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

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 satisfies application performance requirements and optimizes 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 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. In contrast to previous work that compares different routing algorithms under specific network configurations, we construct a detailed model of QoS routing that parameterizes the path-selection algorithm, link-cost function, and link-state update policy. Through extensive simulation experiments with several representative network topologies and traffic patterns, we uncover the effects of stale link-state information and random fluctuations in traffic load on the routing and signalling overheads. We then investigate how the inaccuracy of link-state information interacts with the size and connectivity of the underlying topology. Finally, we show that by tuning the coarseness of the link-cost metric to the inaccuracy of underlying link-state information we can reduce the computational complexity of the path-selection algorithm without significantly degrading performance. 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.

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

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

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

for each flow or connection to satisfy diverse performance requirements and optimize resource usage [1–3]. 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 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 [4] 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 [5, 6]. 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 the scalability of these protocols 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 [7]. To characterize the effects of stale information, comparisons of different QoS-routing algorithms have included simulation experiments that vary the link-state update period [8–10], while other work considers a combination of periodic and triggered updates [11]. In particular, the work in [12] evaluates several variants of triggered updates coupled with hold-down timers. 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 [6, 10]; 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 [8–11, 13–18], 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. 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 large backbone networks. Table 1 summarizes the parameters in our model.

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

Category	Parameters
Path selection	Pruning/not-pruning, cost levels C, and exponent α
Link-state updates	Period, trigger, and hold-down timer
Network topology	Random graph, regular topology, or MCI backbone
Connection characteristics	Uniform random bandwidth b, Pareto or exponential duration ℓ
Traffic pattern	Poisson or Weibull arrivals λ, uniform random destinations
Performance metrics	Blocking, routing vs. set-up failures, and link-state update rate

Table 1: **QoS routing model:** This table summarizes the key parameters in our performance model of QoS routing under stale link-state information.

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.

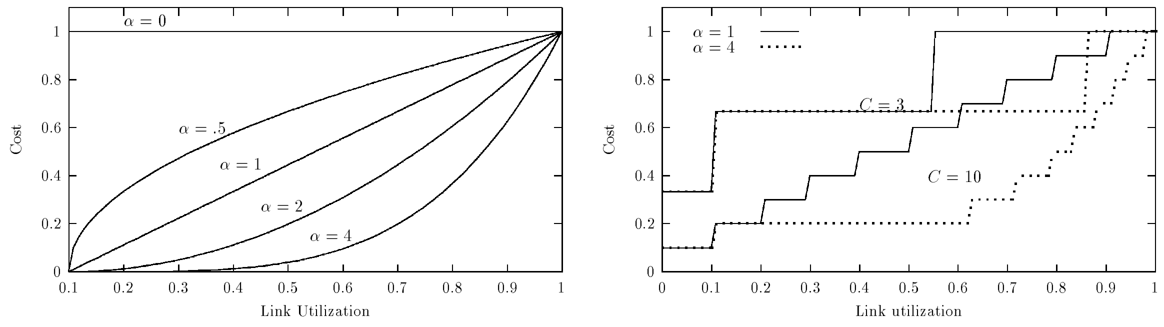
Although networks can employ a wide variety of QoS routing strategies, previous comparative studies have demonstrated that algorithms with a strong preference for minimum-hop routes almost always outperform algorithms that do not consider path length [9, 14–17, 19]. For example, selecting the widest 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 [14].

In our model, the source selects a route based on the bandwidth requirement b and the destination node in three steps:

1. (Optionally) 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 $\sum_i c_i'$.

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 [20] to compute a the shortest path with the smallest total cost[1]. The Dijkstra shortest-path calculation has $O(L \log N)$

[1]A careful assignment of link weights w_i permits a single invocation of the Dijkstra algorithm to produce a shortest-



(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.

complexity when implemented with a binary heap. Although advanced data structures can reduce the average and worst-case complexity [21], 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 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 utilizations. Large values of α loosely correspond to widest shortest-path routing, since the large exponent virtually eliminates heavily-loaded links from consideration. We represent link cost with 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 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.

path route with the minimum cost (in terms of link costs c_i). 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 $h \leq N$.

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 the connection 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 durations. 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 [8–10, 16], 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 update messages for different links, each link introduces a small random component to the generation of successive updates [22]. 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

$$\Delta_i = \frac{|u'_i - u_i|}{1 - u'_i}.$$

Topology	Nodes	Links	Degree	Diameter	Mean path length
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 2: **Topologies used in experiments**: This table lists pertinent parameters of the three topologies considered in our experiments. The random graph is an instance generated using Waxman's model [26]. In the regular topology all nodes have identical connectivity.

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 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

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 a "typical" configuration [23]. 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 [24]. Consequently, our simulation experiments consider a range of network topologies with differences in important parameters such as average path length, number of equal-hop paths between nodes, and network diameter. 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 [24, 25], 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 [9, 10]), 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. The MCI and random graphs in general have relatively few (if any) multiple equal-hop paths between nodes but paths are shorter in the smaller MCI topology. The regular topology has significantly more equal-hop paths but at the expense of having more links and, consequently, higher link-state update distribution overhead. We further assume that the topology remains fixed throughout each simulation experiment; that is, we do not model the effects of link failures.

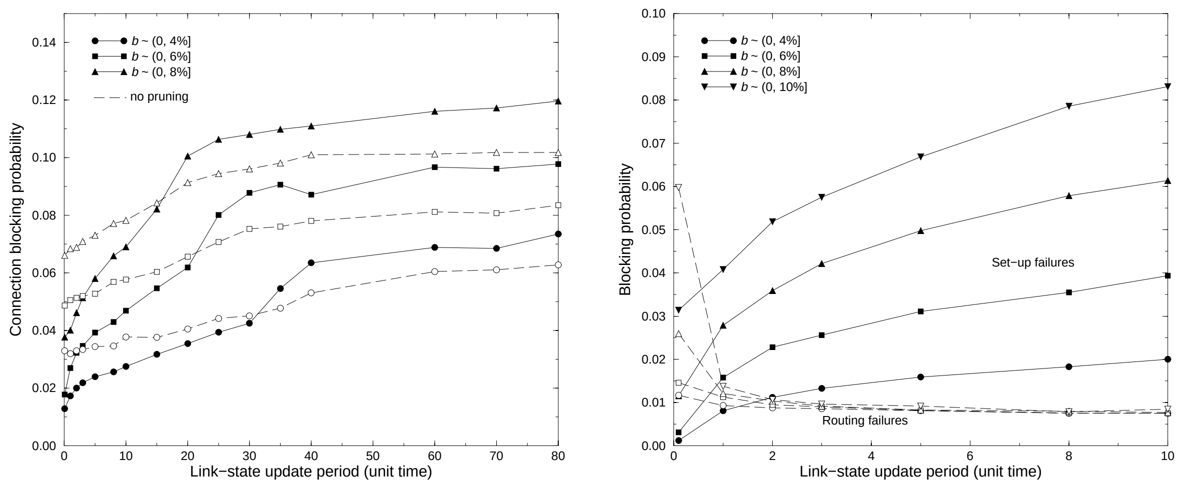
Each node generates connection requests according to a Poisson process with rate λ, with uniform random selection of destination nodes. This results in a uniform traffic pattern in the regular graphs, and a non-uniform pattern on the MCI and random topologies, allowing us to compare QoS routing to static shortest-path routing under balanced and unbalanced loads. In addition, we consider the effect of bursty arrivals where connection interarrival times follow a Weibull distribution [27]. We model connection durations using a Pareto distribution with shape parameter $a = 2.5$ to capture the long-tailed nature of connection durations[2] [23] 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 durations. We denote the mean duration as ℓ. Connection bandwidths are uniformly-distributed within an interval with a spread about the mean $\bar{b}$. 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.

2.6 Performance Metrics

Since we are interested in understanding how performance and overhead scale under stale link-state information, we avoid translating the parameters in our model to specific units. This allows us to decouple the traffic model from assumptions about signalling capacity at network routers or about the type of traffic (e.g., how short- or long-lived) that is assigned to QoS routes. Most network resource requirements are affected by two or more of our performance metrics, where the relative contribution depends on the specific configuration. For example, CPU load is affected by both set-up failures and link-state update messages, where the exact cost depends on the details of the protocol and the implementation. Our approach allows us to locate the operating regions where QoS routing can provide added benefit under reasonable overheads of signalling and link-state updates. For example, the results in Section 3.1 suggest that link-state overhead can be reduced significantly by detecting and assigning only long-lived traffic to QoS routes and tuning the update period accordingly; in contrast, short-lived packets can be routed on static paths with separate, preprovisioned resources.

Similarly, we report link-state update overhead in terms of the number of updates generated per link per unit time. While it may be appealing to express this instead as bytes per second or as a fraction of network capacity, the actual overhead is dependent on the link speed, the routing protocol, and the mechanisms for exchanging link state. For example, a QOSPF advertisement for a single link requires approximately 50 bytes [5], whereas a PNNI topology state packet may require up to a few hundred bytes depending on the resource advertisement format and bundling of updates [4]. In plotting unitless metrics for overhead, we are able to examine the general scaling trends while still permitting estimation of actual overhead for a particular network and protocol configuration. To characterize the performance under stale information, we measure the proportion of connection requests that experience routing or set-up failures.

[2]We use a standard form of the Pareto distribution with shape parameter a, scale parameter β, and cumulative distribution function $F_X(x) = 1 - (\beta/x)^a$.



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

Figure 2: **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 when link pruning is disabled. The right graph focuses on small periods and routing with pruning to show that the blocking is dominated by set-up failures after only a small increase in the period. Both graphs show results for the random topology with $\lambda = 1$, $\alpha = 1$, and $\rho = .75$.

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 set-up failures. The experiments also show that this shift between routing and set-up 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 2(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[3]. 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

[3]The simulator imposes a warm-up period before collecting any final performance statistics. When the measured connection 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 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.

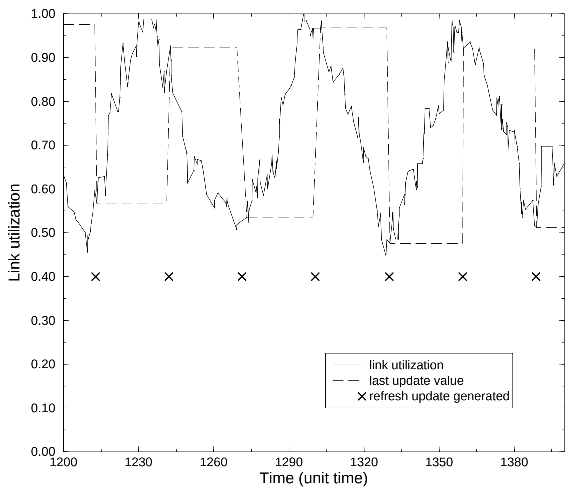
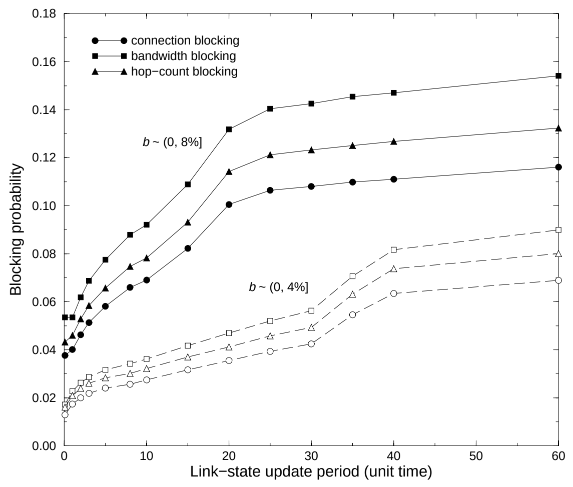


Figure 3: **Link-state fluctuations with periodic updates:** This graph shows link utilization (u_i and u'_i) across time for a single link, with an "×" denoting each link-state update; the experiment corresponds to the curve in Figure 2(a) with $b \sim (0.0, 0.06]$ when the update period is 30 times the connection interarrival time. 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.

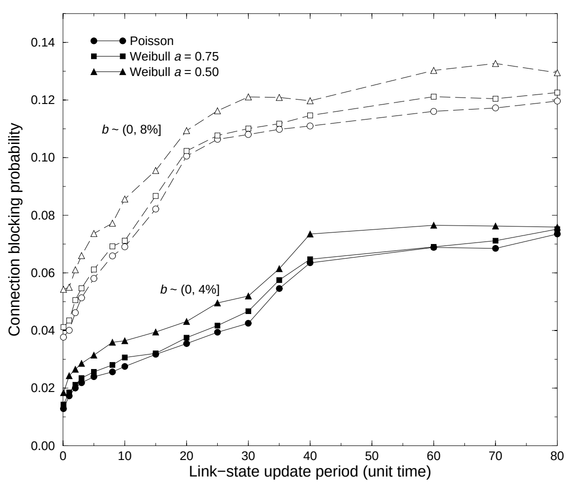
rates. The blocking probability for high-bandwidth connections, while higher, does not appear to grow more steeply as a function of the update period; instead, the three sets of curves remain roughly equidistant across the range of update periods.

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 2(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 2(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 [28].

Route flapping: Although QoS routing performs well for small link-state update periods (significantly outperforming static routing [29]), the blocking probability rises relatively quickly before gradually plateauing for large update periods. In Figure 2(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 2(b) which focuses on a small region of the experiments with pruning in Figure 2(a); when pruning is disabled, routing failures never occur, and set-up failures account for all blocked connections. In



(a) Blocking metrics vs. update period



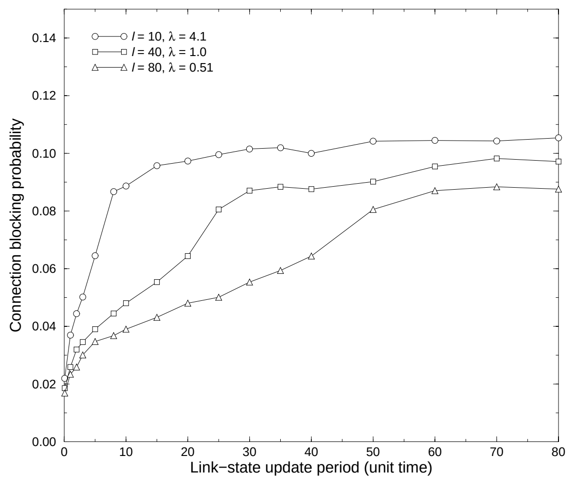
(b) Blocking for varying arrival burstiness

Figure 4: **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 2(a). The right graph shows the blocking probability for varying burstiness in the arrival process. Burstiness is increased with a smaller Weibull shape parameter, a.

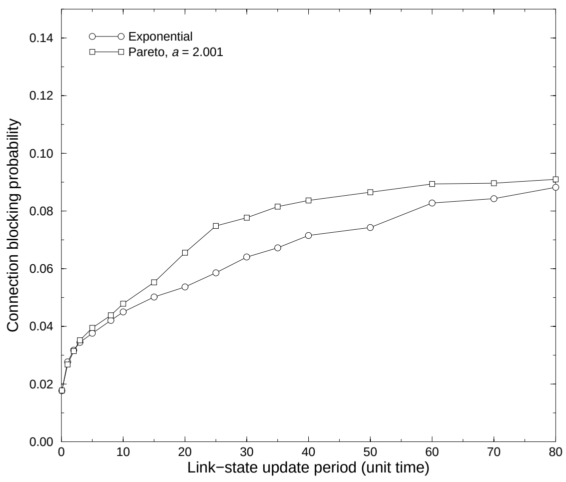
general, periodic updates do not respond quickly enough to variations in link state, sometimes allowing substantial changes to go unnoticed. This suggests that inaccuracy in the link-state database causes the source switch to mistake infeasible links as feasible; hence, the source selects an infeasible 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, as shown in Figure 3. Such link-state flapping has been observed in packet routing networks [30], 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 2(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 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



(a) Blocking for different mean durations



(b) Blocking for different duration distributions

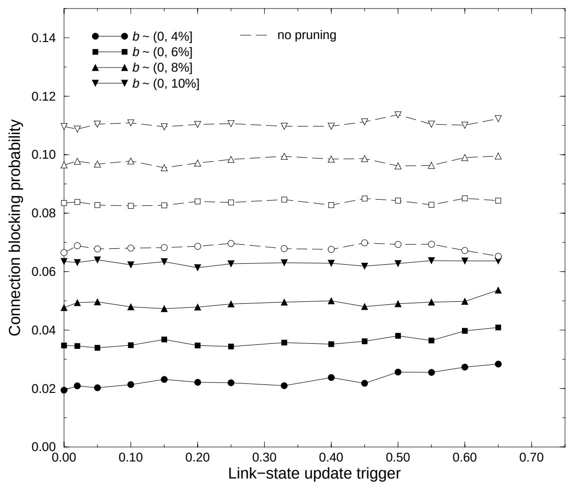
Figure 5: **Effects of connection duration:** The left graph shows the blocking probability for connections with different mean duration; the arrival rate λ varies to keep offered load fixed at $\rho = 0.75$ in the random topology, with $b \sim (0, .06]$, $\alpha = 1$, and pruning enabled. In (b), the mean duration is fixed at $\ell = 40$, and we compare exponentially distributed connection durations to a long-tailed Pareto distribution.

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.

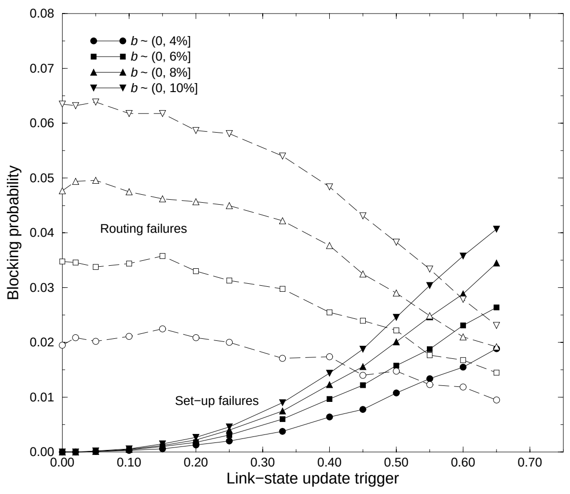
Path length, high-bandwidth requests, and non-Poisson arrivals: Fluctuations in link state have a more pernicious effect on connections between distant source-destination pairs, since QoS routing has a larger chance of mistakenly selecting a path with at least one heavily-loaded link. This is especially true when links do not report their new state at the same time, due to skews in the update periods at different switches. Figure 4(a) illustrates this effect by comparing the connection blocking probabilities from Figure 2(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[4]. 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. Figure 4(b) shows that increased burstiness in the arrival process also increases blocking probability over the range of update periods. For higher bandwidth connections, the effect of burstiness is exacerbated by the nonminimal routes introduced by pruning. Thus, even for the smallest update period, the blocking for bursty traffic is significantly higher than for non-bursty traffic. The effect is not so pronounced for lower-bandwidth connections since they consume fewer resources even when allowed to take nonminimal routes.

Connection durations: 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(a) shows that longer durations allow the use of larger link-state update periods to achieve the same blocking probability. Short-lived connections cause link-state to fluctuate rapidly, particularly for high-bandwidth requests, and thus require frequent

[4]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) Connection blocking vs. update trigger



(b) Routing and set-up failures (with pruning)

Figure 6: **Blocking insensitivity to triggers:** Connection blocking remains fairly constant over a wide range of link-state update triggers, with and without pruning enabled. Across the four curves for different bandwidth ranges, the mean connection holding time ℓ is varied to keep the offered load constant. The experiments evaluate the MCI topology with $\rho = 0.70$, $\lambda = 1$, and $\alpha = 1$.

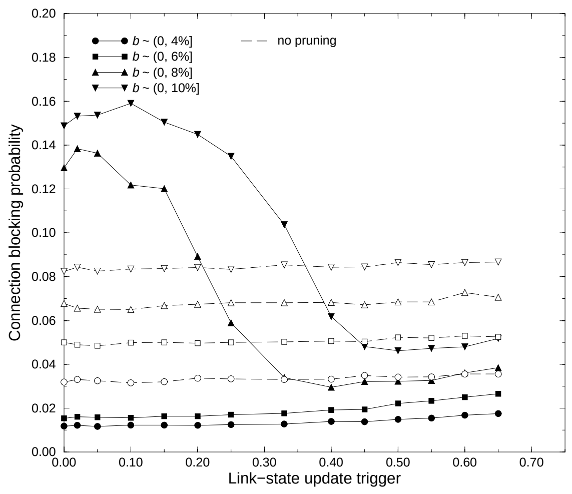
updates to maintain good routing performance.

In Figure 5(b), we find that exponentially distributed connection durations produce a more gradual rise in blocking probability over the same range of update periods (nearly half as fast for some mean holding times) than with the Pareto distribution. The long-tailed Pareto distribution introduces more overall variability in the network load by creating some very long-lived connections (relative to the mean). The increased load fluctuation decreases the effectiveness of periodic link-state updates in identifying feasible paths. The heavier tail of the Pareto distribution also results in 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.

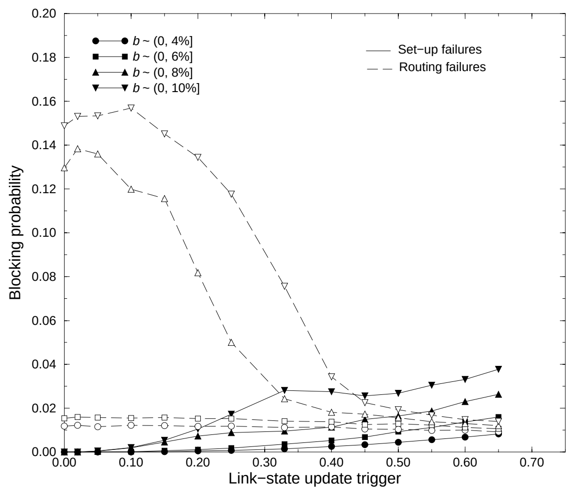
These results suggest that the network could limit QoS routing to the longer-lived traffic that would consume excessive link resources if not routed carefully, while relegating short-lived traffic to preprovisioned static routes. With some logical separation of resources for short-lived and long-lived 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 [31, 32], 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 6(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 [29].



(a) Connection blocking vs. update trigger



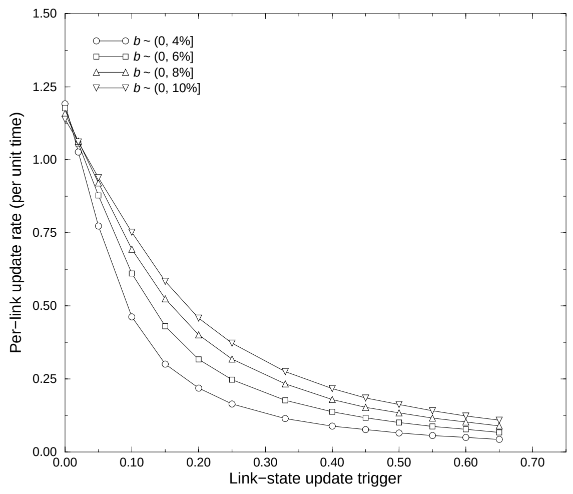
(b) Routing and set-up failures (with pruning)

Figure 7: **Blocking probability with triggers in random topology:** In the larger randomly connected topology, very high bandwidth connections suffer significant blocking when link-state information is relatively accurate, due to circuitous routes introduced by pruning. For lower bandwidth requests and with pruning enabled, though, we observe trends similar to those in Figure 6. In these experiments $\rho = 0.75$, $\lambda = 1$, and $\alpha = 1$.

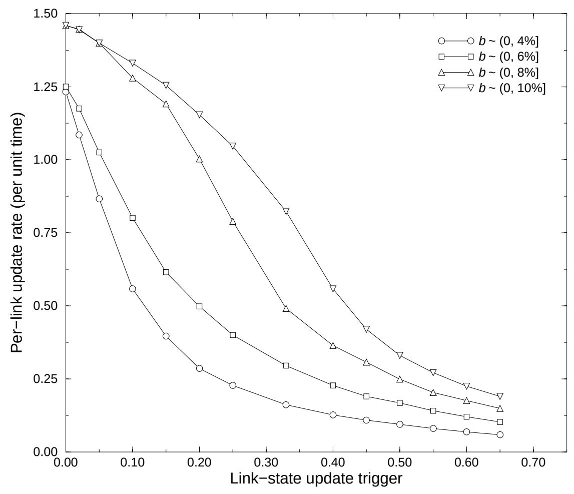
Blocking insensitivity to update trigger: To understand this phenomenon, consider the two possible effects of stale link-state information on the path-selection process when pruning is enabled. Staleness can cause infeasible links to appear feasible, or cause the switch to dismiss links as infeasible 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 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 set-up failure. Figure 6(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 (not shown) 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, Figure 7 shows that the random topology has *higher* blocking rates with *smaller* trigger values when trying to route high-bandwidth connections. Unlike the other two topologies, the random graph typically does not have multiple equal-length



(a) Pruning disabled



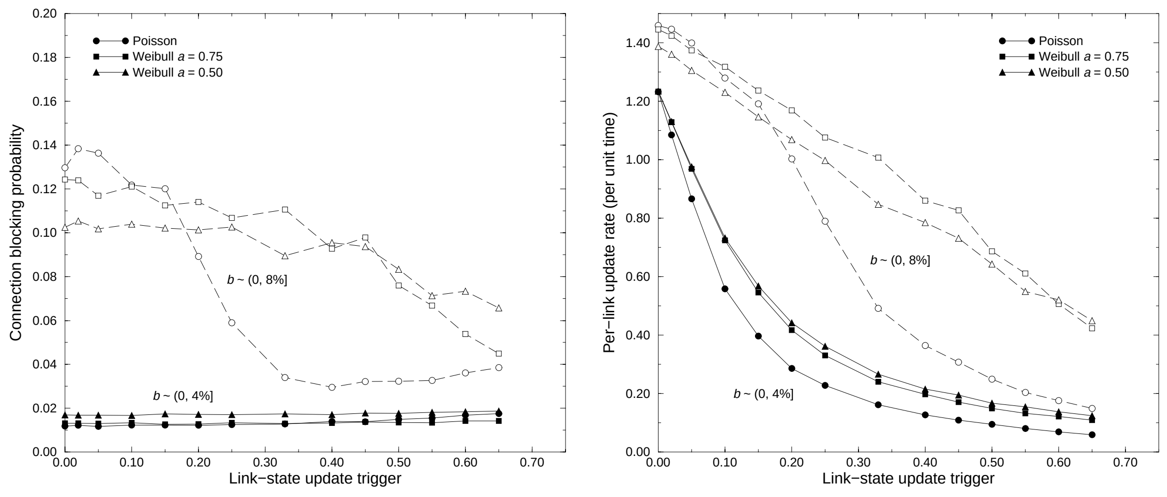
(b) Pruning enabled

Figure 8: **Link-state update rate for different trigger values:** These graphs show the link-state update rate for the random topology with the same parameters as Figure 7. The average rate of link-state update messages decreases as a function of the trigger value.

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.

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 8(a). 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 8(a); 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 8(a), the larger bandwidth values have a slightly smaller 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 generates changes in link state, even if the connection ultimately blocks at a downstream node. The increase in set-up failures for larger triggers slows the reduction in the update rate in Figure 8(a) 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, as shown in Figure 8(b). Overall, though, modest trigger values



(a) Connection blocking (b) Link-state update rate

Figure 9: **Bursty arrivals with triggered updates:** These graphs show the effect of increased burstiness with the same mean arrival rate when using triggered link-state updates. Smaller Weibull shape parameters result in more burstiness in the arrival process. Simulation parameters are the same as those in Figure 7.

are effective at reducing the link-state update rate by about a factor of three to four. Also, for a fixed update rate, triggers are able to 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.

Impact of non-Poisson arrivals: Figure 9(a) further illustrates the problem with nonminimal routes when connection requests arrive in bursts. For both Poisson and non-Poisson arrivals of high-bandwidth requests, blocking is higher when the trigger is smaller. When link-state information becomes more inaccurate, however, the throttling effect is evident for Poisson arrivals, but not for the non-Poisson traffic. When a source chooses nonminimal routes for groups of requests, the network is not able to sufficiently recover from poor allocation of resources. As a result, throttling does not have an effect except for very large triggers, as shown by the gradual decline in blocking probability relative to Poisson arrivals. When pruning is not permitted, we find that blocking is insensitive to the update trigger, but bursty arrivals suffer a higher blocking probability relative to Poisson traffic.

A burst of connection requests may be thought of as a single, high-bandwidth request that, when signalled, results in more link-state fluctuation relative to non-bursty traffic. Figure 9(b) illustrates this through a comparison of the link-state update rate for bursty and non-bursty arrivals. With small update triggers, the bursty traffic has a lower update rate due simply to its higher blocking probability, particularly for the higher-bandwidth requests. That is, fewer connections are admitted to the network and thus link-state triggers are not set off as often. When link-state triggers become coarse, however, the update rate for bursty traffic remains high due to a combination of a higher number of nonminimal routes and increased link-state fluctuations.

In general, bursty traffic increases the blocking due to routing failures, and it remains high even as the update trigger is increased. That is, larger triggers do not shift blocking to set-up failures as in Figure 6. The poor resource allocation caused by bursty arrivals hampers the ability of the source to find feasible routes, especially in the random topology where the probability of having multiple equal-length paths is low. We find, for example, that routing failures remain high

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 3: **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.

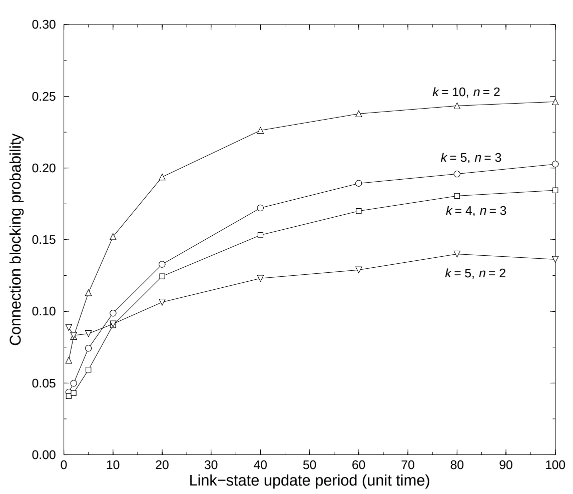
in the random topology as the update trigger increases, but they decline somewhat in the uniform topology where multiple equal-length routes are available. It is generally unwise to apply pruning for high-bandwidth connections when the topology does not have multiple routes of equal (or similar) length. The detrimental effect of nonminimal routes may be limited, however, by explicitly controlling the degree of nonminimality (e.g., at most one extra hop) rather than disabling pruning altogether.

Ultimately, the choice of link-state periods and triggers depends on the relative cost of routing failures, set-up 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. But the benefit of larger triggers and periods must be weighed against the increase in connection blocking, particularly due to more expensive set-up failures. By blocking connections inside the network, set-up failures consume processing resources and delay the establishment of other connections [33]. In addition, a failed connection temporarily holds resources at the upstream links, which may block other connections in the interim [34, 35]. 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. For example, for the experiment shown in Figure 8(b), imposing a hold-down timer two times the connection interarrival time reduces the number of link-state updates by nearly a factor of 3 for fine-grained triggers (around 5%). With coarser triggers, the hold-down timer still reduces the update rate but the decrease is not as dramatic.

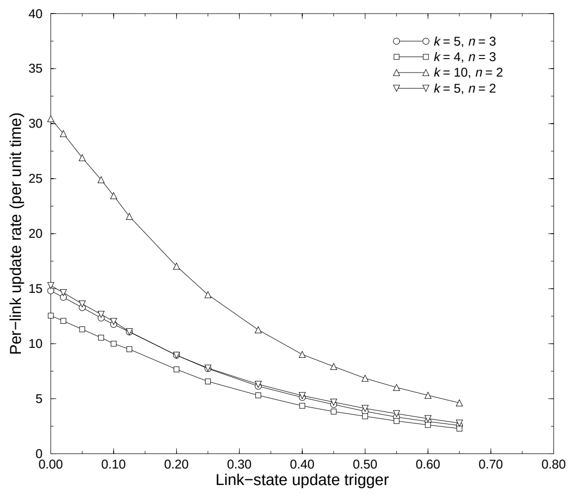
3.3 Network Topology

The impact of stale link-state information depends on the underlying network topology. To study the effects of staleness under a range of topologies, we evaluate a set of regular graphs with similar size and different degrees of connectivity under the same uniform traffic load. The experiments focus on the class of k-ary n-cube graphs with k nodes along each of n dimensions ($N = k^n$ nodes of degree $2n$, with $L = 2nk^n$ links and diameter $D = \lfloor k/2 \rfloor n$), as shown in Table 3. 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 10(a). The graph plots the connection blocking probability over a range of update periods, with offered load kept constant between the four configurations by changing the mean (exponentially distributed) 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 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) Blocking vs. update period



(b) Update rate vs. trigger level

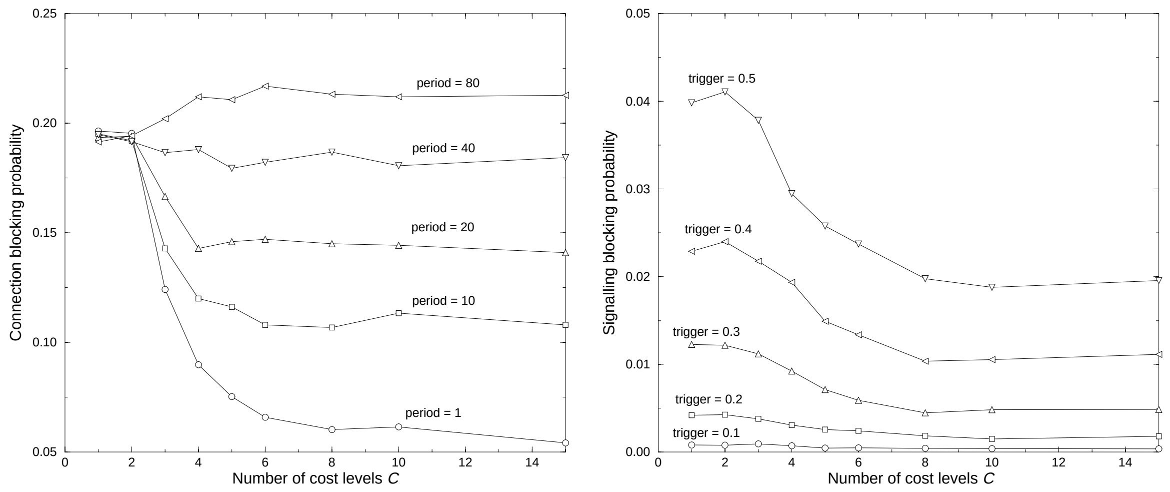
Figure 10: **Topology and link-state accuracy:** Richly-connected topologies have low blocking probabilities under accurate information, although the benefits of multiple routes degrade under large update periods, as shown in the left graph. With triggered updates, the rate of link-state messages is 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 graphs are $\lambda = 1$ and $\lambda = 12.5$, respectively. Load is kept constant across the four topologies by changing ℓ.

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 longer 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 10(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 10(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. In fact, the update rate is approximately $\lambda k/4$ across all of the k-ary n-cube topologies[5]. Increasing the network dimension n corresponds to growing the underlying network in a manner that increase the average path length in proportion to the increase in the number of links.

More generally, a densely-connected topology with a relatively low diameter should trigger

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



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

Figure 11: **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 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 $\alpha = 1$.

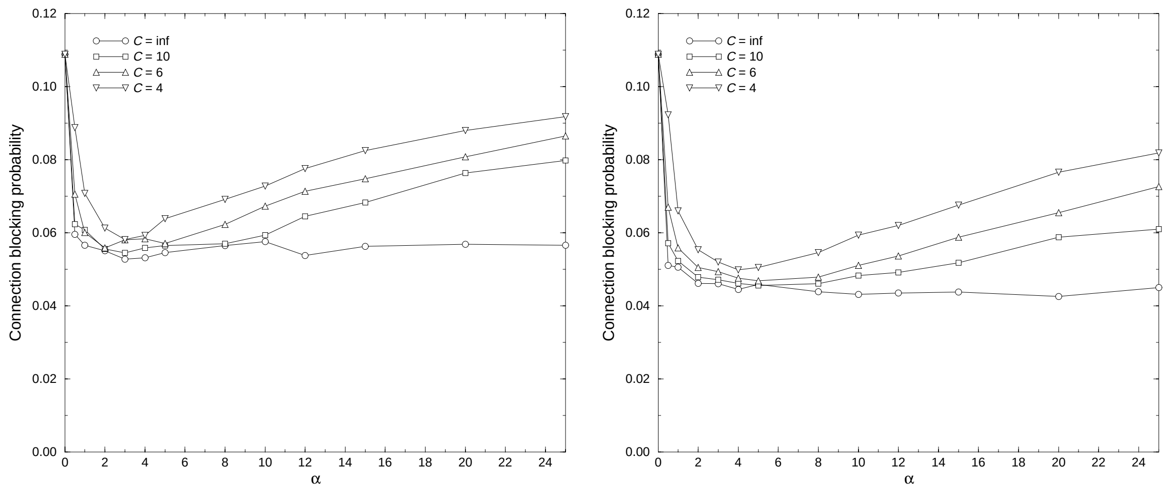
fewer link-state updates since connections are routed on shorter paths. The reduction in overhead, however, may be offset by the cost of distributing the update messages. For example, if link-state messages are flooded throughout the network (as in PNNI and OSPF), then each switch receives the message 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 10(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 an efficient update-distribution mechanism (such as spanning trees, instead of a flooding protocol), 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.

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 space and time complexity of the route computation, allowing the QoS-routing algorithm to scale to larger network configurations. In addition, coarse-grain link costs increase the likelihood of having multiple minimum-cost routes, which allow the network to balance load across alternate routes.

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 11(a)



(a) Blocking for varying C (period = 10) (b) Blocking for varying C (trigger = 0.2)

Figure 12: **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 the random topology with $\rho = 0.75$, $\lambda = 1$, $b \sim (0.0, 0.06]$, $\ell = 40.6$, and pruning disabled.

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 11(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 11(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 11(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), the exponent α does not have a significant impact on performance; values of $\alpha \geq 1$ have nearly identical performance, as shown by the "$C = \infty$" curve in Figure 12(a). 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 has important implications 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, as shown by the other curves in Figure 12. Each plot shows 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. 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 (not shown) 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 flatter curves than the plots in Figure 12(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.

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. Below we categorize our key observations into three areas.

Our key observations regarding QoS routing and update policies 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.

- **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.

Different types of network traffic and QoS requests exhibit significant effects on QoS routing performance and overheads:

- **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.

- **Connection durations:** Longer connection durations 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 distributions, due to the relatively large number of short-lived connections for the same average duration and the additional variability introduced in network load, compared to exponentially distributed durations. 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 short- and long-lived traffic by partitioning link bandwidth for the two classes, and detecting long-lived connections at the edge of the network [32].

- **Connection arrivals:** Bursts of connection requests behave like single arrivals of very high-bandwidth connections, causing higher fluctuation in link-state and greater susceptibility to poor resource allocation. When uncontrolled pruning is enabled, routers choose significantly more nonminimal paths relative to non-bursty traffic, increasing blocking probability and link-state update rate. Effects of bursty arrivals are especially harmful in topologies with a limited number of equal-hop routes.

The network configuration, including routing algorithm parameters, also play an important role:

- **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 [36].

- **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.

References

[1] 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.

[2] 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.

[3] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the Internet." Internet Draft (draft-ietf-qosr-framework-04.txt), work in progress, April 1998.

[4] PNNI Specification Working Group, *Private Network-Network Interface Specification Version 1.0.* ATM Forum, March 1996. Document available at ftp://ftp.atmforum.com/pub/approved-specs/af-pnni-0055.000.

[5] 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.

[6] 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, January 1998. Revised version appeared in *Proceedings of IEEE GLOBECOM*, November 1997.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Quality-of-service based routing: A performance perspective," in *Proceedings of ACM SIGCOMM*, (Vancouver, Canada), pp. 17–28, August 1998.

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

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

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

[23] V. Paxson and S. Floyd, "Why we don't know how to simulate the Internet," in *Proceedings of the Winter Simulation Conference*, (Atlanta, GA), December 1997.

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

[25] A. G. Greenberg and R. Srikant, "Computational techniques for accurate performance evaluation of multirate, multihop communication networks," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 266–277, April 1997.

[26] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, December 1988.

[27] 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.

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

[29] A. Shaikh, J. Rexford, and K. G. Shin, "Dynamics of quality-of-service routing with inaccurate link-state information," Tech. Rep. CSE-TR-350-97, Computer Science and Engineering, University of Michigan, November 1997. (available at `http://www.eecs.umich.edu/~ashaikh/research/`).

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

[31] 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*, vol. 13, pp. 1481–1494, October 1995.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.