A Comparative Study of Quality of Service Routing Schemes That Tolerate Imprecise State Information

Xin Yuan Wei Zheng
Department of Computer Science, Florida State University, Tallahassee, FL 32306
{xyuan,zheng}@cs.fsu.edu

Abstract

In large networks, maintaining precise global network state information is almost impossible. Many factors, including non-negligible propagation delay, infrequent link state update due to overhead concerns, link state update policy, resource reservation, and hierarchical topology aggregation, have impacts on the precision of the global network state information. To achieve efficient Quality of Service (QoS) routing, a practical routing algorithm must be able to make effective routing decisions in the presence of imprecise global network state information. In this paper, we compare five QoS routing algorithms that were proposed to tolerate imprecise global network state information, safety-based routing, randomized routing, multi-path routing, localized routing, and static multi-path routing. The performance of these routing algorithms are evaluated under three different link state update policies, the timer based policy, the threshold based policy and the class based policy. The strengths and limitations of each scheme are identified.

1 Introduction

To support QoS routing, global network state information is typically maintained by either a distance vector algorithm [7] or a link state algorithm [8]. In this paper, we will assume that a link state algorithm is used to maintain the global network state information. Using the link state algorithm, when a node detects a change of the state of its links, it performs a link state update, that is, it informs the change to all other nodes in the network using a reliable flooding algorithm. The rule to govern when to perform an update is called the *link state update policy*. In large networks, maintaining precise global network state information in the dynamic environment is almost impossible. Many factors, including non-negligible propagation delay, infrequent link state update due to overhead concerns, link state update policy, and hierarchical state aggregation, have impacts on the precision of the global network state information [5].

Depending on the reason that causes the imprecise state information, the nature of the imprecision is different. The imprecision caused by non-negligible propagation delay or infrequent link state update is *random* in the sense that routers do not have sufficient information to determine the actual value of the link state. When the imprecision is caused by link state update policies, the imprecision is *deterministic* in that routers can usually infer the range of the actual value of the link state and use this information to perform efficient QoS routing. In practice, the imprecision is usually a combination of deterministic imprecision and random imprecision.

Imprecise global network state information can greatly affect the performance of a QoS routing algorithm. It has been shown that a QoS routing algorithm that treats the stale state information as accurate can degrade drastically when the global network state information is imprecise [1, 10]. Hence, mechanisms must be incorporated into a practical QoS routing algorithm to tolerate the imprecise global network state information and make effective routing decisions in the presence of imprecise state information. A number of QoS routing methods that tolerate imprecise state information have been proposed. These methods include safety–based routing [2], randomized routing

[2], multi–path routing [3] and localized routing [9]. Safety–based routing was designed to deal with the deterministic imprecision caused by the link state update policy. It infers the range of the actual link state value from the link state updates, and finds the path that has the highest probability to satisfy a connection request. Randomized routing was introduced to deal with random imprecision. The idea of randomized routing is to compute a set of feasible paths and randomly select one for a connection request. By randomly selecting a feasible path, the randomized routing avoids using the "best" path that is computed based on the imprecise state information and thus, offsets the impact of the imprecise state information. Multi–path routing simultaneously probes a set of feasible paths instead of the single "best" path. Thus, it also alleviates the impact of the imprecise state information. Localized routing totally eliminates the impact of the imprecise global network state information by making routing decisions based on the information maintained locally at each router.

These techniques were proposed for different purposes. It is unclear how effective each technique is in dealing with different types of imprecision and what the relative performance of each technique is. In this paper, we attempt to answer these questions and find methods that can effectively deal with both deterministic and random imprecision through a comparative simulation study of these methods. We study the performance of these methods under different link state update policies, the timer based policy, the threshold based policy, and the class based policy, compare the effectiveness of the methods in dealing with deterministic imprecision, random imprecision and a combination of both, and identify the advantages and limitations of each algorithm. The main conclusions are the followings. First, multi–path routing is effective in dealing with both random imprecision and deterministic imprecision. Second, randomized routing, which randomly chooses a path from a set of feasible paths computed based on the the imprecise global network state information, is ineffective in most cases. Third, static and localized routing offers better performance than the dynamic routing algorithms when the global network state information is extremely imprecise. Fourth, the performance of safety–based routing depends on the characteristics of the imprecision of the global network state information. Safety–based routing is effective in dealing with deterministic imprecision, especially when the state information is precise. Safety–based routing is ineffective in dealing with random imprecision. When the imprecision is mostly random, the performance of the safety–based routing is similar to that of the basic widest–shortest routing algorithm [6]. Furthermore, when the imprecision is a combination of random imprecision and deterministic imprecision, safety–based routing may result in (much) worse performance than that of the basic widest–shortest routing algorithm.

The rest of the paper is structured as follows. We describe the related work in Section 2 and present the link state update policies in Section 3. In Section 4, we discuss the QoS routing schemes that tolerate imprecise state information. Section 5 reports the performance study. Section 6 concludes the paper.

2 Related Work

QoS routing has attracted much attention recently. An extensive survey can be found in [4]. A number of QoS routing schemes that deal with the imprecise state information have been proposed [1, 2, 3, 5, 9, 10]. The impact of the imprecise global network state information on the performance of QoS routing algorithms was studied in [1, 10]. Guerin [5] proved a number of important theoretical results on routing in networks with imprecise state information and proposed QoS routing schemes based on probability. In [2], the probability based QoS routing scheme in [5] was materialized to be the safety-based routing that can effectively deal with the deterministic imprecision caused by the link state update policies. The randomized routing scheme to deal with random imprecision was also introduced in [2]. Chen [3] studied multi-path QoS routing, which simultaneously probes multiple paths for each connection request. Nelakuditi [9] proposed the localized QoS routing, which makes routing decisions solely based on the information maintained locally at each router and eliminates the problems associated with the imprecise global network state information. In this work, we do not invent new methods to deal with the imprecise global network state information. We compare

the effectiveness of safety–based routing, randomized routing, multi–path routing, and localized routing in dealing with deterministic imprecision, random imprecision, and a combination of both, and identify their strengths and weaknesses.

3 Maintenance of global network state information

Link state update policies, which determine when to perform link state updates, affect not only the precision of the global network state information, but also the nature of the imprecision. In this paper, we study three different link state update policies, the timer based policy, the threshold based policy [2], and the class based policy [2]. We will assume that the QoS metric is the bandwidth and that the propagation delay is negligible.

- **Timer based link state update policy**. In the timer based policy, each router periodically updates the state of its links to the rest of the network. The *link state update interval* is a parameter of this policy.
- **Threshold based link state update policy**. This policy is characterized by a threshold value (*th*). Let b^o be the last advertised value of the available bandwidth for a link, b^c be the current available bandwidth, an update is triggered when $\frac{|b^o - b^c|}{b^o} > th$.
- **Class based link state update policy**. In the class based policy, the range of the potential available bandwidth is partitioned into classes. Whenever the available bandwidth in a link changes from one class to another class, a link state update is triggered. Based on how the range of the available bandwidth is partitioned, there are two types of class based policies.
  - *Equal class based*. This policy is characterized by a constant B that is used to partition the available bandwidth W on a link into multiple equal size classes: $(0, B)$, $(B, 2B)$, $(2B, 3B)$, ...$(\lfloor \frac{W}{B} \rfloor B, W)$.
  - *Exponential class based*. Following the definition in [2], this policy is characterized by two constants B and f, $(f > 1)$, which define unequal size classes: $(0, B)$, $(B, (f + 1)B)$, $((f+1)B, (f^2 + f + 1)B)$, ... In our study, we assume $f = 2$.

To control the overheads, a *hold–down timer* is introduced in the threshold and the class based policies. The hold–down timer specifies the minimum time interval between consecutive updates of the same link.

Using the timer based policy, a small link state update interval results in precise state information while a large link state update interval results in imprecise state information. The imprecision resulted from the large update interval is random. The threshold and class based policies allow more accurate link state to be maintained in comparison to the timer based policy since the link state is updated whenever the change of the link state passes the threshold or the class boundary without waiting for the next link state update period. When the hold–down timer is equal to 0, the threshold and class based policies introduce deterministic imprecision. The imprecision is deterministic in the sense that although the absolute value of the link state cannot be determined, the range of the link state value can be decided. For example, using the threshold based policy with $th = 0.1$. When a link declares that its available bandwidth is $10Mbps$, before the next link state update, we know that the bandwidth of that link is in the range of $[9Mbps, 11Mbps]$. A large hold-down timer will introduce random imprecision. By considering these three link state update policies, we can evaluate the effectiveness of different methods in dealing with deterministic imprecision, random imprecision and a combination of both.

4 Routing methods to tolerate imprecise state information

This section briefly describes the QoS routing methods to tolerate imprecise state information that we study in the paper. The methods include safety–based routing [2], randomized routing [2], multi–path routing [3], proportional sticky routing (a localized routing scheme) [9], and multi–path routing. Among these schemes, safety–based routing, randomized routing, and multi–path routing are dynamic schemes that com-

pute feasible paths dynamically based on the current network state information. Proportional sticky routing is a semi–dynamic scheme. It uses a set of pre–computed feasible paths and selects a path dynamically based on the network state inferred based on the information maintained locally at each router. Static multi–path routing always probes a set of pre–computed paths for connection requests and does not adapt to the network dynamics. Next, we will describe these methods.

4.1 Safety–based routing

Safety–based routing algorithms were proposed in [2] to deal with the deterministic imprecision caused by the threshold based policy and the class based policy. It does not apply when the timer based policy is used. The idea is to infer the range of the potential available bandwidth and use the range to compute the *safety* of a link, that is, the probability that the link can support the requested bandwidth. Assuming that the hold–down timer is 0, the range of the available bandwidth value can be determined for the threshold based policy and the class based policy. For the threshold based policy with threshold th, let the last advertised bandwidth value be b_o, before the next link state update for this link, the range of the potential available bandwidth is in between $(1 - th)b_o$ and $(1 + th)b_o$. For the class based policy, the link state advertisement specifies the bandwidth range (class). Given the range $[b_l, b_u]$ for the potential available bandwidth in a link, the safety of the link can be computed based on some bandwidth probability distribution. Our study uses the uniform distribution suggested in [2]. Using this probability distribution and assuming that the requested bandwidth is b_r, the safety of the link is $\frac{b_u - b_r}{b_u - b_l}$, when b_r is in the range $[b_l, b_u]$. When $b_r < b_l$, the link guarantees to support the request, and the safety of the link is 1. When $b_r > b_u$, the link cannot support the request and the safety of the link is 0. Once the safety of each link is determined, the safety of a path is the product of the safety of the links in the path.

Two safety–based algorithms are proposed in [2], *shortest-safest* and *safest-shortest* routing. Both algorithms can be implemented as variations of the Dijkstra shortest path algorithm. A parameter, s, is used to determine whether a path is worth trying. A path is considered as a good path if the safety of the path is larger than s. Safest-shortest routing selects the min-hop path with maximum safety. Shortest-safest routing selects among the safest paths the min-hop path. We will use shortest-safest routing with $s = 0$ in the evaluation since shortest-safest routing has been shown to perform better than safest-shortest routing in [2].

4.2 Randomized routing

The idea of randomized routing [2] is to compute a set of feasible paths and then randomly select a path for the connection. Thus, the routing does not always select the "best" path that is computed based on the imprecise global network state information, which offsets the impacts of the imprecision. In this study, we use the per pair path selection heuristic [11] to determine the set of feasible paths. The per pair path selection heuristic finds shortest paths between a pair of source and destination while minimizing the number of shared links in the feasible paths. It works as follows. The links whose bandwidths are less than the requested bandwidth are deleted from the graph. After that, each link is assigned a distance of 1 and the widest–shortest path is found. This is the first feasible path. After that, the distance of the links in the feasible path is increased and the process repeats until the number of feasible paths reaches the target or no more new paths can be found. By increasing the distance weights for the links in the selected paths, the heuristic tends to avoid using the links that are in the selected feasible paths and thus, the number of shared links among the paths is minimized. In the experiments, the target for the number of feasible paths is 5.

4.3 Multi–path routing

Multi–path routing [3] probes multiple feasible paths simultaneously. It also offsets the impact of the imprecision since it not only probes the "best" path. In the study, we use the per pair path selection heuristic to select up to 5 feasible paths and probe the paths simultaneously. When multiple paths can satisfy the QoS requirement of a connection, the shortest path is selected for the connection. Notice that the same path selection method is also used in the randomized routing method to select the set of feasible paths. Notice also that multi–path routing introduces more over-

heads, however, since this study focuses on studying the effectiveness of the algorithms in dealing with imprecise information, we will ignore the overhead issue.

4.4 Localized routing

Localized routing [9] totally eliminates the impact of the problems associated with the imprecise global network state information by making routing decisions solely based on the information maintained locally at each router. In this study, the localized routing algorithm that we consider is the proportional sticky routing *(psr)* [9].

In the *psr* scheme, it is assumed that each node has a predefined set of candidate paths to each of the destination nodes. For each connection request, *psr* selects a path in the predefined set based on the flow blocking probability. The *psr* scheme can be viewed to operate in two stages: proportional flow routing and computation of flow proportions.

Psr proceeds in *cycles* of variable lengths. A number of cycles form an *observation period*. During the observation period, flows are routed, that is, paths are selected for connection requests, based on a parameter, called *flow proportion*, associated with each candidate path. At the meantime, the information of the flow blocking probability for each candidate path is collected. At the end of the observation period, a new flow proportion for each path is computed. The flow blocking probability of a path indicates the quality of the path and is used to compute new flow proportions. The detailed relation between the flow blocking probability of a path and the flow proportion of the path is described in [9]. Basically, paths with a lower blocking probability will have larger flow proportions so that they will be utilized more frequently in the next observation period.

Given the flow proportion for each path, *psr* works as follows. During each cycle, incoming flows are routed along paths selected from a set of *eligible* paths. Initially, all the candidate paths are eligible paths. Each candidate path is associated with a variable called maximum permissible flow blocking parameter, which determines how many times this path can block a request before the path becomes *ineligible*. The maximum permissible flow blocking parameter may be dynamically adjusted to adapt to network conditions. When all paths become ineligible, a cycle ends and all parameters are reset to start the next cycle. The probability that an eligible path is selected for a flow depends on its flow proportion. The larger the flow proportion, the larger the probability. Hence, better paths that have lower blocking probability will have larger flow proportions and better chances to be selected to route a flow. In addition, *psr* may reduce the maximum permissible flow blocking parameter for an overly loaded path so that the path will not be selected frequently.

One factor that can affect the performance of *psr* is how to compute the pre–determined sets of paths. In our study, we use the global path selection scheme [11] which has been demonstrated to perform well in comparison of other path selection schemes. Once the set of paths is determined, *psr* basically introduces a nice heuristic to decide which path should be used for each connection quest.

4.5 Static multi–path routing

The static multi–path routing always probes the same set of pre–computed paths when a connection request arrives. In the study, we use the same path selection scheme as that used in the *psr* scheme to compute the set of feasible paths between all pairs of nodes. Essentially, the *psr* scheme adds intelligence into path selection in order to select the right path for a connection request while static multi–path routing tries out all potential candidates.

5 Performance study

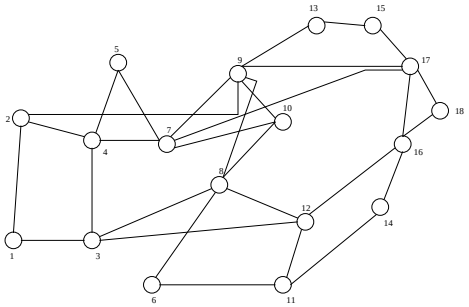


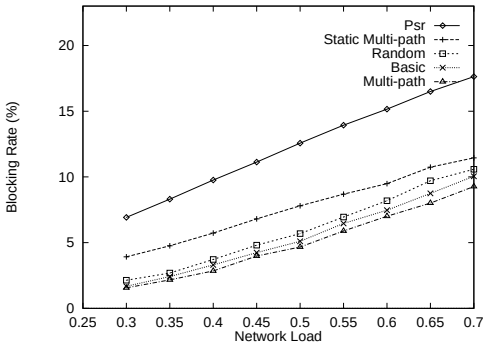
Figure 1: The ISP topology

This section evaluates the performance of the rout-

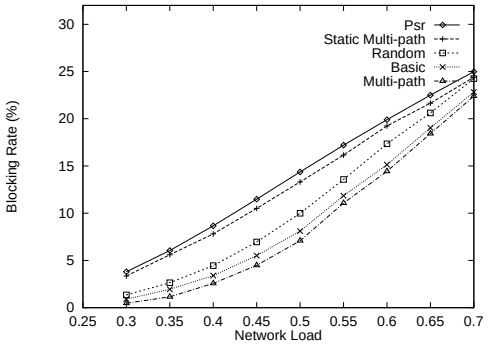
ing algorithms. The topology used is shown in Figure 1. All the links are assumed to be bi-directional and of the same capacity, with C units of bandwidth in each direction. The flow dynamics of the network are modeled as follows. Flows arrive at a node according to a Poisson process with rate λ. The destination node is chosen randomly from all nodes except the source node. The connection holding time is exponentially distributed with mean $1/\mu$. The offered network load is given by $\rho = \lambda N h' B/\mu LC$, where N is the number of source nodes, L is the number of links, h' is the mean number of hops per flow, averaged across all source-destination pairs, and B is the average bandwidth requirement for the flows. The parameters used in this simulation are $C = 20$, $N = 18$, $L = 60$. $h' = 2.36$. The mean connection holding time is 60 seconds, that is, $1/\mu = 60$. Unless specified otherwise, the bandwidth requirement of a flow follows an exponential distribution with a mean value of $B = 3$. The average flow arrival rate, λ, is set depending upon the desired load. Performance of the safety-based routing, multi–path routing, randomized routing and the localized routing is compared with a basic dynamic widest-shortest routing algorithm [6], which will be called the *basic* routing scheme. In the experiments, a blocked flow is dropped without being retried. All the results are obtained with a 95% confidence level and a 5% confidence interval.

Timer based link state update

Figure 2 shows the results when a timer based link state update policy with a small update interval (5 seconds) is used to maintain the link state. Since the safety–based routing does not apply to this method, we only compare randomized routing, multi–path routing, localized routing, static multi–path routing, and the basic scheme. Figure 2 (a) shows the results for the exponentially distributed bandwidth requirement for the flows with a mean value of 3 units. Figure 2 (b) shows the results for the constant 3 units bandwidth requirement for each flow. This experiment shows that when the link state update interval is small, the dynamic schemes in general perform better than the localized routing and static routing schemes. Among the dynamic schemes, multi–path routing performs slightly better than the basic approach, which in turn, performs slightly better than the randomized method.



(a) Exponentially distributed bandwidth requirement



(b) Constant bandwidth requirement

Figure 2: Timer based policy (update interval $= 5$)

As shown in Figure 2 (a), the *psr* scheme does not work well for flows with exponentially distributed bandwidth requirements. This is because *psr* infers the current network state using the blocking probability for the given paths without distinguishing the requests with different bandwidth requirements. Thus, when the bandwidth requirement of the flows is exponentially distributed, the network state inferred is very inaccurate and the performance degrades. When handling flows with the same bandwidth requirement, *psr* gives reasonable results as shown in Figure 2 (b). Although *psr* as it is [9] cannot handle variable bandwidth requirement effectively, we believe that this scheme can be improved. However, it can be expected that no

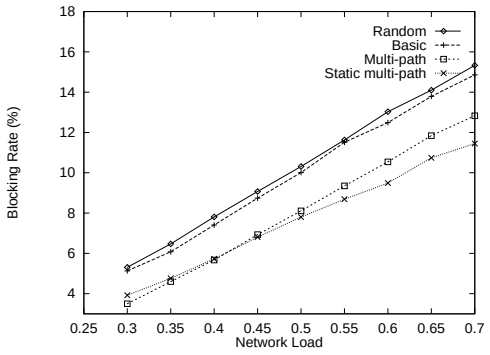


Figure 3: Timer based policy (update interval $= 120$)

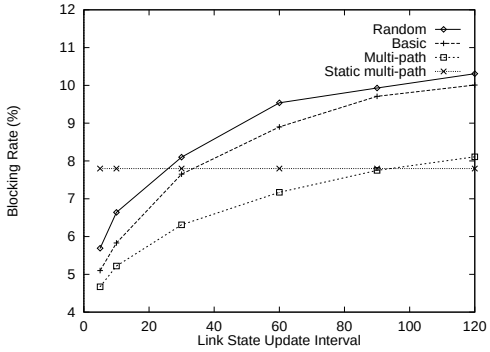


Figure 4: Impact of link state update interval (load $=$ 0.5)

psr type of algorithm can perform better than the static multi–path routing algorithm when both algorithms use the same set of initial feasible paths. In fact, in all the experiments that we performed, *psr* performs much worse than static multi–path routing when the bandwidth requirement of each flow follows the exponential distribution and slightly worse when the bandwidth requirement is a constant. In the rest of the paper, we will omit the results for *psr* and use the static multi–path routing to represent the routing techniques that are not affected by the precision of the global network state information.

Figure 3 shows the results when a timer based policy with a large update interval (120 seconds) is used. The other experimental parameters are the same as those in Figure 2 (a). In this case, static multi–path routing performs the best when the network is under heavy load while the dynamic multi–path routing performs the best when the network is under light load. The multi–path schemes are significantly better than the dynamic single path routing schemes, the basic and the randomized methods, which indicates that multi–path routing is effective in dealing with random imprecision. In this experiment, randomized routing consistently performs worse than the basic method. Actually, in all the experiments, the randomized method either has the similar performance as the basic scheme or performs slightly worse than the basic scheme, which demonstrates that randomly selecting a path from the set of feasible paths computed using the imprecise state information is not effective in dealing with random imprecision. This is because when using the per pair path selection algorithm to compute the set of feasible paths, the lengths of the feasible paths are either the same as the length of the path selected by the basic widest–shortest routing algorithm or slightly longer. When a path is randomly selected from the set of feasible paths, if the quality of the path is not significantly better than that of the path selected by the widest–shortest routing algorithm, the extra length of the randomly selected path will degrade the overall routing performance.

Figure 4 shows the impact of the link state update interval. This experiment assumes the network load to be 0.5 and the bandwidth requirement follows an exponential distribution with a mean value of 3 units. As can be seen from the figure, when the link state update interval becomes larger, the performance of all the dynamic routing algorithms degrades. When the timer based link state update policy is used and the network cannot maintain a high link state update frequency, static routing is preferred.

Threshold based link state update policy

We will first examine the deterministic imprecision resulted from this link state update policy by assuming that the hold-down timer has no effects. Figure 5 shows the results for the threshold based link state update policy with a small threshold value ($th = 0.1$). Thus, the link state is updated when the available bandwidth changes by 10 percent. The other experimental parameters are the same as those for Figure 2 (a). As can be seen from the figure, safety–based routing performs noticeably better than the other routing schemes, which shows that selecting routes based of probability is effective when the network state information is precise. The performance of multi–path routing is similar

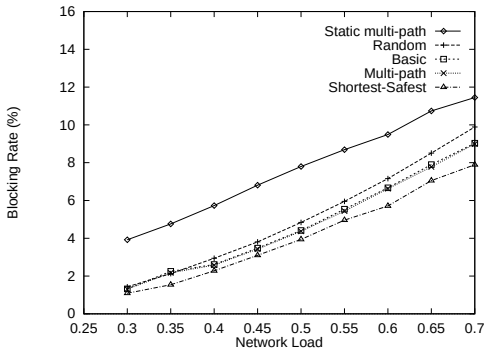


Figure 5: Threshold based link state update policy ($th = 0.1$, $timer = 0$)

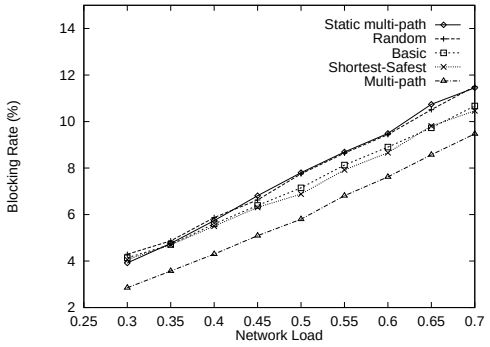


Figure 6: Threshold based link state update policy ($th = 0.9$, $timer = 0$)

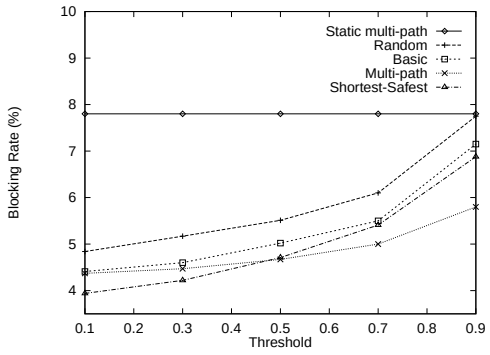


Figure 7: Impact of the threshold, (Load $= 0.5$, timer $= 0$)

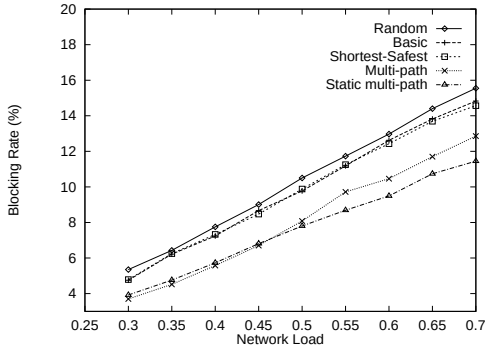


Figure 8: Threshold based link state update policy ($th = 0.1, timer = 120$)

to that of the basic scheme because the global network state information is precise and the basic routing algorithm selects routes effectively.

Figure 6 shows the results for a threshold based policy with a large threshold value ($th = 0.9$). The other experimental parameters are the same as those in Figure 5. In this case, the link state is updated only when the available bandwidth changes by 90%. As can be seen in the figure, the effectiveness of the safety–based routing decreases in comparison to the case when the threshold is small in Figure 5. However, safety–based routing performs slightly better than the basic method and is still the best dynamic uni–path routing method among all the uni–path methods. Multi–path routing performs significantly better than the other algorithms, which indicates that it is very effective in dealing with deterministic imprecision. Figure 7 shows the impact of the threshold on the routing performance assuming that the network load is 0.5. As can be seen in the figure, for all threshold values, safety–based routing consistently performs better than other uni–path routing schemes. This indicates that safety–based routing is effective in dealing with deterministic imprecision. Notice the crossing of curves for the multi–path method and the safety–based method. It shows that the multi–path method can tolerate high deterministic imprecision better than the safety–based method.

Let us now consider the random imprecision in the threshold based link state update policy. Figure 8 shows the results for the threshold based policy with a large hold–down timer (120 seconds) and a small threshold value ($th = 0.1$). In this case, the imprecision is mainly random. These results are similar to those in the timer based policy with a large link state update interval. The multi-path routing algorithms perform the best with static multi–path routing performing better than dynamic multi–path routing under high load and dynamic multi–path routing performing better un-

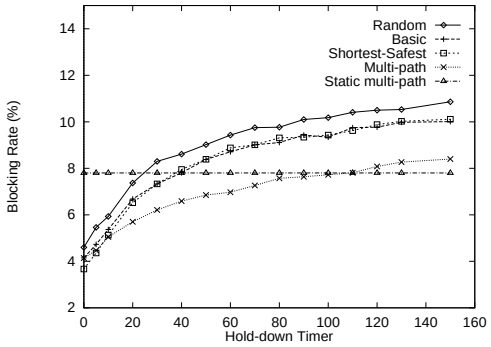


Figure 9: Impact of the hold–down timer ($th = 0.1$, $load = 0.5$)

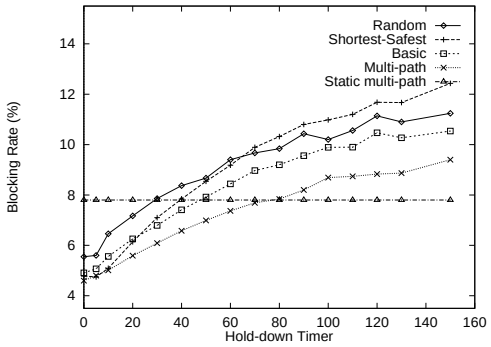


Figure 10: Impact of the hold–down timer ($th = 0.5$, $load = 0.5$)

der low load. The safety–based routing algorithm has the similar performance as the basic method, which indicates that safety–based routing is ineffective in dealing with random imprecision.

Figure 9 shows the impact of the hold-down timer when the threshold is small ($th = 0.1$). The experiment assumes $load = 0.5$. As the hold–down timer becomes larger, the global network state information becomes increasingly randomly imprecise. As can be seen in the figure, safety–based routing is effective only when the hold–down timer is small. When the hold–down timer is larger than 20 seconds, safety–based routing has the similar performance as the basic algorithm. Dynamic multi-path routing is more effective than other dynamic methods. When the hold-down timer is larger than 110 seconds, static multi–path routing yields the best performance.

Figure 10 shows the impact of the hold–down timer when the threshold is larger ($th = 0.5$). This experiment assumes $load = 0.5$. In this case, the global network state information is very imprecise. Both random imprecision and deterministic imprecision are involved. Under this condition, the performance of safety–based routing degrades much faster than all other routing schemes as the hold-down timer becomes larger. When the hold-down timer is large, safety–based routing yields the highest blocking rate. Computing paths based on imprecise safety results in worse results than computing paths based on imprecise link state value. Hence, safety–based routing cannot tolerate the combination of large random imprecision and large deterministic imprecision. This experiment also shows that when the global network state information is imprecise, static multi–path routing becomes more appealing.

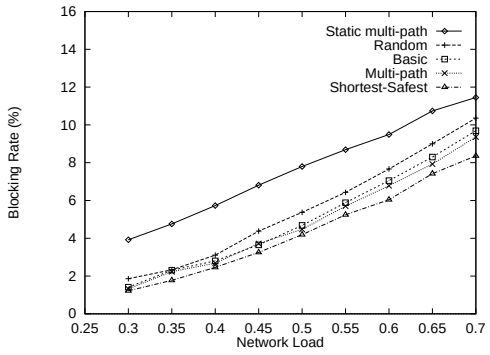


Figure 11: Equal class based link state update policy ($size = 2$, $timer = 0$)

Class based link state update policy

The performance of the exponential class based link state update policy is very similar to that of the threshold based policy. The conclusions made for the threshold based link state update policy also apply to the exponential class based policy. This section will only present the results for the equal class based policy. The main difference between the equal class based policy and the threshold based policy is that the equal class based policy generates more accurate state information for links with a large available bandwidth and less accurate state information for links with a small available bandwidth.

We will first consider the deterministic imprecision introduced in the equal class based policy. Fig-

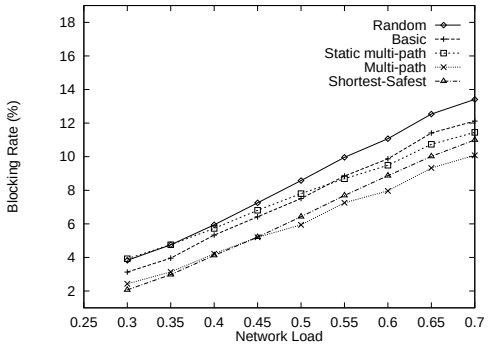


Figure 12: Equal class based link state update policy ($size = 10$, $timer = 0$)

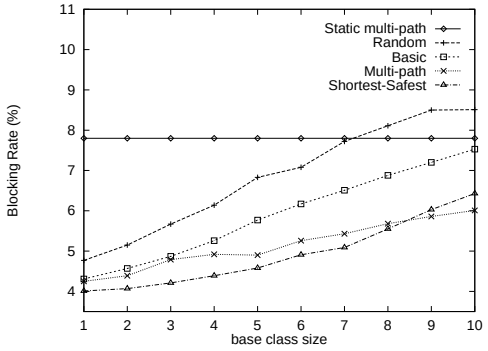


Figure 13: Impact of the class size ($load = 0.5$, $timer = 0$)

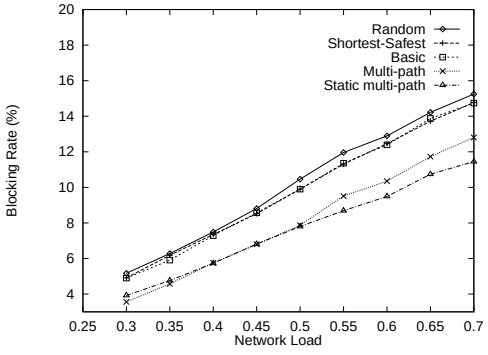


Figure 14: Performance of the equal class based policy with a large hold-down timer ($size = 2$, $timer = 120$)

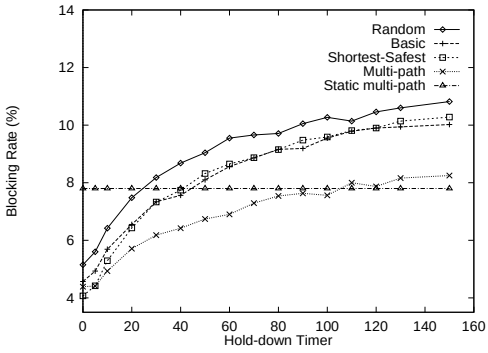


Figure 15: Impact of the hold-down timer ($size = 2$, $load = 0.5$)

ure 11 shows the results for the equal class based link state update policy with a small class size ($size = 2$). The this case, there are ten bandwidth classes, $(0, 2)$, $(2, 4)$, $(4, 6)$, $(6, 8)$,$(8, 10)$, $(10, 12)$,$(12, 14)$, $(14, 16)$,$(16, 18)$, $(18, 20)$. Safety–based routing performs the best for all network loads. Multi–path routing performs slightly better than the basic method and the randomized method is slightly worse than the basic method.

Figure 12 shows the results for the equal class based link state update policy when the class size is large ($size = 10$) and the hold-down timer is zero. There are only two classes in this case, $(0, 10)$ and $(10, 20)$. Safety based routing performs better than the other uni–path method, which indicates that safety–based routing is effective in dealing with the deterministic imprecise caused by the equal class based policy. Under high load, the multi–path method performs the better than safety–based routing.

Figure 13 shows the results for different class sizes. This experiment assumes $load = 0.5$. As can be seen in the figure, the safety–based method is very effective in dealing with the imprecision caused by the large class sizes. Comparing the Figure 7, we can see that the improvement of safety based routing over the basic method is larger in the equal class based policy than that in the threshold based policy.

Figure 14 shows the performance of the equal class based policy with a small class size ($size = 2$) and a large hold-down timer (120 seconds). Figure 15 shows the impact of the hold down timer on the fine grain bandwidth classes ($size = 2$). This experiment assumes $load = 0.5$. Figure 16 shows the impact of the hold down timer on the coarser grain bandwidth classes ($size = 5$). This experiment also assumes $load = 0.5$. The trends in these figures are exactly the same as their corresponding results for the threshold based link state update policy in Figure 8, Figure 9, and Figure 10 re-

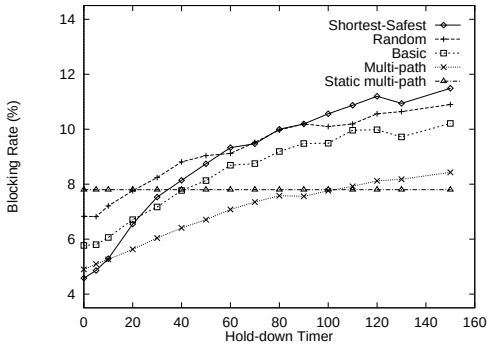


Figure 16: Impact of the hold-down timer ($size = 5$, $load = 0.5$)

spectively and the same conclusions can be drawn for the equal class based policy.

The study of the threshold based link state update policy and the class based link state update policy shows that the performance of safe–based routing depends on the characteristics of the imprecision. It is effective in dealing with the deterministic imprecision resulted from both link state update policies. Using both policies, when the imprecision is mainly random resulted from a large hold–down timer, safety–based routing has similar performance as the basic method. However, in the case when the imprecision is both random and deterministic, safe-based routing may result in worse performance than the basic method. This study also shows that dynamic multi–path routing is effective in deal with deterministic imprecision.

6 Conclusions

In this work, we investigated the Quality of Service routing schemes that tolerate imprecise link state information. Five routing methods, namely safety–based routing, randomized routing, multi-path routing, localized routing, and static multi–path routing, are considered. The interaction between the routing algorithms with three link state update policies, the timer based policy, the threshold based policy and the class based policy, is studied. The conclusions are the followings. First, multi–path routing is effective in dealing with both random imprecision and deterministic imprecision. Second, randomized routing is ineffective in most cases. Third, static and localized routing offers better performance than the dynamic routing algorithms when the global network state information is extremely imprecise. Fourth, the performance of safety–based routing depends on the characteristics of the imprecision of the global network state information. Safety–based routing is effective in dealing with deterministic imprecision and ineffective in handling random imprecision. Furthermore, safety–based routing may result in poor routing performance when the imprecision is a combination of random imprecision and deterministic imprecision. These conclusions suggest that to design effective routing algorithms that can tolerate imprecise state information and make effective routing decisions in the presence of imprecise state information, the characteristics of the global network state information maintained at each router, which depends on many network components, needs to be studied.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat and S. Tripathi "Quality of Service Based Routing: A Performance Perspective", *ACM SIGCOMM*, 1998.

[2] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Improving QoS Routing Performance Under Inaccurate Link State Information." *Proceedings of the 16th International Teletraffic Congress*, June 7-11, 1999.

[3] S. Chen and K. Nahrstedt, "Distributed QoS Routing with Imprecise State Unformation." *ICCCN'98*, October 1998.

[4] S. Chen and K. Nahrstedt, "An Overview of Quality–of–Service Routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Networks*, Special Issue on Transmission and Distribution of Digital Video, Nov./Dec., 1998.

[5] R. Guerin and A. Orda, "QoS–based Routing in Networks with Inaccurate Information: Theory and Algorithms.", *IEEE INFOCOM'97*, April 1997.

[6] Q. Ma and P. Steenkiste, "Quality–of–Service Routing with Performance Guarantees", *4th IwQoS'97*, May 1997.

[7] G. Malkin, "RIP Version 2.", RFC 2453, November 1998.

[8] J. Moy, "OSPF Version 2", RFC 2328, April 1998.

[9] Srihari Nelakuditi, Zhi-Li Zhang, and Rose P. Tsang, "Adaptive Proportional Routing: A Localized QoS Routing Approach", *In IEEE Infocom*, April 2000.

[10] A. Shaikh, J. Rexford and K. Shin, "Evaluating the Overheads of Source-Directed Quality-of Service Routing", *International Conference on Network Protocols* (ICNP), 1998.

[11] X. Yuan and A. Saifee, "Path Selection Methods for Localized Quality of Service Routing." *IC3N'01*, Oct. 2001.