Evaluating the Overheads of Source-Directed Quality-of-Service Routing

Anees Shaikh[†] Jennifer Rexford[‡] Kang G. Shin[†]

[†] Department of Electrical Engineering
and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122
{ashaikh,kgshin}@eecs.umich.edu

[‡] Network Mathematics Research
Networking and Distributed Systems
AT&T Labs – Research
Florham Park, NJ 07932-0971
jrex@research.att.com

Abstract

Quality-of-service (QoS) routing satisfies application performance requirements and optimizes network resource usage but effective path-selection schemes require the distribution of link-state information, which can impose a significant burden on the bandwidth and processing resources in the network. We investigate the fundamental trade-off between network overheads and the quality of routing decisions in the context of the source-directed link-state routing protocols proposed for future IP and ATM networks. Through extensive simulation experiments with several representative network topologies and traffic patterns, we uncover the effects of stale link-state information, random fluctuations in traffic load, and variations of the link-cost metric on the routing and signalling overheads. The paper concludes by summarizing our key results as a list of guidelines for designing efficient quality-of-service routing policies in large backbone networks.

1 Introduction

The migration to integrated networks for voice, data, and multimedia applications introduces new challenges in supporting predictable communication performance. To accommodate diverse traffic characteristics and quality-of-service (QoS) requirements, these emerging networks can employ a variety of mechanisms to control access to shared link, buffer, and processing resources. These mechanisms include traffic shaping and flow control to regulate an individual traffic stream, as well as link scheduling and buffer management to coordinate resource sharing at the packet or cell level. Complementing these lower-level mechanisms, routing and signalling protocols control network dynamics by directing traffic at the flow or connection level. QoS routing selects a path for each flow or connection to satisfy diverse performance requirements and optimize resource usage [6, 16, 28]. However, to support high throughput and low delay in establishing connections in large networks, the path-selection scheme should not consume excessive bandwidth, memory, and processing resources.

In this paper, we investigate the trade-off between these resource requirements and the quality of the routing decisions. We focus on link-state routing algorithms where the source switch or router selects a path based on the connection traffic parameters and the available resources in the network. For example, the ATM Forum's PNNI standard [22] defines a routing protocol for distributing topology and load information throughout the network, and a signalling protocol for processing and forwarding connection-establishment requests from the source. Similarly, proposed QoS extensions to the OSPF protocol include an "explicit routing" mechanism for source-directed IP routing [11, 30]. During periods of transient overload, link failure, or general congestion, these schemes are able to find QoS paths for more flows. However, QoS routing protocols can impose a significant bandwidth and processing load on the network, since each switch must maintain its own view of the available link resources, distribute link-state information to other switches, and compute and establish routes for new connections. To improve scalability in large networks, switches and links can be assigned to smaller peer groups or areas that exchange detailed link-state information.

Despite the apparent complexity of QoS routing, these path-selection and admission control frameworks offer network designers a considerable amount of latitude in limiting overheads. In particular, the network can control the complexity of the routing algorithm itself, as well as the frequency of route computation and link-state update messages. Link-state information can be propagated in a periodic fashion or in response to a significant change in the link-state metric (e.g., utilization). For example, a link may advertise its available bandwidth metric whenever it changes by more than 10% since the previous update message; triggering an update based on a change in available capacity ensures that the network has progressively more accurate information as the link becomes congested. In addition, a minimum time between update messages would typically be imposed to avoid overloading the network bandwidth and processing resources during rapid fluctuations in link bandwidth. However, large periods and coarse triggers result in stale link-state information, which can cause a switch to select a suboptimal route or a route that cannot accommodate the new connection. Hence, tuning the frequency of link-state update messages requires a careful understanding of the trade-off between network overheads and the accuracy of routing decisions.

Several recent studies consider the effects of stale or coarse-grained information on the performance of QoS routing algorithms. For example, analytical models have been developed to evaluate routing in hierarchical networks where a switch has limited information about the *aggregate* resources available in other

peer groups or areas [12]. To characterize the effects of stale information, comparisons of different QoS-routing algorithms have included simulation experiments that vary the link-state update period [2, 17, 18], while other work considers a combination of periodic and triggered updates [21]. However, these studies have not included a detailed evaluation of how the update policies interact with the traffic parameters and the richness of the underlying network topology. Finally, new routing algorithms have been proposed that reduce computation and memory overheads by basing path selection on a small set of discrete bandwidth levels [11, 17]; these algorithms attempt to balance the trade-off between accuracy and computational complexity.

The performance and implementation trade-offs for QoS routing depend on the interaction between a large and complex set of parameters. For example, the underlying network topology not only dictates the number of candidate paths between each pair of nodes or switches, but also affects the overheads for computing routes and distributing link-state information. The effects of inaccurate link-state information depend on the amount of bandwidth requested by new connections. Similarly, the frequency of link-state updates should relate to connection interarrival and holding times. Routing and signalling overheads, coupled with the presence of short-lived connectionless traffic, limit the proportion of traffic that can be assigned to QoS routes; this, in turn, affects the interarrival and holding-time distributions of the QoS-routed connections. Although a lower link-state update rate reduces network and processing requirements, stale load information incurs set-up failures, which may require additional resources for computing and signalling an alternate route for the connection. In addition, controlling overhead in large networks may require strict limits on the frequency of link-state updates and route computation, even though inaccurate information may make it very difficult to successfully reserve resources on long routes.

In this paper, we investigate these performance issues through a systematic study of the scaling characteristics of QoS routing in large backbone networks. In contrast to recent simulation studies that compare different routing algorithms under specific network configurations [2, 9, 10, 14, 17–19, 21, 23, 24], we focus on understanding how routing performance and implementation overheads grow as a function of the network topology, traffic patterns, and link-state update policies. In Section 2, we construct a detailed model of QoS routing that parameterizes the path-selection algorithm, link-cost function, and link-state update policy, based on the PNNI standard and proposed QoS extensions to OSPF, as well as the results of previous performance studies. It should be emphasized that our study focuses on the interaction between link-state staleness and the cost-performance trade-offs of QoS-routing protocols. We consider a mixture of representative topologies, and operating regimes where connection durations and the time between link-state updates are large relative to propagation and signalling delays. Our model permits a realistic evaluation of large backbone networks and the routing of the longer-lived traffic flows that are likely to employ QoS routing.

Since the complexity of the routing model precludes a closed-form analytic expression, we present a simulation-based study that uncovers the effects of stale link-state information on network dynamics. To efficiently evaluate a diverse collection of network configurations, we have developed a connection-level event-driven simulator that limits the computational overheads of evaluating the routing algorithm in large networks with stale information. Based on this simulation model, Section 3 examines the effects of periodic and triggered link-state updates on the performance and overheads of QoS routing. The experiments evaluate several topologies to explore the impact of inaccurate information on how well a richly-connected network can exploit the presence of multiple short routes between each pair of switches. Section 4 studies the impact of stale load information on the choice of link metrics for selecting minimum-cost routes for new connections. The experiments suggest guidelines for tuning link-state update policies and link-cost metrics for efficient QoS routing in high-speed networks. Section 5 concludes the paper with a list of guidelines for designing efficient quality-of-service routing policies in large backbone networks.

2 Routing and Signalling Model

Our study evaluates a parameterized model of QoS routing, where routes depend on connection throughput requirements and the available bandwidth in the network. When a new connection arrives, the source switch computes a minimum-hop path that can support the throughput requirement, using the sum of link costs to choose among feasible paths of equal length. To provide every switch with a recent view of network load, link information is distributed in a periodic fashion or in response to a significant change in the available capacity.

2.1 Route Computation

Since predictable communication performance relies on having some sort of throughput guarantee, our routing model views bandwidth as the primary traffic metric for defining both application QoS and network resources. Although application requirements and network load may be characterized by several other dynamic parameters, including delay and loss, initial deployments of QoS routing are likely to focus simply on bandwidth to reduce algorithmic complexity. Hence, our model expresses a connection's performance requirements with a single parameter b that represents either a peak, average, or effective bandwidth, depending on the admission control policy. In practice, the end-host application may explicitly signal its required bandwidth, or network routers can detect a flow of related packets and originate a signalling request. Each link i has reserved (or utilized) bandwidth u_i that cannot be allocated to new connections. Consequently, a switch's link-state database stores (possibly stale) information u_i' about the utilization of each link i in order to compute suitable routes for new connections. Each link also has a cost c_i (c_i') that is a function of the utilization u_i (u_i'), as discussed in Section 2.2.

Although networks can employ a wide variety of QoS routing strategies, previous comparative studies have demonstrated that algorithms with a strong preference for minimum-hop routes almost always outperform algorithms that do not consider path length [1, 9, 10, 18, 19, 23]. For example, selecting the widest shortest path (i.e., the minimum-hop route with the maximum value of $\min_i\{1 - u_i\}$) increases the likelihood of successfully routing the new connection. Similarly, the network could select the minimum-hop path with the smallest total load (minimum value of $\sum_i u_i$) to balance network utilization. In contrast, non-minimal

routing algorithms, such as shortest widest path, often select circuitous routes that consume additional network resources at the expense of future connections, which may be unable to locate a feasible route. Biasing toward shortest-path routes is particularly attractive in a large, distributed network, since path length is a relatively stable metric, compared with dynamic measurements of link delay or loss rate [10].

In our model, the source selects a route based on the bandwidth requirement b and the destination node in three steps: (i) (Optionally) prune infeasible links (i.e., links i with $u_i' + b > 1$), (ii) compute shortest paths to the destination based on hop-count, and (iii) extract a route with the minimum total cost $\sum_i c_i'$.

This process effectively computes a "cheapest-shortest-feasible," or a "cheapest-shortest" path, depending on whether or not the pruning step is enabled. By pruning any infeasible links (subject to stale information), the source performs a preliminary form of admission control to avoid selecting a route that cannot support the new connection. In an N-node network with L links, pruning has $O(L)$ computational complexity and produces a sparser graph consisting entirely of feasible links. Then, the switch can employ the Dijkstra shortest-path tree algorithm [5] to compute a the shortest path with the smallest total cost [25]. The Dijkstra shortest-path calculation has $O(L \log N)$ complexity when implemented with a binary heap. Although advanced data structures can reduce the average and worst-case complexity [3], the shortest-path computation still incurs significant overhead in large networks. Extracting the route introduces complexity in proportion to the path length.

2.2 Link-Cost Metrics

The routing algorithm uses link cost metrics $\{c_i\}$ to distinguish between paths of the same length. Previous studies suggest several possible forms for the path metric, including sum of link utilizations, maximum link utilization on the path, or sum of the link delays. For a general model of link cost, we employ a function that grows exponentially in the link utilization ($c_i \propto u_i^\alpha$), where the exponent α controls how expensive heavily-loaded links look relative to lightly-loaded links. An exponent of $\alpha = 0$ reduces to load-independent routing, whereas large values of α favor the widest shortest paths (selecting the shortest-path route that maximizes the available bandwidth on the bottleneck link). We define a parameter $u_{\min}$ to be the minimum-cost utilization level; any link utilization below $u_{\min}$ is considered to have the minimum cost. Setting $u_{\min} = 0.5$, for example, results in a routing policy in which all links with less than 50% utilization look the same with regard to cost.

We represent link cost with C discrete values. Small values of C limit the computational and storage requirements of the shortest-path computation. However, coarse-grain link-cost information can degrade performance by limiting the routing algorithm's ability to distinguish between links with different available resources, though the presence of multiple minimum-cost routes provides efficient opportunities to balance load through alternate routing.

2.3 Connection Signalling

When a new connection request arrives, the source switch applies the three-step routing algorithm to select a suitable path. However, the optional step of pruning the (seemingly) infeasible links may actually disconnect the source and the destination, particularly when the network is heavily-loaded. When a feasible route cannot be computed, the source rejects the connection without trying to signal it through the network. Stale link-state information may contribute to these *routing failures*, since the source may incorrectly prune a link that could actually support the new connection (i.e., the link has $u_i + b \leq 1$, although the source determines that $u_i' + b > 1$). Routing failures do not occur when pruning is disabled. In the absence of a routing failure, the source initiates hop-by-hop signalling to reserve bandwidth b on each link in the route. As the signalling message traverses the selected path, each switch performs an admission test to check that the link can actually support the connection. If the link has sufficient resources, the switch reserves bandwidth on behalf of the new connection (i.e., $u_i = u_i + b$) before forwarding the set-up message to the next link in the route.

Once the bandwidth resources are reserved on each link in the route, the network admits the connection, committing bandwidth b on each link in the path for the duration of the call. However, a *set-up failure* occurs if a link does not have enough resources available when the set-up message arrives. To deploy QoS routing with reasonable network overheads, the delays for propagating and processing these set-up messages must be much smaller than the link-state update periods and connection holding times. In assuming that propagation and processing delays are negligible, our model focuses on the primary effects of stale link-state information on establishing connections for the long-lived traffic flows. Finally, we model at most one attempt to signal a connection. Although we do not evaluate alternate routing (or crankback) after a set-up failure, the connection blocking probability provides an estimate of the frequency of crankback operations. In practice, a "blocked" request may be repeated at a lower QoS level, or the network may carry the offered traffic on a preprovisioned static route.

2.4 Link-State Update Policies

Every switch has accurate information about the utilization and cost of its own outgoing links, and potentially stale information about the other links in the network. To extend beyond the periodic link-state update policies evaluated in previous performance studies [2, 17–19], we consider a three-parameter model that applies to the routing protocols in PNNI and the proposed QoS extensions to OSPF. In particular, the model includes a trigger that responds to significant changes in available bandwidth, a hold-down timer that enforces a minimum spacing between updates, and a refresh period that provides an upper bound on the time between updates. The link state is the available link bandwidth, beyond the capacity already reserved for other QoS-routed traffic (i.e., $1 - u_i$). This is in contrast to traditional best-effort routing protocols (e.g., OSPF) in which updates essentially convey only topology information. We do not assume, or model, any particular technique for distributing this information in the network; two possibilities are flooding (as in PNNI and OSPF) or broadcasting via a spanning tree.

The periodic update messages provide a refresh of the link utilization information, without regard to changes in the available capacity. Still, the predictable nature of periodic updates simplifies the provisioning of processor and bandwidth resources for the exchange of link-state information. To prevent synchronization of

Topology	N	L	Deg.	Diam.	$\overline{h}$
Random graph	100	492	4.92	6	3.03
MCI backbone	19	64	3.37	4	2.34
Regular topology	125	750	6	6	3.63

Table 1. The random graph is generated using Waxman's model [27]; nodes in the regular topology have identical connectivity.

update messages for different links, each link introduces a small random component to the generation of successive updates [8]. In addition to the refresh period, the model generates updates upon detection of a significant change Δ_i in the available capacity since the last update message, where $\Delta_i = \frac{|u_i' - u_i|}{1-u_i'}$. These changes in link state stem from the reservation (release) of link bandwidth during connection establishment (termination). By updating link load information in response to a change in available bandwidth, triggered updates respond to smaller changes in utilization as the link nears capacity, when the link may become incapable of supporting new connections. Similarly, connections terminating on a heavily-loaded link introduce a large relative change in available bandwidth, which generates an update message even for very large triggers. In contrast to periodic updates, though, triggered updates complicate network resource provisioning since rapid fluctuations in available capacity can generate a large number of link-state updates, unless a reasonable hold-down timer is used.

2.5 Network and Traffic Model

A key challenge in studying protocol behavior in wide-area networks lies in how to represent the underlying topology and traffic patterns. The constantly changing and decentralized nature of current networks (in particular, the Internet) results in a poor understanding of these characteristics and makes it difficult to define any "typical" configuration [20]. Adding to the challenge are observations that conclusions about algorithm or protocol performance may in fact vary dramatically with the underlying network model. For example, random graphs can result in unrealistically long paths between certain pairs of nodes, "well-known" topologies may show effects that are unique to particular configurations, and regular graphs may hide important effects of heterogeneity and non-uniformity [29]. Consequently, our simulation experiments consider a range of network topologies (see Table 1), and we comment on similarities and differences between the trends in each configuration.

As our study focuses on backbone networks, we consider topologies with relatively high connectivity, an increasingly common feature of emerging core backbone networks [29], that support a dense traffic matrix (with significant traffic between most pairs of core nodes) and are resilient to link failures. Each node can be viewed as a single core switch in a backbone network that sends and receives traffic for one or more sources and carries transit traffic to and from other switches or routers. In addition to studying a representative "well-known" core topology (an early representation of the MCI backbone that has appeared in other routing studies [17, 18]), we also evaluate both random graphs and regular topologies in order to vary important parameters like size, diameter, and node degree in a controlled fashion. Most of our graphs show results for the MCI and random topologies, though we use a set of regular graphs with different degrees of connectivity to evaluate the effects of having multiple shortest-path routes between pairs of nodes [25]. We further assume that the topology remains fixed throughout each simulation experiment; that is, we do not model the effects of link failures.

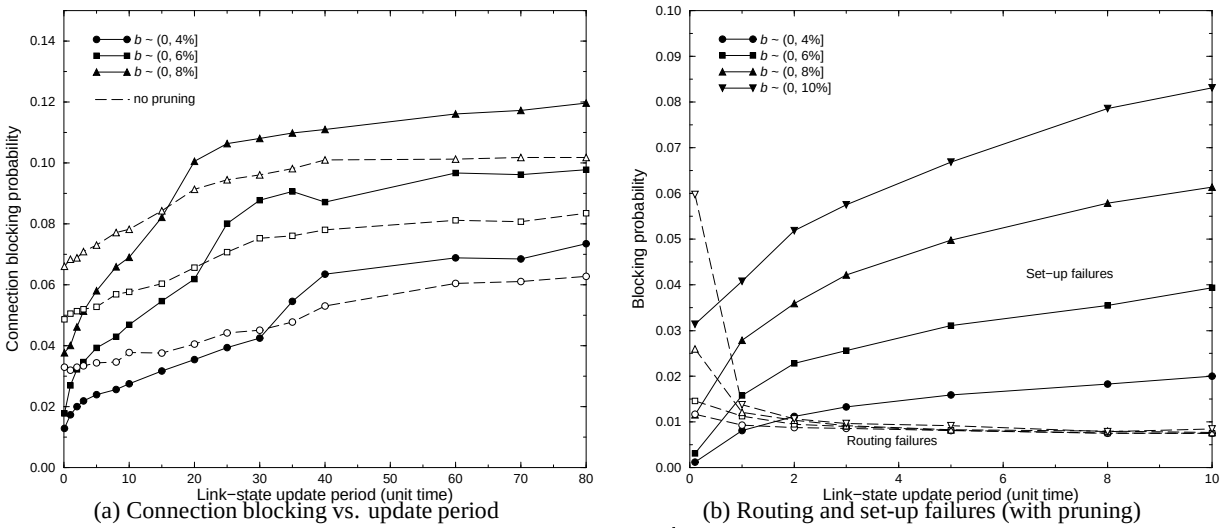
Each node generates connection requests according to a Poisson process with rate λ, with uniform random selection of destination nodes. This results in a uniform traffic pattern in the regular graphs, and a non-uniform pattern on the MCI and random topologies, allowing us to compare QoS routing to static shortest-path routing under balanced and unbalanced loads. We model connection holding times using a Pareto distribution (shape parameter a, scale parameter β, and CDF $F_X(x) = 1 - (\beta/x)^a$) with $a = 2.5$ to capture the heavy-tailed nature of connection durations [20] while still producing a distribution with finite variance making it possible to gain sufficient confidence on the simulation results. For comparison we also conducted experiments with exponentially distributed holding times. We denote the mean holding time as ℓ. Connection bandwidths are uniformly-distributed within an interval with a spread about the mean $\overline{b}$. For instance, call bandwidths may have a mean of 5% of link capacity with a spread of 200%, resulting in $b \sim U(0.0, 0.1]$. Most of the simulation experiments focus on mean bandwidths from 2–5% of link capacity. Smaller bandwidth values, albeit perhaps more realistic, would result in extremely low blocking probabilities, making it almost impossible to complete the wide range of simulation experiments in a reasonable time; instead, the experiments consider how the effects of link-state staleness scale with the $\bar{b}$ parameter to project the performance for low-bandwidth connections. With a connection arrival rate λ at each of N switches, the offered network load is $\rho = \lambda N \ell \bar{b} \bar{h} / L$, where $\bar{h}$ is the mean distance (in number of hops) between nodes, averaged across all source-destination pairs.

3 Link-State Update Policies

Our initial simulation experiments focus on the effects of inaccurate link-state information on the performance and overheads of QoS routing by evaluating periodic and triggered updates in isolation.

3.1 Periodic Link-State Updates

The connection blocking probability increases as a function of the link-state update period, as shown in Figure 1(a). The experiment evaluates three bandwidth ranges on the random graphs with an offered load of $\rho = 0.75$; the connection arrival rate remains fixed at $\lambda = 1$, while the Pareto scale parameter, β, is used to adjust the mean holding time to keep load constant across the three configurations. For comparison, the graph shows results with and without pruning of (seemingly) infeasible links. We vary the update periods from almost continuous updates to very long periods of 200 times (graphs show up to 80 times) the average connection interarrival time (further details on simulation setup are available in [25]). Due to their higher resource requirements, the high-bandwidth connections experience a larger blocking probability than the low-bandwidth connections across the range of link-state update rates. The blocking probability for high-bandwidth connections, while higher, does not appear to grow more steeply as a function of the update period; instead, the three sets of curves remain roughly equidistant across the range of update periods.



(a) Connection blocking vs. update period (b) Routing and set-up failures (with pruning)

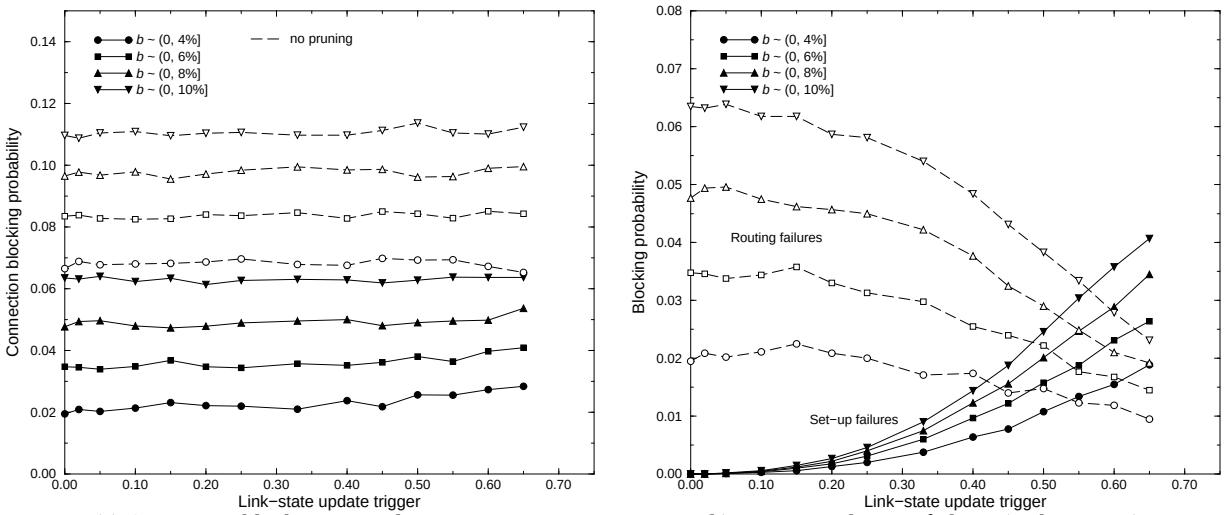
Figure 1. Random topology with $\lambda = 1$, $\alpha = 1$, and $\rho = .75$.

Pruning vs. not pruning: In experiments that vary the offered load, we see that pruning reduces the blocking probability under small to moderate values of ρ by allowing connections to consider nonminimal routes. However, pruning degrades performance under heavy load since these nonminimal routes consume extra link capacity, at the expense of other connections. Stale link-state information reduces the effectiveness of pruning, as shown in Figure 1(a). With out-of-date information, the source may incorrectly prune a feasible link, causing a connection to follow a nonminimal route when a minimal route is available. Hence, the staleness of the link-state information narrows the range of offered loads where pruning is effective, though other techniques can improve the performance. Experiments with different α values and topologies show the same basic trends as Figure 1(a), though the results with pruning disabled show a weaker dependence on the link-state update period in the MCI topology. Since the MCI topology has relatively low connectivity, most source-destination pairs do not have multiple minimum-length routes; hence, when pruning is disabled, the the route computation does not react much to changes in link load. In general, the network can control the negative influence of nonminimal routes by limiting the number of extra hops a connection can travel, or reserving a portion of link resources to connections on minimal routes. To address staleness more directly, the pruning policy could more conservative or more liberal in removing links to balance the trade-off between minimal and nonminimal routes [13].

Route flapping: Although QoS routing performs well for small link-state update periods (significantly outperforming static routing [25]), the blocking probability rises relatively quickly before gradually plateauing for large update periods. In Figure 1(a), even an update period of five time units (five times the average connection interarrival time) shows significant performance degradation. By this point, set-up failures account for all of the call blocking, except when the update period is very small (e.g., for update periods close to the interarrival time), as shown in Figure 1(b) which focuses on a small region of the experiments with pruning in Figure 1(a); when pruning is disabled, routing failures never occur, and set-up failures account for all blocked connections. In general, periodic updates do not respond quickly enough to variations in link state, sometimes allowing substantial changes to go unnoticed. This suggests that inaccuracy in the link-state database causes the source switch to mistake infeasible links as feasible; hence, the source selects an infeasible path, even when there are other feasible routes to the destination. We see that routing failures occur only with very accurate information since the source learns about link infeasibility very quickly. When link-state can fluctuate significantly between updates the source is virtually certain to find at least one seemingly feasible path, thus avoiding a routing failure.

Under large update periods, relative to the arrival rates and holding times, the links can experience dramatic fluctuations in link state between successive update messages. Such link-state flapping has been observed in packet routing networks [15], where path selection can vary on a packet-by-packet basis; the same phenomenon occurs here since the link-state update period is large relative to the connection arrival rates and holding times. When an update message indicates that a link has low utilization, the rest of the network reacts by routing more traffic to the link. Blocking remains low during this interval, since most connections can be admitted. However, once the link becomes saturated, connections continue to arrive and are only admitted if other connections terminate. Blocking stays relatively constant during this interval as connections come and go, and the link remains near capacity. For large update periods, this "plateau" interval dominates the initial "climbing" interval. Hence, the QoS-routing curves in Figure 1(a) flatten at a level that corresponds to the steady-state blocking probability during the "plateau" interval.

Eventually, QoS routing starts to perform worse than static routing, because the fluctuations in link state begin to exceed the random variations in traffic load. In searching for (seemingly) underutilized links, QoS routing targets a relatively small set of links until new update messages arrive to correct the link-state database. In contrast, under static routing, the source blindly routes to a single group of links, though this set is typically larger than the set identified by QoS routing. Thus, when the update period grows quite large, static routing is more successful at balancing load and reducing connection blocking. The exact crossover point between the two routing algorithms is very sensitive to the distribution of



(a) Connection blocking vs. update trigger (b) Routing and set-up failures (with pruning)

Figure 2. The experiments evaluate the MCI topology with $\rho = 0.70$, $\lambda = 1$, and $\alpha = 1$.

traffic in the network. For example, in the presence of "hot-spots" of heavy load, QoS routing can select links that circumvent the congestion (subject to the degree of staleness). Under such a non-uniform load, QoS routing continues to outperform static routing even for large update periods. For example, experiments with the non-homogeneous MCI backbone topology show that QoS routing consistently achieves lower blocking probability than static routing over a wide range of update rates.

Path length and bandwidth requirements: Fluctuations in link state have a more pernicious effect on connections between distant source-destination pairs, since QoS routing has a larger chance of mistakenly selecting a path with at least one heavily-loaded link. This is especially true when links do not report their new state at the same time, due to skews in the update periods at different switches. In other experiments, we compared connection blocking probability to several alternative measures of blocking [25]. The hop-count blocking probability is defined as the ratio of the hop-count of blocked connections to the hop-count of all connections; bandwidth blocking is defined analogously relative to requested bandwidth. Compared to conventional connection blocking, these metrics grow more quickly in the presence of stale information. In general, bandwidth blocking exceeds hop-count blocking, suggesting that high-bandwidth connections are even harder to route than high hop-count connections, though link-state staleness does not seem to affect one metric more than the other.

Connection holding times: Despite the fact that staleness due to periodic updates can substantially increase connection blocking, the network can limit these effects by controlling which types of traffic employ QoS routing. For example, other experiments show that longer holding times allow the use of larger link-state update periods to achieve the same blocking probability [25]. Also, in comparing the trends to an identical experiment with exponentially distributed holding times, we found that the Pareto distribution produces a significantly more rapid rise in blocking probability over the same range of update periods (nearly twice as fast for some mean holding times). The heavier tail of the Pareto distribution results in many more shorter-lived connections than an exponential distribution with the same mean, implying that these shorter connections require very frequent updates to achieve acceptably low blocking probability. These results suggest that the network could limit QoS routing to the longer-lived traffic that would consume excessive link resources if not routed carefully, while relegating short-lived traffic to preprovisioned static routes. With some logical separation of resources for short-lived and long-lived traffic, the network could tune the link-state update policies to the arrival rates and holding times of the long-lived connections. With appropriate mechanisms to identify or detect long-lived traffic, such as flow detection schemes for grouping related packets [4], the network can assign this subset of the traffic to QoS routes and achieve good routing performance with a lower link-state update rate.

3.2 Triggered Link-State Updates

Although periodic updates introduce a predictable overhead for exchanging link-state information, triggered updates can offer more accurate link-state information for the same average rate of update messages. The graph in Figure 2(a) plots the connection blocking probability for a range of triggers and several bandwidth ranges in the MCI topology. In contrast to the experiments with periodic link-state updates, we find that the overall blocking probability remains relatively constant as a function of the trigger, across a wide range of connection bandwidths, cost metrics, and load values, with and without pruning, and with and without hold-down timers. Additional experiments with the well-connected regular topology show the same trend [25].

Blocking insensitivity to update trigger: To understand this phenomenon, consider the two possible effects of stale link-state information on the path-selection process when pruning is enabled. Staleness can cause infeasible links to appear feasible, or cause the switch to dismiss links as infeasible when they could in fact support the connection. When infeasible links look feasible, the source may mistakenly choose a route that cannot actually support the connection, resulting in a set-up failure. However, if the source had accurate link-state information, any infeasible links would have been pruned prior to computing a route. In this case, blocking is likely to occur because the source cannot locate a fea-

sible route, resulting in a routing failure. Instead of increasing the connection blocking probability, the stale information changes the nature of blocking from a routing failure to a set-up failure. Figure 2(b) highlights this effect by plotting the blocking probability for both routing and set-up failures. Across the range of trigger values, the increase in set-up failures is offset by a decrease in routing failures.

Now, consider the other scenario in which staleness causes feasible links to look infeasible. In this case, stale information would result in routing failures because links would be unnecessarily pruned from the link-state database. Although this case can sometimes occur, it is very unlikely, since the triggering mechanism ensures that the source switch has relatively accurate information about heavily-loaded links. For example, a connection terminating on a fully-utilized link would result in an extremely large change in available bandwidth, which would activate most any trigger. Moreover, a well-connected topology often has more than one available route between any two nodes; the likelihood of pruning links incorrectly on *all* of the feasible routes is quite low. Hence, the blocking probability is dominated by the previous scenario, namely mistaking infeasible links as feasible. Additional experiments illustrate that the trade-off between routing and set-up failures persists even in the presence of hold-down timers, though the hold-down timer increases the overall blocking probability and rate of signalling failures.

However, in a loosely-connected networks, an incorrect pruning decision can cause the source to erroneously consider nonminimal routes. For example, the random topology has *higher* blocking rates with *smaller* trigger values when trying to route high-bandwidth connections [25]. Unlike the other two topologies, the random graph typically does not have multiple equal-length paths between a pair of nodes. As a result, pruning an infeasible link along the shortest path results in the selection of a nonminimal route. In the end, this increases the overall blocking probability, since these nonminimal routes consume extra resources. If, instead, the source chose not to prune this infeasible link (say, due to stale link-state information), then the connection would attempt to signal along the shortest path. Although the connection would block upon encountering the infeasible link, the network would benefit by deciding not to accept the connection. Having slightly out-of-date information has a throttling effect in this case; in fact, the use of a small hold-down timer has a similar effect, resulting in much flatter curves for blocking as a function of the trigger. Still, it is generally unwise to apply pruning for high-bandwidth connections when the topology does not have multiple routes of equal (or similar) length. For most realistic scenarios, blocking remains largely insensitive to the trigger value.

Link-state update rate: Despite the increase in set-up failures, large trigger values substantially reduce the number of update messages for a given blocking probability, as shown in Figure 3. For very fine-grained triggers, every connection establishment and termination generates an update message on each link in the route, resulting in an update rate of $2\lambda N\bar{h}/L$ in a network with N switches, L links, and an average path length of $\bar{h}$ hops. For the parameters in this experiment, the expression reduces to 1.23 link-state update messages per unit time, which is close to the y-intercept in Figure 3; additional experiments show that the link-state update rate is not sensitive to the connection holding times,

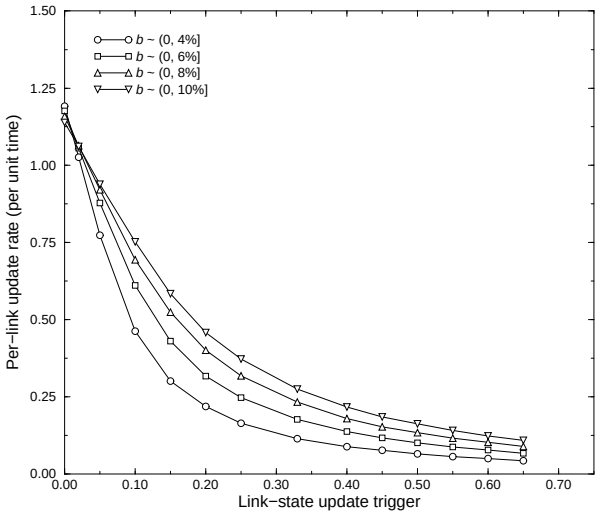


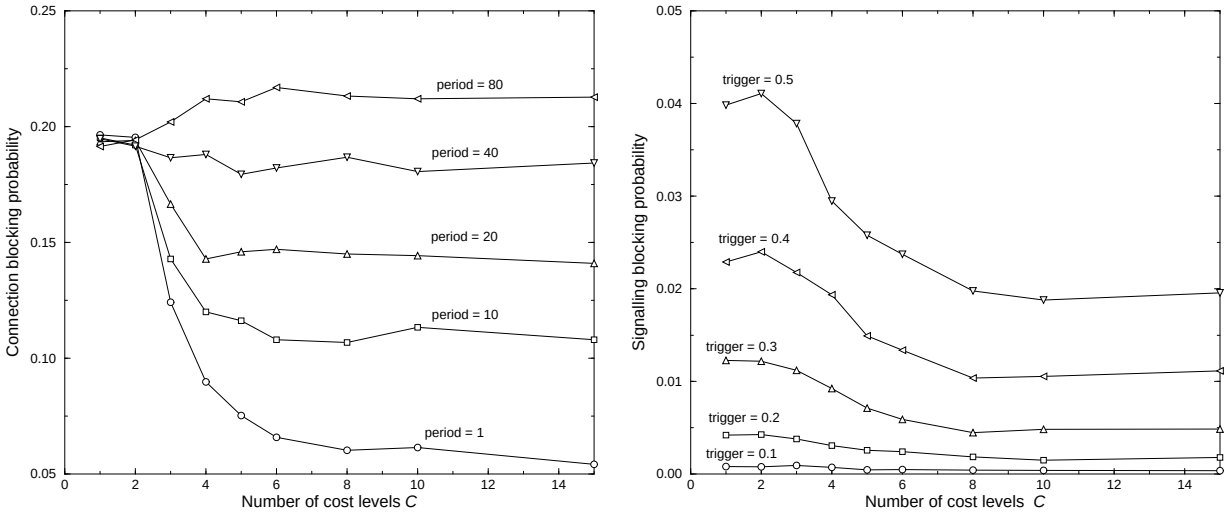
Figure 3. Link-state update rate for random topology with $\rho = 0.75$, $\lambda = 1$, $\alpha = 1$, and no pruning.

consistent with the $2\lambda N\bar{h}/L$ expression. In Figure 3, the larger bandwidth values have a slightly lower link-state update rate for small triggers; high-bandwidth connections experience a higher blocking rate, which decreases the proportion of calls that enter the network and generate link-state messages. When triggers are coarse, however, more connections are signalled in the network (due to fewer routing failures), and the high-bandwidth connections trigger more updates since they create greater fluctuation in link state.

Unlike routing failures, set-up failures can generate link-state update messages, since reserving and releasing link resources changes link state, even if the connection ultimately blocks at a downstream node. The increase in route flapping and set-up failures for larger triggers slows the reduction in the update rate in Figure 3 as the trigger grows. The exact effect of set-up failures depends on the number of successful hops before the connection blocks. Also, if the network supports crankback operations, the attempt to signal the connection on one or more alternate routes could generate additional link-state update messages. Finally, as a secondary effect, pruning infeasible links at the source switch can inflate the update rate by selecting nonminimal routes that reserve (and release) resources on extra links. Overall, though, modest trigger values are effective at reducing the link-state update rate by about a factor of three to four. Also, for a fixed update rate, triggers can significantly reduce the proportion of set-up failures when compared with periodic updates. For instance, setting the trigger to around 0.30 results in an average update interarrival of 3 (for $b \sim (0, 0.06]$) and 17% of the blocking occurs in signalling. When using periodic updates at the same frequency, set-up failures account for 74% of the blocked connections, and the blocking rate is much higher.

4 Link-Cost Parameters

In this section we consider the impact of the link-cost parameters (C and α) of the routing algorithm on blocking probability and route computation complexity.



(a) Blocking for periodic updates (b) Signalling failures for triggered updates

Figure 4. Discretized costs with stale link-state information: Both experiments simulate the 5-ary 3-cube with $\rho = 0.85$, $b \sim (0.0, 0.1]$, $\lambda = 1$, ℓ is exponentially distributed with mean 28, and $\alpha = 1$.

4.1 Number of Cost Levels (C)

The experiments in Section 3 evaluate a link-cost function with a large number of cost levels, limited only by machine precision. With such fine-grain cost information, the path-selection algorithm can effectively differentiate between links to locate the "cheapest" shortest-path route. Figure 4(a) evaluates the routing algorithm over a range of cost granularity and link-state update periods. To isolate the effects of the cost function, the routing algorithm does not attempt to prune (seemingly) infeasible links before invoking the shortest-path computation. The C cost levels are distributed throughout the range of link utilizations by setting $u_{\min} = 0$. Compared to the high blocking probability for static routing ($C = 1$), larger values of C tend to decrease the blocking rate, particularly when the network has accurate link-state information, as shown in the "period=1" curve in Figure 4(a).

Fine-grain cost metrics are less useful, however, when link-state information is stale. For example, having more than four cost levels does not improve performance once the link-state update period reaches 20 times the average interarrival time. Although fine-grain cost metrics help the routing algorithm distinguish between links, larger values of C also limit the number of links that the routing algorithm considers, which can cause route flapping. In contrast, coarse-grain cost information generates more "ties" between the multiple shortest-path routes to each destination, which effectively dampens link-state fluctuations by balancing the load across several alternate routes. In fact, under stale information, small values of C can sometimes outperform large values of C, but this crossover only occurs once the update period has grown so large that QoS routing has a higher blocking probability than static routing. The degradation in performance under high update periods is less significant in the MCI and random topologies, due to the lower likelihood of having multiple minimum-hop paths between pairs of nodes.

The appropriate number of cost levels depends on the update period and the connection-bandwidth requirements, as well as the overheads for route computation. Larger values of C increase the complexity of the Dijkstra shortest-path computation without offering significant reductions in the connection blocking probability. Fine-grain cost information is more useful in conjunction with triggered link-state updates, as shown in Figure 4(b). We still find, however, that experiments with a finite number of C values are consistent with the results in Section 3.2; that is, the connection blocking probability remains constant over a wide range of triggers. Since the trigger value does not affect the overall blocking probability, Figure 4(b) plots only the signalling failures. In contrast to the experiment with periodic updates, increasing the number of cost levels beyond $C = 4$ continues to reduce the blocking rate. Since triggered updates do not aggravate fluctuations in link state, the fine-grain differentiation between links outweighs the benefits of "ties" between shortest-path routes. Although larger values of C reduce the likelihood of signalling failures by a factor of two, increasing the number of cost levels eventually offers diminishing returns.

4.2 Link-Cost Exponent (α)

To maximize the utility of coarse-grain load information, the cost function should assign each cost level to a critical range of link utilizations. Under fine-grain link costs (large C), our experiments show that the exponent α does not have much impact on performance; values of $\alpha \geq 1$ have nearly identical performance. These results hold across a range of link-state update periods, suggesting that large values of α do not introduce much extra route flapping. This is important for path selection algorithms, since it suggests that widest shortest-path and cheapest shortest-path should have similar performance under stale link-state information. However, the choice of exponent α plays a more important role in cost-based routing with coarse-grain link costs. These experiments showed a sharp drop in the blocking probability due to the transition from static routing ($\alpha = 0$) to QoS routing ($\alpha > 0$), followed by an increase in blocking probability for larger values of α [25]. When α is too large, the link-cost function concentrates most of the cost information in a very small, high-load region.

For large α and small C, some of the cost intervals are so narrow that the arrival or departure of a single connection could

change the link cost by one or more levels. For example, when $\alpha = 8$ and $C = 10$, the link-cost function has four cost levels in the 90–100% range. This sensitivity exacerbates route flapping and also limits the routing algorithm's ability to differentiate between links with lower utilization. Further experiments demonstrate that pruning lowers the differences between the curves for different C values. This occurs because pruning provides additional differentiation between links, even for small values of C. We also explored the effects of the link-state update period on the connection blocking probability as α is increased, for a fixed value of C. Interestingly, larger update periods dampen the detrimental effects of large values of α, resulting in a flatter blocking probability curve. Although large values of α limit the granularity of the cost information, the drawback of a large value of α is largely offset by the benefit of additional "ties" in the routing algorithm when information is stale. Hence, the selection of α is actually more sensitive when the QoS-routing algorithm has accurate knowledge of link state.

5 Conclusions and Future Work

The performance and complexity of QoS routing depends on the complex interaction between a large set of parameters. This paper has investigated the scaling properties of source-directed link-state routing in large core networks. Our simulation results show that the routing algorithm, network topology, link-cost function, and link-state update policy each have a significant impact on the probability of successfully routing new connections, as well as the overheads of distributing network load metrics. Our key observations are:

Periodic link-state updates: The staleness introduced by periodic link-state update messages causes flapping that substantially increases the rate of set-up failures. This increases connection blocking and also consumes significant resources inside the network, since most of the failures occur during connection set-up instead of during path selection. In extreme cases with large update periods, QoS routing actually performs worse that load-independent routing, due to excessive route flapping. Our results show that a purely periodic link-state update policy cannot meet the dual goals of low blocking probability and low update overheads in realistic networks.

Bandwidth and hop-count: Connections with large bandwidth requirements experience higher blocking probabilities, but the effects of increasing link-state staleness are only slightly worse relative to lower-bandwidth connections. However, stale link-state information has a strong influence on connections between distant source-destination pairs, since long paths are much more likely to have at least one infeasible link that looks feasible, or one feasible link that looks infeasible. These effects degrade the performance of QoS routing in large network domains, unless the topology is designed carefully to limit the worst-case path length.

Holding times: Longer connection holding times change the timescale of the network and allow the use of larger link-state update periods. Stale information has a more dramatic effect under heavy-tailed holding-time distributions, due to the relatively large number of short-lived connections for the same average holding time. Our findings suggest that the networks should limit QoS routing to long-lived connections, while carrying short-lived traffic on preprovisioned static routes. The network can segregate the traffic in short- and long-lived flows by partitioning link bandwidth for the two classes, and detecting long-lived flows at the edge of the network [7].

Triggered link-state updates: Triggered link-state updates do not significantly affect the overall blocking probability, though coarse-grain triggers do increase the amount of blocking that stems from more expensive set-up failures. Triggers reduce the amount of unnecessary link-state traffic but require a hold-down timer to prevent excessive update messages in short time intervals. However, larger hold-down timers increase the blocking probability and the number of set-up failures. Hence, our findings suggest using a combination of a relatively coarse trigger with a modest hold-down timer.

Pruning infeasible links: In general, pruning infeasible links improves performance under low-to-moderate load by allowing connections to consider nonminimal routes, and avoiding unnecessary set-up failures by blocking more connections in the route computation phase. However, under heavy load, these nonminimal routes consume extra link resources, at the expense of other connections. Pruning becomes less effective under stale link-state information, loosely-connected topologies, and high-bandwidth connections, since these factors increase the amount of traffic that follows a nonminimal route, even when a minimal route is feasible. These results suggest that large networks should disable pruning, unless most source-destination pairs have multiple routes of equal (or near equal) length. Alternatively, the network could impose limits on the resources it allocates to nonminimal routes.

Rich network topologies: The trade-off between routing and set-up failures also has important implications for the selection of the network topology. Although dense topologies offer more routing choices, the advantages of multiple short paths dissipate as link-state information becomes more stale. Capitalizing on dense network topologies requires more frequent link-state updates, and techniques to avoiding excessive link-state traffic. For example, the network could broadcast link-state updates in a spanning tree, and piggyback link-state information in signalling messages.

Coarse-grain link costs: Computational complexity can be reduced by representing link cost by a small number of discrete levels without significantly degrading performance. This is especially true when link-state information is stale, suggesting a strong relationship between temporal and spatial inaccuracy in the link metrics. In addition, coarse-grain link costs have the benefit of increasing the number of equal-cost routes, which improves the effectiveness of alternate routing. We exploit these characteristics in our recent work on precomputation of QoS routes [26].

Exponent alpha: Under fine-grain link costs (large C), routing performance is not very sensitive to the exponent α; an exponent of 1 or 2 performs well, and larger values do not appear to increase route flapping, even under very stale information. These results suggest that selecting routes based widest shortest-paths or cheapest shortest-paths would have similar performance under stale link state. Coarse-grain link costs require more careful selection of α, to ensure that each cost level provides useful information, and that the detailed cost information is focused in the expected load region.

These observations represent an initial step in understanding and controlling the complex dynamics of quality-of-service routing

under stale link-state information. We find that our distinction between routing and set-up failures, and simulation experiments under a wide range of parameters, provide valuable insights into the underlying behavior of the network. Our ongoing work focuses on exploiting these trends to reduce the computational and protocol overheads of QoS routing in large backbone networks.

Acknowledgment

The authors wish to thank Anja Feldmann, Pawan Goyal, Albert Greenberg, Gisli Hjalmtysson, and Balachander Krishnamurthy for their helpful feedback on earlier versions of this paper.

References

[1] H. Ahmadi, J. S. Chen, and R. Guerin. Dynamic routing and call control in high-speed integrated networks. In A. Jensen and V. B. Iversen, editors, *Teletraffic and Datatraffic in a Period of Change: Proc. of the International Teletraffic Congress*, volume 14 of *Studies in Telecommunication*, pages 397–403. North-Holland, Copenhagen, Denmark, June 1991.

[2] L. Breslau, D. Estrin, and L. Zhang. A simulation study of adaptive source routing in integrated services networks. Technical Report 93-551, Computer Science Department, University of Southern California, 1993.

[3] B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest-path algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73(2):129–174, May 1996.

[4] K. C. Claffy, H.-W. Braun, and G. C. Polyzos. A parameterizable methodology for Internet traffic flow profiling. *IEEE Journal on Selected Areas in Communications*, 13(8):1481–1494, October 1995.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press (McGraw-Hill), Cambridge, MA (New York), 1990.

[6] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A framework for QoS-based routing in the Internet. Internet RFC (RFC 2386), August 1998.

[7] A. Feldmann, J. Rexford, and R. Caceres. Reducing overhead in flow-switched networks: An empirical study of web traffic. In *Proceedings of IEEE INFOCOM*, April 1998.

[8] S. Floyd and V. Jacobson. Synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking*, 2(2):122–136, April 1994.

[9] R. Gawlick, C. Kalmanek, and K. Ramakrishnan. Online routing for virtual private networks. *Computer Communications*, 19(3):235–244, March 1996.

[10] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and admission control in general topology networks. Technical Report CS-TR-95-1548, Stanford University, May 1995.

[11] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS routing mechanisms and OSPF extensions. Internet Draft (draft-guerin-qos-routing-ospf-03.txt), work in progress, March 1998.

[12] R. Guerin and A. Orda. QoS-based routing in networks with inaccurate information. In *Proceedings of IEEE INFOCOM*, April 1997.

[13] R. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *Proceedings of IEEE GLOBECOM*, November 1997.

[14] A. Iwata, R. Izmailov, H. Suzuki, and B. Sengupta. PNNI routing algorithms for multimedia ATM internet. *NEC Reserach & Development*, 38(1), January 1997.

[15] A. Khanna and J. Zinky. The revised ARPANET routing metric. In *Proceedings of ACM SIGCOMM*, pages 45–56, Austin, TX, 1989.

[16] W. C. Lee, M. G. Hluchyj, and P. A. Humblet. Routing subject to quality of service constraints in integrated communication networks. *IEEE Network Magazine*, pages 46–55, July/August 1995.

[17] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *Proceedings of IEEE International Conference on Network Protocols*, Atlanta, GA, October 1997.

[18] Q. Ma and P. Steenkiste. Quality-of-service routing for traffic with performance guarantees. In *Proc. IFIP International Workshop on Quality of Service*, pages 115–126, Columbia University, New York, May 1997.

[19] I. Matta and A. U. Shankar. Dynamic routing of real-time virtual circuits. In *Proceedings of IEEE International Conference on Network Protocols*, pages 132–139, Columbus, OH, 1996.

[20] V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. In *Proc. of the Winter Simulation Conference*, Atlanta, GA, December 1997.

[21] M. Peyravian and R. Onvural. Algorithm for efficient generation of link-state updates in ATM networks. *Computer Networks and ISDN Systems*, 29(2):237–247, January 1997.

[22] PNNI Specification Working Group. *Private Network-Network Interface Specification Version 1.0*. ATM Forum, March 1996.

[23] C. Pornavalai, G. Chakraborty, and N. Shiratori. QoS based routing in integrated services packet networks. In *Proceedings of IEEE International Conference on Network Protocols*, Atlanta, GA, October 1997.

[24] S. Rampal and D. Reeves. Routing and admission control algorithms for multimedia data. *Computer Communications*, October 1995.

[25] A. Shaikh, J. Rexford, and K. G. Shin. Dynamics of quality-of-service routing with inaccurate link-state information. Technical Report CSE-TR-350-97, Computer Science and Engineering, University of Michigan, November 1997.

[26] A. Shaikh, J. Rexford, and K. G. Shin. Efficient precomputation of quality-of-service routes. In *Proceedings of Workshop on Network and Operating System Support for Digital Audio and Video*, July 1998.

[27] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.

[28] Z. Whang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, September 1996.

[29] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM*, pages 594–602, March 1996.

[30] Z. Zhang, C. Sanchez, B. Salkewicz, and E. S. Crawley. Quality of service extensions to OSPF or quality of service path first routing (QOSPF). Internet Draft (draft-zhang-qos-ospf-01.txt), work in progress, September 1997.