Impact of Resource Reservation on Distributed Multi–path Quality of Service Routing Schemes

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

Abstract

This paper considers the impact of resource reservation on the multi–path Quality–of–Service (QoS) routing schemes that use global network state to make routing decisions. Incorporating resource reservation into multi–path QoS routing algorithms can greatly change the communication characteristics in a network system and affect the performance of the routing algorithms. In this paper, we develop a new QoS routing protocol that combines resource reservation with the ticket–based distributed multi–path QoS routing scheme, evaluate the new routing protocol through extensive simulation, study the impacts of other network components, such as the link state update mechanisms, on the performance of multi-path QoS routing schemes with resource reservation, and compare multi-path routing schemes with other routing protocols including single–path routing schemes and flooding based routing schemes.

1 Introduction

The migration to integrated networks for voice, data and multimedia application introduces new challenges in supporting predictable communication performance. Multimedia applications require the communication to meet stringent requirement on delay, delay–jitter, cost and/or other quality of service (QoS) metrics. To support such applications, the network must be able to provide communication channels with QoS guarantees. Two key issues, *QoS routing*, which identifies paths that meet the QoS requirement and selects the one that leads to high overall resource efficiency, and *resource reservation*, which reserves the resources in the path, must be addressed to support communications with QoS guarantees.

Although QoS routing and resource reservation [15] are two closely related network components, traditionally, these two tasks are separated into two steps. First, a route is selected, then the route is set–up and the resources are reserved along the route. Separating routing and resource reservation simplifies the protocol design. However, in B–ISDN, resource availability may change rapidly and the route information may be outdated. In such environment, a route that was computed in the first step may lack the resources in the second step. Combining the two steps was suggested to overcome this problem [5, 8]. Combining resource reservation and

QoS routing can greatly change the communication characteristics in the network system and affect the performance of the routing algorithms.

Many QoS routing algorithms have been designed to determine the route to establish a connection [2, 5, 6, 7, 9, 10, 12, 13, 14]. Among these algorithms, multi–path QoS routing schemes [3] that use global network information to make routing decisions are promising for future networks for the following reasons. First, by using the global network information, these protocols will incur less messaging overhead compared to the flooding based protocols that do not use the global network information to make routing decisions. Second, by exploring multiple paths simultaneously in search of the path that satisfies the QoS requirement of a connection, multi–path QoS routing algorithms are more effective, in terms of both the blocking probability and the path establishment time, than single–path QoS routing schemes that explore one path at a time.

Resource reservation may, however, affect the performance of the multi–path QoS routing algorithms that use global network state information more than it affects the performance of the single–path QoS routing algorithms or the flooding based algorithms. In multi–path routing, multiple paths are probed simultaneously, which may require reserving resources on multiple paths for each connection request. This problem is called the *over reservation problem*. Furthermore, reserving resources on multiple paths can greatly change the resource availability characteristics in the network system and decrease the precision of the global network state information. Since the algorithms rely on the global network state to make effective routing decisions, the impact of resource reservation on the precision of the global network state information can greatly affect the performance of the algorithms. Thus, incorporating resource reservation in multi–path QoS routing algorithms not only requires the design of new efficient protocols that combine resource reservation and QoS routing, but also requires the re–evaluation of the performance issues in the routing algorithms.

In this paper, we develop a new QoS routing protocol that combines resource reservation and a variation of the ticket–based QoS routing scheme [3], which is a distributed multi–path QoS routing scheme designed to deal with imprecise state information. We evaluate the performance of new protocols through extensive simulation. One unique feature of the ticket–based QoS routing scheme is that the number of paths to be probed in parallel is controlled by the number of tickets generated for each connection request. By manipulating the number of tickets for each request, the ticket–based routing scheme can emulate a wide range of QoS routing schemes including flooding based QoS routing schemes and single–path QoS routing schemes. Thus, introducing resource reservation into the ticket–based QoS routing enables us to study the impact of resource reservation on a wide range of QoS routing algorithms.

Our results show that multi–path QoS routing schemes (with resource reservation) are more effective in finding paths that satisfy the QoS requirement of a connection than single–path QoS routing schemes when the network is under light traffic load. When the network is under heavy traffic, multi–path routing is better than the single–path routing when the resource requirement of each connection is low and the network is not saturated. Single–path QoS routing schemes are more efficient when the network is under heavy load and the resource requirement of each connection is high. Our results also show that the ticket–based QoS routing scheme is much better than the flooding based routing schemes in terms of the scalability. We conclude that it is possible to explore multiple paths in search of the paths that satisfy the QoS requirement without introducing excessive protocol overhead and the ticket–based QoS routing algorithm with resource reservation is an effective QoS routing protocol.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the ticket based QoS routing with resource reservation. Section 4 reports the performance evaluation. Section 5 concludes the paper.

2 Related work

QoS routing has attracted much attention recently. An extensive survey can be found in [4]. Most existing QoS routing schemes decouple the routing issues from the resource reservation issues [2, 3, 6, 7, 10, 13, 14]. Few schemes [5, 8, 9, 12] combine the routing with the resource reservation. However, these routing schemes are either flooding based [5, 9, 12], where the global network information does not affect the routing performance, or single–path routing schemes [8], where the over reservation problem is not severe. This paper studies the impact of resource reservation on the multi–path routing schemes that use global network state information to make routing decisions. Studies on other network components, such as the link state update mechanisms and network topology, are reported in [1, 11]. Resource reservation adds another dimension to the network components and requires most of the performance issues to be revisited. In [3], the authors proposed a ticket–based routing algorithm to solve the delay–constrained least–cost routing without considering resource reservation. We incorporate resource reservation into a variant of this algorithm [3] and study the impact of resource reservation.

3 Ticket–based distributed QoS routing with resource reservation

This section presents the path establishment protocol that incorporates resource reservation into a ticket–based QoS routing scheme. Our algorithm uses a link state algorithm to maintain the global state information. The basic idea of the ticket–based QoS routing is discussed next. A detailed description of the ticket–based routing algorithm for the delay–constrained least–cost routing problem can be found in [3]. We will use the bandwidth constraint as an example to illustrate the protocol. The protocol can be modified to deal with other QoS metrics, such as delay, delay jitter and cost.

3.1 Ticket–based distributed routing algorithm

When a connection request arrives at the source node, a certain number (t) of tickets are generated and a reservation packet with the t tickets are sent to the destination in search of paths that satisfy the QoS constraints. Each reservation packet carries one or more tickets. When an intermediate node receives a reservation packet, it determines the next hops that can potentially establish a connection for the request, calculates the number of tickets to be distributed for each of the next hops, and sends a reservation packet with its share of the tickets to each of the next hops. The detail routing and ticket distribution algorithm will be discussed in the next subsection. Along the path, a reservation packet with more than one ticket may split into multiple reservation packets to explore different paths. Since intermediate nodes only distribute and do not generate any new tickets, the maximum number of reservation packets at any time is bounded by the total number of tickets, t, which was generated at the source node when the connection request arrived. Since each reservation packet searches a path, the maximum number of paths searched is also bounded by t.

The ticket–based QoS routing limits the routing overhead by controlling the number of tickets for each request. It can deal with imprecise state information since it explores multiple paths instead of only the "optimal" path that is computed based on the imprecise global network state information. In addition, the ticket–based QoS routing algorithm can emulate other QoS routing schemes by properly generating the number of tickets for each request. For example, the ticket–based QoS routing algorithm can realize the flooding based QoS routing schemes by generating an infinite number of tickets for each request. It can also realize single–path QoS routing schemes that rely on the global link state information to make routing decisions by generating one ticket for each request. This feature is the major reason that we choose this algorithm as our base algorithm.

3.2 Routing and ticket distribution

In large networks, the global network state information maintained by the link state algorithm is imprecise. Our ticket–based algorithm includes a number of methods when making routing and ticket distribution decisions to cope with the imprecise global state information. On the one hand, the ticket–based algorithm explores multiple paths instead of only the 'optimal' path computed based on the imprecise global state information. This offsets the impact of the imprecise state information. On the other hand, the algorithm attempts to explore better routes with higher probability by distributing more tickets to probe the routes that have better chance of satisfying the QoS constraints.

Routing and ticket distribution is done as follows in our algorithm. First, for each router and each destination, the algorithm uses the global state information to rank the neighbors of the router based on their capability to provide QoS routes to that destination. In the case of a tie, the hop counts of the paths are used to break the tie. If two paths can provide the same bandwidth with the same hop counts, their ranks are assigned randomly. Consider the example in Figure 1. The numbers in the figure represent bandwidth. The ranking of the neighbors of router B to destination F is shown in Table 1. Nodes C and A can both provide paths with bandwidth 4 (from node B to node F), they are ranked higher than nodes D and E. Since the path from nodes B to F needs 3 hops when the path goes through node A and 2 hops when the path goes through node C. Thus, node C is ranked No. 1 and node B is ranked No. 2.

To avoid exploring the paths that have little chance of satisfying the QoS constraints, a cut–off percentage value $cf\%$ is introduced. Assuming that a connection requires a minimum bandwidth of B, the algorithm will only probe the paths that can provide bandwidth $B - B * cf\%$. The value $B - B * cf\%$ will be called the cut–off QoS. Notice that since the state information is imprecise, the paths that can only provide bandwidth $B - B * cf\%$ at the time

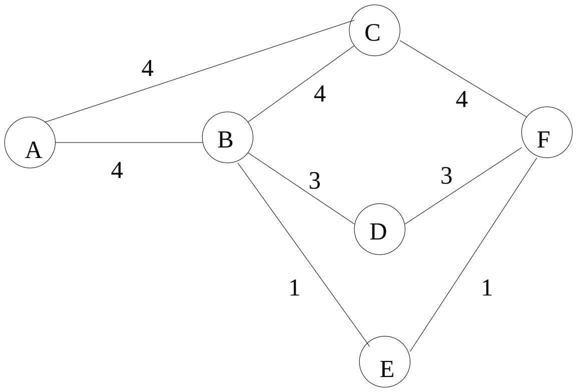


Figure 1: Routing and ticket distribution

neighbor	rank	bandwidth
A	2	4
C	1	4
D	3	3
E	4	1

Table 1: Routing table entry from node B to node F

the global state is computed may have the resources to provide bandwidth B at the time the request is made. The value $cf\%$ is a parameter of the protocol and will be evaluated in the performance study.

Every time a link state packet is received, the ranking information in the routing table is recomputed. This ranking information is used to guide the routing and ticket distribution in our algorithm. Specifically, when an intermediate router receives a reservation packet from its neighbor A, the router determines the next hops that the reservation packet can be forwarded. All neighbors except node A that can provide at least the cut–off QoS are included in the next hop list with their ranks. Ticket distribution is based on the ranking information. How to distribute tickets in order to achieve the best performance is hard to determine. We will evaluate a number of ticket distribution heuristic in the performance study. One simple heuristic is as follows. The first ranked next hop node gets 50% of the tickets, The second ranked next hop node gets 25% of the tickets and so on. The lowest ranked next hop node gets the remaining of the tickets. The rational behind this ticket distribution heuristic is that more tickets will go toward the paths that have better chance of satisfying the QoS constraints. Other tickets will be distributed to explore other not so good paths to allow the algorithm to adapt to the imprecision of the global state information.

Consider the example in Figure 1. When a reservation packet requesting a connection with bandwidth 4 from node A to node F arrives at node B with 4 tickets. Let the cut–off percentage be 50%. Node B will determine that the outgoing nodes include nodes C and D. Node A is not included in the next hop node list since it is where the packet comes from. Node E is not included in the next hop node list since it can only provide bandwidth 1 which is less than the cut–off QoS, $4 - 4 * 50\% = 2$. Thus, the reservation packet will split into two packets toward nodes C and D. The router will then compute the number of tickets for the two packets. Using the simple heuristic discussed earlier, the packet towards node C will get 50% of the tickets since it is ranked No. 1. The other packet towards node D will get the rest of the tickets.

3.3 Incorporating resource reservation in the ticket–based QoS routing scheme

The following control packets are used in the protocol.

- REQ packet. The information in this packet includes the connection identification, number of tickets, the QoS requirement and the current QoS. The connection identification is specified as a triple (src, dst, seq), where src is the source node number, dst is the destination node number and seq is a sequence number assigned by the source node. The current QoS is the QoS that can be satisfied by the partial path, that is, the QoS that can be satisfied from the source node to the router that receives the REQ packet. A REQ packet will be denoted as $REQ((src, dst, seq), ticket, QoS, cQos)$. Here, (src, dst, seq) is the connection identification, $ticket$ is the number of tickets carried by the REQ packet, QoS is the QoS requirement of the connection, and $cQoS$ is the current QoS.
- $REJECT$ packet. The information in this packet includes the connection identification. This packet travels in the opposite direction in the path of the corresponding REQ packet. A REJECT packet will be denoted as $REJECT((src, dst, seq))$.
- $ACCEPT$ packet. This packet contains the connection identification information, the granted QoS and the path for the connection. An ACCEPT packet will be denoted as $ACCEPT((src, dst, seq), QoS, path)$. The routers record the path for the connection in the ACCEPT packet.
- $RELEASE$ packet. This packet contains the connection identification and the path for the connection. This packet will travel through the path and release all the resources for the connection. The RELEASE packet will be denoted as $RELEASE((src, dst, seq), path)$.

Each router maintains precise state information for all outgoing links from the router. Since our algorithm focuses on bandwidth, each router will maintain for each outgoing link the amount of available bandwidth, the amount of used bandwidth, which is the bandwidth that is currently used by some connections, and the amount of reserved bandwidth, which is the bandwidth that is reserved but is not currently used. We will use notation $AVAILABLE$, $USED$ and $LOCKED$ to denote the three states of the bandwidths. The router maintains a routing table. For each node in the network, routing table contains the ranking information of each neighbor and the QoS that can be provided when a connection to the destination goes through that neighbor. The router also records the seq of the latest connection that have been established for each source/destination pair whose connection went through the router. This information is used to deal with the strayed REQ packets. In addition, a router maintains the following information for each connection that passes the router.

- The parent node and the number of REQ packets received from the parent node. The parent node of a connection is where the REQ packet comes from. The REJECT packet and the ACCEPT packet will follow the parent node from the destination to the source. The protocol allows multiple REQ packets to be sent over the same link. However, resources reservation is done once for a connection request on one link.

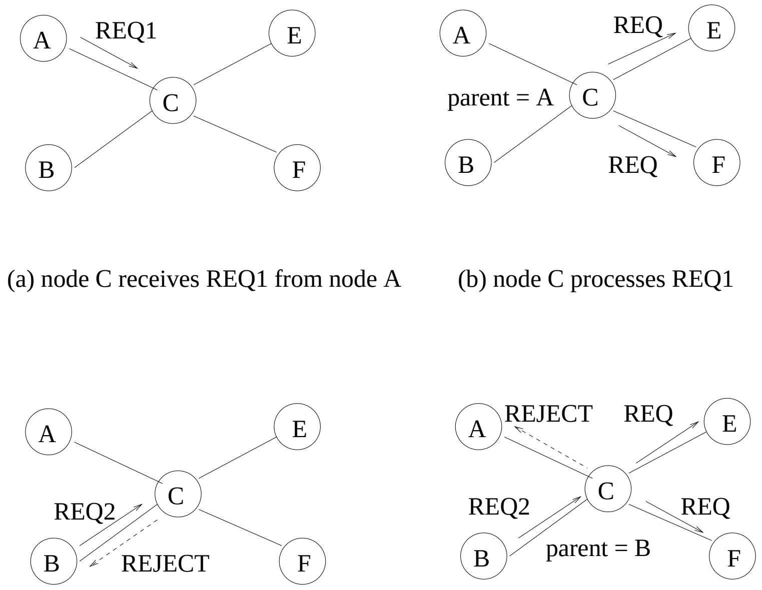
- Current QoS of the connection. This information is used to compare to the current QoS in a REQ packet to determine whether the new REQ packet can provide better QoS.
- The number of QoS packets sent over each of the outgoing links and the associated resource reserved. This information is used to determined whether all REQs from a node or over a link have been rejected. If all REQ packets over a link have been rejected, the resources reserved on the link can be released. If all REQ packets from a node have been rejected, REJECT packets must be sent to the parent.
- The path for each connection that starts from the router.

The operations in the source nodes, the intermediate nodes and the destination nodes are different in the protocol. Next, we will describe the operations for the three different kinds of nodes.

- *Source node.* When the source node wants to establish a connection, it constructs a REQ packet with proper number of tickets and the QoS requirement of the connection. It then sends the REQ packet towards the destination. The source then waits for either the ACCEPT/REJECT packets. If all the REQ packets sent from the source are rejected, the source will start another round, trying to establish a connection for the request. If the source node receives an ACCEPT packet for the request, the connection has been established and the source can start sending data. In this case, the source records the path for the connection from the ACCEPT packet. Once the source node finishes sending the data, a RELEASE packet with the path information for the connection will be sent to release the resources for the connection.
- *Intermediate nodes.* The following operations are performed at intermediate nodes when they receive different control packets. Figure 2 summarizes the operations in the intermediate nodes.
  - REQ packet. When an intermediate router receives a REQ packet, it does the followings. If the connection has been established, the router rejects the request since there is no meaning to reserve resources for an established connection. If resources have not been reserved for the connection (the REQ packet is the first one that reaches the router), the router will check whether the it has sufficient resources to satisfy the connection using the routing and ticket distribution algorithm. The router will distribute the tickets for all the outgoing links that may satisfy the QoS constraints and forward the REQ packets. If none of the outgoing links has sufficient resources for the connection, the router rejects the request. If the router has received a connection request for the connection, that is, the current REQ packet is not the first one to reach the router, the router checks whether the current REQ can provide better QoS for the connection. If the current REQ carries better $cQoS$, the router would then forward the REQ. Otherwise, the router will reject the REQ packet. When a router forwards the second REQ packet, it needs to change the parent node

```
IF (REQ((src, dst, seq), ticket, QoS, cQoS) is received from node A) THEN
   IF ((src, dst, seq) has been established)) THEN
      send REJECT((src, dst, seq)) to node A
   ELSE IF (REQ packets for connection (src, dst, seq) have been processed) THEN
      IF (the new REQ packet has better cQoS) THEN
         send the REQ further
         Update the information of the outgoing link
         let node B be the old parent node
         IF (A ≠ B) THEN
            send REJECT packets to B
            set parent node to be A
         END IF
      ELSE
         send REJECT packet to node A
      END IF
   ELSE IF ((No REQ packets for connection (src, dst,seq) have been processed before) THEN
      Compute next nodes and ticket distribution
      IF (any next nodes exists) THEN
         forward the REQ packets with proper number of tickets reserve the resources in the link.
      END IF
   END IF
ELSE IF (REJECT((src,dst,seq) is received from link L)) THEN
   IF (all REQ packets for connection (src, dst, seq) over link L are rejected) THEN
      release the resources locked for the connection on link L
   ELSE
      decrease the counter of the number of REQ packets sent over L.
   END IF
   IF (all REQ packets sent for the connection from the router are rejected) THEN
      send REJECT packets to the parent
      remove the information about connection (src, dst, seq) from the router.
   END IF
ELSE IF (ACCEPT(src, dst, seq, QoS, path) is received) THEN
   mark the reserved resources USED.
   release additional resources that are locked but will not be used
   forward the packet to the parent node
   record the partial path for the connection in the ACCEPT packet
ELSE IF (RELEASE((src, dst, seq), path) is received) THEN
   release the resources used by (src, dst, seq)
   forward the packet along path
END IF
```

Figure 2: Operations in the intermediate nodes



(a) node C receives REQ1 from node A

(b) node C processes REQ1

(c) node C receives REQ2 from node B. It rejects REQ2 if REQ2 cannot provide better QoS.

(d) node C receives REQ2 from node B and REQ2 can provide better QoS. node C rejects REQ1 from node A and changes the parent for the connection to node B.

Figure 3: Processing REQ packets

for this connection and reject the old REQ if the new parent node is not the same as the old parent node. When the second REQ is sent through a link, the router will just increase the counter of the number of REQ packets sent over the link, but will not reserve extra resources.

The cases when REQ packets are processed in a router are depicted in Figure 3. Figure 3 (a) and (b) show the case when the first REQ packet is processed by node C. Assuming that links (C, E) and (C, F) can provide at least the cut–off QoS. Two REQ packets are sent to nodes E and F respectively with proper number of tickets and the parent for this connection is set to be node A. Figure 3 (c) shows the case when a second REQ packet arrives node C from node B and does not carry better $cQoS$. In this case, the REQ packet is rejected. Figure 3 (d) shows the case when the second REQ packet from node B carries better $cQoS$. In this case, the old REQ is rejected, new REQ packets are sent further and the parent of the connection is set to be node B.

– REJECT packet. When an intermediate node receives a REJECT packet, the intermediate node checks the number of REQ packets sent through that link. When the all REQ packets sent through that link are rejected, the router releases the resources locked for the connection over that link. Otherwise, the router decreases the number of REQ packets sent through the link. When all REQ packets from this router (all REQ packets for all outgoing links) are rejected, REJECT packets are sent to the parent node for this connection in the network. The number of REJECT packets to be sent to the parent node is equal to the number of REQ packets it received from the parent node.

  - ACCEPT packet. When an ACCEPT packet is received, the intermediate node changes the state of the resources from the LOCKED state to the USED state, releases the extra resources reserved but is not granted and forwards the ACCEPT packet to the parent node for this connection. The router also records the path for the connection in the ACCEPT packet.
  - RELEASE packet. The router releases the resources used by the connection and forwards the packet to the next hop according to the information in the *path* field carried in the RELEASE packet.

- Destination node. When the destination node receives a REQ packet, it checks whether the connection has been established. If the connection has been established, the REQ packet is rejected. Otherwise, an ACCEPT packet is constructed is sent to the source from the path in the opposite direction. The destination will not receive REJECT or ACCEPT packets.

3.4 Discussion

Fast and slow algorithm

Like the flooding based algorithm in [5], the ticket based QoS routing algorithm with resource reservation can also have the fast and slow variations. In the fast algorithm, the destination can accept the first REQ packet arrived, while in the slow algorithm, the destination can wait for more time before it accepts a REQ packet in order to find a better route for the connection. We only adopt the fast algorithm option for the following two reasons. First, whether the fast algorithm is more effect or the slow algorithm is more effect is debatable. Although the fast algorithm does not try to find the best path to satisfy the connection, it releases the overly reserved resources quicker. Second, there is a practical problem with the slow algorithm. The destination may not be able to determine how long it has to wait before it accepts a request. One choice is that it waits until all tickets arrive. This scheme is originally proposed in the ticket base algorithm [3] (in this case, even when a reservation fails, a packet carrying the tickets is forwarded to the destination). However, due to the inconsistency of the network state, a loop may be formed in the network and some of the control packets may not be able to reach the destination within a reasonable amount of time.

loops

Due to the imprecise global network information, loops can be formed when the reservation packet tries to reach the destination and a control packet may get strayed and show up later after its connection has been established (and released). Our protocol can deal with both cases. First, The protocol rejects all the REQ packets that involve in a loop. When the second time a REQ packet reaches a node (forming a loop), it will be rejected since it cannot provide better QoS than the first time it reached the node. Second, when a strayed packet shows up after a connection has been release, it cannot reestablish the connection since each router keeps the latest *seq* of each connection. Thus, when the strayed packet reaches the destination, the destination will know that the connection has been established and will reject the request. Notice that the REJECT, ACCEPT and RELEASE packets always follow a route without loops.

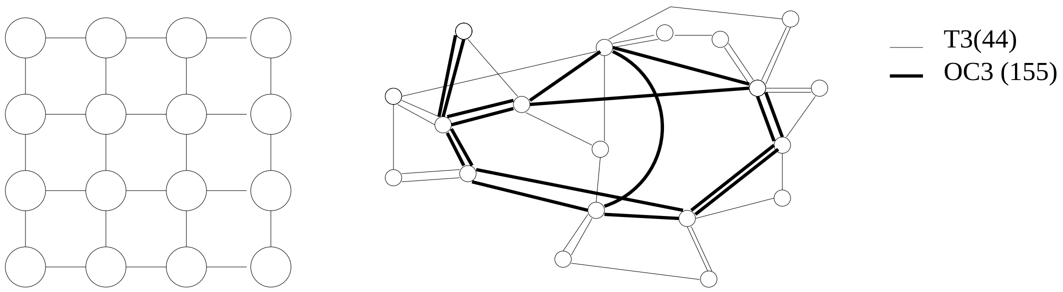
4 Performance evaluation

To evaluate the performance of the protocol and to study the impact of resource reservation on the multi–path QoS routing, we develop a cycle–by–cycle network simulator that simulates the QoS routing protocol discussed in the previous sections. Bandwidth is used as the QoS metric for the QoS routing protocol. The major features of the network simulator include the followings.

- The simulator simulates the new QoS routing protocol that combines resource reservation with the ticket–based QoS routing scheme. By manipulating the number of tickets for each request, the simulator can simulate the flooding based routing schemes that do not use the global network state information to make routing decisions, multi–path QoS routing schemes which probe different number of paths for each connection request based on the global network state information, and single–path QoS routing schemes that probe one path at a time.
- The control network is logically separated from the data network. The control network is used to exchange the control messages and the link state distribution messages while the data network is used to exchange data messages.
- The simulator simulates various link state update mechanisms including the periodical link state update mechanism and the sudden–change link state update mechanism [1] which broadcasts link state information when the state of a link changes drastically.
- The simulator parameterizes many network components. For example, the network load is specified using three parameters, the connection request generation rate, the connection duration and the bandwidth requirement of each connection. By manipulating the parameters for each network component, we are able to study the impact of each network component on the performance of the protocol.

The system parameters used in the simulations are listed below.

- *Network size.* This specifies the size of the network. The simulations are performed on 4×4, 6×6 and 8×8 and 10×10 meshes and the MCI backbone topology. Figure 4 shows a 4×4 mesh and the MCI backbone topology. The bandwidth in each link of the meshes is assumed to be $100Mbps$.
- *Connection request generation rate, r.* The connection request generation rate, r, specifies the traffic in the network. The connection requests at each node are assumed to have a Poisson inter-arrival distribution with an average of r messages generated every time



(a) mesh

(b) MCI backbone

Figure 4: The network topologies used in the experiment

unit. When a request is generated at a node, the destination of the request is generated randomly among the other nodes in the system. A generated request is discarded when the request is blocked.

- *Connection duration, d.* The connection duration specifies the time duration of the connection after the connection is established.
- *Link state information update frequency, uf.* This frequency determines the accuracy of the global network state information in each router. The higher the frequency, the more accurate the global network state. Higher frequency also results in higher processing requirement in each router and higher traffic load in the control network.
- *Cutoff value, cf.* This value determines the requirement (considered by the protocol) of a path to satisfy the QoS constraints.
- *Number of the outstanding requests, o.* This number indicates the number of requests that each router can attempt to establish the connection simultaneously.

We use two performance metrics, the blocking probability and the average number of connections granted, to evaluate the protocols. The blocking probability is the ratio of the number of connection requests that are rejected (blocked) and the total number of connection requests processed. Under light traffic, both the blocking probability and the number of connection requests granted are good indicators of the performance of the protocols. Under heavy traffic, the number of connection requests granted is the better indicator of the performance. The simulation time is measured in time cycles, which is the basic unit of the network activities. All other network activities, such as the packet processing and packet propagation over a link, take a multiple of the time cycle to complete. In the simulations, we assume that the propagation delay in each link 1 cycle and the packet processing time in each router is also 1 cycle.

Figure 5 shows the blocking probability as a function of the number of tickets for each connection. This experiment is done on a 6×6 mesh with the following assumptions. $uf = 500$, $c = 0$, $o = 1$. Four cases are considered in the figure. The first case is when the connection

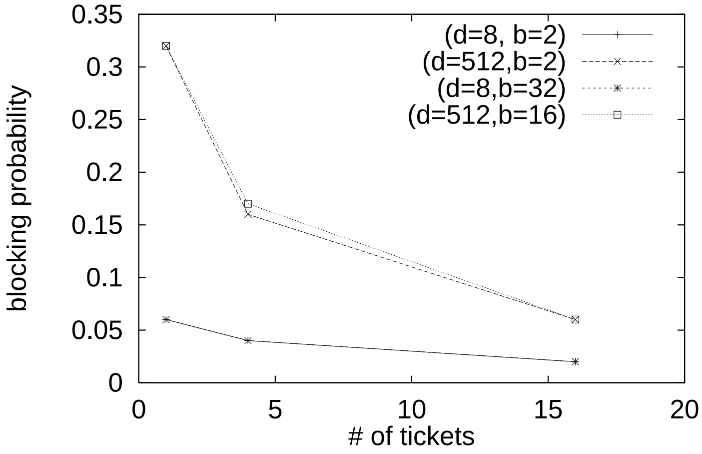
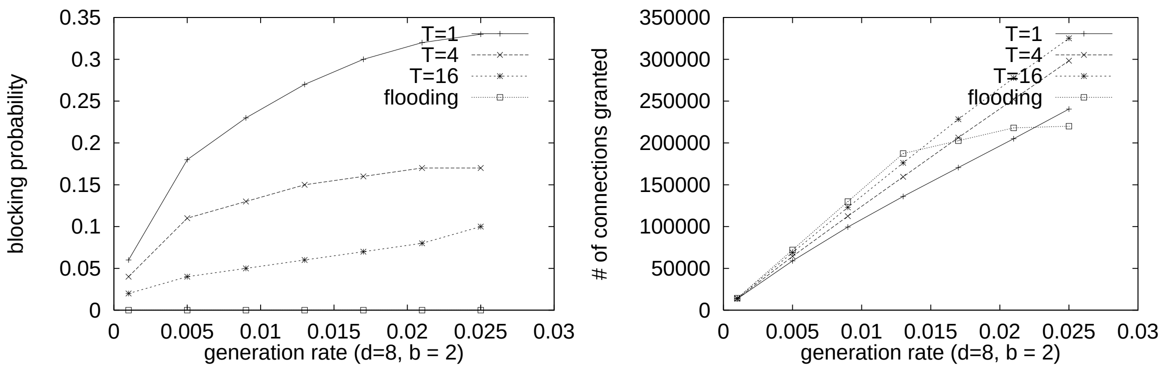


Figure 5: The blocking probability for light traffic

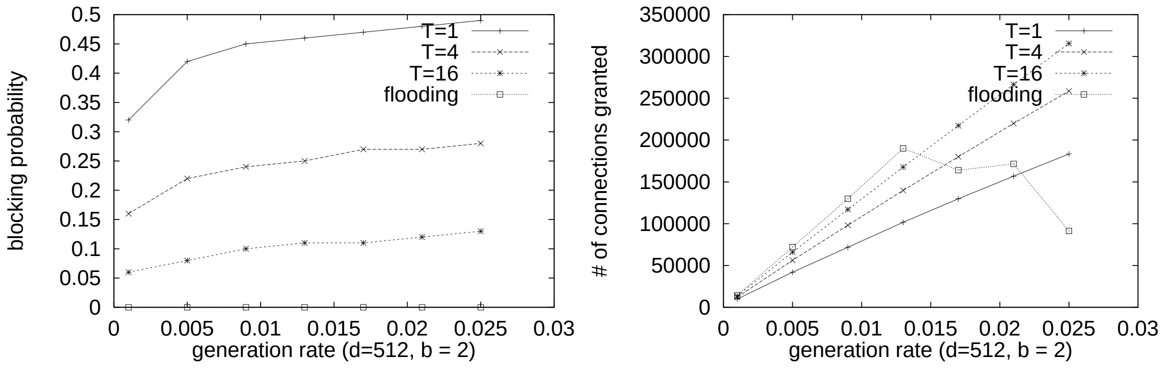
duration is small and the bandwidth requirement for each request is low ($d = 8, b = 2$). The second case is when the connection duration is small and the bandwidth requirement is high ($d = 8, b = 32$). The third case is when the connection duration is large and the bandwidth requirement is small ($d = 512, b = 2$) and the last case is when both the connection duration and the bandwidth requirement are high ($d = 512, b = 16$). These four cases represent different traffic patterns when the network is under light load. As shown in Figure 5, the blocking probability decreases as the number of tickets increases, which indicates that under light network load, the protocols that can probe more multiple paths perform better than the protocols that probe less number of paths. This result is consistent for all the four cases we studied in this experiment. The figure also shows that under light load, the connection duration affects the blocking probability while the bandwidth requirement has little impact. When the network load is low, the over reservation problem does not really affect the performance of the protocols since the network always has the resources to support the connection. Thus, the protocols that probe more paths have the higher probability to establish a connection.

Figures 6, 7, 8, 9 show the performance of the protocols when the network load is higher. As in the previous experiment, the same four cases are studied. Figure 6 shows that case when connection duration is small ($d = 8$) and the bandwidth requirement of each request is small ($b = 2$). Figure 7 shows that case when connection duration is large ($d = 512$) and the bandwidth requirement of each request is small ($b = 2$). Figure 8 shows that case when connection duration is small ($d = 8$) and the bandwidth requirement of each request is large ($b = 32$). Figure 6 shows that case when connection duration is large ($d = 512$) and the bandwidth requirement of each request is large ($b = 16$). The number of connections granted is obtained by counting the number of connections that are granted during a period of 200000 cycles when the network traffic is under stable state. Figure 6 and Figure 7 have the similar trend, which indicates that connection duration does not affect the relative performance of the protocols. Exploring more paths simultaneously can still achieve better performance till the number of paths becomes very big. In these two figures, the performance degradation only occurs for the flooding based scheme. This result holds when the bandwidth requirement ($b = 2$) is small and the over reservation problem is not very severe. Figure 8 and Figure 9



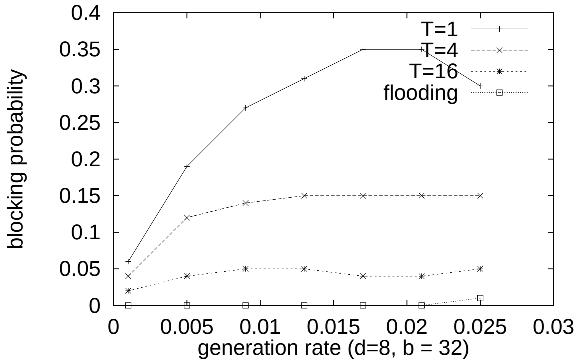
(a) Blocking probability (b) Number of connections granted

Figure 6: Performance of the protocols for $(d = 8, b = 2)$

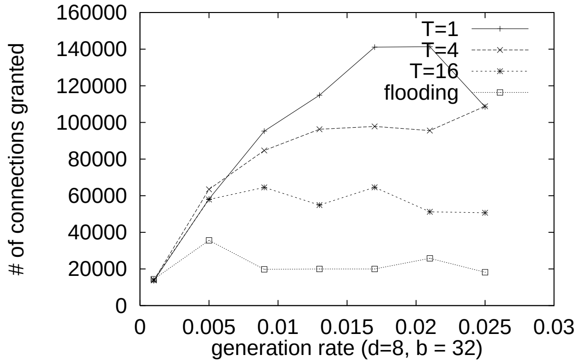


(a) Blocking probability (b) Number of connections granted

Figure 7: Performance of the protocols for $(d = 512, b = 2)$

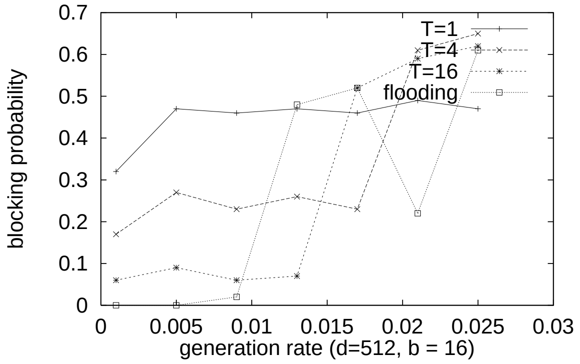


(a) Blocking probability

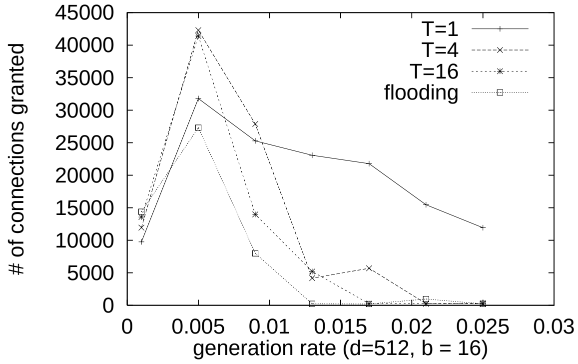


(b) Number of connections granted

Figure 8: Performance of the protocols for $(d = 8, b = 32)$



(a) Blocking probability

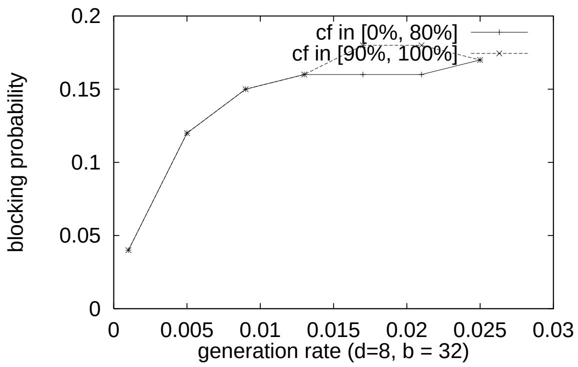


(b) Number of connections granted

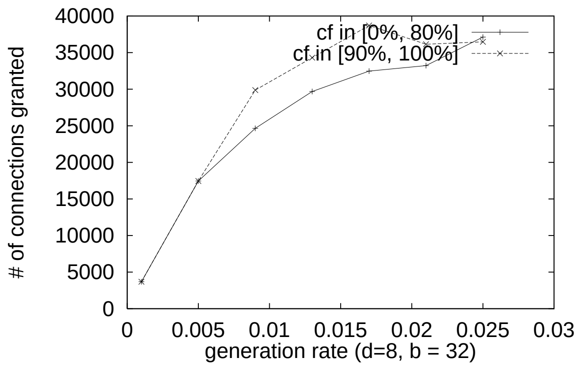
Figure 9: Performance of the protocols for $(d = 512, b = 16)$

show the case when the bandwidth requirement is high ($b = 32$). Note here that each link in the network can support $100Mbps$. In the case when the bandwidth requirement is high, the multi–path QoS routing performs better than the single–path QoS routing schemes in terms of the blocking probability. However, in terms of the number of connections granted, which is directly related the network system throughput, multi–path routing performs much worse. The reason for this is that when the bandwidth requirement is high, the over reservation problem becomes very severe and the performance of the multi–path routing degrades as the number of paths to be probed simultaneously increases (which also increase the amount of resources reserved for each connection).

In summary, multi–path QoS routing schemes with resource reservation perform better than single–path QoS routing schemes when the network load is lower regardless of the connection duration and bandwidth requirement. When the network is under high load, the protocols behave different according to the reasons that cause the network load to be high. There are three reasons that can cause the network to be under high load, (1) high request generation rate, (2) long connection duration and (3) large bandwidth requirement. Among the three reasons, the performance of multi–path QoS routing is more sensitive to the bandwidth requirement and less sensitive to the request generation rate and the connection duration. In the case when the bandwidth requirement is large, although multi–path QoS routing can still have lower blocking probability when the network is not saturated, the path establishment time is significantly larger and the throughput of the system degrades. When the network is saturated, the multi–path routing has higher blocking probability and lower throughput as shown in Figure 9.



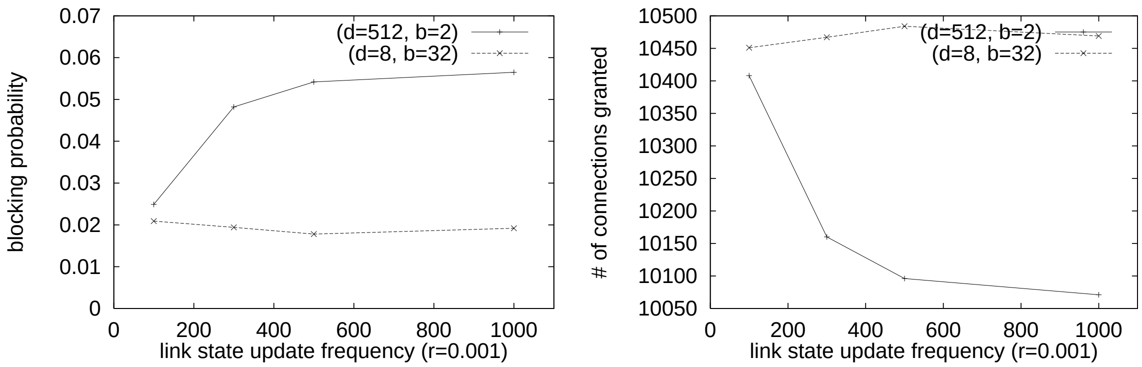
(a) Blocking probability



(b) Number of connections granted

Figure 10: Impact of the cutoff value

We studied the impact of the cut–off value on the performance of the protocols. The cutoff value determines the kind of paths that are considered by the protocol as the potential paths to establish the QoS connection. The smaller the cutoff value, the higher the requirement. Our experiment results, while not shown in a figure, show that when the bandwidth requirement is low, the cut–off value has little impact on the performance. Figure 10 shows the impact of the cutoff value on the performance of the protocols when the bandwidth requirement is high. This figure is done for a 6×6 mesh and assumes $d = 8$, $b = 32$, $t = 4$ and $ut = 500$. The figure shows that when the network is under low load, the cutoff value has little impact on the protocols even when the bandwidth requirement is high. When the network is under heavier load and not saturated, using very large cutoff value (close to 0.9 to 1), which means that considering almost all paths as candidates for a connection request achieves higher throughput. The reason for this is that when the bandwidth requirement is high, the resource reservation renders the global network state to be very imprecise. As a result, making routing decisions based on the global network state does not give better performance than not making such decisions. When the network is saturated, the performance of the protocols with different cut–off value is similar since the performance of the protocols is mainly determined by the amount of resource reserved for each connection and will not be affected significantly by the cut–off value.

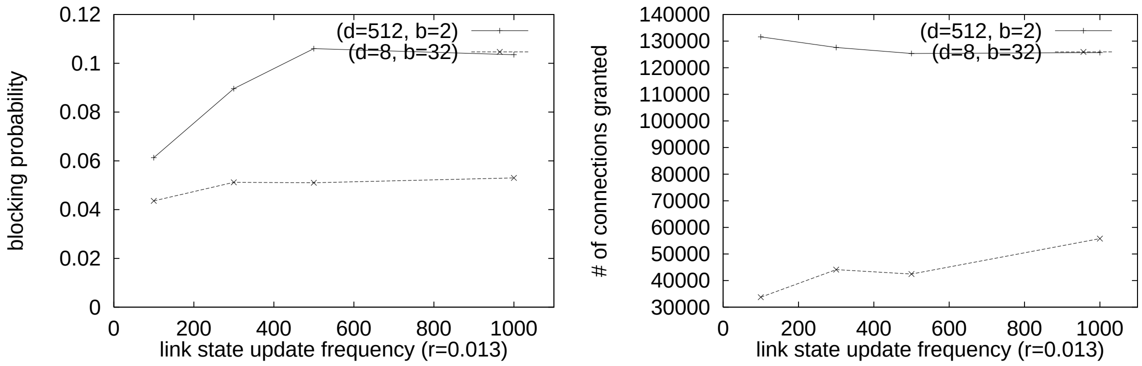


(a) Blocking probability

(b) Number of connections granted

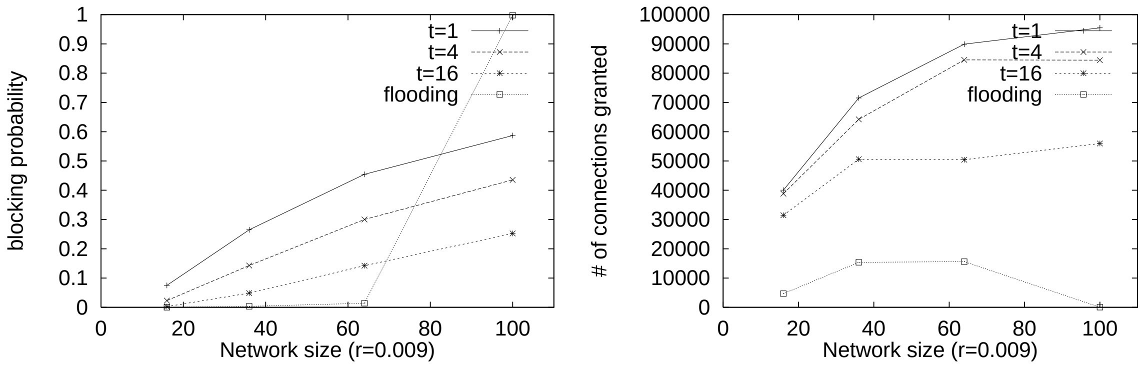
Figure 11: Impact of link state update frequency (low load)

Figures 11 and 12 show the impact of link state update frequency on the multi–path QoS routing algorithm. This experiment is done on 6×6 meshes and assumes $t = 16$. The results show the two contradictory effects of the link state update frequency. For long duration and low bandwidth requirement communications ($d = 512, b = 2$), the global link state maintained by the link state algorithm is fairly accurate and the performance of the protocols improves (in terms of both the blocking probability and the number of connections granted) when the link state update frequency increases as shown in both figures. However, for short duration and large bandwidth requirement ($d = 8, b = 32$) communications, the performance of the protocols decreases as the link state update frequency increases. The reason is that when the bandwidth requirement is high, the global state information maintained by the link state algorithm is always inaccurate even when the link state is updated frequently ($uf = 100$). As a result, the performance gained by using more accurate global link state information is limited and the control overhead incurred by using higher link state update frequency decreases the overall performance of the protocol.



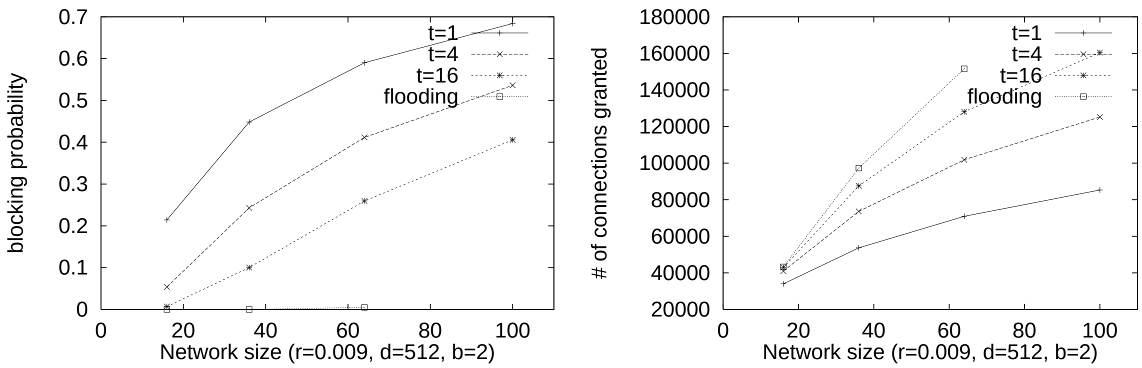
(a) Blocking probability (b) Number of connections granted

Figure 12: Impact of link state update frequency (high load)



(a) Blocking probability (b) Number of connections granted

Figure 13: Impact of network size (large bandwidth requirement)



(a) Blocking probability

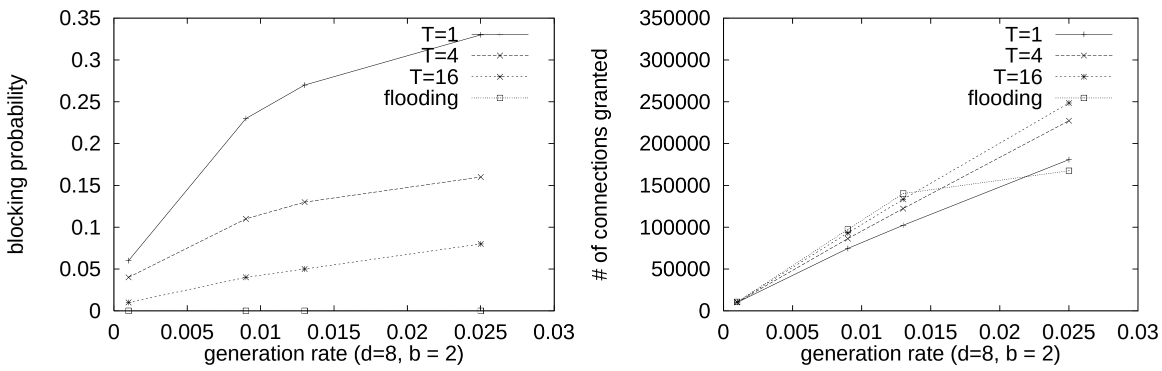
(b) Number of connections granted

Figure 14: Impact of network size (large duration)

Figures 13 and 14 show the impact of network size on the protocols. Two cases are depicted in the figures. The first case, shown in Figure 13, is for short connection duration ($d = 8$) and large bandwidth requirement ($b = 32$). The second case, shown in Figure 14, is for long connection duration ($d = 512$) and small bandwidth requirement ($b = 2$). The network size amplifies the performance differences, but does not affect the relative performance of the four schemes studied, single–path routing, multi–path routing ($t = 4$ and $t = 16$) and flooding. Figure 13 (b) also shows that the flooding scheme is not scalable with respect to the network size while the multi–path routing schemes ($t = 4$ and $t = 16$) scale much better.

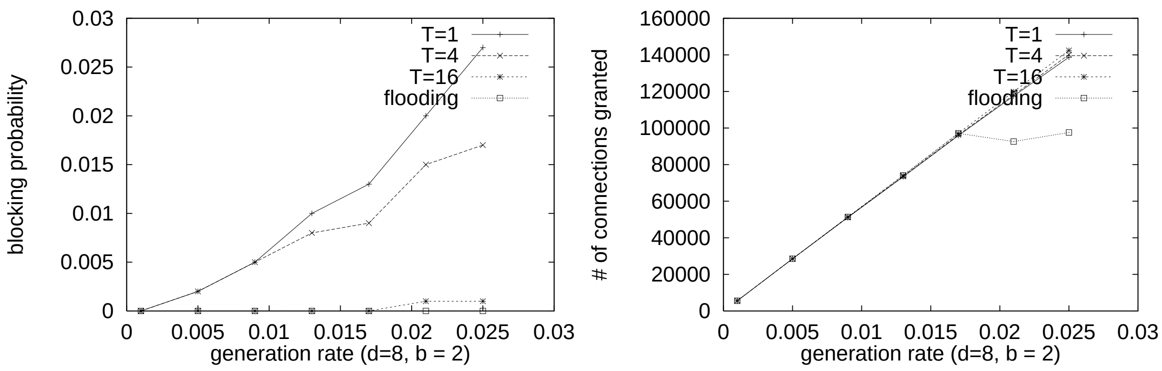
To determine the effect of the routing and ticket distribution methods, the following modification is made to the simulator. At an intermediate node, after the next hop candidates have been chosen, tickets are distributed equally among the candidates. Using this new scheme, ticket distribution does not bias toward the better routes computed based on the imprecise global state information. Figure 15 shows the performance of the new ticket distribution scheme. All other settings are the same as the settings used in Figure 6. Comparing Figure 15 with Figure 6, we have the following observations. First, both figures show the same trend in the blocking probability and the number of connections granted. Second, in terms of the number of connections granted, equally distributing tickets is much worse than the old ticket distribution method, which distributes more tickets to explore better paths. This result indicates that making routing decisions using the imprecise global state information can still be effective, at least for some cases.

Figures 16, 17 and 18 show the simulation results for the MCI backbone topology. The settings for these experiments are similar to the settings for Figures 6, 7 and 8, respectively. The trend shown in the performance results for the MCI backbone is similar to the trend for the mesh topology, except that the impact of the multi–path routing (different number of tickets for each connection request) is different. Since the MCI network is a smaller network than a 6×6 mesh, the performance of flooding on the MCI network is similar to the performance of



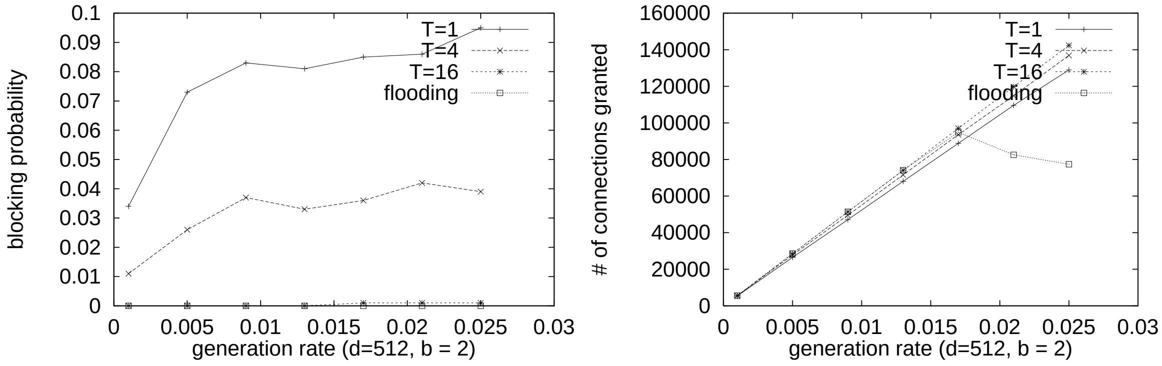
(a) Blocking probability (b) Number of connections granted

Figure 15: Performance of a ticket distribution variation



(a) Blocking probability (b) Number of connections granted

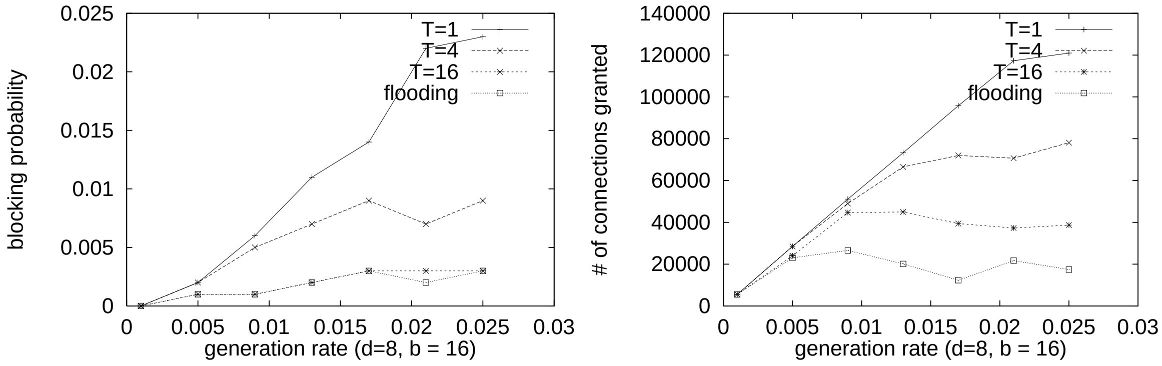
Figure 16: MCI backbone, short connection duration and small bandwidth requirement



(a) Blocking probability

(b) Number of connections granted

Figure 17: MCI backbone, long connection duration and small bandwidth requirement



(a) Blocking probability

(b) Number of connections granted

Figure 18: MCI backbone, short connection duration and large bandwidth requirement

the 16 tickets multi–path routing scheme. Other conclusions we drawn for the mesh topology also apply to the MCI network. For example, in MCI network, the single–path routing scheme is more efficient when the bandwidth requirement for each connection is large as shown in Figure 17.

5 Conclusion

In this paper, we develop a new QoS routing protocol that combines resource reservation with the ticket–based distributed multi–path QoS routing scheme, evaluate the new routing protocol through extensive simulation, study the impacts of other network components, such as the link state update mechanisms, on the performance of the multi-path QoS routing with resource reservation and compare multi-path routing schemes with other routing protocols including single–path routing schemes and flooding based routing schemes.

Our results show that multi–path QoS routing schemes are more effective in finding paths that satisfy the QoS constraints than single–path routing schemes when the network is under light load. When the network is under heavy load, single–path routing schemes can achieve better performance, especially when each resource reservation reserves a large amount of resources (e.g. bandwidth). Our results also indicate that multi–path QoS routing with resource reservation exhibits much better scalability than the flooding based routing schemes when the number of paths to be probed for each request is limited. We conclude that the ticket–based multi–path QoS routing scheme with resource reservation is an effective QoS routing scheme.

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 (ITC'16)*, Edinburgh, United Kingdom, June 7-11, 1999.

[2] S. Chen and K. Nahrstedt, "On Finding Multi–Constrained Paths", *IEEE International Conference on Communications*, June 1998.

[3] S. Chen and K. Nahrstedt, "Distributed QoS routing with Imprecise State Information." *ICCCN'98*, 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 Network*,....

[5] I. Cidon, R. Rom and Y. Shavitt, "Multi–Path Routing Combined with Resource Reservation", In *IEEE INFORMCOM'97*, 1997.

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

[7] Q. Ma and P. Steenkiste, "Quality–of–Service Routing with Performance Guarantees." *4th International IFIP Workshop on Quality of Service*, May 1997.

[8] *ATM Forum, Private network network interface (PNNI) v1.0 specifications*, June 1996.

[9] H.K. Pung, J. Song and L. Jacob, "Fast and Efficient Flooding Based QoS Routing Algorithm", *IEEE International Conference on Computer Communications and Networks (IC3N'99)*, October, 1999.

[10] H.F. Salama, D. S. Reeves and Y. Viniotis, "A Distributed Algorithm for Delay–Constrained Unicast Routing." *IEEE INFOCOM'97*, April 1997.

[11] A. Shaikh, J. Rexford and K. Shin, "Dynamics of Quality–of–Service Routing with Inaccurate Link–state Information." *University of Michigan Technical Report CSE–TR–350–97*, Nov. 1997.

[12] K. G. Shin and C.C. Chou, "A Distributed QoS Routing Using Bounded Flooding", *University of Michigan CSE technical report*, CSE–TR–388–99.

[13] Z. Wang and J. Crowcroft, "QoS Routing for Supporting Resource Reservation." *IEEE Journal on Selected Areas in Communications*, September, 1996.

[14] X. Yuan, "On the Extended Bellman–Ford Algorithms to Solver Two–Constraints Quality–of–Service Routing Problems", *IEEE IC3N*, 1999.

[15] L. Zhang, S. Deering, D. Estrin, S. Shenker and D. Zappala, "RSVP, A New Resource ReSerVation Protocol", *IEEE Network*, September 1993.