An Effective Randomized QoS Routing Algorithm On Networks with Inaccurate Parameters[1]

JIANXIN WANG[2], JIANER CHEN[3], SONGQIAO CHEN[2], JIANHUA XIAO[2]

Abstract

This paper develops an effective randomized on-demand QoS routing algorithm on networks with inaccurate link-state information. Several new techniques are proposed in the algorithm. First, the maximum safety rate and the minimum delay for each node in the network are pre-computed, which simplify the network complexity and provide the routing process with useful information. The routing process is dynamically directed by the safety rate and delay of the partial routing path developed so far and by the maximum safety rate and the minimum delay of the next node. Randomness is used at the link level and depends dynamically on the routing configuration. This provides great flexibility for the routing process, prevents the routing process from overusing certain fixed routing paths, and adequately balances the safety rate and delay of the routing path. We have established a network testing environment and introduced five parameters to measure the performance of QoS routing algorithms. Experimental results demonstrate that in terms of the proposed parameters, our algorithm improves over existing QoS algorithms appearing in the literature.

Keywords. QoS, network routing, randomized algorithm, link-state information, performance measure

1 Introduction

Today's Internet only provides best-effort service. Traffic is processed as quickly as possible, but there is no guarantee as to timeless or actual delivery. Integrated services networks are expected to support multiple and diverse applications, with various quality of service (QoS) requirements. The actual QoS metrics of interest are likely to vary from one application to another, but are projected to include such measures as cell loss, delay, delay jitter, and bandwidth guarantees. There are many aspects in the QoS study, including traffic shaping, packet scheduling, QoS routing, resource reservation, call admission control, and so on. QoS routing is a process of selecting a path to be used by the packets of a flow based on its QoS requirements, e.g., bandwidth or delay [6, 4, 13, 11, 9].

Network routings in general rely heavily on network's state information on resource availability at network nodes and links. In order to keep network state information accurate, many parameters associated with QoS requirements must update in time. Unfortunately, this is impractical when the network is highly dynamic and changes are frequent, because it brings out high computational cost and protocol overhead [6]. Thus, network routing algorithms must deal with the situations in which network state information is inaccurate.

[1]This work is supported by the China National Natural Science Foundation for Distinguished Young Scholars and by the Changjiang Scholar Reward Project.

[2]College of Information Engineering, Central-South University of Technology, ChangSha, Hunan, P. R. China.

[3]Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, USA.

In this paper, we primarily focus on routing algorithms with inaccurate network state information. There have been two major techniques in the study of QoS routing algorithms: deterministic and randomized. Deterministic routing algorithms have been popular and it finds a path according to determinate criteria, such as the shortest-safest and safest-shortest algorithms [12]. However, these algorithms ignore the quick dynamic network changes. For example, if in a short time period, there are many QoS requests and the network state information does not update accordingly, then the deterministic routing algorithms will keep on using the "good" paths, most likely quickly overloading these paths with more traffic than they can handle. On the other hand, randomized QoS routing algorithms bring in randomness and select paths with probability. Recent studies provide evidence that deterministic routing algorithms are of better performance when network state information is relatively accurate, while randomized algorithms can balance network load, prevent performance degradation, and improve service performance of the entire network [10, 8, 11, 9, 7].

The additional cost of QoS routing has 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 relevant to the path selection, e.g., available link bandwidth [1]. In general, we hope to obtain good performance with low cost.

In this paper, we consider QoS routing algorithms in which the routing path selection and computation are "on-demand", i.e., they are upon receipt of a request for the QoS routing. The benefit of such an approach is that on-demand route computation in general yields better routes by using most recent network state information. Another benefit of on-demand path computation is the associated storage saving, i.e., there is no need for a QoS routing table [5]. We point out that there has also been extensive research that studies the "off-line" computation for QoS routings.

The structure of this paper is as follows. In section 2, we briefly describe our QoS routing network model, and review the current status of QoS routing algorithms. In section 3, we present in detail our randomized on-demand QoS routing algorithm. In section 4, we describe the details of our simulation environment and discuss the experimental results. Conclusions and further remarks are given in section 5.

2 Network Model and QoS Routing

In this section, the network model of QoS routing and the performance of deterministic path selection and randomized path selection will be described.

Each QoS routing request R is a 5-tupe $R = (u_0, v_0, b, N_0, S_0)$, where u_0 and v_0 are the source node and the destination node of the QoS routing path, respectively, b is the bandwidth requirement, S_0 is the probability that the bandwidth requirement b must be satisfied by the routing path, and N_0 is the delay requirement that specifies the maximum number of hops the routing path may have. Note that we have somehow simplified the delay requirement and assume that all hops have the same delay and the total delay of a routing path is simply the additive sum of delays of the hops in the path. This will greatly simplify our discussion. On the other hand, our algorithm can be extended to include more complicated computation formulas for QoS path delay, such as the one suggested in [6].

We focus on QoS routing environment with inaccurate link state information, where the available bandwidth of each link l is advertised as B_{adv}. The link state information is updated according to a pre-specified update triggering policy. So, 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 by the triggering policy, using the advertised bandwidth value B_{adv} and the threshold value h. The lower bound B_l and the upper bound B_u of the range can be computed using the formulas

$$B_l = B_{adv}(1-h) \qquad \text{and} \qquad B_u = B_{adv}(1+h)$$

According to this, S_l, which describes the probability of link's bandwidth, can be defined as follows: for a given value of requested bandwidth B_{req}, if $B_{req} \geq B_u$, we know that link l does not have enough bandwidth, so $S_l = 0$. If $B_{req} \leq B_l$, link l then definitely has enough bandwidth, so $S_l = 1$. For $B_l < B_{req} < B_u$, we need to know the distribution of actual values of available bandwidth within $[B_l, B_u]$ in order to compute S_l. As proposed in [6], if we assume that the values of actual available bandwidth are uniformly distributed between B_l and B_u, then $S_l = (B_u - B_{req})/(B_u - B_l)$.

Previous research [6, 4, 13] has indicated that deterministic routing algorithms can achieve good performance when advertised available bandwidths are almost accurate. Safest-shortest routing and shortest-safest routing [12] are typical algorithms of this kind. In the 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, here the "safest" one is referred to the one that satisfies the bandwidth with the highest probability. In the shortest-safest routing, all the safest paths between the source and the destination are found and the shortest one is used. Pruning policy can be used in these algorithms where a link is pruned if its safety below a certain cut-off value and path selection will be done in the residual graphs.

As we have seen, the accuracy of the link safety information has a direct impact on the performance of the routing algorithms, and update triggering policy directly impacts the accuracy of the link state information. Link state information will be more accurate when update is done more frequently, which will introduce much more overhead to the protocol. Generally, we can trade off between information accuracy and protocol overhead. There are several update triggering policies, such as the Threshold Triggering, the Equal Class Triggering, and the Exponential Class Triggering [12]. No matter which update triggering is used, the advertised information will be lag. So the safety of a link will remain the same value during the period of adjacent update times. Thus, in the time period between two consecutive update times, a deterministic routing algorithm, for example the safest-shortest algorithm, will keep on selecting the same path when requests on the same source and destination are given. This will most likely overload this path quickly with more traffic than it can handle, resulting in poor QoS routing performance.

Randomized routing algorithms try to avoid this problem. With randomized path selection, a path is selected randomly from a set of candidate paths. An example is the pre-computation algorithm proposed in [1], in which there is a list of paths $[S_1, S_2, \ldots, S_k]$ for each pair of source and destination nodes. Each set S_d contains paths of certain hop length h_d. Instead of selecting the path with the smallest h_d, the selection of a path in the set S_d is associated with a probability P_d. Apparently, this somehow avoids the unbalance caused by the deterministic routing algorithms.

3 A New Randomized On-demand QoS Routing Algorithm

In this section, we present a new randomized on-demand QoS routing algorithm.

Let $G = (V, E)$ be a network, where $V = \{v_1, \ldots, v_n\}$ is the set of hops in G and $E = \{e_1, \ldots, e_m\}$ is the set of links in G. Each link e_j is associated with a function f_j, such that $f_j(b)$ is the probability that the link e_j has bandwidth at least b. Consider a QoS request $R = (u_0, v_0, b, N_0, S_0)$, where u_0 and v_0 are the source node and the destination node, respectively, b is the bandwidth requirement, N_0 is the "delay requirement", i.e., the maximum number of hops allowed in the routing path, and S_0 is the "safety rate requirement", i.e., the minimum probability with which the routing path has bandwidth at least b.

Remark. Instead of setting an individual safety rate for each QoS request, we may also set a global safety rate that all QoS requests must satisfy.

For a given QoS request $R = (u_0, v_0, b, N_0, S_0)$, we say that a routing path P from the source node u_0 to the destination node v_0 *satisfies* the QoS request R if it has the following properties:

1. The probability that the bandwidth of the path P is at least b is not smaller than the safety rate requirement S_0:
$$\prod_{e_j \in P} f_j(b) \geq S_0$$
2. The number $|P|$ of hops in the path P is not larger than the delay requirement N_0:
$$|P| \leq N_0$$

Our goal is for each QoS request R, we construct a routing path satisfying the request when such a routing path exists, or report that no such a path exists. It can be proved that this problem is NP-hard by showing that the NP-hard problem KNAPSACK [3] can be polynomial time reduced to the problem. Thus, it is unlikely that there is a deterministic polynomial time algorithm that solves the problem optimally. Our approach is an algorithm that can use randomness and solves the problem efficiently *in most cases*. Our algorithm is presented in Figure 1.

We first look at Step 1–Step 3 of the algorithm. Step 1–Step 3 pre-compute for each node v in the network the maximum safety rate and the minimum delay from the node v to the destination node v_0. This pre-computation serves for two purposes. First, it provides the routing algorithm with useful information on bandwidth and delay that will help the routing algorithm to decide which link should be included in the routing path. Secondly, it finds the nodes that definitely do not lead to a valid routing path (either because the safety rate is too small or because the delay is too large). In step 3, these useless nodes are removed and the network is simplified. Note that Step 1–Step 3 may need to be repeated because removing useless nodes may result in new useless nodes. For example, suppose the shortest path from v to v_0 is $[v, v_1, \ldots, v_0]$ which is smaller than N_0, and the safety rate of v is larger than S_0. Therefore, in the first execution of Steps 1–3, the node v is not removed. However, in Steps 1–2 we may find out that the safety rate for v_1 is too small so we remove v_1 in Step 3. After removing v_1, the delay of vertex v may become too large. Thus, repeating Steps 1–3 may remove further nodes from the network. The more nodes can we

Algorithm. RandomRoute

Input: a network $G = (V, E)$ and a QoS request $R = (u_0, v_0, b, N_0, S_0)$.

Output: a routing path P that satisfies the QoS request R.

```
repeat
1.  for each node v in G, find the maximum safety rate from v to v0:
        maxsafe[v] = max{∏_{e_j ∈ P} f_j(b) | P is a path in G from v to v0};
2.  for each node v in G, find the length of the shortest path from v to v0:
        mindelay[v] = min{|P| | P is a path in G from v to v0};
3.  remove all nodes v with either maxsafe[v] < S0 or mindelay[v] > N0;
until step 3 cannot remove any more node;

4.  (* search for a routing path satisfying the QoS request R. *)
    now = u0;   nowdelay = 0;   nowsafe = 1;
    repeat
      if there are links e1, ..., ek, where for each j, e_j = [now, v_j] and
          nowdelay + 1 + mindelay[v_j] ≤ N0  and  nowsafe · f_j(b) · maxsafe[v_j] ≥ S0
      then with probability p_j  (* the probability p_j will be explained below *):
          pick the link e_j;   now = v_j;
          nowdelay = nowdelay + 1;   nowsafe = nowsafe · f_j(b);
      else backtracking
    until either now = v0 or all paths are tried;

5.  if now = v0 then output the routing path
    else report ("QoS request cannot be satisfied").
```

Figure 1: The randomized QoS routing algorithm **RandomRoute**.

remove in Step 1–Step 3, the simpler will the network be. Thus, Step 1–Step 3 may greatly reduce the network complexity.

We explain how the probability p_j is decided. Suppose that in the **repeat** loop in Step 4 of the algorithm, a partial path P from u_0 to the node *now* has been constructed, such that the length (i.e., the delay) of the path P is

$$nowdelay = |P|$$

and the safety rate of the path P is

$$nowsafe = \prod_{e_j \in P} f_j(b)$$

We also assume that there are links $e_1, \ldots, e_k$, where for each j, $e_j = [now, v_j]$ and

$$nowdelay + 1 + mindelay[v_j] \leq N_0 \quad \text{and} \quad nowsafe \cdot f_j(b) \cdot maxsafe[v_j] \geq S_0$$

that is, extending the path P from the node *now* to the node v_j still makes a valid partial path. Then the probability p_j with which we pick the link $e_j = [now, v_j]$ is

$$\begin{aligned} p_j \quad = \quad & \alpha \cdot \log(1 + maxsafe[v_j]) / \sum_{i=1}^{k} \log(1 + maxsafe[v_i]) \\ & + \ \beta \cdot (1/(1 + mindelay[v_j])) / \sum_{i=1}^{k} (1/(1 + mindelay[v_i])) \end{aligned}$$

Here we let $\alpha = a/(a+b)$ and $\beta = b/(a+b)$, where the values a and b are given by

$$a = N_0 - (nowdelay + mindelay[now]) \quad \text{and} \quad b = N_0 \log((nowsafe \cdot maxsafe[\, now])/S_0)$$

Since $\alpha + \beta = 1$, it is easy to see that $\sum_{j=1}^{k} p_j = 1$. Moreover, from the safety term

$$\log(1 + maxsafe[v_j]) / \sum_{i=1}^{k} \log(1 + maxsafe[v_i])$$

and the delay term

$$(1/(1 + mindelay[v_j])) / \sum_{i=1}^{k} (1/(1 + mindelay[v_i]))$$

we can see that the link leading to a higher safety rate and shorter delay has a higher probability to be picked. Moreover, the coefficients α and β give different weights to the safety rate and the delay of the path, as follows. The value

$$a = N_0 - (nowdelay + mindelay[now])$$

is the difference between the delay requirement N_0 and $nowdelay + mindelay[now]$, the latter is the minimum delay we may get by extending the current partial path P to the destination node v_0, while the value

$$b = N_0 \log((nowsafe \cdot maxsafe[\, now])/S_0)$$

measures the difference between the maximum safety rate $(nowsafe \cdot maxsafe[now])$ we may obtain be extending the current partial path P to the destination node v_0 and the safety rate requirement S_0. We point out that here we have multiplied the value $\log((nowsafe \cdot maxsafe[now])/S_0)$ by a coefficient N_0 to make the values a and b comparable. The intuition is that when the value a is much larger than the value b, then delay is a less important issue, while when the value b is much larger than the value a, then safety is a less important issue. Therefore, for the safety term

$$\log(1 + maxsafe[v_j]) / \sum_{i=1}^{k} \log(1 + maxsafe[v_i])$$

we multiply by the coefficient $\alpha = (1/b)/((1/a) + (1/b)) = a/(a+b)$, while for the delay term

$$(1/(1 + mindelay[v_j])) / \sum_{i=1}^{k} (1/(1 + mindelay[v_i]))$$

we multiply by the coefficient $\beta = (1/a)/((1/a) + (1/b)) = b/(a+b)$.

According to the above construction of the probability p_j, we can see that the algorithm **RandomRoute** dynamically justifies its routing strategy based on the routing configurations. In general, a routing path with large safety rate and short delay has a high probability to be picked. However, the routing strategy may change when the routing configuration becomes unbalanced. When the delay requirement becomes tighter, the algorithm gives higher probability to short paths than to safe paths, while when the safety rate becomes tighter, the algorithm gives higher priority to safe paths than to short paths.

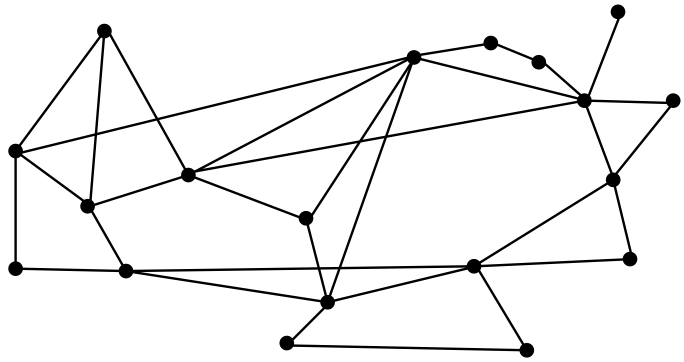

Figure 2: The network topology for the experiments

We analyze the complexity for the algorithm. Step 1 of the algorithm can be done using a modified Dijkstra's algorithm [2], in which we reverse the link directions in the network G and find a maximum safety rate path from v_0 to each of the nodes in the network. This takes time $O(m+n\log n)$. Step 2 can be similarly done in time $O(m)$ by a single Breadth First Search process [2] from the node v_0. Since each repeat of Steps 1–3 removes at least one node from the network, the time complexity of Steps 1–3 is bounded by $O(hn\log n + hm)$, where h is the number of times Steps 1-3 are repeated. In general, the value h is very small (in particular when there are many valid routing paths). In the main body of the algorithm, Step 4, it takes time $O(d)$ to extend the partial path $|P|$ by one link, where d is the maximum number of links connected to a node in the network. Therefore, if the number of paths we try is g, then the time complexity of this step is bounded by $O(d \cdot g \cdot N_0)$. In practice, the values of d and g are small, so our algorithm finds a routing path satisfying the QoS request very efficiently.

4 Analysis and Simulation Results

4.1 Simulating network environment

In order to measure the performance of the on-demand randomized algorithm **RandomRoute** so that it can be compared with other QoS routing algorithms, for example the safest-shortest routing algorithm [12], a simulation testing environment has been developed. The topology used in our experiments is the typical nationwide network of United States of America based on MCI ISP, which is shown in Figure 2.

To simplify the implementation of the simulation, some assumptions have been made as follows:

1. The link capacities are between 10 and 50 Mbps, which are chosen in proportion to a uniform traffic load on the network.

2. The request arrival process at each node is assumed to be Poisson and independent of arrival processes at other nodes. The average request inter-arrival time is about 20 seconds for normal traffic and 3 seconds between the hot-spot nodes.

3. Call duration is exponentially distributed with a mean of 3 minutes.

4. Bandwidth requirements are uniformly distributed between 64k bps and 6 Mbps.

In the second assumption, the average inter-arrival time is less than the one in practical network. In other words, the simulated loading is heavier than real loading. Although it does not match exactly the realistic situation, it can keep the blocking in a definite range so as to benefit to the measurement of how path selection impacts the request acceptance. Otherwise, if network loading is too light, then many paths can satisfy the request, and routing becomes a matter of easy job. The hot-spot nodes are randomly changed during the simulation in order to avoid excessive dependence on network topology details.

4.2 Performance measures

In order to evaluate and compare the performance of QoS routing algorithms, we introduce five parameters to measure the effectiveness of network configuration and QoS routing algorithms. These parameters are computed based on six testing values obtained from our experiments on the network topology given in Figure 2. Note that since the network state information is inaccurate and given in probability, it is possible that a QoS routing algorithm returns a QoS routing path satisfying the given QoS request but the path turns out to be invalid based on the actual configuration of the network, or that the QoS algorithm finds no routing path satisfying the QoS request while there actually exists such a path in the network.

The six testing values are defined as follows:

1. *#Requests* is the total number of QoS requests received during the simulation.

2. *#Successes* is the total number of QoS requests on which the routing algorithm successfully returns a valid QoS routing path.

3. *#FailRoutes* is the total number of QoS requests on which the routing algorithm returns an invalid QoS routing path under the condition that there is no QoS routing path in the network that can satisfy the QoS request.

4. *#WrongRoutes* is the total number of QoS requests on which the routing algorithm returns an invalid QoS routing path under the condition that there is a QoS routing path in the network that satisfies the QoS request.

5. *#WrongRejects* is the total number of QoS requests on which the routing algorithm rejects under the condition that there is a QoS routing path in the network that satisfies the QoS request.

6. *#RightRejects* is the total number of QoS requests on which the routing algorithm rejects under the condition that there is no QoS routing path in the network that can satisfy the QoS request.

Our experiments compute these six values from the simulation, and use the following relative ratios as parameters to measure the performance of QoS routing algorithms.

1. The Successful Routing Ratio: $SRR = \#Successes/\#Requests$

2. The Failing Routing Ratio: FRR = $\#FailRoutes/\#Requests$

3. The Wrong Routing Ratio: WRR = $\#WrongRoutes/\#Requests$

4. The Wrong Rejecting Ratio: WJR = $\#WrongRejects/\#Requests$

5. The Right Rejecting Ratio: RJR = $\#RightRejects/\#Requests$

It is easy to see that $SRR + FRR + WRR + WJR + RJR = 1$. These parameters measure the effectiveness of the QoS routing algorithm and reflect the quality of the update triggering policy and network load configuration. For example, in case the values FRR and RJR are high, the network is overloaded and many QoS requests will be rejected no matter which QoS routing algorithm is used. The values WRR and WJR measure the accuracy of network state information and the consistence of the routing algorithm and the underlying update triggering policy: high values of WRR and WJR indicate that the QoS routing algorithm is often provided with wrong network state information that leads the algorithm to make incorrect QoS routing decisions. The ratio SRR is the major parameter measuring both the effectiveness of the underlying QoS routing algorithm and the feasibility of the network load configuration. A good QoS system should always guarantee a high ratio SRR.

4.3 Protocol simulation environment and experiment results

Our experiments are based on link state information table, which is established by running network layer protocol. The table keeps the values of available bandwidth for each link. Routing algorithms can perform path selection using various strategies according to these values. As we pointed out in Section 2, update triggering policy influences the accuracy of link state information and the protocol cost. Our first experiment is to compare our algorithm **RandomRoute** against other typical QoS routing algorithms in terms of the Successful Routing Ratio SRR based on different update triggering policies. Three QoS routing algorithms have been compared with our algorithm in the experiments: the shortest path routing (i.e., the algorithm that always returns a shortest path regardless the bandwidth safety rate); and the safest-shortest path routing and the shortest-safest path routing algorithms proposed in [12]. Two update triggering policies have been considered: the Threshold Triggering Policy and the Exponential Class Triggering Policy [12]. For both triggering policies, a hold-down time of 100 seconds is used. The experiment results are given in Figure 3.

From Figure 3, we can see that the Successful Routing Ratio SRR of our randomized QoS routing algorithm **RandomRoute** is higher than the other three QoS routing algorithms, under both triggering policies. Under the Threshold Triggering Policy, the Successful Routing Ratio becomes pretty bad for all the algorithms when the threshold value gets large, while the performance of the algorithms looks better under the Exponential Class Triggering Policy.

We have also performed experiments measuring the parameters FRR, WRR, WJR, and RJR, under the Threshold Triggering Policy with hold-down time of 100 seconds for the four algorithms. As shown in Figure 4, our randomized QoS routing algorithm **RandomRoute** gets low Failing Routing Ratio FRR and low Wrong Routing Ratio WRR in most cases. Recall that the ratios FRR and WRR reflect the consistency of a QoS routing algorithm and the underlying update triggering policy. The experiments for the Wrong Rejecting Ratio WJR and for the Right Rejecting Ratio RJR are shown in Figure 5. The ratio WJR is mainly affected by the accuracy of link-state information and the consistency of the update triggering policy and the underlying routing algorithm. If the ratio WJR is high, we should update the link-state information more frequently or select a more consistent routing algorithm for that environment. The ratio RJR reflects the network load, as well as the consistency of the update triggering policy and the underlying routing algorithm.

5 Conclusions

We have developed an effective randomized on-demand QoS routing algorithm on networks with inaccurate link-state information. Several new techniques are proposed in our algorithm. First, the maximum safety rate and the minimum delay for each node in the network are pre-computed, which simplify the network complexity and provide the routing process with useful information. The routing process is dynamically directed by the safety rate and delay of the partial routing path developed so far and by the maximum safety rate and the minimum delay of the next node. Randomness is used at the link level and depends dynamically on the routing configuration. This provides great flexibility for the routing process, prevents the routing process from overusing certain fixed routing paths, and adequately balances the safety rate and delay of the routing path. We have established a network testing environment and introduced five parameters to measure the performance of QoS routing algorithms, the network load configuration, and the consistency of the update triggering policy and the underlying QoS routing algorithm. Experimental results show that in terms of the proposed parameters, our algorithm improves over existing QoS algorithms appearing in the literature.

References

[1] G. Apostolopoulos, R. Guérin, S. Kamat, and S. Tripathi, Quality of Service Based Routing: A Performance Perspective, *Proc. of ACM SIGCOMM'98*, (1998), pp. 17-28, Vancouver, Canada.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, Mass. 1990.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[4] R. Guérin and A. Orda, QoS Routing in Networks with Inaccurate Information: Theory and Algorithms, *IEEE/ACM Trans. Networking*, vol. **7**, (1999), pp. 350-364.

[5] S. Kamat, R. Guérin, A. Orda, and T. Przygienda, QoS Routing Mechanisms and OSPF Extensions, *RFC 2676*, IETF, Aug. 1999.

[6] D. H. Lorenz and A. Orda, QoS Routing in Networks with Uncertain Parameters, *IEEE/ACM Trans. Networking*, vol. **6**, (1998), pp. 768-778.

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

[8] A. Orda, Routing with End to End QoS Guarantees in Broadband Networks, *IEEE/ACM Trans. Networking*, vol. **7**, (1999), pp. 365-374.

[9] B. Rajagopalan and H. Sandick, A Framework for QoS-based Routing in the Internet, *RFC 2386*, IETF, Aug. 1998.

[10] A. Shaikh, J. Rexford, and K. Shin, Dynamics of Quality-of-Service Routing with Inaccurate Link-State Information, *Tech. Rep. CSE-TR-350-97*, Univ. of Michigan, Ann Arbor, MI, Nov. 1997.

[11] S. Shenker, C. Partridge, and R. Guérin, Specification of Guaranteed Quality of Service, *RFC 2212*, IETF, Sept. 1997.

[12] S. Tripathi Improving QoS Routing Performance Under Inaccurate Link State Information, *Proc. ITC'16*, (1998), pp. 1351-1362.

[13] Zhi-Li Zhang, End-To-End support for Statistical Quality-of-Service Guarantees in Multimedia Networks, *PhD thesis*, University of Massachusetts at Amherst, Amherst, MA, Feb. 1997.

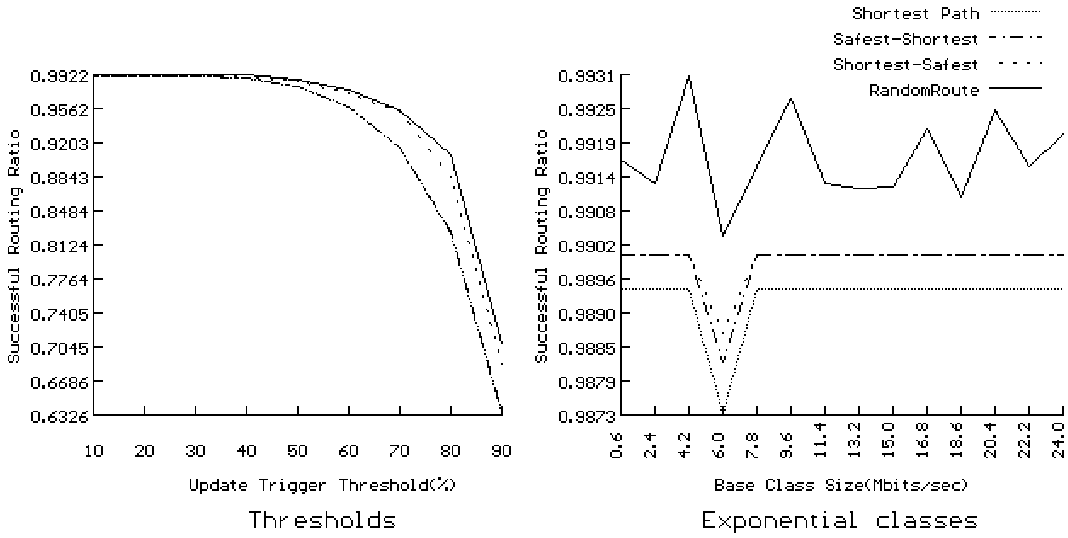

Figure 3. The Successful Routing Ratio SRR under different triggering policies

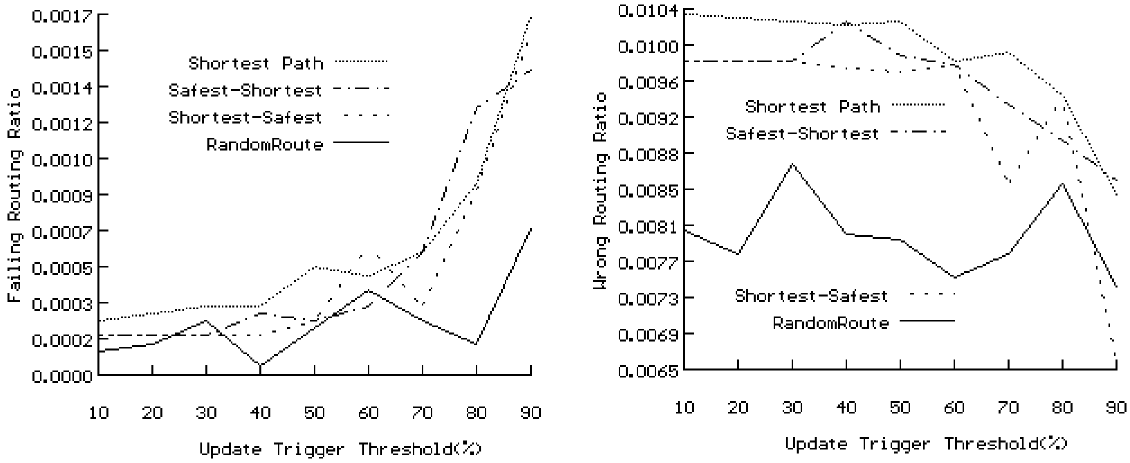

Figure 4. The Failing Routing Ratio and the Wrong Routing Ratio

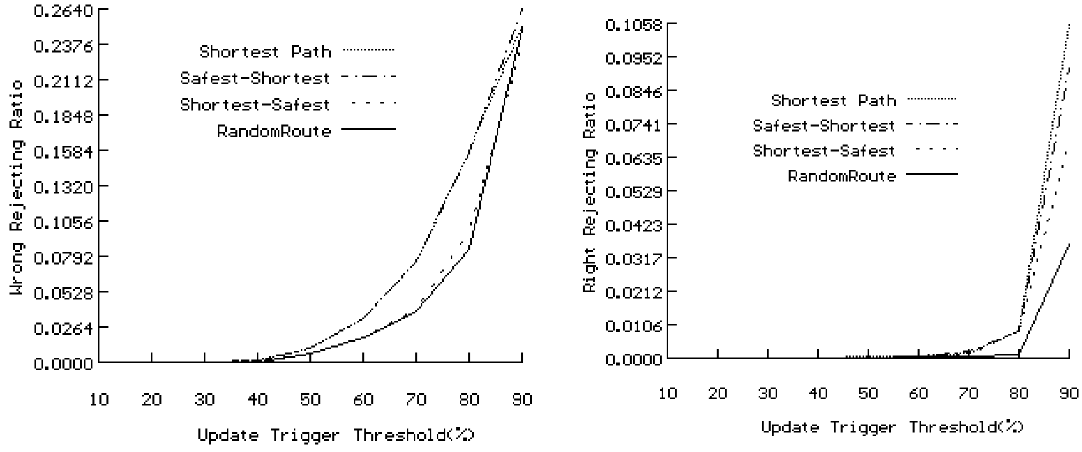

Figure 5. The Wrong Rejecting Ratio and the Right Rejecting Ratio