

Improving QoS Routing Performance Under Inaccurate Link State Information

George Apostolopoulos^a, Roch Guérin^b, Sanjay Kamat^c, Satish K. Tripathi^d

^aComputer Science Department, University of Maryland, College Park, MD 20742

^bUniversity of Pennsylvania, 200 S. 33rd Street, Philadelphia, PA 19104

^cIBM T. J. Watson, Research Center, Yorktown Heights, NY 10598

^dBourns College of Engineering, University of California, Riverside, CA 92521

Several recent works have investigated the costs associated with QoS routing, in terms of both additional processing and increased protocol overhead, in an attempt to identify operating conditions that achieve a good trade-off between cost and performance. In this paper, we concentrate on improving this trade-off by devising methods that increase the performance of QoS routing, when cost constraints, i.e., limitations on link state update traffic, affect the accuracy of the information on which it operates. In particular, we study the performance of “safety-based” routing, which incorporates knowledge of the underlying inaccuracy in computing a “safety” measure for links, and uses it when computing paths. Using detailed simulations, we evaluate the effectiveness of safety-based routing and investigate the dependence of its performance on the assumptions made when computing link safety. Our findings show that for many operating conditions, safety-based routing is successful in improving routing performance while maintaining low update traffic volumes.

1. Introduction

QoS routing is the process of selecting the path to be used by the packets of a flow based on its QoS requirements, e.g., bandwidth or delay. Several recent research results [2,4,6–8] as well as research on state and event dependent routing algorithms for telephony networks [1] have pointed out the potential benefits of QoS routing in packet networks with arbitrary topology and telephony networks. These benefits relate to improvement in the service that users receive as well as in the utilization of network resources. These improvements, however, do not come for free. QoS routing is typically “blamed” for introducing additional costs, that have two major components: *computational cost* and *protocol overhead*. The former is due to the more sophisticated and more frequent path selection computations, and the latter is caused by the need to distribute updates on the state of network resources that are of relevance to path selection, e.g., available link bandwidth. While increases in computational complexity can usually be offset by leveraging the technology curve, i.e., faster processors and bigger memories, an increased volume of protocol traffic contributes to higher cost along multiple dimensions, e.g., bandwidth, storage, update processing, and the associated context switching overheads. Thus, it is important to study solutions that achieve satisfactory routing performance

with a reduced level of link state update traffic.

The volume of update traffic depends significantly on the conditions for triggering a new link state update, i.e., when does a node decide to inform the rest of the network about changes in the state of one or more of its links. A variety of *triggering* policies can be defined for this purpose. In addition, *hold-down* timers may be used to further control the rate of updates, by imposing a minimum spacing between consecutive updates originated by the same node. By tuning the parameters of the update triggering policy it is possible to control the volume of update traffic. Clearly, when the sensitivity of the triggering policy is reduced, so is the accuracy of the information available at the routers. Since operating with reduced update traffic loads is highly desirable, we attempt to identify methods capable of achieving good routing performance while operating with relatively insensitive triggering policies and, therefore, inaccurate information. The main vehicle for achieving the above will be “safety-based” routing. Safety-based routing and the notion of link safety was introduced in [3,9]. Given the requested amount of bandwidth, the last advertised value of available bandwidth, and the knowledge of the triggering policy parameters, it is possible to compute the range of feasible values for the actual available bandwidth on a link. Knowledge of this range, coupled with assumptions about the distribution of the real bandwidth within this range, can then be used to compute a probability that the requested amount of bandwidth is indeed available on a given link. This probability can be incorporated in the routing algorithm in an attempt to compensate for the inaccuracy in the link state information.

The structure of this paper is as follows. In Section 2, we describe the details of our evaluation environment and introduce the routing algorithm that we use throughout the paper. In Section 3, we illustrate the negative effects on routing performance of update triggering policies with reduced sensitivity, motivating in this manner the need for more sophisticated methods for coping with link state information inaccuracy. In Section 4, we describe the details of safety-based routing. Section 5 describes and compares two potential methods for performing safety-based routing, one based on measurements of link bandwidth distribution and the other on the assumption of uniform distribution of link bandwidth. In Section 6, we discuss the feasibility of safety-based routing in an environment where hold-down timers are used to contain the routing update overhead during transient periods of overload. Finally, in Section 7, we present some initial conclusions.

2. Evaluation Environment

We focus on a link state routing environment where paths are calculated at the source and requests are routed using explicit routing. We concentrate on the case of requests for rate guarantees, for which the routing algorithm selects paths based on information about available bandwidth on network links. We assume an *on-demand* path computation environment, where a single path is computed for an incoming request at the time of the request arrival.

2.1. Routing Algorithms

Recent simulation studies ([5]) indicate that from among the many heuristics proposed for routing requests with bandwidth requirements, shortest (with respect to the number of links in the path) path heuristics [5,9,10] perform better than widest path heuristics [6]. The width of a path, also called bottleneck capacity, is defined as the minimum available bandwidth over all the links in the path. We will use the widest-shortest path heuristic [9] as the basis of the (non

safety-based) path selection algorithms used in this paper. Widest-shortest paths are computed as follows: Links that have insufficient available bandwidth for the request that is being routed are pruned from the network topology before the path is computed. Then the minimum hop count paths between the source and the destination are discovered and the widest one is used to route the request. If there is more than one widest-shortest path, one of them must be chosen. In order to improve load balancing, the path is chosen at random with a probability that is weighed according to the bandwidth available on the first hop of the path.

2.2. Triggering Policies

There is a variety of methods for determining when to initiate the flooding of a link state update. In this work, we concentrate on three alternatives that cover a wide range of different options and operational characteristics. Specifically, we consider the following triggering policies: **Threshold based updates:** This policy is characterized by a constant threshold value (th). At some node, if bw_i^o is the last advertised value of available bandwidth for interface i and bw_i^c is the current value, an update is triggered when $\|(bw_i^o - bw_i^c)\|/bw_i^o > th$. **Equal class based updates:** This policy is characterized by a constant B which is used to partition the available bandwidth on a link into multiple equal size classes: $(0, B)$, $(B, 2B)$, $(2B, 3B)$, \dots , etc. An update is triggered when the available bandwidth on an interface changes so that it belongs to a class that is different from the one to which it belonged at the time of the previous update. **Exponential class based updates:** This policy is characterized by two constants B and f ($f > 1$) which are used to define unequal size classes: $(0, B)$, $(B, (f+1)B)$, $((f+1)B, (f^2+f+1)B)$, \dots , etc. Unlike the previous policy, the class sizes grow geometrically by the factor f . In this work, we will only use a growth factor f of 2. Updates are triggered as before, i.e., when a class boundary is crossed. With all the above policies, hold down timers can be used to better control the volume of update traffic.

In class based policies, triggering an update each time the available bandwidth value crosses a class boundary has the undesirable effect of generating somewhat meaningless updates when the available bandwidth fluctuates around a class boundary. In order to dampen such oscillatory behavior, class based policies are augmented with a *hysteresis* mechanism that requires that, in addition to the class boundary crossing, the magnitude of the change in the available bandwidth is significant. In addition, the benefit of advertising a fixed value for each class when using class based triggering policies was demonstrated in [12] and we, therefore, follow this approach here as well and advertise a bandwidth value that corresponds to the middle of the class.

2.3. Simulation Details

The topology used in our experiments is shown in Figure 1(a). This “isp” topology is a popular topology used in many QoS routing studies ([5,11]), and is typical of the nationwide network of a US based ISP. We determine link capacities so that the topology is dimensioned for uniform traffic. This is achieved by applying a uniform traffic load on the network and using shortest path routing. The corresponding load on each individual link is then measured, and link capacities are chosen in proportion to this load. The resulting link capacities are between 30 and 80 Mbits/sec.

We believe that QoS routing is most effective when a temporary mismatch exists between traffic pattern and network topology. Conditions of load imbalance are not uncommon and can occur either as a result of a network failure (link failure) or simply because of changing traffic patterns or inaccurate/outdated network dimensioning. This is why we mainly focus on non-

uniform traffic. Such traffic is generated by allowing increased levels of traffic between selected pairs of *hot-spot* nodes. The hot-spot nodes are changed during the simulation in order to avoid excessive dependence on topology details. Bandwidth requirements are uniformly distributed between a minimum value of 64 Kbits/sec and a maximum value of 6 Mbits/sec, while call duration is exponentially distributed with a mean of 3 minutes. The request arrival rates are chosen in such a way so that blocking ratios are kept in the 2%-22% range. The request arrival process at each node is assumed to be Poisson and independent of the arrival processes at other nodes. In our experiments, the average request inter-arrival time was about 20 seconds for background traffic and 3 seconds between the hot-spot nodes.

2.3.1. Performance Measures

The main quantities of interest in this paper are routing performance and volume of update traffic. We measure routing performance using the bandwidth acceptance ratio, i.e., the sum of the bandwidth requirements of accepted connection requests over the sum of bandwidth requirements of all connection requests. We measure link state update traffic using the number of update messages generated in the network. This number includes all the link state update packets generated by all routers over the duration of the simulation. The 95% confidence interval for the routing performance results reported is less than .005%.

3. Effects of the Sensitivity of the Triggering Policies

In this section, we briefly review how the choice of the parameters of triggering policies affects both the update traffic cost and the performance of the routing algorithm. For threshold based triggering, we vary the value of the threshold from a minimum of 10% to a maximum of 90%. When using class-based policies, the size of the first class is varied from a minimum 0.6 Mbits/sec to a maximum of 24 Mbits/sec.

Figure 1(b) shows the trade-off between the routing performance and the update traffic volume achieved by the different triggering policies as their sensitivity is varied. Routing performance is shown on the y-axis in terms of the bandwidth acceptance ratio, while update traffic is shown on the x-axis using the total number of update messages generated over the duration of the simulation. The points of each curve correspond to different sensitivity settings of the policies; sensitivity increases as we move to the right.

For both threshold and class based triggering, update traffic is significantly reduced as the triggering policies become less sensitive. Threshold triggering policies and exponential classes achieve roughly similar routing performance and update traffic trade-offs (with threshold policies performing slightly better), while equal class triggering policies require significantly more link state updates for the same level of routing performance. This is intuitive since partitioning links using equal classes results in a fairly large number of classes, many of them in the region of high available bandwidth, that convey information that is not very important for the path computation algorithm. Due to this inherent inefficiency of equal class triggering policies, we do not consider them any further in this paper.

Overall, reducing the sensitivity of triggering policies can significantly lower the volume of update traffic, but these savings come at the cost of a fairly severe reduction in routing performance. We need solutions that will allow the routing algorithm to operate with triggering policy settings that yield low update overheads, while still achieving good routing performance.

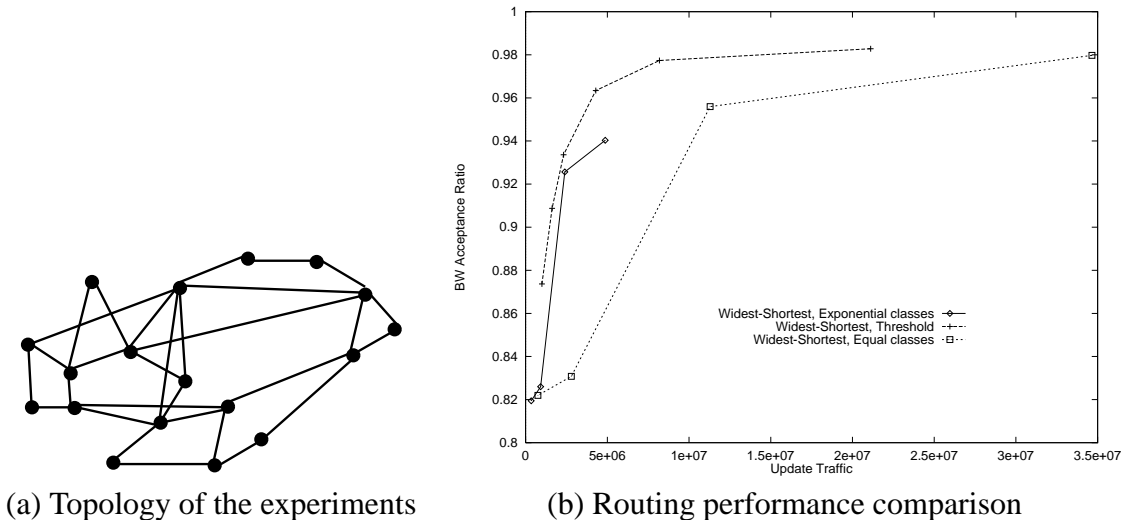


Figure 1. Topology and routing performance / protocol traffic trade-off

4. Safety-Based Routing

As was first proposed in [3,9], the inaccuracy introduced by the triggering policy can be modeled when the operating details of the policy are known. For all change-based triggering policies, given the last known value of available bandwidth on a link b_{adv} , it is possible to determine the *range* of values that the actual bandwidth can take as long as a new update has not been triggered. The bounds, b_l for the minimum value, and b_u for the maximum value, can then be used to estimate the likelihood that a given amount of requested bandwidth is indeed available on this link. For example, for the threshold policy, $b_l = b_{adv} * (1 - th)$ and $b_u = b_{adv} * (1 + th)$. If class based triggering is used, then it is sufficient to determine the class that the advertised value belongs to and then, if this is class c_i , b_l can be set to the boundary value between c_{i-1} and c_i and b_u can be set to the boundary between c_i and c_{i+1} . This range is appropriately adjusted to take into account the hysteresis mechanism used.

For a given value of requested bandwidth b_{req} , we can then assign a probability that this amount of bandwidth is indeed available on the link. We call this probability the *safety* of the link. Clearly, if $b_{req} \geq b_u$ then we can be sure that the link does not have enough bandwidth and it has a safety of 0. If $b_{req} \leq b_l$, the link has definitely enough bandwidth and it has a safety of 1. If $b_l < b_{req} < b_u$, we need information about the distribution of actual values of available bandwidth within $[b_l, b_u]$ in order to compute its safety. For example, if we assume that values for the actual link bandwidth are uniformly distributed between b_l and b_u , then the safety of the link is given by $(b_u - b_{req}) / (b_u - b_l)$ if $b_l \leq b_{req} \leq b_u$, 0 if $b_{req} > b_u$, and 1 if $b_{req} < b_l$. This safety information can then be used as a means of ranking links and consequently paths. Links with larger safety should be preferred since they are more likely to have the requested amount of bandwidth. In the next section, we present several methods for incorporating this safety information into the routing algorithm.

Clearly, knowledge of the distribution of actual bandwidth values in the $[b_l, b_u]$ interval, is an essential component of safety based routing. Intuitively, it seems unlikely that a specific distribution can emerge as being characteristic for any arbitrary link. This is because there are numerous factors that can affect it, e.g., topology, traffic patterns, arrival rates, request sizes, link size, and also routing. Identifying if and how actual link bandwidth distributions can be reliably estimated and used by routing, is in itself a very complex problem. In this paper, we do not attempt to fully address it, and instead try to obtain some practical insight into the validity of several approximations.

4.1. Safety-Based Routing Algorithms

Assuming mutual independence among the safety of different links, the safety of a path can be determined as the product of the safety of the links in the path. Path safety can then be handled exactly as other path metrics. In correspondence to the well known heuristics for the bottleneck capacity metric, we can define *safest-shortest* and *shortest-safest* routing. Again, a shortest path is a path with the minimum number of links. In safest-shortest routing, all the minimum hop paths between the source and the destination are determined and the one with the largest safety is used. In shortest-safest routing, all the safest paths between the source and the destination are found and the shortest one is used.

Using link safety information allows for a large range of link pruning policies, i.e., which links not to consider when searching for a feasible path. A family of pruning policies can be specified, where a link is pruned only if its safety s is below a certain cut-off value. For example, a conservative pruning strategy is to prune all links with safety less than one ($s < 1$); that is, we keep only those links that can *guarantee* the availability of the requested bandwidth. On the other hand, pruning all the links with zero safety ($s = 0$) is an aggressive pruning strategy where we only prune links that we know for sure that have inadequate bandwidth to satisfy the request. In this paper, we limit our investigation to these two representative pruning policies, i.e., the ones corresponding to $s = 0$ and $s < 1$.

5. Implementation and Performance of Safety-Based Routing

In this section, we describe and compare methods for implementing safety-aware routing. First we discuss a method for performing safety-based routing without making assumptions about the distribution of available bandwidth. We argue that although this method appears to be effective, it is hard to implement. Then we present a method that is based on the assumption that available bandwidth is uniformly distributed in the ranges and compare its performance with that of the first method.

5.1. Measurement Based Safety-Based Routing

In this method, the “distribution” of actual values of available bandwidth is measured over a given time interval, and then used to compute the safety of links. Specifically, we measure this distribution within a range by sampling the actual value of available bandwidth on each network link at regular time intervals that are set to half the average inter-request arrival interval. The sampled value is normalized to the range $[0, 1]$ based on the bounds obtained from the triggering policy, and is used to update a histogram of the frequency of the values observed so far for this link.

When an exponential class triggering policy is used, a different histogram is maintained

for each class. For each sampled value v_{sample} , the class to which it belongs is determined, and the upper and lower bounds for this class are used to normalize the value: $v_{norm} = (b_u - v_{sample}) / (b_u - b_l)$. As a result, the histogram reported for any interval gives the distribution of bandwidth values between the lower and upper bounds of the interval. A similar approach is used for threshold policies, where the boundaries of the feasible range of bandwidth values are determined based on the threshold value and the last advertised value. As before, values are normalized so that the histogram reports relative distribution between the lower and upper bounds. With a threshold policy there can be a very large number of possible ranges, each corresponding to a single value of advertised bandwidth. As a result we aggregate all the histograms corresponding to advertised values that fall within a given interval, with intervals chosen so that they coincide with used in a class based triggering policy.

The bins of the histograms are “emptied” periodically and the distributions are measured afresh to capture shifts in link load. Measured distribution information is propagated to all routers where it is used to compute link safety so that routers can compute paths for incoming requests.

Using the above measurement based approach, a comparison of the performance of various safety-based routing heuristics is shown in Figure 2. Safest-shortest routing does not perform well when only links with safety 0 are pruned. In this case, its performance is even worse than that of non safety-aware routing. This is because, under this link pruning policy, safest-shortest routing can, because of its preference for shortest paths, end-up using links with very low safety. On the other hand, non safety-aware, i.e., bottleneck routing does not consider links that do not appear to have enough bandwidth to accommodate the incoming request. This effectively results in avoiding links that have very low safety. When the pruning policy ignores all links with safety $s < 1$, the performance of safest-shortest routing improves and exceeds that of non safety aware routing, since links with very low safety are now avoided.

However, Figure 2, shows that the shortest-safest heuristic yields the best overall performance, when the link pruning policy used is to prune only links with safety = 0. In this case, choosing the shortest-safest path in the network significantly improves the chances of successful resource reservation. When the pruning policy used is to prune all links with safety < 1 , the shortest-safest heuristic behaves identically to the safest-shortest heuristic as all paths now have a safety of 1, and both approaches amount to selecting a minimum hop count path.

5.2. Uniform Distribution Assumption

Although the measurement based approach was shown to perform well, maintaining accurate measurement information for all network nodes requires the exchange of a large number of routing messages, which is exactly what we want to avoid by using safety aware routing in the first place. A more practical approach will be to simply assume, as in [3], a uniform distribution for link bandwidth on all links. In Figure 3 we show how the performance of safety aware routing under the uniformity assumption compares to that of the measurement based approach, when a pruning policy that prunes links with safety 0 is used.

For threshold based policies, there is a small loss in performance for very large threshold values, but overall the performance difference is relatively small, and more important, safety-based routing maintains its performance advantage over non safety-aware routing. For exponential classes, the performance difference between the measurement based approach and the use of a uniform distribution assumption is even less significant, and confirms the effectiveness and

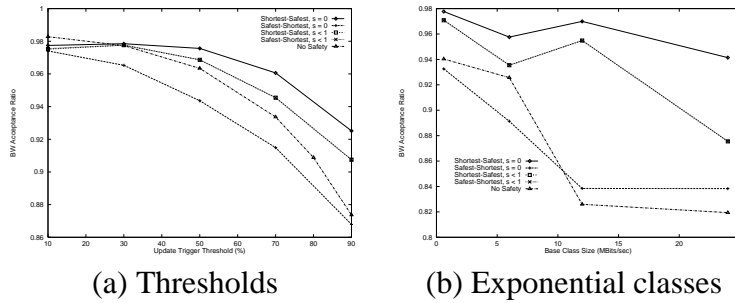


Figure 2. Routing performance of safety-based routing

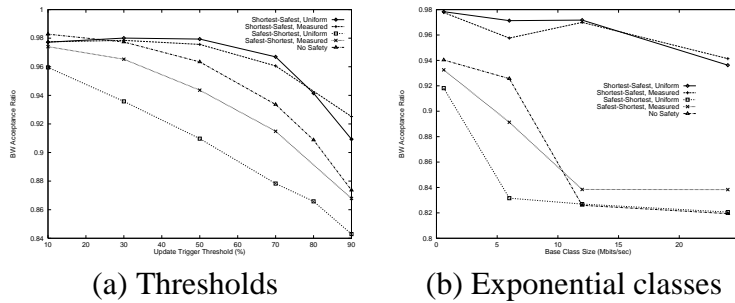


Figure 3. Effects of the uniformity assumption

robustness of assuming a uniform distribution for link bandwidth values.

It is interesting to note that for exponential classes, the shortest-safest algorithm assuming uniform distribution of actual bandwidth is effective even when the class sizes are very large (larger than the maximum request size). In this case, there can only be two values for link safety. All (reasonably) loaded links have bandwidth values that remain confined to the same first class, and therefore yield the same safety value. All other links have bandwidth values in higher classes, and as a result have a safety of 1. In this setting, the shortest-safest algorithm reduces to picking paths that have the smallest number of links with advertised bandwidth value within the first class. Even this simplistic mode of operation is sufficient to substantially improve routing performance over non safety aware routing.

6. Safety-Based Routing Under Hold-Down Timers

In what preceded, we assumed that triggering policies were not complemented with hold-down timers to further control the volume of update traffic. However, in practice it is most likely that hold-down timers will be used to ensure the stability of routing protocols during periods of transient overload. These timers can affect the accuracy of link safety computations.

This is because timers interfere with the basic trigger mechanisms and can, therefore, distort the ranges of possible values used to compute link safeties. In particular, because the generation of updates can now be delayed (by the timer), it is possible for link bandwidth values to extend well outside the range of values defined by the triggering policy. In the following section, we investigate whether safety-based routing can remain effective in such an environment.

6.1. Performance under Hold-Down Timers

In this section, we investigate the effects of hold-down timers on the routing performance of both safety aware and non safety aware routing. In Figure 4, we show the relative performance of safety routing for hold-down timers of 100 seconds. This setting of the hold down timer results in a significant reduction of the update traffic, in the order of 80%. For safety aware routing, link safeties are computed assuming uniform distribution of advertised bandwidth and links with safety of 0 are pruned. Comparing with Figure 2 we see that large hold-down timers result in routing performance loss for both safety-based and non-safety-based routing. However, safety-based routing (using the shortest-safest heuristic) still manages to improve routing performance over non safety aware routing. With threshold policies, we observe that with both timer settings, the routing performance of the safety-based heuristic initially improves with increasing coarseness. This is because fine thresholds result in highly inaccurate link safety values in an environment where hold-down timers are used. As the threshold increases, the link safety values computed become more reliable, and safety-based routing performs better. However, with very large threshold settings, the safety values computed hardly convey any useful information since the corresponding ranges are now very large and span a large fraction of the link capacity. As a result, the performance of safety based heuristics ultimately decreases as the thresholds get larger. The same behavior can be observed when class based triggers are used. Routing performance increases initially before it decreases again for large class sizes. Note that these results are similar in nature to the ones reported in [12]: in the presence of hold-down timers, advertising coarse routing information can improve routing performance.

6.2. Randomized Path Selection

An important effect of hold-down timers, is that link state information is updated less frequently, therefore forcing routing to always select the same path for extended periods of time. As was discussed in [12], such a deterministic path selection based on outdated information can result in substantial routing performance loss. In the case of safety-based routing, assuming that we use the shortest-safest routing algorithm, routing will keep on using the apparently safest path to the destination, most likely overloading this path with more traffic than it can handle. [12] proposed randomized path selection as a means for coping with this problem. With randomized path selection, a path is selected randomly from a set of candidate paths, avoiding overloading the same (apparently best) path. The same approach can be used in the context of safety-based routing.

Randomized path selection requires the existence of a set of candidate paths to a destination, but the safety-based algorithms discussed so far only discover a single path. [9,12] have presented a Bellman-Ford based path finding algorithm that can discover and maintain multiple paths per destination in the context of a bottleneck metric. This algorithm can be easily adapted to use the link safety metric. The modified algorithm first determines the safest shortest path(s) to the destination. Then, it continues expanding paths, by increasing hop count. A path will be maintained in a QoS routing table if its safety is larger than that of previously discovered (and

as a result shorter) paths, or equal to that of a path of the same hop count. When the algorithm terminates, a series of paths of increasing hop count and safety have been discovered for each destination. In this sequence of paths, the shortest path will have the smallest safety value and the longest path the largest one.

Randomized path selection allows the occasional selection of shorter and less safe paths instead of always selecting the safest one. There is a variety of methods for randomizing path selection. A sample method that we use in our experiments, is to make the selection decision solely based on the relative degradation in the safety of the paths. In this method, we consider the paths to a destination in order of decreasing safeties (or longer to shorter). For each path, we make a random decision to either use the path, or move to the next shorter one. This decision is weighted by the difference in safety between the current path and the next one. If this difference is small, then we will move to the next path with high probability. If this difference is large though, the chance of moving to the next path is lower, since that path typically has much lower safety.

Randomizing the selection of the safest path has mixed results for the different triggering policies as can be seen in Figure 4. In the case of exponential class policies, randomization appears to make little difference when compared to non-randomized shortest-safest routing. This can be attributed to the nature of the safety values computed when link capacity is divided into classes. Since class boundaries are identical on all links, the safety values computed for a link will be 0, 1 or a value s between 0 and 1, depending on whether the request amount of bandwidth falls above, below or inside the last advertised class for the link bandwidth. When classes are very small, the chances of the requested bandwidth falling within the last advertised class of a link will be smaller, and there will be few links with safety s in the network, with all other links having safety of 0 or 1. As a result, the shortest-safest path will most probably be the shortest path of safety 1, and due to the small number of links with safety s there will be a small number of shorter paths with smaller safety. On the other hand, when classes are very large, most of the links will have safety s since there will a large probability that the requested bandwidth will fall into the large last advertised class. As a result, path safety is likely to rapidly decrease with hop count (it decreases as s^k , where k is the number of links with safety s), and the safest path is likely to have a small hop count with a small number of shorter paths. Since randomization uses paths that are shorter and less safe than the shortest-safest path, in both of the above cases, there will be few paths available for randomizing the path selection, and the effects of randomization will be limited.

In contrast, when a threshold based triggering policy is used, link safeties tend to have distinct values since the ranges depend on the last advertised bandwidth value. As a result, there is a potential for a larger number of paths for randomization to choose from. Unfortunately, it turns out that for small threshold values, randomization has a negative impact. This is because the combined effect of a small threshold and the use of hold-down timers renders very inaccurate the advertised values, and hence the corresponding link safety. On the other hand, for large thresholds, which are recommended when hold down timers are large, the safety information is likely to be more accurate because of the wider ranges around advertised values. Hence, selecting a less safe path is less risky and the resulting load balancing effect can help improve performance, albeit only slightly.

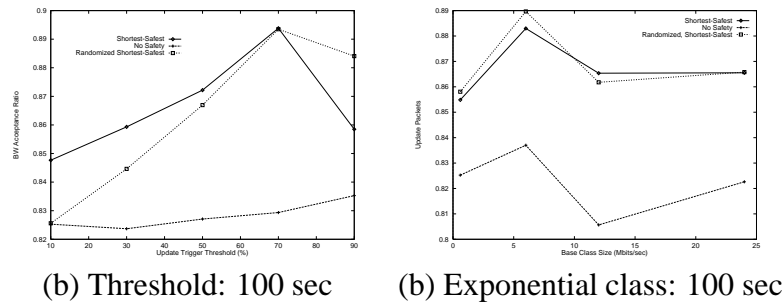


Figure 4. Effects of hold-down timers on safety-based routing

7. Conclusions

In this work, we attempted to investigate a variety of issues related to using link safety information for improving routing performance when link state information is inaccurate. We evaluated the influence of a number of parameters such as a) triggering policy, b) link pruning strategy, c) method for determining the distribution of actual bandwidth, d) type of safety-based routing algorithm, and e) impact of hold-down timers. Our results verify the effectiveness of safety-based routing for improving routing performance, when insensitive triggering policies are used to achieve low update traffic volumes. In particular, choosing the shortest-safest path proved to be the most effective algorithm for safety-based routing. This algorithm was effective with all types of triggering policies. Furthermore, we were able to demonstrate that assuming a uniform distribution of bandwidth values within the advertised ranges is effective, and does not result in significant routing performance loss when compared to an “accurate” measurement based model. In addition, our results show that the benefits of safety-based routing can extend to cases where moderate hold-down timers are used. In this context, randomization in the selection of the safest path can further improve routing performance, if a threshold based policy is used and the threshold is set to a large value.

REFERENCES

1. G. R. Ash, “Dynamic Routing in Telecommunication Networks”, McGraw-Hill, 1997
2. V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, Two Distributed Algorithms for the Constrained Steiner Tree Problem, In Proceedings of 2nd International Conference on Computer Communication and Networking, pages 343-349, 1993.
3. R. Guérin, and A. Orda, QoS Based Routing in Networks With Inaccurate Information: Theory and Algorithms, in proceedings of INFOCOM, 1997.
4. H. Ahmadi, J. S.-C. Chen, and R. Guérin, Dynamic Routing and Call Control in High-Speed Integrated Networks, Proc. Workshop Sys. Eng. Traf. Eng., ITC’13, 1991.
5. Q. Ma and P. Steenkiste, On Path Selection for Traffic with Bandwidth Guarantees, Proceedings of IEEE International Conference on Network Protocols, Atlanta, Georgia, October 1997.

6. Z. Wang, and J. Crowcroft, Quality of Service Routing for Supporting Multimedia Applications, *IEEE Journal Selected Areas in Communications*, 14(7):1228-1234, 1996.
7. W. C. Lee, M. G. Hluchyj, and P. A. Humblet, Routing Subject to Quality of Service Constraints in Integrated Communication Networks, *IEEE Networks*, pages 46-55, July/August 1995.
8. R. Widyonon, The Design and Evaluation of Routing Algorithms for real-time Channels, Technical Report TR-94-024, University of California at Berkeley, June 1994.
9. R. Guérin, D. Williams, and A. Orda, QoS Routing Mechanisms and OSPF Extensions, in proceedings of GLOBECOM 1997.
10. Q. Ma, P. Steenkiste, and H. Zhang, Routing High-Bandwidth Traffic in Max-Min Fair Share Networks, in *ACM SIGCOMM'96*, 206-217, 1996.
11. A. Shaikh, J. Rexford, and K. Shin, Dynamics of quality-of-service routing with inaccurate link-state information, in proceedings of International Conference on Networking Protocols, Austin Texas, 1998
12. G. Apostolopoulos, R. Guérin, S. Kamat, and S. K. Tripathi, QoS Routing: A Performance Perspective, in proceedings of *ACM SIGCOMM* 1998.