Dynamics of Quality-of-Service Routing with Inaccurate Link-State Information

Anees Shaikh†[*], Jennifer Rexford‡, and 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 can satisfy application performance requirements and optimize network resource usage by selecting paths based on connection traffic parameters and link load information. However, effective path-selection schemes require the distribution of link-state information, which can cause a significant burden on the bandwidth and processing resources in the network. We investigate the fundamental tension between network overheads and the quality of routing decisions in the context of source-directed QoS routing algorithms. In contrast to previous performance studies that compare different routing algorithms under specific network configurations, we characterize how the performance and overheads of QoS routing relate to the link-state update policies, as a function of the underlying traffic load, network topology, and link-cost metrics. We explore the interplay between stale link-state information and random fluctuations in traffic load through a broad set of simulation experiments on a parameterized model of QoS routing. These results suggest ways to tune the frequency of link-state update messages and the link-cost functions to strike a careful balance between high accuracy and low complexity.

Keywords: quality-of-service, source-directed routing, explicit routing, signalling, link state

[*]The work of this author was performed while visiting AT&T Labs – Research

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 [1,2]. 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 has the potential to satisfy diverse performance requirements and optimize resource usage by selecting a path for each flow or connection based on the traffic parameters and network load [3–5]. 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 fundamental tension 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 [6] 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, the proposed QoS extensions to the OSPF protocol include an "explicit routing" mechanism for source-directed IP routing [7,8]. Despite the attractiveness of these schemes, 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 the scalability of these protocols in large networks, switches and links can be assigned to smaller peer groups or areas that exchange detailed topology and 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 tension 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 [9]. To characterize the effects of stale information, comparisons of different QoS-routing algorithms have included simulation experiments that vary the link-state update period [10–12], while other work considers a combination of periodic and triggered updates [13]. 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 [8,12]; 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 subtle interplay between a large 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. 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 routes with a large number of links.

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 [10–19], 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. To guide the evaluation, Section 2 introduces a parameterized model of route computation, link-state metrics, and update policies; we also address how our model relates to previous work on QoS routing. In this context, we evaluate an efficient algorithm that selects routes based on bandwidth requirements, path lengths, and link utilization. Since the model's complexity 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 an 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 discussion of future research directions.

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, each link distributes information about its resources in a periodic fashion or in response to a significant change in the available capacity. Using this model, we characterize the effects of stale link-state information and random fluctuation in traffic load on the performance and overheads of QoS routing in a well-provisioned backbone network.

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. Similarly, 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 [11, 15–18, 20]. 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 [15].

In our model, the source selects a route based on the bandwidth requirement b and the destination node in three steps, effectively computing a "cheapest-shortest-feasible" path:

1. Prune infeasible links (i.e., links i with $u'_i + b > 1$)

2. Compute shortest paths to the destination based on hop-count

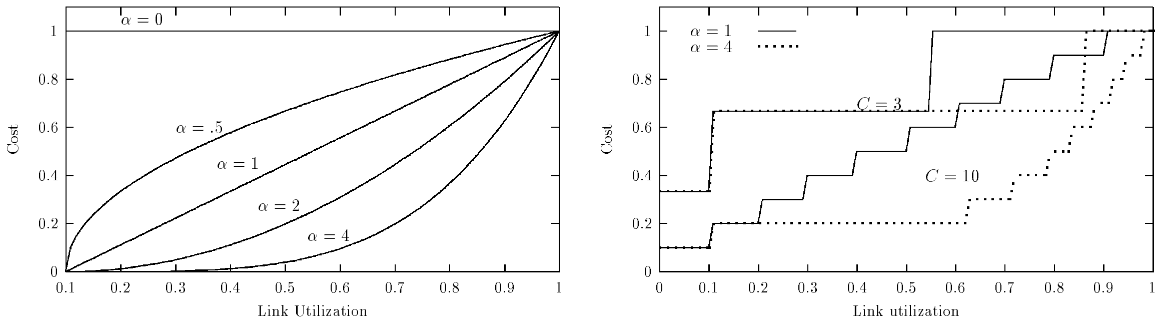
3. Extract a route with the minimum total cost $\sum_i c'_i$.

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 [21] to compute a minimum-hop path with the smallest total cost[1]. 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 [22], the shortest-path computation still incurs significant overhead in large networks. Extracting the route from the tree introduces complexity in proportion to the path length with a maximum complexity of $O(N)$ for an N-hop route.

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,

[1] A careful assignment of link weights w_i permits a single invocation of the Dijkstra algorithm to produce a minimum-cost, shortest path. In a network with N switches and $0 < c_i \leq 1$, the link weights $w_i = N + c_i$ ensure that paths with h links always appear cheaper than paths with $h+1$ links. In particular, h-hop routes have a maximum cost of $h(N+1)$, while any $(h+1)$-hop route has a cost that exceeds $(h+1)N$, where $h \leq N$.



(a) Continuous cost functions (b) Discrete cost functions

Figure 1: **Link-cost metrics**: These graphs show the basic shape of the exponential link-cost function, in its continuous and discrete forms, for $u_{\min} = 0.1$ and several values of the exponent α.

maximum link utilization on the path, or sum of the link delays. Defining the path cost as the sum of link utilization reduces call blocking probability and results in less route oscillation by adapting slowly to changes in network load [17]. Other studies have shown that assigning each link a cost that is exponential in its current utilization results in optimal call blocking probability [23]. 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. We define the 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.

Figure 1(a) plots link-cost functions for several values of α with $u_{\min} = 0.1$. When $\alpha = 0$ the path-selection scheme reduces to load-independent routing, while $\alpha = 1$ selects a shortest-path with the minimum sum of link utilization. Large values of α loosely correspond to widest shortest-path routing, since the large exponent virtually eliminates heavily-loaded links from consideration. Since it is expensive for the link-state database to support an arbitrary number of cost levels, our model converts the continuous cost function to C discrete values

$$c_i = \begin{cases} \dfrac{\left\lceil \left(\frac{u_i - u_{\min}}{1 - u_{\min}}\right)^{\alpha} \cdot (C-1) \right\rceil + 1}{C} & u_i > u_{\min} \\ 1/C & \text{otherwise,} \end{cases}$$

as shown in Figure 1(b). Small values of C reduce network overhead by decreasing the number of bits in the link-state database. More importantly, limiting the number of cost levels can substantially lower the complexity of the Dijkstra shortest-path computation; relatively simple algorithms have $O(L + CN)$ complexity [21], while more complicated approaches offer even further reduction [22]. However, coarse-grain link-cost information can degrade performance by limiting the routing algorithm's ability to distinguish between links with different available resources.

2.3 Call Signalling

When a new connection request arrives, the source switch applies the three-step routing algorithm to select a suitable path. However, 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 the call 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 $u'_i + b > 1$). 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 call. 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 *signalling failure* occurs if a link does not have enough resources available when the set-up message arrives. Our simulation experiments assume that the propagation and processing delays for signalling messages are small, relative to connection durations. Instead of modeling the latency in establishing and terminating connections, we characterize the effects of stale link state on QoS routing for longer-lived traffic streams. Still, by modeling the staleness of link-state information, we capture the basic effects of inaccuracy on connection signalling. Since these link-state update messages occur on a coarser time scale than connection set-up delay, the staleness of link-state information is likely to have a more significant influence on routing and signalling failures. Finally, we model at most one attempt to signal a connection. Although we do not evaluate alternate routing (or crankback) after a signalling failure, the connection blocking rate provides an estimate of the frequency of crankback operations.

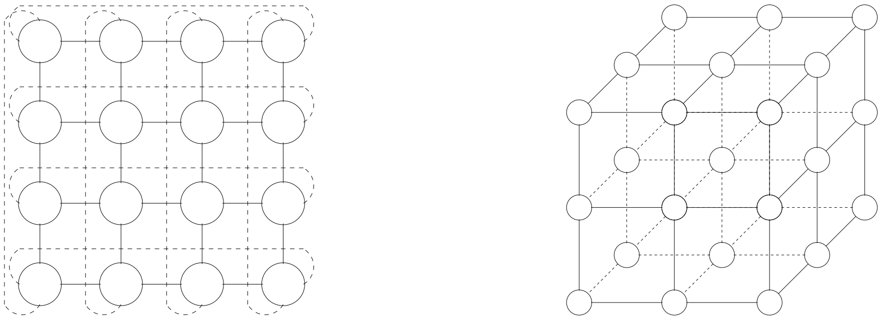
2.4 Link-State Update Policies

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 [10–12, 17], 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. To improve the efficiency of the simulation, we avoid devoting any events to link-state updates. Instead, we perform a "lazy" evaluation that updates link state only when a particular link is used by a connection.

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 update messages for different links, each link introduces a small random component to the generation of successive updates [24]. 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



(a) 4-ary 2-cube (b) 3-ary 3-cube

Figure 2: **Sample k-ary n-cube topologies:** The left figure shows a 4-ary 2-cube with the wrap links indicated by dotted lines, whereas the right figure shows a 3-ary 3-cube with the wrap links omitted for clarity. Each edge in the figures represents two unidirectional links.

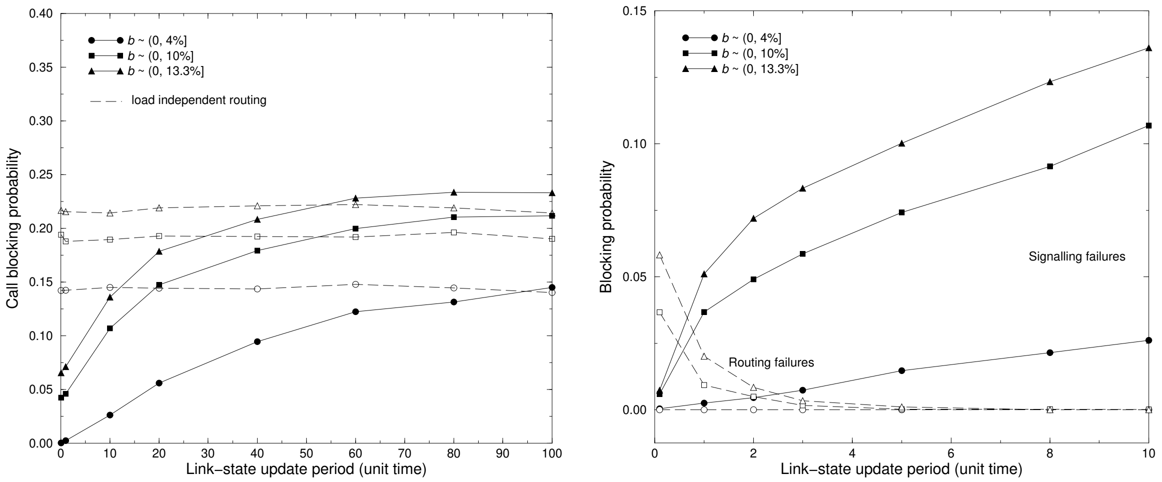
generates an update message even for very large trigger thresholds. In contrast to periodic updates, though, triggered messages complicate the provisioning of network resources since rapid fluctuations in available capacity can generate a large number of link-state updates, unless a reasonable hold-down timer is used.

2.5 Network and Traffic Model

For a systematic study of link-state routing under different network configurations, we employ a parameterized model of network topology and traffic load. Performance evaluation on small "well-known" networks, such as the NSFnet or MCI backbone, may reveal trends that are particular to the specific topology [25]. In addition, depending on the simulated traffic pattern, these non-homogeneous topologies may unfairly bias the evaluation against static routing algorithms. As an alternative, random graphs could provide more uniform connectivity, although these networks are difficult to study in a systematic manner and may result in unrealistically long paths between certain pairs of nodes [25]. Instead, our experiments focus on regular graphs, which permit us to change the network size, diameter, and node degree in a controlled fashion. To verify the trends in the simulation results, we occasionally refer to the results of similar experiments on other "well-known" topologies.

The simulation experiments evaluate the class of k-ary n-cube topologies, with k nodes in each of n dimensions, as shown by the examples in Figure 2. 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. For simplicity, we assume that links are bidirectional, with unit capacity in each direction. A k-ary n-cube has $N = k^n$ nodes of degree $2n$, with $L = 2nk^n$ links and diameter $D = \lfloor k/2 \rfloor n$. These graphs have relatively dense connectivity, common in emerging core backbone networks. Most of the experiments in the paper evaluate a 125-node 5-ary 3-cube topology, though Section 3.3 compares the effects of stale link-state information on different k-ary n-cube networks. We further assume that the topology remains fixed throughout each simulation experiment; that is, we do not model the effects of link failures.

Instead of simulating a non-uniform traffic pattern that would favor QoS-routing algorithms, the experiments evaluate a homogeneous traffic pattern, with a uniform random selection of source and destination nodes. For simplicity, we assume that connection interarrival and holding times are exponentially-distributed with means $1/\lambda$ and ℓ, respectively. 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 $b \sim U(0.0, 0.1]$.



(a) Connection blocking vs. update period (b) Routing and signalling failures

Figure 3: **Staleness due to periodic updates:** The left graph shows that the blocking probability grows rapidly as the link update period grows for several ranges of requested bandwidth; the dashed lines indicate the performance of static routing ($\alpha = 0$ and no pruning). The right graph shows that the blocking is dominated by signalling failures after only a small increase in the period. In both graphs $\lambda = 1$, $\alpha = 1$, and $\rho = .85$.

Most of the simulation experiments focus on mean bandwidths from 2–10% 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

The 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. With a periodic update policy, large periods substantially increase connection blocking, ultimately outweighing the benefits of QoS routing. In contrast, experiments with triggered updates show that coarse-grain triggers do not have a significant impact on the overall blocking probability, although larger triggers shift the type of blocking from routing failures to more expensive signalling failures. The experiments also show that this shift between routing and signalling failures degrades the performance of QoS routing in richly-connected network topologies.

3.1 Periodic Link-State Updates

The connection blocking probability increases as a function of the link-state update period, as shown in Figure 3. The experiment evaluates three bandwidth ranges at an offered load of $\rho = 0.85$; the connection arrival rate remains fixed at $\lambda = 1$, while the holding times are adjusted to keep load constant across the three configurations. For comparison, the graph also plots the performance of static shortest-path routing ($\alpha = 0$ with no pruning). We vary the update periods from almost

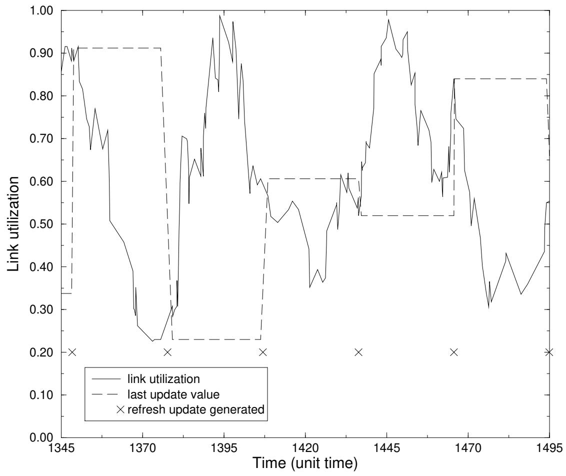


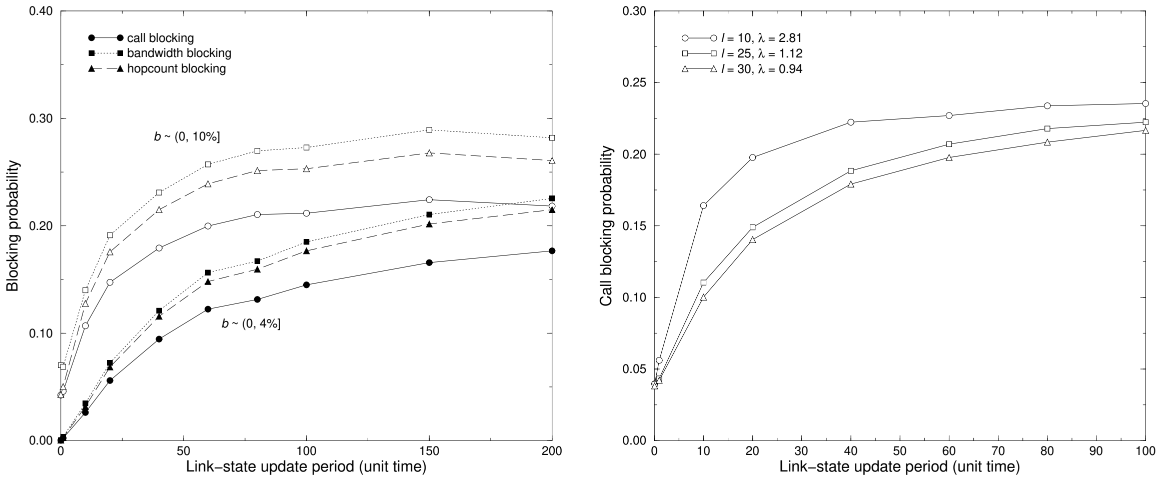
Figure 4: **Link-state fluctuations with periodic updates:** This graph shows link utilization (u_i and u_i') across time for a single link, with an "$\times$" denoting each link-state update; the experiment corresponds to the curve in Figure 3(a) with $b \sim (0.0, 0.1]$ when the update period is 30 time units. An update indicating low utilization causes a rush of traffic to the link, causing the utilization to remain very high until the next update; then, new traffic avoids the link and the utilization drops as existing connections terminate, and the cycle repeats.

continuous updates to very long periods of 100 times the average connection interarrival time[2]. Experiments with different α values, with and without pruning, show similar performance trends. Furthermore, we find that pruning does not significantly affect the blocking probability, even with very stale information, consistent with the results in [17]. Due to fragmentation of link resources [11, 12, 20], 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 curves remain roughly equidistant across the entire graph.

Although QoS routing clearly outperforms static routing for smaller link-state update periods, the blocking probability rises relatively quickly before gradually plateauing for large update periods. In fact, static routing becomes competitive with QoS routing once the update period grows beyond 60 times the average connection interarrival time. In general, though, periodic updates do not respond quickly enough to variations in link state, sometimes allowing substantial changes to go unnoticed. Signalling failures account for all of the call blocking, except when the update period is very small (e.g., for periods close to the arrival rate), as shown in Figure 3(b). 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, however, 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 experi-

[2]The simulator imposes a warm-up period before collecting any final performance statistics. When the measured call blocking probability is within a 99% confidence interval subject to a specified threshold, we begin collecting actual statistics and we terminate the simulation when the new call blocking probability is within a 99% confidence interval subject to a tighter threshold. When evaluating periodic link-state updates, we ensure that the warm-up and data-collection intervals are each at least several times larger than the update period, even if the confidence intervals initially suggest that the simulation has converged.



(a) Blocking metrics vs. update period (b) Blocking for different holding times

Figure 5: **Blocking for different traffic types:** The left graph shows several different blocking metrics to highlight the effects of staleness on connections with different bandwidth requirements and route lengths, with the same simulation parameters as in Figure 3(a). The right graph shows the blocking probability for connections with different mean holding times; the arrival rate λ varies to keep offered load fixed at $\rho = 0.85$, with $b \sim (0, .1]$ and $\alpha = 1$.

ence dramatic fluctuations in link state between successive update messages, as shown in Figure 4. Such link-state flapping has been observed in packet routing networks [26], 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 3(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 topology view. In contrast, under static routing, the source switches blindly route 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 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, further experiments with the non-homogenous MCI backbone topology used in [11,12] show that QoS routing consistently achieves lower blocking probability than static routing over a wide range of update rates.

Fluctuations in link state have a more pernicious effect on connections between distant source-destination pairs, since QoS routing has a large chance of mistakenly selecting 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. Figure 5(a) illustrates this effect by comparing

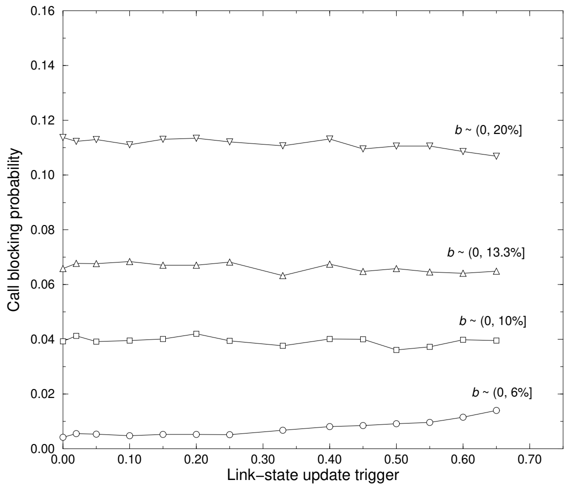


Figure 6: **Blocking insensitivity to triggers:** Call blocking remains fairly constant over a wide range of link-state update triggers. Across the four curves for different bandwidth ranges, the mean connection holding time ℓ is varied to keep the offered load constant at $\rho = 0.85$; here $\lambda = 12.5$ and $\alpha = 1$.

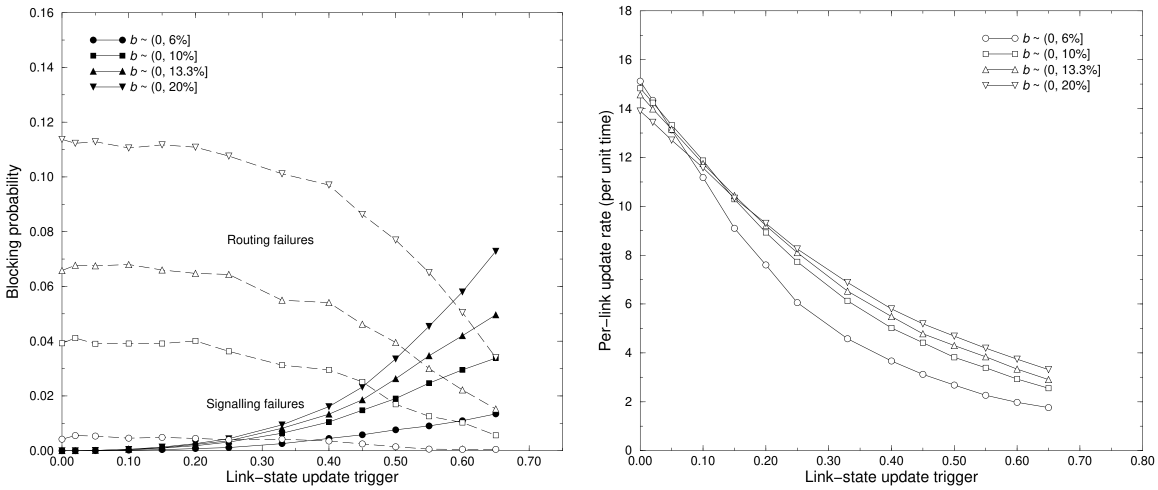
the call blocking probabilities from Figure 3(a) to several alternative measures of blocking. The hop-count blocking probability is defined as the ratio of the hop-count of blocked connections to the hop-count of all connections; bandwidth blocking is defined analogously relative to requested bandwidth[3]. 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-hopcount connections, though link-state staleness does not seem to affect one metric more than the other.

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, Figure 5(b) shows that longer holding times allow the use of larger link-state update periods to achieve the same blocking probability. In particular, 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 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, 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. Using similar simulation parameters as the experiment in Figure 3, the graph in Figure 6 plots the connection blocking probability for a range of triggers and several bandwidth ranges. 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. In conducting further experiments on the asymmetric MCI backbone topology used in [11, 12], we find, again, that

[3]For the hop-count blocking metrics, the number of hops derives from the shortest-path distance between the source and destination nodes, independent of the actual (possibly longer) path selected for the connection.



(a) Routing and signalling failures (b) Link-state update rate vs. trigger

Figure 7: **Connection blocking and link-state updates for different trigger values:** Expanding the simulation results from Figure 6, the left graph shows that a decrease in routing failures compensates for the increase in signalling failures for larger triggers. The right graph shows how the average frequency of link-state update messages decreases as a function of the trigger value.

the blocking probability is not significantly influenced by the trigger level, despite our intuition that staler information should increase the likelihood of blocking new connections.

To understand this phenomenon, consider the two possible effects of stale link-state information on the path-selection process at the source switch. 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 signalling 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 feasible 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 signalling failure. Figure 7(a) highlights this effect by plotting the blocking probability for both routing and signalling failures; the experiments on the MCI topology show similar trends, with the decrease in routing failures compensating for the increase in signalling failures as the trigger grows.

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, the richly connected topology of the 5-ary 3-cube has many available routes 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. Networks with sparser topologies and longer propagation delays, however, may experience more inadvertent routing failures.

Despite the increase in signalling failures, large trigger values substantially reduce the number of update messages for a given blocking probability, as shown in Figure 7(b). 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 15.1 link-state update messages per unit time, which is close to the y-intercept in Figure 7(b); additional experiments show that the link-state update rate is not sensitive to the connection holding times, consistent with the $2\lambda N\bar{h}/L$ expression. In Figure 7(b), the larger bandwidth values have a slightly smaller link-state update rate for small triggers; the higher blocking probability for high-bandwidth connections decreases the proportion of calls that enter the network and generate link-state messages. When triggers are coarse, however, more calls 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.

Although routing failures do not generate link-state updates, signalling failures can trigger updates, since reserving bandwidth as part of the attempt to signal the connection can change the status of upstream links, even if the connection eventually blocks at a downstream link. Hence, the increase in signalling failures in Figure 7(a) serves to slow the reduction in the update rate in Figure 7(b) as the trigger grows. The exact effect of signalling 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 four.

Ultimately, the choice of link-state periods and triggers depends on the relative cost of routing failures, signalling failures, and update messages, as well as the importance of having a predictable link-state update rate. Coarse triggers and large periods can substantially decrease the processing and bandwidth requirements for exchanging information about network load. Still, the benefit of larger triggers and periods must be weighed against the increase in connection blocking, particularly due to more expensive signalling failures. By blocking connections inside the network, signalling failures consume processing resources and delay the establishment of other connections [27]. In addition, a failed connection temporarily holds resources at the upstream links, which may block other connections in the interim [28, 29]. These effects become more important when connection set-up delay is large, relative to connection holding times, since the state of a downstream link may change while the signalling message propagates through the network. In contrast, routing failures are purely local and do not consume any resources beyond the processing capacity at the source switch. These trade-offs suggest a hybrid policy with a moderately large trigger value to provide load-sensitive information when it is most critical, as well as a relatively small hold-down timer to bound the peak link-state update rate without suppressing these important messages.

3.3 Network Topology

The impact of stale link-state information depends on the underlying network topology. Varying the parameters in the k-ary n-cube topology model allows us to examine the relative performance of networks of similar size but different node degree and diameter, as shown in Table 1. A higher dimension (n) typically implies a "richer" topology with more flexibility in selecting routes, whereas a large number of nodes (k), with a fixed n, increases the average length of routes, which requires connections to successfully reserve resources on a larger number of links. These differences between the topologies have a significant influence on how well the QoS-routing algorithm's performance scales with the staleness of link-state information, as shown in Figure 8(a). The graph plots the connection blocking probability over a range of periods, with offered load kept constant between the four configurations by changing the mean holding time.

Under accurate link-state information, the 10-ary 2-cube and 5-ary 3-cube topologies have good performance, despite the longer average distance between pairs of nodes. For small update

Topology	Nodes	Links	Degree	Diameter	Mean path length
10-ary 2-cube	100	400	4	10	5.05
5-ary 3-cube	125	750	6	6	3.63
4-ary 3-cube	64	384	6	6	2.95
5-ary 2-cube	25	100	4	4	2.50

Table 1: **Characteristics of k-ary n-cubes**: This table lists pertinent parameters of four k-ary n-cube topologies that vary in their size and richness of routes.

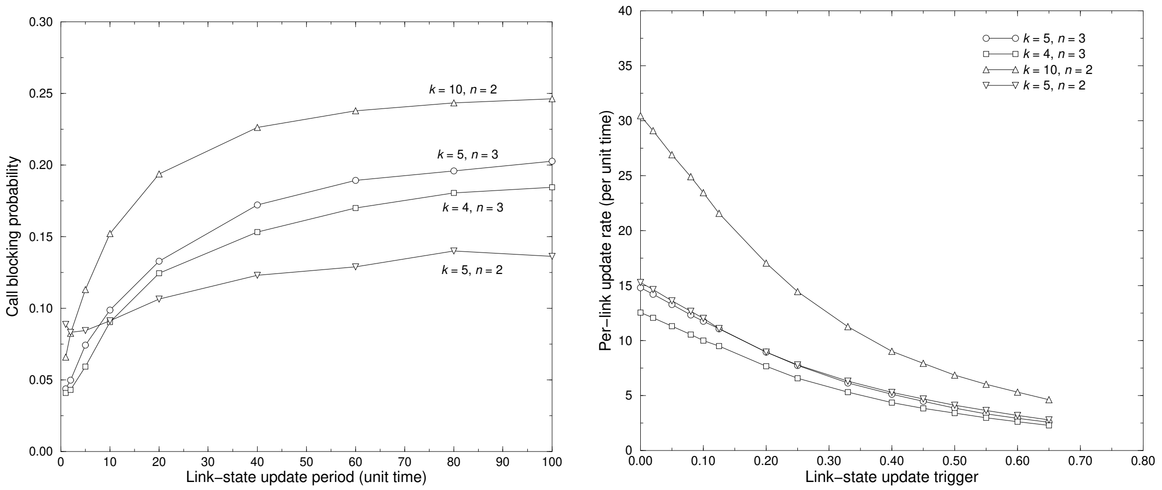
periods, the higher connectivity of the 5-ary 3-cube and 4-ary 3-cube results in a large number of possible routes, which reduces the likelihood of a routing failure. Similarly, the 10-ary 2-cube has a large number of routes, though fewer than the 5-ary 3-cube. However, the performance of these richer topologies degrades more quickly under stale load information. For higher link-state update periods, blocking stems mainly from signalling failures, which are more likely when a connection has a longer path through the network. Once the routing algorithm selects a single path, based on stale information, the new connection can no longer capitalize on the presence of other possible routes. The performance of the 5-ary 2-cube degrades more slowly, since the shorter route lengths increase the chance that the routing algorithm selects a feasible path. Similarly, though they have identical connectivity, the 4-ary 3-cube outperforms the 5-ary 3-cube, due to its smaller average path length.

Direct comparisons between the four topologies are somewhat difficult, due to differences in the number of switches and links. For example, the crossover in Figure 8(a) occurs because the 5-ary 2-cube has a lower average path length, despite the topology's poorer connectivity. Still, varying k and n lends insight into the effects of stale information. Figure 8(b) shows the overheads for triggered link-state updates in the four topologies. Although the 10-ary 2-cube has fewer switches than the 5-ary 3-cube topology, the 10-ary 2-cube generates substantially more link-state update messages than the other two networks. The larger update rate stems from the large path lengths, relative to the number of switches. Interestingly the 5-ary 3-cube and the 5-ary 2-cube have nearly identical link-state update rates. Drawing on the analytic expression from Section 3.2 and the average path length in a k-ary n-cube network, the link-state update rate should be

$$\text{update rate} = \frac{2\lambda N\overline{h}}{L} = \frac{2\lambda k^n \frac{k^2-1}{4k} n \frac{N-1}{N}}{2nk^n} = \frac{\lambda(k^2-1)}{4k}\frac{N-1}{N} \approx \frac{\lambda(k^2-1)}{4k} \approx \frac{\lambda k}{4},$$

for a fine-grain trigger, assuming odd values of k. This update expression is proportional to k and independent of n. A similar expression holds for even values of k.

More generally, a densely-connected topology with a relatively low diameter should trigger fewer link-state updates since connections are routed on shorter paths. The reduction in overhead, however, may be offset by the cost of distributing the update messages. For example, if link-state messages are flooded throughout the network (as in PNNI and OSPF), then each switch receives the message on each incoming link. As a result, each switch receives $2n$ copies of every link-state update. Hence, the advantages of a richer topology are partially overshadowed by the cost of flooding the link-state messages. Also, the experiment in Figure 8(a) shows that stale information limits the benefits of richer connectivity, though the use of update triggers, instead of periods, can mitigate these effects. With a prudent update-distribution mechanism, a richly-connected topology, coupled with a reasonable trigger level, can retain the advantage of having many routing choices and a low link-state update overhead.



(a) Blocking vs. update period (b) Update rate vs. trigger level

Figure 8: **Topology and link-state accuracy:** Richly-connected topologies have low blocking probabilities under accurate information, although the benefits of multiple routes degrade under large update periods, as shown in the left graph. With triggered updates, the rate of link-state messages is proportional to k and independent of n, as shown in the right graph. In both experiments, $\rho = 85\%$, $b \sim (0, 0.1]$, and $\alpha = 2$ (with pruning). The arrival rates in left and right grahps are $\lambda = 1$ and $\lambda = 12.5$, respectively. Load is kept constant across the four topologies by changing $\bar{\ell}$.

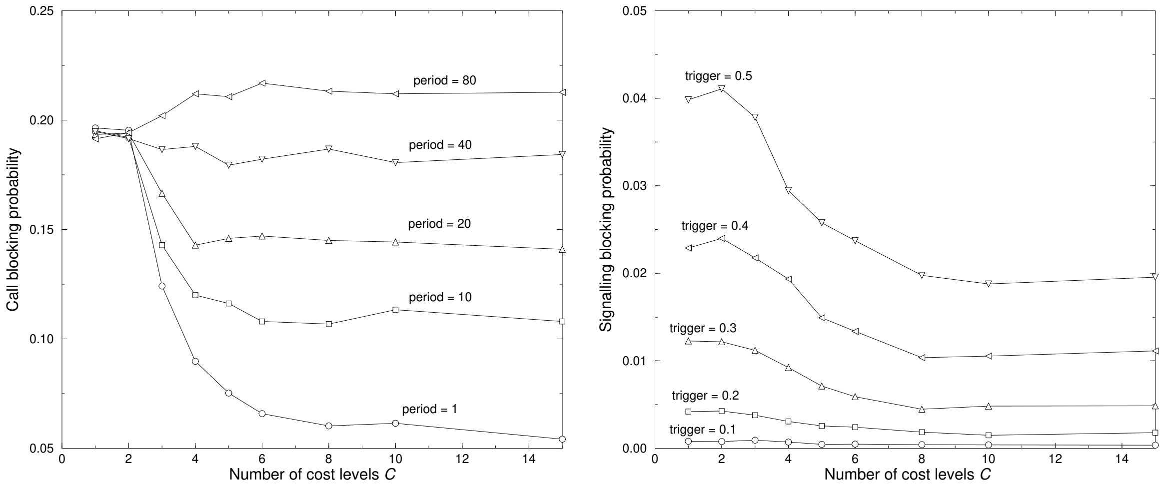
4 Link-Cost Parameters

The link-state update rate also impacts the choice of the link-cost parameters (C and α) in the routing algorithm. Fine-grain cost metrics are much less useful, and can even degrade performance, in the presence of stale link-state information. With a careful selection of the exponent α, the path-selection algorithm can reduce the number of cost levels C without increasing the blocking probability. Smaller values of C reduce the size of the link-state database and lower the computational complexity of the path-selection algorithm, allowing the QoS-routing algorithm to scale to larger network configurations.

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 9(a) evaluates the routing algorithm over a range of cost levels 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 9(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 time units. 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



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

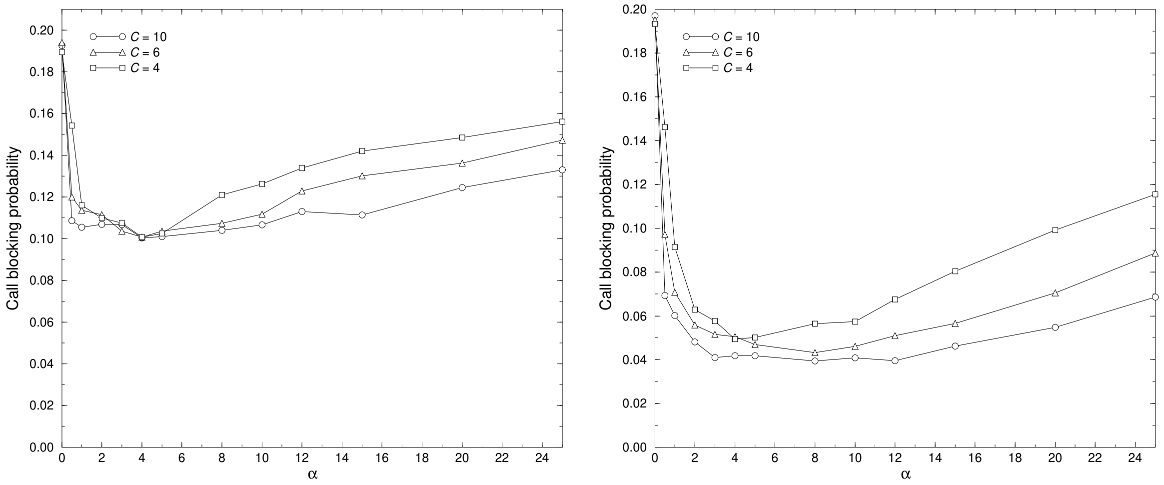
Figure 9: **Discretized costs with stale link-state information:** With periodic updates, connection blocking drops significantly with more cost levels when link-state information is relatively accurate; however, stale information counteracts the benefits of fine-grain costs, as shown in the left graph. Additional cost levels decrease the rate of signalling failures when using triggered updates, as shown in the right graph. Both experiments simulate a 5-ary 3-cube with $\rho = 0.85$, $b \sim (0.0, 0.1]$, $\lambda = 1$, $\ell = 28.1$, and $\alpha = 1$.

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 appropriate number of cost levels depends on the update period and the connection-bandwidth requirements, as well as the overheads for route computation. Ultimately, though, 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 9(b). Since the trigger value does not affect the overall blocking probability, Figure 9(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. A large value of α concentrates most of the cost information in the sensitive, high-utilization region. However, as shown in Figure 10, large values of α can degrade performance after an initial sharp drop due to the transition from static routing ($\alpha = 0$) to QoS routing ($\alpha > 0$). Setting α larger than 4 for periodic updates or 10 for triggered updates increases the blocking rate, particularly for small values of C. If α is too high, some of the cost intervals are so narrow that the arrival or departure of a single connection could change the



(a) Blocking for varying C (period = 10) (b) Blocking for varying C (trigger = 0.4)

Figure 10: **Exponent α with stale link-state information:** A larger exponent α decreases the blocking probability, until α grows so large that the cost intervals become too narrow, as shown in the left graph. Large values of α have a more negative influence on performance under accurate link-state information, as shown in the right graph. Both experiments evaluate a 5-ary 3-cube network with $\rho = 0.85$, $\lambda = 1$, $b \sim (0.0, 0.1]$, and $\ell = 28$.

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.

Small values of C exacerbate this effect, as seen by the higher blocking rates for large values of α in the $C = 4$ and $C = 6$ curves in Figure 10. 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 flatter curves than the plots in Figure 10(a). 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. For triggered updates, experiments with different values of C are consistent with the results in Section 3.2; that is, the blocking probability remains constant over a wide range of triggers.

5 Conclusions and Future Work

The performance and complexity of QoS routing depend on the interaction between a large set of parameters. In this paper we investigated the scaling properties of source-directed link-state routing in large backbone networks. Our simulation results show that the routing algorithm, network topology, link-cost function, and link-state update policy have a significant impact on the probability of successfully routing new connections, as well as the overhead required to distribute network load metrics. In this context, one of our key observations was that stale link-state information introduces a basic trade-off between the number of signalling failures and the frequency of link-state update messages. With periodic updates, reducing the rate of link-state messages increases the blocking probability dramatically and results in a large number of expensive signalling failures. The network can reduce these effects by limiting QoS routing to long-lived connections, while carrying short-lived traffic on static preprovisioned routes.

In contrast, triggering link-state updates on a significant change in available bandwidth does not affect overall blocking, although coarse triggers increase the number of signalling failures. Our findings suggest a combination which uses a relatively coarse trigger with a hold-down timer to mitigate the unpredictable nature of triggered updates when link state fluctuates rapidly. The trade-off between routing and signalling failures also has important implications for the selection of the network topology. Although dense topologies offer more options for choosing good routes, the advantages of multiple short paths dissipate as link-state information becomes more stale. While large, dense topologies have high overheads for distributing load information and computing routes, this 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.

To further investigate QoS-routing dynamics for a range of topology and traffic models, we plan to extend our experiments to a broader class of transit-stub networks [25] and consider more realistic traffic models, such as bimodal bandwidth requests (e.g., audio and video traffic), a mixture of short-lived and long-lived connections, and non-uniform traffic patterns. We also plan to include more detailed models of call set-up delay and propagation delay to gauge the impact of reserving link resources for connections that ultimately fail at a downstream switch, and to study the effects of inaccurate information about links that are further from the source. Also, selecting alternate routes after signalling failures introduces interesting performance trade-offs between connection blocking and set-up delay. Finally, since performance degrades when many connections are routed on nonminimal paths, we plan to evaluate other routing algorithms that incorporate some form of trunk reservation to prevent the overuse of constrained network resources.

In addition to these extensions to our simulation model, we plan to explore new QoS-routing policies that consume less network resources. For example, support for precomputed routes can reduce computation overheads and connection set-up delay [8], yet these policies must reconcile the timescale for route computation with the expected connection arrival and link-state update rates. Some initial simulation experiments with precomputed routes for a few bandwidth classes show promising results [12]. We are especially interested in finding ways to precompute routes, while still incorporating new link-state information in routing decisions. Finally, the intrinsic staleness of link-state information suggests new policies for alternate routing after a signalling failure. For example, introducing a small delay before trying an alternate route for a connection [29, 30] is particularly appropriate when the arrival of fresh link-state information has the potential to improve the routing decision.

References

[1] J. J. Bae and T. Suda, "Survey of traffic control schemes and protocols in ATM networks," *Proceedings of the IEEE*, vol. 79, pp. 170–189, February 1991.

[2] D. Towsley, "Providing quality of service in packet switched networks," in *Performance Evaluation of Computer and Communication Systems*, pp. 560–586, Springer Verlag, 1993.

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

[4] Z. Whang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1228–1234, September 1996.

[5] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the Internet." Internet Draft (draft-ietf-qosr-framework-00.txt), March 1997.

[6] PNNI Working Group, *ATM Forum 94-0471R13 PNNI Draft Specification.* Document available at ftp://ftp.atmforum.com/pub/contributions.

[7] 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), September 1997.

[8] R. Guerin, S. Kamat, and A. Orda, "QoS routing mechanisms and OSPF extensions." Internet Draft (draft-guerin-qos-routing-ospf-01.txt), March 1997. To appear in *Proceedings of IEEE GLOBECOM*, November 1997.

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

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

[11] Q. Ma and P. Steenkiste, "Quality-of-service routing for traffic with performance guarantees," in *Proc. IFIP International Workshop on Quality of Service*, (Columbia University, New York), pp. 115–126, May 1997.

[12] 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.

[13] M. Peyravian and R. Onvural, "Algorithm for efficient generation of link-state updates in ATM networks," *Computer Networks and ISDN Systems*, vol. 29, pp. 237–247, January 1997.

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

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

[16] R. Gawlick, C. Kalmanek, and K. Ramakrishnan, "Online routing for virtual private networks," *Computer Communications*, vol. 19, pp. 235–244, March 1996.

[17] I. Matta and A. U. Shankar, "Dynamic routing of real-time virtual circuits," in *Proceedings of IEEE International Conference on Network Protocols*, (Columbus, OH), pp. 132–139, 1996.

[18] 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.

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

[20] H. Ahmadi, J. S. Chen, and R. Guerin, "Dynamic routing and call control in high-speed integrated networks," in *Teletraffic and Datatraffic in a Period of Change: Proceedings of the International Teletraffic Congress* (A. Jensen and V. B. Iversen, eds.), vol. 14 of *Studies in Telecommunication*, pp. 397–403, Copenhagen, Denmark: North-Holland, June 1991.

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

[22] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest-path algorithms: Theory and experimental evaluation," *Mathematical Programming*, vol. 73, pp. 129–174, May 1996.

[23] S. Plotkin, "Competitive routing of virtual circuits in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1128–1136, August 1995.

[24] S. Floyd and V. Jacobson, "Synchronization of periodic routing messages," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 122–136, April 1994.

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

[26] A. Khanna and J. Zinky, "The revised ARPANET routing metric," in *Proceedings of ACM SIGCOMM*, (Austin, TX), pp. 45–56, 1989.

[27] E. Gelenbe, S. Kotia, and D. Krauss, "Call establishment overload in ATM networks," *Performance Evaluation*, 1997.

[28] R.-H. Hwang, J. Kurose, and D. Towsley, "On call processing delay in high speed networks," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 628–639, December 1995.

[29] D. J. Mitzel, D. Estrin, S. Shenker, and L. Zhang, "A study of reservation dynamics in integrated services packet networks," in *Proceedings of IEEE INFOCOM*, pp. 871–879, April 1996.

[30] A. Feldmann, "Impact of non-poisson arrival sequences for call admision algorithms with and without delay," in *Proceedings of IEEE GLOBECOM*, pp. 617–622, November 1996.