Path Selection Methods for Localized Quality of Service Routing

Xin Yuan Arif Saifee
Department of Computer Science
Florida State University
Tallahassee, FL 32306

Abstract

Localized Quality of Service (QoS) routing was recently proposed as an alternative to the QoS routing algorithms that use global network state information to make routing decisions. In localized QoS routing, each router maintains a predetermined set of candidate paths for each of the destinations. For a connection request, a router decides the path for the request based on the information maintained locally at the router. Hence, localized QoS routing does not use any global network state information in the routing process and avoids the problems associated with the maintenance of the global network state information. To achieve good routing performance, localized QoS routing must select the predetermined set of candidate paths between the source and the destination effectively. This paper studies path selection methods for localized QoS routing. Five path selection heuristics, namely breadth-first search path selection, per-pair shortest path selection, global path selection, hybrid per-pair/global path selection, and per-pair path selection with global tuning, are proposed and their performance is evaluated through simulation. We conclude that path selection methods can greatly affect the performance of localized QoS routing and that an effective path selection algorithm must consider various factors, including path length and load balancing in the whole network.

1 Introduction

Quality of Service (QoS) routing identifies paths that meet the QoS requirement of a connection and selects the one that leads to high overall resource efficiency. Most of the existing QoS routing algorithms rely on the global network state information to make routing decisions [4, 6, 10, 11, 12, 13]. Such algorithms will be called *global QoS routing algorithms*. Global QoS routing algorithms require the global network state information to be exchanged periodically using either the link state algorithm [8] or the distance vector algorithm [7]. For such algorithms to be effective, the global network state information must be precise and the maintenance of the global network state information may incur large overheads in both communication and CPU processing.

The localized QoS routing scheme [9] attempts to overcome the problems associated with the maintenance of the global network state information by making routing decisions based solely on the information maintained locally at each router. In the localized QoS routing scheme, each router has a predetermined set of candidate paths to each of the destinations. The source node infers the network QoS state based on the information collected locally, such as the flow blocking probability, and selects a path from the predetermined set of candidate paths for a connection request. In this paper, we will use the words *flow* and *connection* interchangeably. Localized QoS routing does not require the exchange of global network state information and eliminates all the problems associated with it.

While it has been demonstrated that localized QoS routing is simple, stable, adaptive and effective in comparison to global QoS routing schemes [9], a number of issues remain to be addressed. How many candidate paths do we need to have for each pair of nodes? How to select the candidate paths? Since the predetermined sets of candidate paths govern how the localized QoS routing algorithm works, it is crucial to select "good" candidate paths for the routing algorithm to achieve good performance. These important issues were ignored by the previous study of localized QoS routing [9] where an ad hoc method was used to select candidate paths. This paper fills in the gap by investigating path selection schemes that systematically determine candidate paths for a given network.

To achieve the best performance, a number of factors must be considered in the path selection process. First, a path selection method should give higher priority to shorter paths since using a longer path for a connection requires more network resources. Second, a path selection method should try to balance the load of the network so that the network can accommodate more connections. Third, the number of links shared by the paths for a given source-destination pair should be minimized to allow the routing algorithm to be adaptive. When designing path selection schemes, we may not be able to optimize all these factors. Thus, trade-offs must be made. In this paper, we will focus on studying the following five path selection heuristics that give priority to different factors.

- *Breadth-first search path selection* (BFS). This heuristic is based on the breadth-first search algorithm and is able to find many minimum-hop paths between a source and a destination. This heuristic ignores the global load balancing and shared link issues.

- *Per-pair path selection* (PP). PP selects the shortest paths for each source-destination pair while minimizing the number of shared links. This heuristic ignores the global load balancing issue.

- *Global path selection* (GP). GP selects paths for all pairs of sources and destinations, considering all the three factors at the same time. However, by considering the global load balancing issue, the paths for a given source-destination pair may be longer than those selected by PP and BFS.

- *Hybrid per-pair/global path selection* (PPGP). PPGP uses PP to select the first candidate path for each source-destination pair and then uses GP to select the rest of the candidate paths. PPGP guarantees a shortest path between a source-destination pair to be selected as a candidate path.

- *PP with global tuning* (PPGT). PPGT uses PP to decide the initial set of candidate paths. PPGT then tries to achieve global load balancing by eliminating the paths that use the most loaded links.

By studying these heuristics, we hope to determine which factors are more important and which factors are less critical and to find an efficient way to perform path selection for localized QoS routing. We analyze static measurements of the paths selected by the heuristics and perform extensive simulations to compare the performance of the heuristics. Our main conclusions include the followings. First, path selection methods can greatly affect the performance of localized QoS routing. Second, localized QoS routing with multiple candidate paths between each source-destination pair performs better than that with only one path between each source-destination pair. Third, in order for localized QoS routing to be effective, only the paths whose lengths are close to the minimum-hop (between the source and the destination) should be selected as candidate paths. Fourth, global load balancing is an important factor in the path selection process.

The rest of the paper is structured as follows. Section 2 introduces a localized QoS routing algorithm, proportional sticky routing, which is the only localized QoS routing algorithm proposed. We evaluate the path selection schemes using this algorithm. Section 3 describes the path selection heuristics. Section 4 studies the performance of the heuristics. Section 5 concludes the paper.

2 Localized QoS routing

In this section, we will briefly describe the *proportional sticky routing* (psr) algorithm, which is a localized QoS routing scheme. Details about this algorithm can be found in [9]. This is the only existing localized QoS routing algorithm and our path selection schemes are developed for this algorithm.

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 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, the proportional flow routing 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.

The *psr* scheme maintains self-adaptivity by controlling the number of flows routed along a path in each cycle using the maximum permissible flow blocking parameter and by re-adjusting flow proportions after every observation period. For example, if the load along a path increases, causing the number of flows blocked to quickly reach the maximum permissible blocks, the source will automatically back off from this path by eliminating it from the eligible path set for the rest of the cycle. If this situation persists, at the end of the observation period, the new flow proportion for the path will be reduced. Likewise, if the load on the path decreases, its new flow proportion will be increased at the end of the observation period. Furthermore, because the length of each cycle is not fixed, but determined by how fast each eligible path reaches its maximal permissible blocks, the length of an observation period also varies. This self-adjusting observation period allows the *psr* scheme to respond to the system load fluctuations in an elastic manner. If the system load changes suddenly, the old flow proportions would result in rapid termination of cycles, which would in turn lead to faster ending of the current observation period. New flow proportions will thus be re-computed to adapt to the system load. On the other hand, if the system load is stable, the observation periods will also be stabilized, with increasingly accurate calibration of the flow proportions. As a result, flow proportioning will eventually converge to the equilibrium state.

As can be seen from the above discussion, psr has the ability to self-adapt to the network condition by using more frequently good paths and avoiding bad paths. This leads to a fundamental question: Is the selection of candidate paths important for this routing algorithm? If yes, how many candidate paths do we need for each source and destination pair? How to select the candidate paths for this algorithm? The rest of the paper will attempt to answer these questions.

3 Path selection methods

This section will present a number of path selection algorithms that systematically determine the set of candidate paths to be used by the proportional sticky routing algorithm. In the candidate path selection process, a number of factors need to be considered:

- *Path length.* As discussed in [6], QoS routing algorithms that favor shorter paths yield better routing performance. Since the *psr* scheme only selects paths from the set of candidate paths, it is critical to select short paths as candidate paths for *psr* to achieve good performance. This factor implies that the shortest path algorithm is a good algorithm to select the candidate paths.

- *Load balancing.* Since the set of candidate paths is predefined, it is critical to evenly distribute the load by carefully selecting the candidate paths. To achieve load balancing, the traffic pattern must be considered. In this paper, we assume uniform traffic. Load balancing must be done in the global scale, that is, the aggregate effect of all candidate paths on all links in the network must be considered. The shortest path algorithm does not take the load balancing issue into account. Thus, other mechanisms must be incorporated in the path selection process to achieve global load balancing. Notice that due to the self-adaptivity of psr, the static load balancing, that is, load balancing at the path selection process, may not result in dynamic load balancing.

- *Shared links.* For a given source-destination pair, multiple candidate paths are to be selected. We should minimize the number of shared links in the candidate paths for a given source-destination pair so that *psr* can be adaptive to the network state. Essentially, when a link is under heavy load, *psr* should select other paths that do not use the link for connection requests.

An efficient path selection scheme should optimize all these factors. In practice, however, a perfect algorithm cannot be obtained and compromises must be made. In the following, we will present a number of path selection heuristics, which focus one or more of the factors while ignoring others. Five path selection heuristics will be described. We assume that the network load is uniform and all links have the same bandwidth for simplicity. The heuristics can be modified to handle non-uniform traffic and un-even link bandwidth. We also assume the bandwidth as the QoS metric and that all links in the network can satisfy the QoS requirement. Notice that this is not a restriction, we can always obtain a network that contains only the links that can satisfy the bandwidth constraint by eliminating all the links that cannot satisfy the constraint.

Breadth first search path selection *(BFS)*

This heuristic focuses on selecting shortest paths between each source-destination pair as candidate paths. The heuristic uses a breadth first search algorithm to find *all* minimum-hop paths, as well as some alternative paths with a limited extra path length (e.g. 1 hop more than the minimum-hop), for each source-destination pair. It then selects from the found paths the set of candidate paths. This heuristic ignores the global load-balancing and shared link issues.

Per-pair path selection *(PP)*

This heuristic attempts to find shortest paths between each source-destination pair while minimizing the number of shared links in the candidate paths for a given source-destination pair. This heuristic selects candidate paths for each source-destination pair in the following manner. We initially assign the same weight, 1, to all links in the network and use the Dijkstra shortest path algorithm to find the first candidate path. After the heuristic finds a candidate path, it increases the weights on all the links along the path and then repeats the process to find the next candidate path until the number of candidate paths reaches the target or no more new paths can be found. By increasing the weights for the links in the selected path, the heuristic tends to avoid using the links that are in the selected candidate paths and thus, the number of shared links among the paths for a source-destination pair is minimized. However, this heuristic selects candidate paths for each source-destination pair and does not consider the global load balancing issue. By avoiding using links in the candidate paths, the paths selected by PP may be longer than those selected by BFS.

Global path selection *(GP)*

The basic idea of GP is to select paths for all source-destination pairs such that the load is evenly distributed to all links. This is done as follows: We first assign the same weight, 1, to all links in the network. We then consider each pair of nodes in a round-robin fashion. For the first pair of nodes, we select one shortest path using the Dijkstra shortest path algorithm and increase the weights on the links along the path. We do the same for the second source-destination pair and so on. Since the weights of the links on the paths chosen previously have been made higher, it is likely that the links in the selected paths will not be chosen in the future. Since we select candidate paths for all the source-destination pairs, it is likely that, at the end, the load on all links will be balanced since the less loaded links are more likely to be included in a new candidate path. Like in PP, the candidate paths selected by GP may not be as short as those selected by BFS. However, the average load on each link is likely to be more even across all links.

Algorithm	Path Length	Global Load Balancing	Shared Links
BFS	Yes	No	No
PP	Yes	No	Yes
GP	Yes	Yes	Yes
PPGP	Yes	Yes	Yes
PPGT	Yes	Yes	Yes

Table 1: Summary of the heuristics

Hybrid per-pair/global path selection *(PPGP)*

This heuristic tunes the path selection process of GP. The idea is to include at least one shortest path in the candidate path for each source-destination pair while balancing the global load. In PPGP, we first use the PP to select one path for each source-destination pair. The paths selected by PP are minimum-hop paths. We then assign the weights according to the network load and find alternative paths using the global path selection method.

PP with global tuning *(PPGT)*

PPGT tries to achieve global load balancing using a different approach. In this heuristic, we first select the set of candidate paths using PP. Since PP does not take global load balancing into consideration, it is possible that the load on some link is much higher than the load on other links. Thus, the heuristic tries to balance the load by removing paths that use the most loaded links. The process is repeated until no more paths can be removed, that is, the number of paths between each source-destination pair is below a target value. In this heuristic, if we want to find x paths between two nodes, we will first use PP to find $2x$ paths and then start the removing process.

Table 3 summarizes the heuristics. Notice that although GP, PPGP and PPGT consider all the three factors, they have different emphases. Two parameters are incorporated into all the heuristics, the *maximum number of paths between two nodes* and the *extra path length*. The maximum number of paths parameter controls the number of candidate paths to be found. The extra path length parameter is incorporated to the heuristics so that the heuristics will only find paths that are within the extra path length of the minimum hop path. In other words, the lengths of all candidate paths found by the heuristics are less than minimum-hop plus the extra path length parameter.

4 Performance study

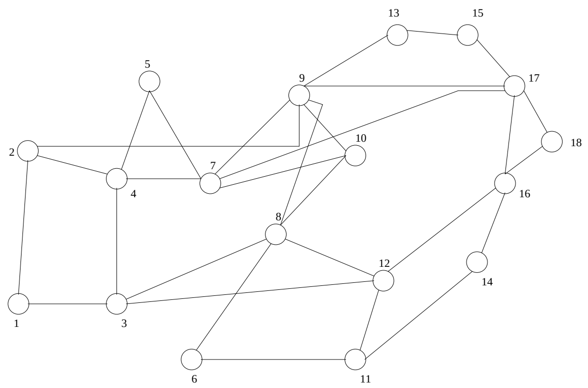


Figure 1: The ISP topology

This section evaluates the performance of the proposed path selection algorithms. In this study, we use the same simulation environment as that used in [9]. Figure 1 shows the *isp* topology used in the study. All the links are assumed to be bi-directional and of the same capacity, with C units of bandwidth in each direction. Flows arriving into the network are assumed to require one unit of bandwidth. Hence each link can accommodate at most C flows simultaneously. The flow dynamics of the network are modeled as follows. Flows arrive at a source node according to a Poisson process with rate λ. The destination node of a flow is chosen randomly from the set of all nodes except the source node. The holding time of a flow is exponentially distributed with mean $1/\mu$. The offered network load is given by $\rho = \lambda N h'/\mu L C$ where N is the number of source nodes, L is the number of links and h' is the mean number of hops per flow, averaged across all source-destination pairs. The parameters used in this simulation are $C = 20$, $N = 18$, $L = 60$. $h' = 2.36$ and $1/\mu = 60$ seconds. The average arrival rate at a source node λ is set depending upon the desired load. In *psr*, 3 cycles form an observation period and the maximum permissible flow blocking parameter is initially set to 5.

4.1 Static measurements

We first examine the static measurements of the paths selected by the heuristics. Figure 2 shows the average number of paths found using each of the heuristics. In this experiment, the extra path length is assumed to be 1 if the minimum-hop between the source and the destination is 1 or 2, and 2 if the minimum-hop is more than 2. The x-axis is the maximum number of paths between two nodes. The

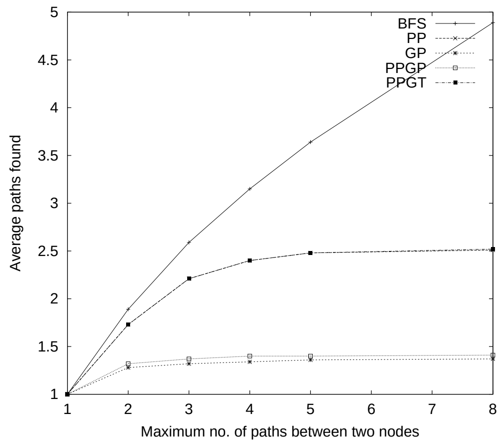


Figure 2: The average number of paths between a source and a destination

y-axis is the average number of paths actually found by the heuristics. This figure shows the capability of each heuristic to find different paths between a pair of nodes. As can be seen from the figure, the average number of paths found by BFS grows almost linearly with the maximum number of paths, which indicates that BFS can found many different paths. PP and PPGT find on average around 2.5 paths even when the maximum number of paths between two nodes is 8. GP and PPGP are the worst in terms of finding different paths. They can only find an average of 1.4 paths between two nodes since they repeatedly select the same candidate paths and are unable to find other potential candidate paths.

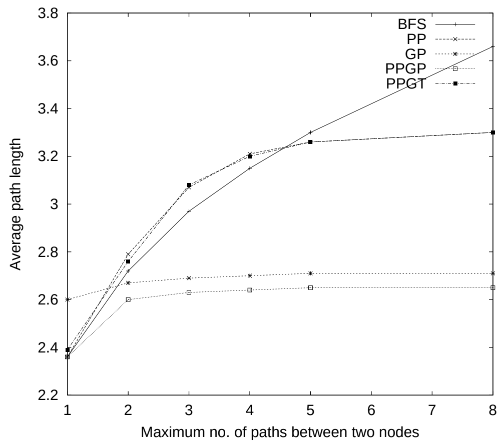


Figure 3: The average path length

Figure 3 shows the average path length for all the paths found using each of the heuristics with the same experimental setting as that in Figure 2. The paths found by PPGP is the shortest, followed by GP. This indicates that the candidate paths found by PPGP and GP are mostly good quality minimum-hop paths although these two heuristics cannot find many candidate paths. When the maximum number of paths is less than 4, BFS is better than PP and PPGT. When the maximum number of paths is larger than 4, BFS results in the largest average path length. The reason is that BFS can find more paths, including both minimum-hop paths and alternative paths. When the total number of paths between a pair of nodes is small (< 4), BFS will mostly select minimum-hop paths as candidate paths. When the total number of paths is large (> 4), more alternative paths will be included, resulting in a larger average path length.

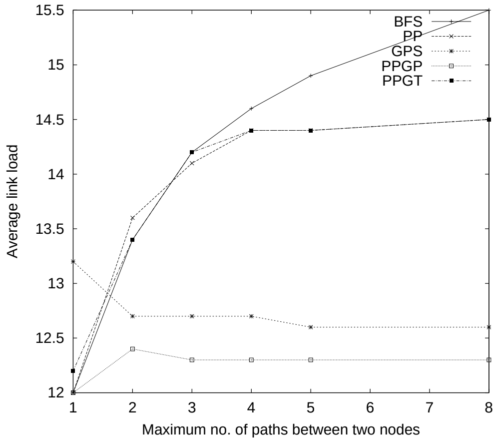


Figure 4: Average link load

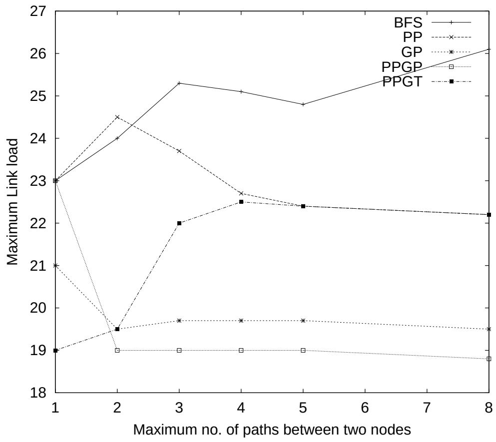
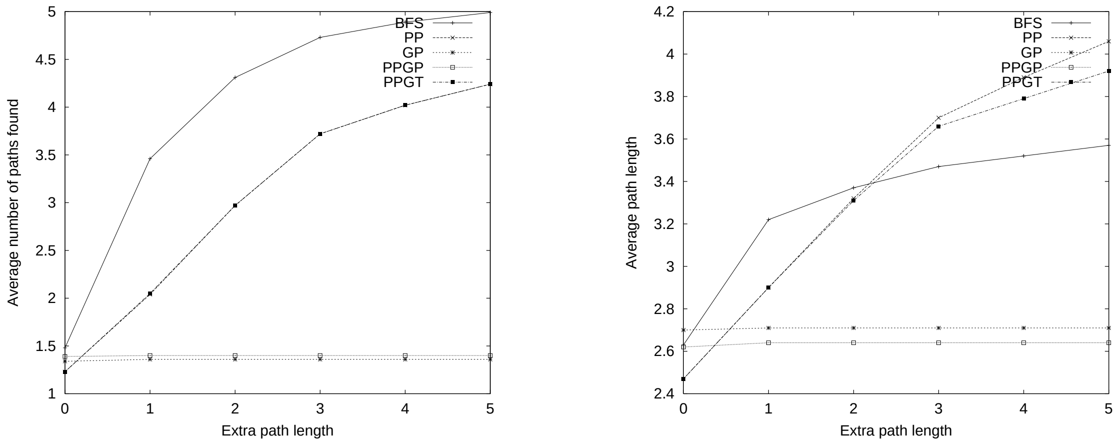


Figure 5: Maximum link load



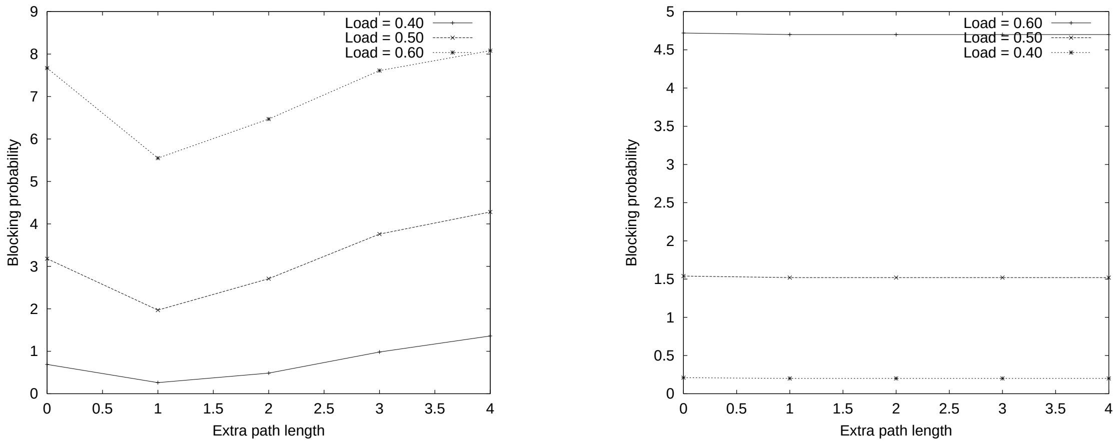
(a) Average number of paths

(b) Average path length

Figure 6: Static measurements for different extra path lengths

Figure 4 shows the average static link load and Figure 5 shows the maximum static link load using the same experimental setting as that in Figure 2 and Figure 3. The static link load is computed by assuming that there is 1 unit of data flowing between each pair of nodes and the unit is evenly distributed on all paths between the pair of nodes. When the maximum number of paths between two nodes is larger than 2, GP and PPGP have lower average load in comparison to BFS, PP and PPGT. This is because GP and PPGP tend not to select alternative paths as candidate paths while BFS, PP and PPGT select alternative paths as candidate paths. Comparing PP and PPGT, we can see that when the maximum number of paths between two nodes is small (< 5), PPGT yields lower average link load by considering the global load balancing issue. When the maximum number of paths between two nodes is larger (≥ 5), by limiting extra path length, PPGT and PP virtually find the same set of candidate paths since both heuristics find an average of 2.5 paths between each pair.

Figure 6 shows the average number of paths found and the average path lengths for different extra path lengths, assuming that the maximum number of paths per pair of nodes is 5. BFS. PP and PPGT can find more paths when a larger extra path length is allowed while GP and PPGP do not have this ability. This indicates that GP and PPGP only find good paths. BFS, PP and PPGT can find more paths with longer path lengths.



(a) PP

(b) GP

Figure 7: Impact of the extra path length

Since many factors can affect the routing performance, the static measurements do not give a clear picture about the heuristics. On the one hand, BFS finds more paths than other heuristics, which allow the localized routing algorithm to be more adaptive. On the other hand, the paths found by BFS are longer than those found by other heuristics, which can result in higher network load and degrade the performance. GP and PPGT cannot find many paths. Yet the paths found by these two heuristics are shown to be of high quality. The next section studies the overall effects by comparing the dynamic measurement of the blocking probability for all the path selection algorithms.

4.2 Dynamic measurements

We first consider the impact of the extra path length parameter. Figure 7 shows the impact of the extra path length parameter on PP and GP. The impact on PPGT and BFS is similar to that on PP, the impact on PPGP is similar to that on GP. In this figure, the maximum number of paths between a source-destination pair is 5. As can be seen in the figure, for PP, an extra path length of 1 to 2 results in the best performance. When the extra path length is 0, the algorithm can only select minimum hop paths as candidate paths. In this case, although the average path length is small, the number of paths between each pair is also small (as observed in Figure 6) and the self-adaptivity of psr cannot be exploited. When the extra path length is too large, the average path length is too long and the routing performance decreases. For GP, the extra path length parameter does not make much difference. The reason for this is that GP only selected a very small set of paths as candidate regardless of the extra path length parameter. As observed in Figure 6, the extra path length parameter does not affect the path selection process of GP that much. This experiment shows that in order for localized QoS routing to be effective, only the paths whose lengths are very close to the minimum-hop should be selected as candidate paths. Since the best performance is obtained when the extra path length is between 1 and 2, for the rest of the experiments, we will assume that the extra path length is equal to 1 if the minimum hop is 1 or 2, and 2 if the minimum hop is more than 2.

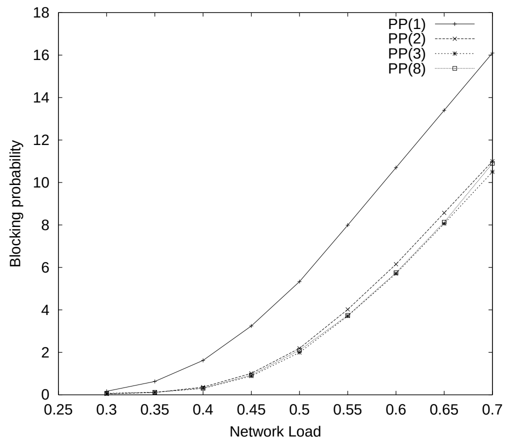


Figure 8: PP with different maximum numbers of paths per pair of nodes

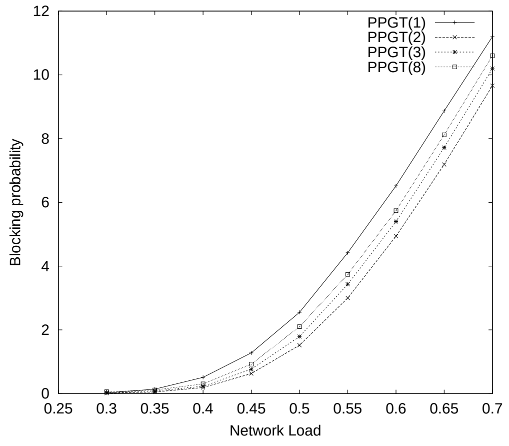


Figure 9: PPGT with different maximum numbers of paths per pair of nodes

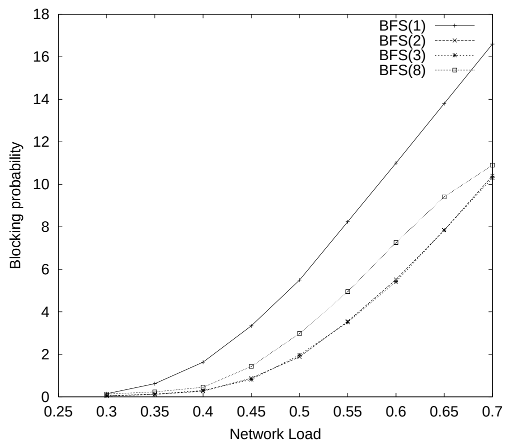


Figure 10: BFS with different maximum numbers of paths per pair of nodes

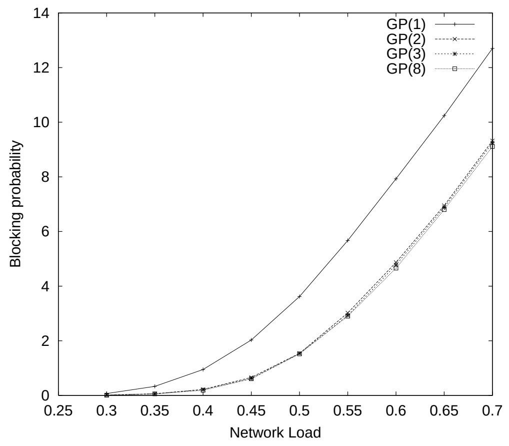


Figure 11: GP with different maximum numbers of paths per pair of nodes

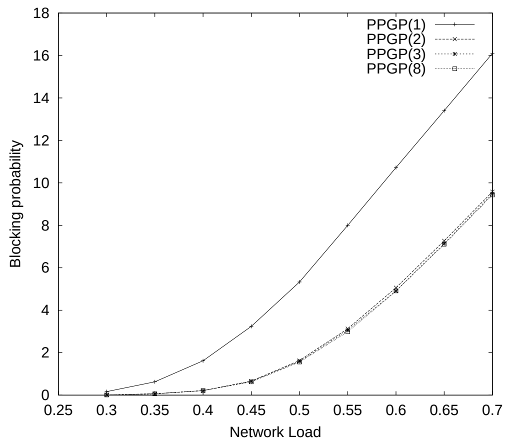


Figure 12: PPGP with different maximum numbers of paths per pair of nodes

Figure 8 shows the performance of PP with different maximum number of paths for each source-destination pair. We use the legend *heuristic*(*maxpath*) to denote the *heuristic* with *maxpath* being the maximum number of paths per pair of nodes. For example, PP(2) means PP with a maximum of 2 paths for each pair of nodes. In this figure, we observe that allowing more than one path per pair of nodes performs much better than only use a path between each pair of nodes. PP(3) results in the best performance. Figure 9 shows the results for PPGT. For PPGT, allowing more paths per pair of nodes noticeably decreases the performance. PPGT(8) performs worse than PPGT(2). This indicates that when trying to select a large number of candidate paths, PPGT starts to select more alternative paths and the overall performance decreases. Figure 10 shows the results for BFS. The trend is similar to that for PPGT. Figures 11 and 12 show the results for GP and PPGP respectively. As discussed previously, GP and PPGP are not able to find many candidates, thus, allowing more paths between two nodes does not affect these two heuristics. When the number of paths between two nodes is larger than 1, the performance of the heuristics are similar. Allowing more paths improves routing performance slightly. The results in this set of experiments show that regardless of the path selection schemes, localized QoS routing needs multiple paths for each source-destination pair to be effective. The results also indicate that the optimum value for the number of paths between two nodes depends on the path selection scheme. When the path selection scheme, such as PP and PPGT, tends to find more paths at the cost of finding longer alternative paths, trying to use a large number of paths between two nodes may not be effective.

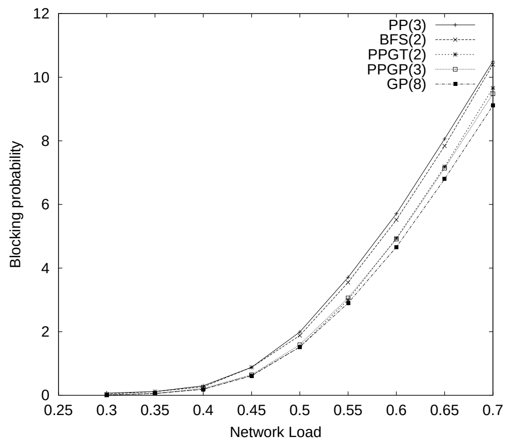


Figure 13: Comparison of the heuristics

Figure 13 shows the comparison of all the five heuristics. We use the best maximum number of paths per pair of nodes parameter for each heuristic in the comparison. The result shows that different path selection heuristics result in difference in the routing performance. GP performs better than PPGP and PPGT which in turn perform better than BFS and PP. The heuristics (GP, PPGP and PPGT) that consider global load balancing perform better than the ones that do not consider global load balancing (PP and BFS) and the one (GP) that emphasizes on global load balancing performs better than the ones that do not emphasize on global load balancing (PPGP and PPGT). One interesting observation of this experiment is that GP perform better than PPGP. Almost all the static measurements would suggest that PPGP should perform better than GP since PPGP has shorter path length, lower average load and lower maximum load. Due to the self-adaptivity of psr, GP performs slightly but consistently better than PPGP.

5 Conclusions

In this paper, we investigate the path selection schemes for localized QoS routing. Five path selection heuristics, namely breadth-first search path selection, per-pair path selection, global path selection, hybrid per-pair/global path selection, and per-pair path selection with global tuning, are proposed and studied. Following are the conclusions we draw from this study.

- Although localized QoS routing has the self-adaptive capability, path selection schemes still have significant impact on its routing performance.
- Regardless of the path selection schemes, allowing multiple paths between two nodes always performs better than allowing only a single path between two nodes in localized QoS routing.
- The optimum value for the maximum number of paths between two nodes depends on the path selection scheme. When the path selection scheme tends to find more paths at the cost of finding longer alternative paths, trying to use a large number of paths between two nodes may not be effective.
- In order for localized QoS routing to be effective, only the paths whose lengths are very close to the minimum-hop should be selected as candidate paths.

- Global load balancing is an important factor in the path selection process.
- Among the five path selection schemes studied, GP, which considers three factors, path length, global load balancing and shared links, with emphasis on global load balancing, achieves the best routing performance.

References

[1] 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*, Edinburgh, United Kingdom, June 7-11, 1999.

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

[3] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, "Introduction to Algorithms", *MIT Press*, 1999.

[4] R. Guerin, S. Kamat, A. Orda, T. Przygienda and D. Williams, "QoS Routing Mechanisms and OSPF Extensions", *Work in Progress*, Internet Draft, March 1997.

[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 International IFIP Workshop on Quality of Service*, 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] Z. Wang and J. Crowcroft, "Quality of Sercive Routing for Supporting Multimedia Applications", *IEEE JSAC*, Sept 1996.

[11] X. Yuan and X. Liu, "Heuristic Algorithms for Multi–Constrained Quality of Service Routing", *IEEE INFOCOM*, April 2001.

[12] Z. Zhang, C. Sanchez, B. Salkewicz, E. Crawley, "Quality of Sercie Extensions to OSPF", *Work in Progress*, Internet Draft, September 1997.

[13] Y. Zhong and X. Yuan, "Impact of Resource Reservation on the Distributed Multi-path Quality of Service Routing Scheme." The *Eighth International Workshop on Quality of Service (IwQoS2000)*, pages 95-104, Pittsburgh, PA, June 2000.