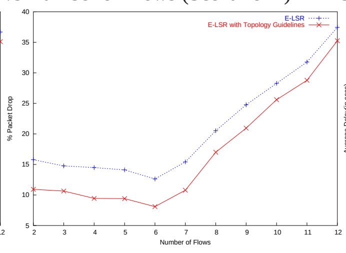

Fig. 8. Percentage Packet Drop vs Number of Flows (Scenario A with topology guideline)

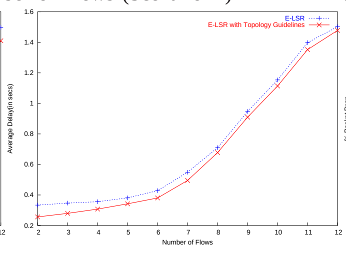

Fig. 9. Average Delay vs Number of Flows (Scenario B with topology guideline)

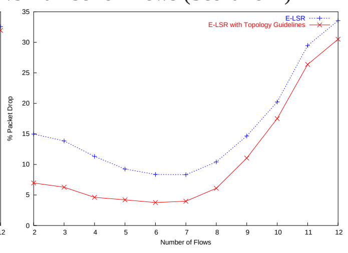

Fig. 10. Percentage Packet Drop vs Number of Flows (Scenario B with topology guideline)

not exhibited by LSR algorithm because it chooses an alternate path randomly. This leads LSR algorithm to choose inefficient alternate paths. The same trend is observed across the three algorithms in *Scenario B* (Figure 6).

In order to show the effectiveness of *topology guidelines*, we specified $nodes\_to\_be\_protected(G) = \{D, E, F\}$ (since they do not have alternate path). The closest neighbor of destination node G which is not in $ospfPath(A, G)$ is node K (or node Q). Since node K is not an eligible alternate next hop of node F (which is the closest node to the destination node G among the list of $nodes\_to\_be\_protected(G)$), the E-LSR coefficient is recalculated. The new E-LSR coefficient corresponds to a better *objective function* value. Using the new E-LSR coefficient we find that only node D, from the list of $nodes\_to\_be\_protected(G)$, can have node K as an eligible alternate next hop. So the algorithm suggests to establish a new link from node D to node K. The results after adding the $Link(D, K)$ are as follows. Figures 7 and 8 show average delay and PPD respectively in *Scenario A*, after a link from node D to node K is added. The maximum reductions in average delay and PPD are 23% and 36% respectively. Similarly, Figures 9 and 10 show average delay and PPD respectively corresponding to *Scenario B*. The figures clearly show that average delay and PPD improves after following the *topology guideline*.

VI. Conclusion and Future Work

We have presented an efficient load sensitive QoS routing algorithm (E-LSR) that provides loop free routing via alternate paths in the event of congestion. We reported a more efficient method of calculating the operational coefficient of the algorithm. Further, the optimization function chosen was such that the total number of alternate paths is maximized while maximum number of nodes have atleast one alternate path. We have shown, through simulation, that E-LSR performs better than LSR and OSPF in terms of delay and percentage packet drop. We provided topology guidelines for implementers so that they can change their network topology to make E-LSR more effective.

We intend to look at the effect of route flapping in the performance of E-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 want to 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. In this paper, we provided topology guideline for one ingress-egress pair. We would like to extend it to multiple ingress-egress pairs.

Acknowledgement

This work was partially supported by Indian Institute of Technology, Bombay, under grant number 03IR059 and by Sybase Software (India) Private Limited.

References

[1] "SIP: Measurement-Based CAC for SIP." `http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcacsip.htm`.

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

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

[4] 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.

[5] J. L. Sobrinho, "Algebra and algorithms for qos path computation and hop-by-hop routing in the internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 541–550, 2002.

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

[7] 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.

[8] A. Segall,P. Bhagwat and A. Krishna, "QoS Routing Using Alternate Paths," *Journal of High Speed Networks*, vol. 7, no. 2, pp. 141–158, 1998.

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

[10] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, A. Przygienda, and D.Williams, "QoS routing mechanisms and OSPF extensions," *Internet Request for Comments (RFC2676)*, April 1999.

[11] A. Tiwari, *Providing Quality of Service Support in OSPF Based Best Effort Network*. M.Tech thesis, IIT Bombay, 2005.

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

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

[14] D. J. Mogul and S. Deering, "Path MTU discovery," *Internet Request for Comments (RFC1191)*, November 1990.