Developing Scalable Protocols for Three-Metric QoS-Routing

Luís Henrique M. K. Costa [a,b] Serge Fdida [a]
Otto Carlos M. B. Duarte [b]

[a] *LIP6 – Université Pierre et Marie Curie*
4, place Jussieu - 75252 - Paris Cedex 05 - France

[b] *Grupo de Teleinformática e Automação – GTA*
Universidade Federal do Rio de Janeiro – COPPE/EE
P.O. Box 68504 - 21945-970 - Rio de Janeiro - RJ - Brasil [1]

Abstract

Quality of Service (QoS) based routing provides QoS guarantees to multimedia applications and an efficient utilization of the network resources. Nevertheless, the QoS routing algorithm must be simple because a costly procedure does not scale with the size of the network. This paper proposes and analyzes the performance of the Single Mixed Metric (SMM) routing protocol in two versions: one based on distance-vectors and the other on link-states. SMM takes into account three metrics: propagation delay, available bandwidth, and loss probability. A heuristic based on the residual loss probability and metric-combination is used to turn the algorithm scalable and solvable in polynomial time. The simulation results show that SMM outperforms other solutions that implement QoS routing based on three metrics.

Key words: Routing, Quality of Service, Scalability

1 Introduction

The next generation of the Internet will offer new classes of service and Quality of Service (QoS) based routing is one of the main components to provide QoS guarantees to the applications. QoS routing helps providing these guarantees since the QoS delivered to flows is taken into account to make route decisions [1]. Other approaches such as to implement adaptive applications [2] or to

Email addresses: Luis.Costa@lip6.fr, Serge.Fdida@lip6.fr, otto@gta.ufrj.br.
1 FUJB, CNPq, CAPES/COFECUB, and IST GCAP N^0 1999-10504.

use the multicast service [3] can be combined to further improve the received quality of service as well as the network resource utilization.

Classical routing generally considers a single metric in route computation, such as delay or hop count. QoS routing is more complex because the routing decisions are based on n metrics [4,5]. The routing algorithms may treat the metrics in different ways depending on the metric correlation. According to the traffic model employed, the end-to-end transmission delay depends on the link capacity, reserved bandwidth, propagation delay, and link loss probability. The metric relationship depends on the queuing discipline that routers use [5]. The other approach, used in this paper, is to consider metrics that are independent. For instance, the propagation delay is independent from the available bandwidth. In this case, the computational complexity of the routing algorithm is the main issue. The proposals found in the literature treat the metrics in one of three ways: individually, as an overall cost (or single mixed metric), or by means of heuristics.

Multiple individual metrics - The complexity of the algorithm is function of the number of metrics treated and of their composition rules. Wang and Crowcroft [4] proved that the problem of finding a path subject to constraints on n additive and m multiplicative metrics is NP-complete if $n + m \geq 2$. Bandwidth can be taken into account by pruning infeasible links. Therefore, algorithms of acceptable complexity can only handle bandwidth and one additional metric.

Single mixed metric - Since algorithms based on a single metric are well known, a direct approach is to combine the different QoS parameters in a single mathematical function such as $f(p) = \frac{B(p)}{D(p) \times L(p)}$, where $B(p)$, $D(p)$, and $L(p)$ are respectively the bandwidth, delay, and loss rate of path p [6,7]. This simple solution has two drawbacks: computing a path based on $f(p)$ does not guarantee each QoS parameter individually and, as the composition rules of the different parameters are different, the composition rule of this measure is complex to define.

Heuristics - There are different heuristic proposals for QoS routing [8,9,7,10]. The most performing heuristics in terms of network resource utilization suppose centralized routing. One node knows the entire network topology and the set of flows to be carried. This approach introduces, however, a single point of failure in the network and requires a large amount of memory and processing capacity as the network size grows. Distributed route-computation is more adequate for large networks and flow sets, and when the flow set is dynamic.

To simplify the protocols, some authors considered the use of precomputed paths to minimize the setup overhead [11,12,13], or the reduction of the resource information accuracy to minimize the message overhead [14,15,16], for example by setting the frequency updates are produced in a link-state protocol. A general consensus is that per-flow state in routers should be avoided whenever possible because this approach does not scale [17,18]. A more suitable solution is to use per-flow state only for special flows such as the ones that require very stringent QoS parameters. Less demanding flows can be provided a connectionless service, where routing is done per class of service instead of per flow. We believe that combining the class-of-service approach with per-flow routing for a few flows may increase the routing efficiency. We therefore propose two different approaches. The first one is a distance-vector routing protocol that considers a connection-less service and provides per class-of-service QoS guarantees. The latter is a link-state protocol that provides per-flow guarantees within a connection-oriented resource reservation mechanism. This paper briefly presents two algorithms for unicast route computation based on distance-vectors (SMM-DV)[19] and link-states (SMM-LS)[20]. The key idea of SMM is to prune the links that do not have enough bandwidth and to combine the use of a single mixed metric with a simple heuristic that turns the routing algorithm scalable. This single mixed metric combines the propagation delay with the link loss probability.

The performance analysis compares SMM-DV and SMM-LS with other possible solutions for DV distance-vector (DV) and LS link state (LS) QoS routing. The performance metric used to evaluate the stateless DV protocols is the computational complexity. For the LS protocols, which keep per-flow state, the call acceptance probability was evaluated. The results show that both algorithms are promising solutions for three-metric route computation as they outperformed the alternative approaches.

This paper is organized as follows. Section 2 presents the SMM-DV and SMM-LS algorithms. Sections 3 and 4 present the results obtained through simulation. Section 5 concludes the paper.

2 SMM Routing Algorithms

Our objective is to propose simple algorithms that can scale instead of trying to find the optimal solution to the routing problem. Therefore, the proposed SMM routing algorithms consider three independent metrics: the propagation delay, the available bandwidth, and the link loss probability. The key idea of SMM is to prune links (or paths) that do not have enough bandwidth and then to run a modified version of classical routing algorithms based on a single mixed metric. This single mixed metric combines delay and loss probability, but uses the absolute value of the logarithmic transmission-success probability function (*slog*) instead of the loss probability to avoid complex composition rules.

$$slog(p) = |log(1 - L(p))|. \tag{1}$$

Furthermore, we assume that routes can not have more than 90% of loss probability, that is, $0 < slog(p) < 1$. As ms unit is appropriate [21] to represent delay, we assume that delay is an integer. As a consequence, we have a simple single-metric representation of delay and loss, where the integer part is delay and the decimal part represents loss. This property allows the construction of simple algorithms capable of finding paths taking into account the three QoS requirements mentioned. The next sub-sections give the details of the DV and LS variants of SMM.

2.1 The SMM-DV Algorithm

This section presents the Single Mixed Metric - Distance Vector (SMM-DV) algorithm for hop-by-hop routing. The usual approach in current hop-by-hop routing algorithms is to compute the best path to every destination. For a single metric, the best path is easily defined, for multiple metrics, the best path with optimal values for all parameters may not exist at all. Actually, the objective of QoS routing is not to find the optimal path, but a feasible path given some QoS constraints. Nevertheless, finding a feasible path is inadequate for distributed route computation based on distance-vectors. Permanent loops may be formed since routers do not know the entire topology. For example, consider a class of service defined by a maximum delay of 6 time units and a minimum of 3 bandwith units, and the following topology:

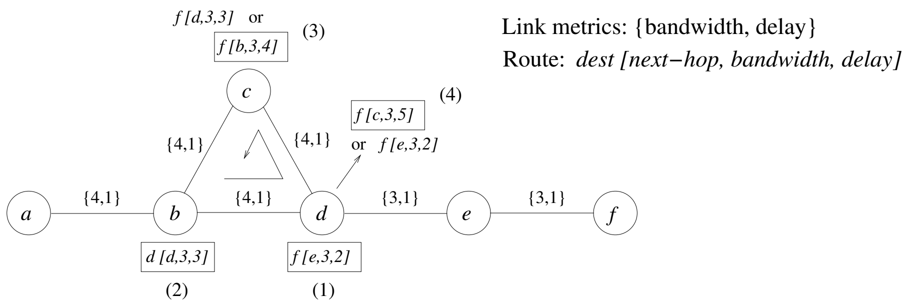


Figure 1. Example of loop creation.

Consider the route from the different nodes to node f. Node d reaches f through e with metrics {3,2} (Figure 1(1)) and announces this route to its neighbors, c and b. Thus b decides it can reach f through d, with metrics {3,3} (Figure 1(2)). Node b also announces to its neighbors that it has a route to f through d. Node c receives both announcements (from d and b and can choose between b{3,4} and d{3,3}. Suppose c chooses b (Figure 1(3)), this is possible since the two routes are constrained by the maximum delay and minimum bandwidth. Now d has two options, $c\{3,5\}$ and $e\{3,2\}$. Nothing prevents d from choosing c and creating a loop (Figure 1(4)), since the two routes meet the QoS constraints.

To avoid this problem, we adopted the approach proposed by Wang and Crowcroft [4] where the shortest-widest path is loop-less. In fact, the "shortest" property avoids loops. This design decision guarantees the absence of loops at the expense of not finding some feasible paths. [2] In the algorithm that follows, the bandwidth precedes the delay metric. This centralized version of the algorithm is based on the Bellman-Ford's algorithm for distance-vector route computation.

Let $G = (N, A)$ be a network with N nodes and A arcs. Each arc (i, j) is assigned two real numbers: the available bandwidth, b_{ij}, and the single mixed metric, $u_{ij} = slog_{ij} + d_{ij}$, where $slog_{ij}$ is the logarithmic transmission-success probability and $d_{i,j}$ is the propagation delay. When arc (i, j) is inexistent $b_{ij} = 0$ and $u_{ij} = \infty$. Given a path $p = (i, j, k, ..., q, r)$, the path width, $w(p)$, is $min[b_{ij}, b_{jk}, ..., b_{qr}]$ and its length is $l(p) = u_{ij} + u_{jk} + ... + u_{qr}$. Given three constants, the minimum bandwidth, **B**, the maximum delay, **D**, and the maximum logarithmic transmission-success probability, **P**, where the maximum single mixed metric is $\mathbf{U} = \mathbf{D} + \mathbf{P}$; the routing problem is to find a path, p, from i to r such that $w(p) \geq \mathbf{B}$ and $l(p) \leq \mathbf{U}$: in our case, the shortest-widest path with three constraints.

Suppose node 1 is the source and h the number of arcs away from this node. Let $B_i^{(h)}$ and $U_i^{(h)}$ be the width and the length of the shortest-widest path from node 1 to node i within h hops. By convention, $B_1^{(h)} = \infty$ and $U_1^{(h)} = 0$ for all h. Adding a length checking, when multiple widest paths are found, produces the shortest-widest path algorithms. The SMM-DV algorithm is as follows:

`Step 1:` $\forall i \neq 1 : h = 0$ `and` $B_i^{(0)} = 0$

`Step 2:` $\forall i \neq 1$, `find`
$K = \{k \mid w(1, ..., k, i) = \max_{1 \leq j \leq N}[\min[B_j^{(h)}, bji]] \wedge w(1, ..., k, i) \geq \mathbf{B}\}$.

`Step 3:` $\forall i \neq 1$, `if` $\#K > 1$, `find` $k \in K \mid$
$l(1, ..., k, i) = \min_{1 \leq j \leq N}[U_j^{(h)} + u_{ji}] \wedge l(1, ..., k, i) \leq \mathbf{U} \wedge slog(1, ..., k, i) \leq \mathbf{P}$.

`Step 4:` $B_i^{(h+1)} = w(1, ..., k, i)$ `and` $U_i^{(h+1)} = l(1, ..., k, i)$.

`Step 5: If` $h \geq A$, `the algorithm stops.`
`If not,` $h = h + 1$ `and go to step 2.`

Step 2 finds the maximum-bandwidth paths from node 1 to node i within h

[2] E.g., the shortest-widest route may not meet the delay constraint, while another route with less bandwidth (although still enough) meets the delay requirement.

hops (and tests if the bandwidth is enough). When multiple widest paths are found, the minimum-length path is chosen (step 3). Step 3 also checks if the chosen path has a *slog* value smaller than **P**, where $slog_{ij}$ is the decimal part of u_{ij}. Thus the loss probability is guaranteed with no additional complexity, thanks to the specific mixed metric representation we proposed. Step 4 updates width and length of the path from node 1 to node i.

The optimal approach takes into account the metrics separately. This leads to a similar algorithm, where step 2 chooses the widest path, step 3 chooses the shortest path (minimum delay) and an additional step chooses the path with minimum loss probability. This additional search turns the approach unscalable. The optimal algorithm uses two path lengths instead of one, so $U_i^{(h)}$ is replaced by $D_i^{(h)}$ and $Slog_i^{(h)}$. The optimal algorithm is as follows:

`Step 1:` $\forall i \neq 1 : h = 0$ `and` $B_i^{(0)} = 0$
`Step 2:` $\forall i \neq 1$`, find`
$K = \{k \mid w(1,...,k,i) = \max_{1 \leq j \leq N}[\min[B_j^{(h)}, bji]] \wedge w(1,...,k,i) \geq \mathbf{B}\}$.
`Step 3:` $\forall i \neq 1$`, if` $\#K > 1$`, find`
$Y = \{y \in K \mid d(1,...,y,i) = \min_{1 \leq j \leq N}[D_j^{(h)} + d_{ji}] \wedge d(1,...,y,i) \leq \mathbf{D}\}$.
`Step 4:` $\forall i \neq 1$`, if` $\#Y > 1$`, find` $y \in Y \mid$
$slog(1,...,y,i) = \min_{1 \leq j \leq N}[Slog_j^{(h)} + slog_{ji}] \wedge slog(1,...,y,i) \leq \mathbf{P}$.
`Step 5:` $B_i^{(h+1)} = w(1,...,y,i)$`,` $D_i^{(h+1)} = d(1,...,y,i)$ `and` $Slog_i^{(h+1)} = slog(1,...,y,i)$.
`Step 6: If` $h \geq A$`, the algorithm stops.`
`If not,` $h = h + 1$ `and go to step 2.`

The distributed version of these algorithms is simple. Each router periodically receives distance-vectors from its neighbors and chooses the best route among the possible options. SMM-DV takes advantage of the proposed single metric to find the route consuming fewer steps.

2.2 The SMM-LS Algorithm

This section presents the version of SMM based on link-states (SMM-LS) [20]. This algorithm assumes that routes are fixed after connection setup using some kind of explicit routing or pinning. The key idea of SMM-LS is to execute a search in the network graph to eliminate all arcs that do not meet the bandwidth restriction and then to run a modified version of Dijkstra's algorithm. In this scheme, the source knows the network topology.

Consider the same network representation, $G = (N, A)$, of the previous section. If arc (i, j) does not exist $b_{ij} = 0$ and $u_{ij} = \infty$. Given a path $p = (i, j, k, ..., q, r)$, the path width, $w(p)$, is $min[b_{ij}, b_{jk}, ..., b_{qr}]$ and its length is $l(p) = u_{ij} + u_{jk} + ... + u_{qr}$. Given three constants, **B** the minimum bandwidth,

$\mathbf{D}$ the maximum delay, and $\mathbf{P}$ the maximum logarithmic transmission-success probability ($\mathbf{U} = \mathbf{D} + \mathbf{P}$ is the maximum single mixed metric), the problem is to find a path, p, between i and r such that $w(p) \geq \mathbf{B}$ and $l(p) \leq \mathbf{U}$.

Given a path p with x hops, if $slog_{ij}$ for each link (i, j) in p is smaller than $\mathbf{P}/x$, then $slog(p)$ is smaller than $\mathbf{P}$. Nevertheless, x is unknown during the path computation. The idea is then to restrict $slog_{ij}$ to $\mathbf{P}/C$, where C is the length of the longest path on the network graph (this is a pessimistic hypothesis). If $slog_{ij}$ is limited to $\mathbf{P}/C$ at each step, then $slog(p)$ is smaller than $\mathbf{P}$. This limitation may be, however, too strong because the route may have a hop-length smaller than C.

Thus, an alternative scheme is proposed to relax this restriction. At each algorithm step, a hop remainder (SS_i) and a logarithmic probability remainder (SP_i) are associated with node i. The hop remainder is the difference between the length of the longest path in the graph and the length of the candidate path from the source to node i. The probability remainder is the difference between the required limit and the present value of the logarithmic transmission success of the candidate path from the source to node i. The maximum logarithmic probability that can be added at each step is limited to SP_i/SS_i instead of $\mathbf{P}/C$. This value can be larger than $\mathbf{P}/C$, in the case where SP_i is smaller than $SP_i \times (\mathbf{P}/C)$. Therefore, the probability of finding a path in the network graph grows. Supposing that node 1 is the source, SMM-LS is as follows:

Step 1: $\forall i, j : u_{ij} = \infty$ if $b_{ij} < \mathbf{B}$.

Step 2: $L = \{(1, 1)\}$; $SS_1 = C$; $SP_1 = \mathbf{P}$.
    $\forall i, i \neq 1$:
        $U_i = u_{1i}$ and $ph_i = 1$
        if $(U_i == \infty)$, $SS_i = C$, $SP_i = \mathbf{P}$;
            else $SS_i = C - 1$, $SP_i = \mathbf{P} - (u_{1i} - \lfloor u_{1i} \rfloor)$.

Step 3: Find $k \notin L \mid (U_k = \min_{i \notin L} U_i$ and $U_i - \lfloor U_i \rfloor \leq \mathbf{P})$.
    If $U_k > \mathbf{U}$, a path cannot be found and the algorithm stops.
    $L := L \cup \{(k, ph_k)\}$.
    If L contains node m, a path was found and the algorithm stops.

Step 4: $\forall i \notin L$:
    If $((U_k + u_{ki} < U_i)$ and $(u_{ki} - \lfloor u_{ki} \rfloor < SP_i/SS_i))$,
      or if $((U_k + u_{ki} < \mathbf{U})$ and $(\frac{SP_k - \lfloor u_{ki} \rfloor}{SS_k - 1} < \frac{SP_i}{SS_i}))$,
      do
        { $U_i := U_k + u_{ki}$.
          $ph_i := k$.
          $SS_i := SS_k - 1$.
          $SP_i := SP_k - \lfloor u_{ki} \rfloor$. }
    Go to step 3.

Step 4 checks if node k is a better previous hop to the i nodes that are not in the path being constructed (L). If choosing k provides a smaller length, than it is chosen as previous hop according to the original Dijkstra's algorithm. The second test in line 2 of Step 4 guarantees the remainder ratio of our heuristic. Line 3 provides a second possibility of choosing k as previous hop: even if the path length is larger passing through k, it may be chosen as previous hop if this new length is feasible and the remainder ratio is better. This heuristic increases the probability of finding a feasible path.

Step 1 consists of $O(N^2)$ operations and is executed once. Step 2 requires $O(N)$ operations and is also executed once. Step 3 and 4 require $O(N)$ operations and are repeated $N - 1$ times in the worst case. Thus the complexity of the above algorithm is $O(N^2)$, the same as the original Dijkstra's algorithm.

The SMM-LS heuristic can be improved trading off some additional complexity. The idea is to repeat the Dijkstra search for different values of C (say $\mathbf{C} = \{C_i\}$), thus increasing the probability of finding a feasible path. This means that steps 2 to 4 are repeated $|\mathbf{C}|$ times. The complexity of the algorithm then turns $O(|\mathbf{C}| \times N^2)$. As long as $|\mathbf{C}|$ is small the complexity of SMM-LS is acceptable.

3 SMM-DV Simulation

We implemented the SMM-DV algorithm inside NS [22]. This simulator already provides a dynamic implementation of the Bellman-Ford routing algorithm that is the base of our work. The implementation consists of two new routing modules and a modified link implementation to admit three metrics: propagation delay, available bandwidth, and packet loss probability. The packet classifier takes into account the packet type and maps it to one of the classes of service.

Each class of service (CoS) is characterized by specific QoS guarantees on bandwidth, delay, and loss probability. Within a specific class of service one requirement is privileged.[3] Table 1 presents the classes of service used in our implementation. The QoS values are not based on real applications but arbitrarily chosen to differentiate the classes of service. Each service has one strong guarantee: delay (CoS-0), bandwidth (CoS-1), or loss probability (CoS-2).

[3] For example, an application may have a strong requirement on the end-to-end delay but low bandwidth consumption (e.g., voice traffic). We can also consider that for some applications the most important is to have high bandwidth available (e.g., video transmission), and real-time applications that demand high reliability (e.g., telemedicine) for which the loss probability must be minimized.

Table 1
Classes of Service.

	QoS guarantees		
CoS	Delay	Bandwidth	Loss probability
0	**10 ms**	1 $Mbps$	0.04
1	40 ms	**5 Mbps**	0.04
2	40 ms	1 $Mbps$	**0.01**

The optimal algorithm for CoS-1 is exactly the one presented in Section 2.1. Bandwidth is optimized, then delay and loss probability are minimized. The optimal algorithm for CoS-0 is obtained by inverting the order of steps 2 and 3 (so delay is first minimized); for CoS-2 step 4 precedes step 2 (so loss probability is first optimized).

The SMM-DV algorithm for CoS-1 is the one presented in Section 2.1, where bandwidth is first optimized, then the single mixed metric is minimized. The CoS-0 algorithm first finds the paths that minimize the single mixed metric (they also minimize delay) and, among these paths, it chooses the one with maximum bandwidth. SMM-DV for CoS-2 finds the paths with minimum loss probability (the rational part of the single mixed metric) and afterwards the widest path among these. Delay is not minimized but the path is delay-constrained.

Our proposal was evaluated through the implementation of two protocols. The first one (3M) uses the optimal algorithm that searches the route taking into accont all the three metrics one by one. The second protocol uses the SMM-DV algorithm. SMM-DV does not find the route with optimal values for all three metrics, as does 3M, but finds a route constrained by the QoS requirements defined for each class of service. Therefore the routes may be different for some source-destination pairs.

The performance metric is the time complexity [23] expressed in steps: when a node receives routing update messages it recomputes its routes. A probe counts the number of loops needed by the routing algorithm to compute routes. This measure reflects the processing requirements of the algorithm. In the experiments link failures produced routing updates.

3.1 Topologies

Several topologies were produced by the GT-ITM software [24]. The flat model was used. Nodes are placed on a plane with a grid: with a probability p a link is put between a pair of nodes; link delay is proportional to the distance between the nodes in this grid. The generator was modified to produce values for link delay, bandwidth, and loss probability. These values are just used for route computation. Physically, all links have the same bandwidth, propagation delay, and loss probability. The objective is to measure the algorithm complexity. We generated topologies with 10, 15, 20, 25, 30, and 35 nodes (Table 2). The parameter p was decreased for the 35 node topology due to the memory limitations imposed by the simulator.

Table 2
Characteristics of the simulated topologies.

n of nodes	p - GT-ITM	n of links	NxA
10	0.2	12	120
15	0.2	18	270
20	0.2	35	700
25	0.2	54	1350
30	0.2	68	2040
35	0.1	61	2135

The used model does not represent the Internet structure since it is purely random. Nevertheless, it can model one routing domain. This restriction is coherent with the simulation of a DV routing protocol. DV routing is used in real situations with topologies that are not too large because of its intrinsic problems, such as slow convergence [25].

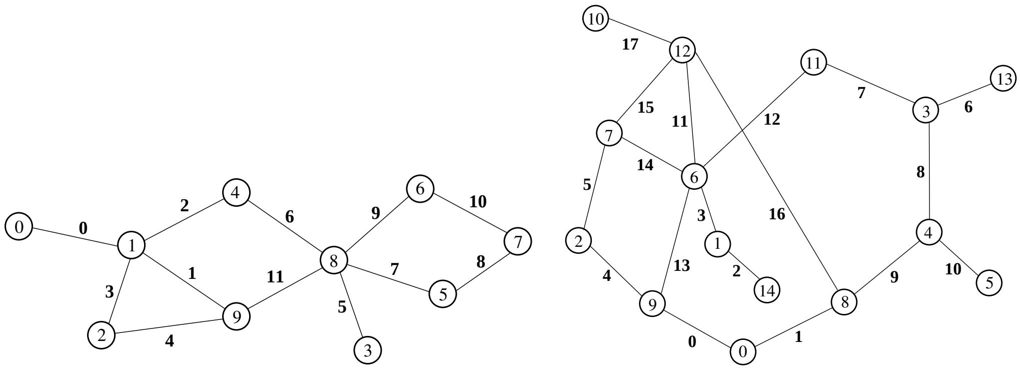
3.2 Experiments

The objective of our experiments is to measure the processing requirements of QoS routing. This is done through the analysis of the trace files produced by the probe mentioned before. Each simulation lasts for 2.5 seconds. During the first moments, all routers exchange messages that enable them to compute routes for every destination in the network. Then, at $t = 1.0s$, one link in the topology goes down, it goes up at $t = 2.0s$. The experiment is repeated for each link in the topology.

The Bellman-Ford algorithm has a problem of convergence: counting to infinity [25]. Protocol configuration limits "infinity" to a finite number (16 is usual with RIP [26]). For metrics other than the hop-count (which is an integer) the definition of infinity is not so evident. Our first experiments presented problems due to too large values of "infinity". With slow convergence, a big number of messages is exchanged between the nodes, this may lead to congestion, at this point routing messages may be lost, this leads to slower convergence and to the production of more routing messages; this leads to more congestion and so on. The choices of the infinity value and of the metric granularity define the convergence speed. Therefore infinity was empirically chosen: 50 for delay, 0 for bandwidth and 0.05 for loss probability. [4] With this approach it was possible to obtain simulations with topologies up to 25 nodes without routing-packet losses. With 30 nodes, there were some losses; with 35 nodes, as the chosen limits for "infinity" became short, therefore some source-destination became unreachable. An option to setting various infinity values is to add the hop-count metric just for counting to infinity. Metric granularity also influences link-state QoS routing since it affects the amount of update messages produced by the routing protocol [11,27].

3.3 Analysis

Figure 2 represents the ten and fifteen-node topologies use in the experiments. Figure 3 plots the processing requirement at each node, each graph corresponds to a single link failure. [5]



(a) 10 nodes. (b) 15 nodes.

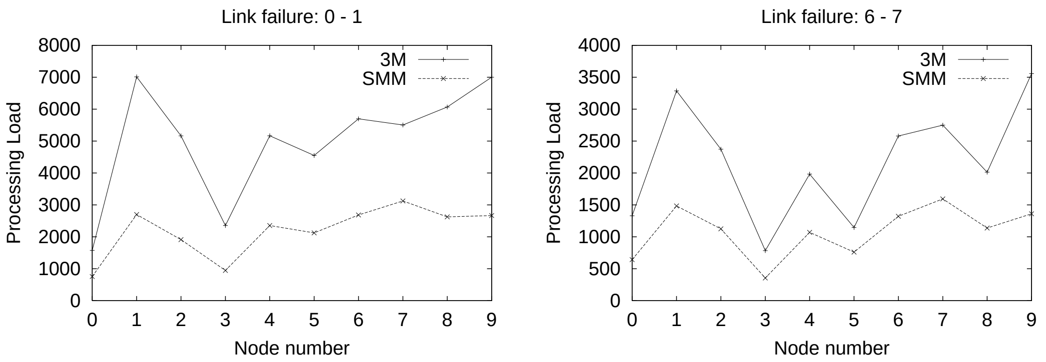
Figure 2. Two of the test topologies.

Nodes that have more neighbors experience a higher processing charge because they have more options for choosing the next hop for each destination and receive more routing messages than nodes with just one neighbor. This is the case of nodes **8**, **1**, and **9** for the 10-node topology and of nodes **6**, **7**, and **12** for the 15-node topology. On the other hand, nodes that have just one neighbor have the lightest loads: **0** and **3** in Figure 2(a) and **5**, **10**, **13**, and

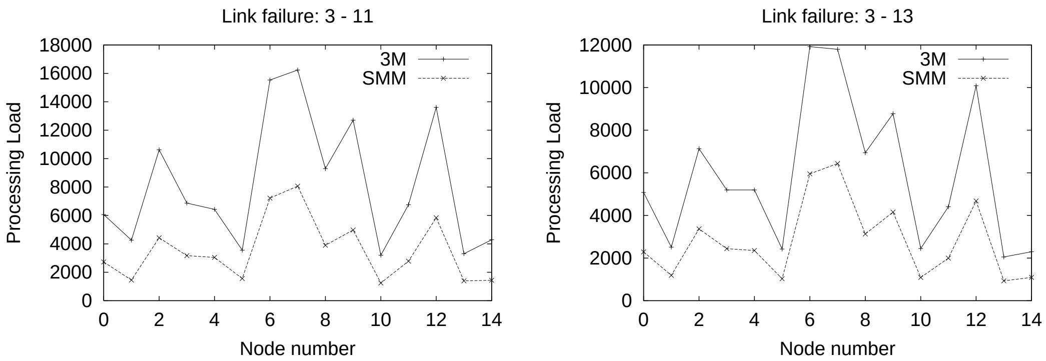
[4] Infinity equals ten times the metric mean value for delay and loss probability. For example, link delay ranges from 1 to 10; infinite delay is 5×10. Bandwidth has no problems since it is concave.

[5] For better visualization the graphs are presented as a continuous line.

14, as well as **1** (that has two neighbors, but one of them is node **14** which is only connected to node **1** itself) in Figure 2(b).



(a) 10-node topology.



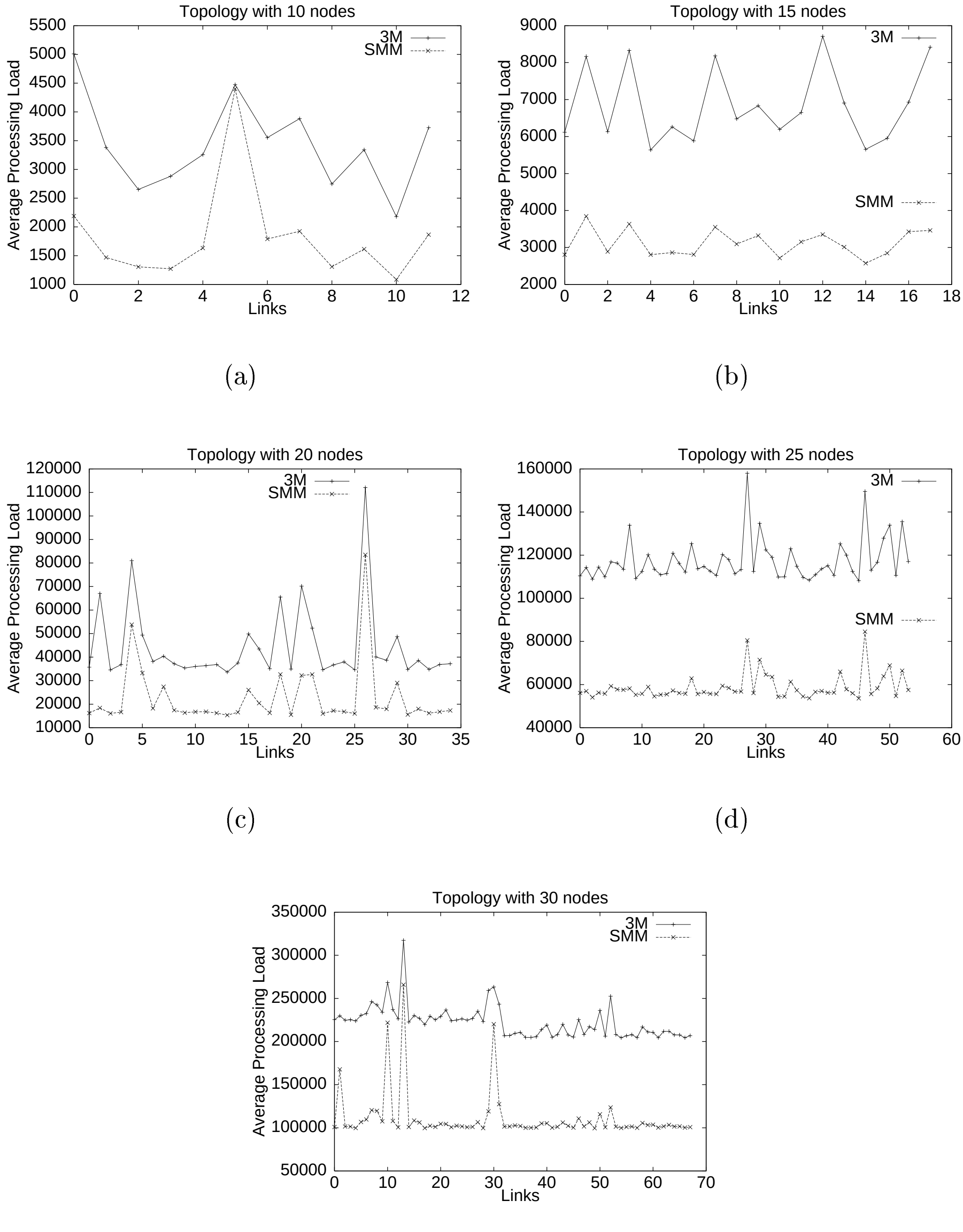
(b) 15-node topology.

Figure 3. Processing requirements at each node.

The failure of link 0_1 produces a higher processing charge than that of link 6_7 in the 10-node topology (the graphs in Figure 3(a) have different scales). In the first case the network is disconnected: node **0** becomes unreachable forcing the other nodes to count to infinity.

Figure 4 plots one graph per topology, each one present the average processing load (on all nodes in the topology) versus the link failure.[6] Topologies with 10, 15, 20, 25 and 30 nodes were randomly genereated by the GT-ITM package. The effect of different link failures is observed. Observe the topologies in Figure 2 and the results in Figures 4(a) and 4(b). The cases that have higher processing requirements are:

[6] Links are numbered as in Figure 2: starting from node zero, all links connected to the node are numbered in the decreasing order of neighbor numbers (e.g. 0_1=0; 1_9=1; 1_4=2; 1_2=3; 2_9=4...).



(a) (b)

(c) (d)

(e)

Figure 4. Global processing requirements.

- when the topology is disconnected (links 0 and 5 in the 10-node topology, and links 3 and 17 in the 15-node topology);
- when the link failure adds to the graph diameter, slowing the routing convergence (links 7 and 9 in the 10-node topology and links 1, 7, and 12 in the 15-node topology).

Although SMM-DV discovers routes more quickly, it may present a slower convergence if 3M uses the delay metric for counting to infinity and SMM-DV uses the single mixed metric which has a finer granularity. Nevertheless, this is not likely to happen as our experiments showed.

Figures 4(a) through 4(e) show that the global processing requirements increase with the topology size. The advantage of SMM-DV over 3M, the optimal algorithm, is as large as 50%.

4 SMM-LS Simulation

To simulate SMM-LS we added a link-state mechanism to NS similar to the one used by OSPF (Open Shortest-Path First) [28]. This basic mechanism is used with different routing algorithms to simulate different protocols. We used the Integrated Services module available in NS to simulate the admission control, connection setup, and bandwidth reservation. By measuring the call blocking probability we evaluated the efficiency of the different algorithms in finding feasible paths and using the network resources.

We compared SMM-LS with five other algorithms. The first one is the classical Dijkstra's shortest-path (DSP) algorithm using the number of hops as metric. Theoretically DSP should have the worst acceptance probability since it does not consider the QoS requirements of the connections. Therefore the results obtained by DSP are the reference to compare the other algorithms. The BandWidth pruning and Delay Dijkstra (BWDD) algorithm implements the proposal of Wang and Crowcroft [4]. The BandWidth pruning and Mixed metric Dijkstra (BWMD) algorithm is the same as BWDD, but the Dijkstra search is based on a single mixed metric of delay and loss probability. SMM-LS is our proposed algorithm. The Overall Cost (OC) algorithm implements a Dijkstra search based on an overall cost function. The Scaled OC (SOC) algorithm is a modified version of the overall cost approach, where the weights of the three metrics are scaled so as to have the same importance in the cost function. The scaling factors were chosen according to the link metrics.

4.1 Experiments

Figure 5 shows the topology used in our experiments. This topology is typical of a large Internet Service Provider (ISP) network and similar to that used in [11]. Link propagation delays and loss probabilities are uniformly distributed on the interval $[min, max]$ as presented in Table 3. Link bandwidth is homogeneous and equal to 10 $Mbps$.

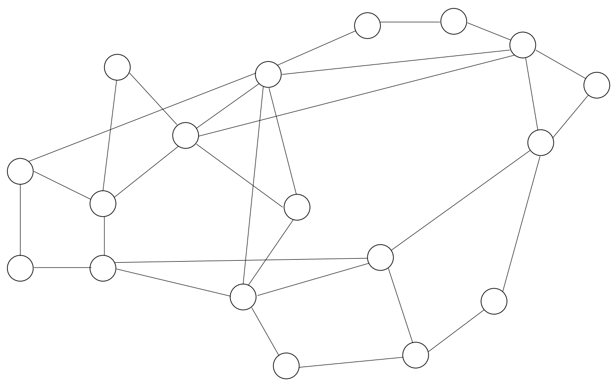


Figure 5. The test topology.

Table 3
Characteristics of the simulated topology.

Link bandwidth	10 $Mbps$
Number of nodes	18
Number of links	30

	min	max
Link delay	1 ms	10 ms
Link loss	0.001	0.005

Each call in our traffic model consists of a source-destination pair, plus the delay, bandwidth, and loss probability requirements. The calls are supposed to arrive independently at each node, following a Poisson distribution. Call inter-arrival time is exponentially distributed with average $0.4s$. The call holding time is exponentially distributed with $3s$ average. QoS requirements varied in the different experiments. Since we are not concerned with the data traffic of flows, the amount of bandwidth available on a link is determined based on the bandwidth reservations on this particular link. Each simulation lasts for 1500 seconds and is repeated 10 times, leading to approximately 35,000 connection requests for each configuration in the experiments described hereafter.

We first executed three experiments where one QoS requirement is varied and the other two are weak and fixed. The objective is to evaluate the sensitivity of the algorithms to each particular requirement. Table 4 summarizes the characteristics of the connection requests in the different experiments.

Figures 6 through 9 show the results obtained from the different experiments. The X-axis represents the upper bound limit of the QoS requirement interval. The Y-axis is the call acceptance probability in the left-side graphs and the difference from DSP's acceptance probability in the right-side graphs.

In Experiment 1, the bandwidth and loss probability required by the connections are negligible, so the delay requirement dominates the path choice. The delay required is uniformly distributed on $[dl_i, dl_f]$. The upper bound of the interval (dl_f) is varied from min to max as shown in Table 4. The larger is this interval, the looser are the QoS requirements of the calls. In experiment 2, delay and loss probability requirements are fixed. The bandwidth required

Table 4
Characteristics of the connection requests.

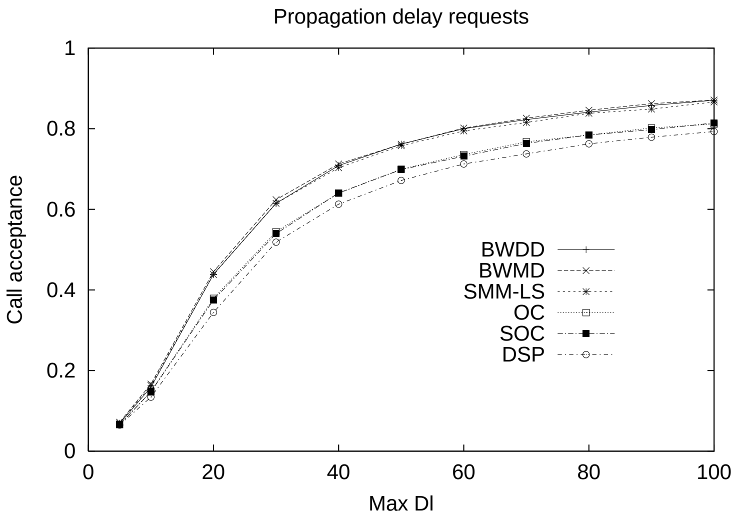
	QoS requirements								
	Delay (ms)			Bandwidth ($\times 64\ kbps$)			Loss probability ($\times 10^{-3}$)		
Experiment	dl_i	dl_f		bw_i	bw_f		lp_i	lp_f	
		min	max		min	max		min	max
1 (dl-req)	1	5	100	$fixed$			$fixed$		
2 (bw-req)	$fixed$			1	10	160	$fixed$		
3 (lp-req)	$fixed$			$fixed$			1	5	100
4 (par-req)	1	5	100	1	10	160	1	5	100

by the connections is uniformly distributed on $[bw_i \times 64\ kbps, bw_f \times 64\ kpbs]$. The parameter bw_f is varied, but in this case, the larger the interval, the stronger the QoS requirements. The left side of Figure 7 shows this effect. The network load is relatively low, as the call acceptance is above 70% all the time. Experiment 3 has delay and bandwidth requirements fixed whereas the loss probability requested is uniformly distributed in the interval $[lp_i, lp_f]$, as shown in Table 4. Here again, as lp_f grows, the QoS requirements are looser so the call acceptance probability increases (Figure 8). In the fourth experiment, the three series of QoS requests are mixed as shown in Table 4. We chose to invert the order of the bandwidth requests, in this way the low-numbered QoS levels in the X-axis of Figure 9 represent tighter requirements (QoS level 0 groups the three tightest requirements on delay, bandwidth, and loss whereas QoS level 10 groups the three loosest requests on the three metrics.).

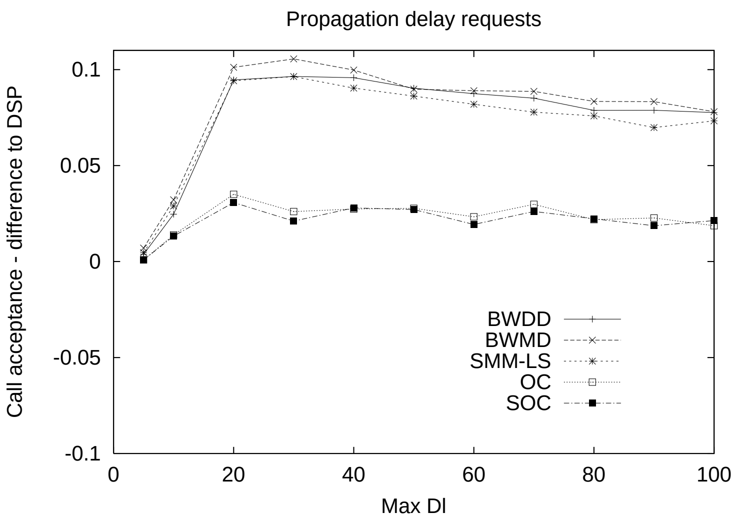
4.2 Analysis

Delay requirement

Figure 6 shows the results obtained from Experiment 1. As expected, DSP yields the worst call acceptance probability. BWDD, BWMD and SMM-LS have approximately the same behavior. This is because the three approaches reduce to a delay Dijkstra search in this experiment. SMM-LS has a little disadvantage because the remainder heuristic may not consider some paths in route computation. OC and SOC present the worst performance showing that, at least for this topology and metrics, the overall cost approach is not well suited. Both OC and SOC suffer from the effect of unbalanced QoS requirements - the minimum limit required for the overall cost $f(p)$ from the route computation algorithm does not reflect the QoS demanded. The reason is that



(a)



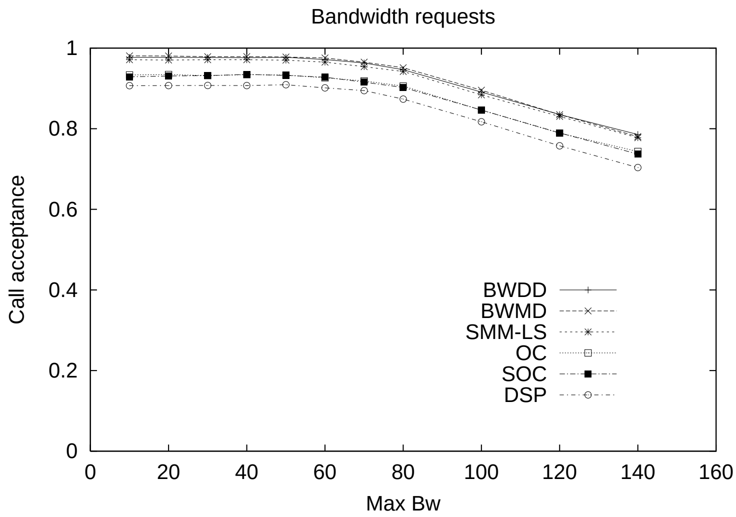
(b)

Figure 6. Delay variation.

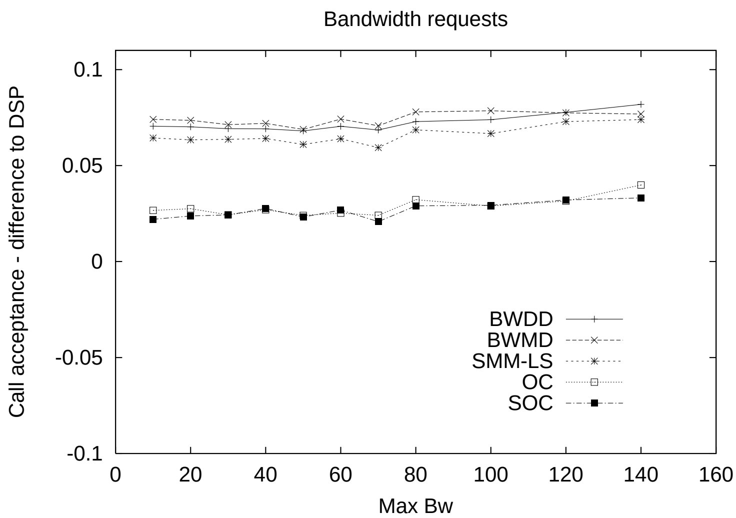
the same $f(p)$ may be obtained with a large $D(p)$ and small $L(p)$ or with a small $D(p)$ and large $L(p)$. The overall cost hides the relative importance of the different QoS requirements.

Bandwidth requirement

Figure 7 shows Experiment 2, where the bandwidth requirement is varied. DSP has the worst acceptance probability since it does not consider bandwidth in route computation. BWDD, BWMD and SMM-LS have similar performance.



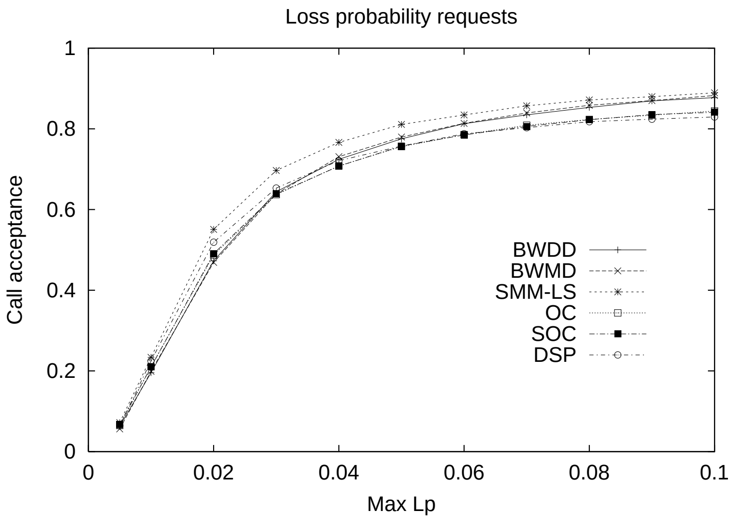
(a)



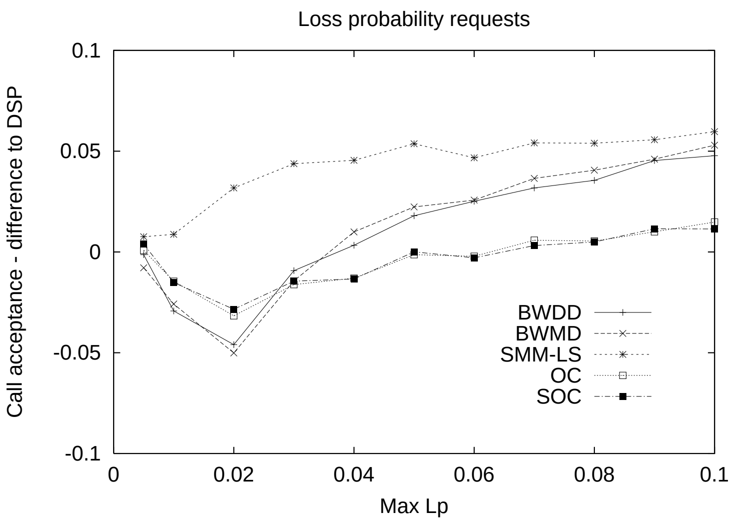
(b)

Figure 7. Bandwidth variation.

This is expected since the three algorithms treat the bandwidth requirement in the same way - by pruning infeasible links. Another time the single-function approaches, OC and SOC, present the worst performance among the QoS-aware algorithms. The reason of this poor performance is the same as in Experiment 1. QoS requirements are very different in scale causing the overall cost to hide the real needs of the connection request.



(a)



(b)

Figure 8. Loss probability variation.

Loss probability requirement

Figure 8 presents the results from the third experiment, where bandwidth and delay requirements are small and fixed whereas the loss probability is varied. This graph shows that the heuristic employed by SMM-LS is effective in guaranteeing the loss probability requirement. When this requirement is too tight (lp_f is small) all algorithms perform bad, even when compared to DSP. BWDD and BWMD perform nearly the same, showing that the use of the single mixed metric in BWMD is not effective. This is due to the domination of delay in the mixed metric. The performances of OC and SOC are similar, but slightly better when compared to Experiments 1 and 2. This is because the loss probability requirement is numerically small, which causes the disequilibrium among the metrics to be smaller and the overall-cost algorithms to perform better.

Three-requirement requests

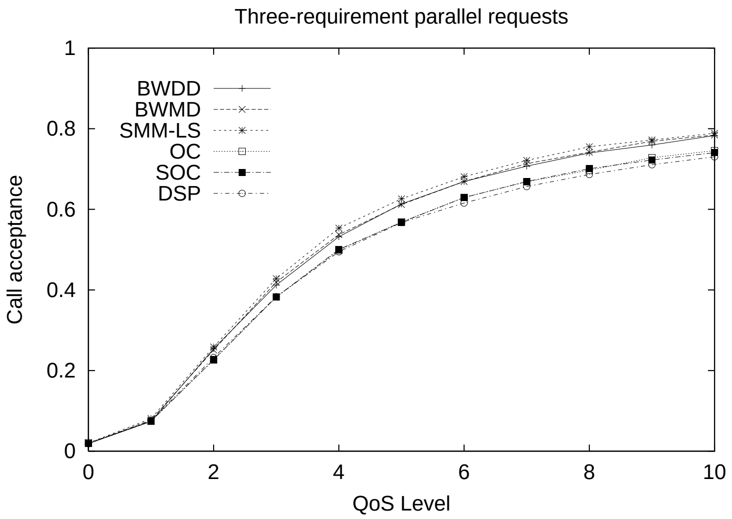
Figure 9 presents the results of Experiment 4. Figure 9(a) shows that the QoS levels in the left side are too tight, thus the call rejection is very high. The looser QoS levels yield a call acceptance probability near 80 %. When the QoS levels are very loose (e.g. QoS levels 9 and 10 in Figure 9(b)), BWDD and BWMD have a performance similar to SMM-LS. This is due to the weak loss probability requirement. On the other hand, when the QoS requirements are too tight (QoS levels 0 and 1) the performances of all algorithms are equivalent. Almost all connection requests are infeasible in this scenario.

The best performance of SMM-LS is in the middle of the QoS-level interval. In this band SMM-LS is able to carry a larger number of flows than all other approaches, guaranteeing a high level of network resource utilization. It is worth noting that this higher acceptance probability is achieved through an additional processing overhead, as we explained in Section 2.1. This processing penalty is fortunately not heavy. In our experiments with the ISP topology, the size of $\mathbf{C}$ equals 9. Nevertheless, $|\mathbf{C}|$ may be easily reduced for this topology, since the larger C_i's were rarely used. Reducing $\mathbf{C}$ will thus weakly impact the call acceptance probability.

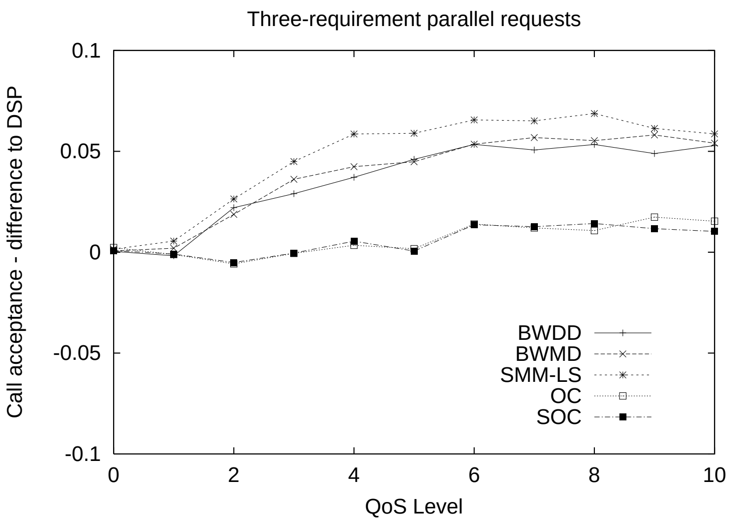
5 Conclusions

QoS routing provides better QoS guarantees to applications and improves the network resource utilization. Nevertheless, routing algorithms have to be carefully designed or the complexity will compromise their implementation. This paper presented two variants of the SMM algorithm which take into account three metrics: propagation delay, available bandwidth, and link loss probability.

SMM-DV is an algorithm for distance-vector QoS routing that computes routes per class of service. SMM-DV is as complex as a two-metric algorithm. We compared the processing requirements of SMM-DV with that of an optimal algorithm, 3M. The average performance gain of SMM-DV over 3M is as large as 50%, and grows with the network size. The network topology influences the processing requirements of DV QoS routing. Nodes that have more neighbors experience bigger processing charge. On the other hand, the



(a)



(b)

Figure 9. Varying QoS levels.

link failure is more critical when the network is disconnected and when the network diameter is increased. Therefore DV QoS routing takes advantage of highly connected networks.

SMM-LS is the link-state protocol which is based on a modified version of Dijkstra's shortest-path algorithm and on the utilization of the remainder-based heuristic proposed. The simulation results show that SMM-LS outperforms all of the other approaches when treating combined QoS requests. The additional complexity of SMM-LS is compensated by the increased call acceptance probability and better network resource utilization.

Combining the approaches used in SMM-DV and SMM-LS can improve the scalability of QoS routing. One possibility is to use an algorithm such as SMM-LS inside a routing domain - where the network topology is known - and SMM-DV for inter-domain since the internal topologies are unknown. A second approach is to modify SMM-LS to keep per-flow state only for long-lived or resource-intensive flows. This approach reduces the size of routing tables and the frequency of routing updates. It also couples QoS routing with traffic engineering, potentially leading to a more efficient utilization of the network resources.

References

[1] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, *A Framework for QoS-Based Routing.* RFC 2386, Aug. 1998.

[2] M. D. de Amorim, O. C. M. B. Duarte, and G. Pujolle, "Multi-criteria arguments for improving the fairness of layered multicast applications," in *Broadband Communications – Networking 2000, Lecture Notes in Computer Science 1815*, pp. 1–11, May 2000.

[3] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte, "Hop by hop multicast routing protocol," in *ACM SIGCOMM'2001*, pp. 249–259, Aug. 2001.

[4] Z. Wang and J. Crowcroft, "Quality of service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, Sept. 1996.

[5] Q. Ma and P. Steenkiste, "Quality of service routing for traffic with performance guarantees," in *IFIP International Workshop on Quality of Service*, pp. 115–126, May 1997.

[6] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *IEEE GLOBECOM'97*, vol. 3, pp. 1903–1908, Nov. 1997.

[7] L. L. H. Andrew and A. A. N. A. Kusuma, "Generalised analysis of a QoS-aware routing algorithm," in *IEEE GLOBECOM'98*, pp. 118–123, 1998.

[8] W. C. Lee, M. G. Hluchyj, and P. A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Network Magazine*, vol. 9, no. 4, pp. 46–55, July-Aug. 1995.

[9] R. Vogel, R. G. Herrtwich, W. Kalfa, H. Wittig, and L. C. Wolf, "QoS-based routing of multimedia streams in computer networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1235–1244, Sept. 1996.

[10] H. F. Salama, D. S. Reeves, and Y. Viniotis, "A distributed algorithm for delay-constrained unicast routing," in *IEEE INFOCOM'97*, vol. 1, pp. 84–91, 1997.

[11] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *ACM SIGCOMM'98*, pp. 17–28, Sept. 1998.

[12] N. Taft-Plotkin, B. Bellur, and R. Ogier, "Quality-of-Service routing using maximally disjoint paths," in *IEEE/IFIP IWQoS'99*, pp. 119–128, June 1999.

[13] A. Orda and A. Sprintson, "QoS routing: The precomputation perspective," in *IEEE INFOCOM'00*, vol. 1, pp. 128–136, Mar. 2000.

[14] B. Rajagopalan and R. Nair, "Multicast routing with resource reservation," *Journal of High Speed Networks*, vol. 7, no. 2, pp. 113–139, July 1998.

[15] D. H. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," in *IEEE INFOCOM'98*, vol. 1, pp. 3–10, 1998.

[16] V. Fayet, D. A. Khotimsky, and T. Przygienda, "Hop-by-hop routing with node-dependent topology information," in *IEEE INFOCOM'99*, vol. 1, pp. 79–87, Mar. 1999.

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

[18] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *ACM SIGCOMM'99*, pp. 81–94, Sept. 1999.

[19] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte, "Distance-vector QoS-based routing with three metrics," in *IFIP High Performance Networking – Networking 2000, Lecture Notes in Computer Science 1815*, pp. 847–856, May 2000.

[20] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte, "A scalable algorithm for link-state QoS-based routing with three metrics," in *IEEE International Conference on Communications*, June 2001.

[21] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on delay and hop-count of the Internet," in *IEEE GLOBECOM'98 - Internet Mini-Conference*, 1998.

[22] K. Fall and K. Varadhan, *The ns Manual.* UC Berkeley, LBL, USC/ISI, and Xerox PARC, Jan. 2001. Available at http://www.isi.edu/nsnam/ns/ns-documentation.html.

[23] J. J. Garcia-Luna-Aceves and J. Behrens, "Distributed, scalable routing based on link-state vectors," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1383–1395, Oct. 1995.

[24] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *IEEE INFOCOM'96*, vol. 2, pp. 594–602, Mar. 1996.

[25] C. Huitema, *Routing in the Internet.* Prentice Hall, 2nd ed., 1999.

[26] G. Malkin, *RIP Version 2.* RFC 2453, Nov. 1998.

[27] A. Shaikh, J. Rexford, and K. G. Shin, "Evaluating the overheads of source-directed quality-of-service routing," in *International Conference on Network Protocols*, pp. 42–51, 1998.

[28] J. Moy, *OSPF version 2.* RFC 2328, Apr. 1998.