Evaluating the Impact of Stale Link State on Quality-of-Service Routing

Anees Shaikh, *Member, IEEE*, Jennifer Rexford, *Member, IEEE*, and Kang G. Shin, *Fellow, IEEE*

***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, distributing link state imposes significant bandwidth and processing overhead on the network. This paper investigates the performance tradeoff between protocol overhead and the quality of the routing decisions in the context of the source-directed link-state routing protocols proposed for IP and ATM networks. 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 network topologies and traffic patterns, we uncover the effects of stale link-state information and random fluctuations in traffic load on the routing and setup overheads. We then investigate how inaccuracy of link-state information interacts with the size and connectivity of the underlying topology. Finally, we show that tuning the coarseness of the link-cost metric to the inaccuracy of underlying link-state information reduces the computational complexity of the path-selection algorithm without significantly degrading performance. This work confirms and extends earlier studies, and offers new insights for designing efficient quality-of-service routing policies in large networks.**

***Index Terms*—Explicit routing, link-state, modeling, quality-of-service, signaling, source-directed routing.**

I. 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 signaling protocols control network dynamics by directing traffic at the flow or connection level. QoS routing selects a path for each flow or connection to satisfy diverse performance requirements and optimize resource usage [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 tradeoff 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 private network–network interface (PNNI) standard [4] defines a routing protocol for distributing topology and load information throughout the network, and a signaling protocol for processing and forwarding connection establishment requests from the source. Similarly, proposed QoS extensions to the open-shortest-path-first (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. Computational overhead depends on the complexity of the routing algorithm and the frequency of route computation, whereas protocol overhead depends on the frequency of 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, imposing a minimum time between update messages avoids 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. Tuning the frequency of link-state update messages requires a careful understanding of the tradeoff between network overheads and the accuracy of routing decisions.

Manuscript received December 7, 1998; revised September 27, 2000; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor E. Zegura. This work was previously presented in part at the IEEE International Conference on Network Protocols, October, 1998.

A. Shaikh is with the IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA (e-mail: aashaikh@watson.ibm.com).

J. Rexford is with AT&T Labs–Research, Florham Park, NJ 07932-0971 USA (e-mail: jrex@research.att.com).

K. G. Shin is with the University of Michigan, Ann Arbor, MI 48109 USA (e-mail: kgshin@eecs.umich.edu).

Publisher Item Identifier S 1063-6692(01)03228-9.

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 for three small topologies. However, these studies have not included a detailed evaluation of how the update policies interact with realistic traffic parameters and the key properties 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 address the tradeoff between the granularity of the link-state metrics and computational complexity. However, earlier studies do not consider how the coarse-grain metrics interact with out-of-date link-state information.

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]–[17], we focus on the interplay between link-state update policies, traffic patterns, and network topology. Our contributions include:

- **Routing versus setup failures:** In evaluating the connection blocking probability, we draw an important distinction between *routing* failures and *setup* failures. A routing failure occurs when the source switch cannot compute a feasible path for the new connection. In contrast, a setup failure occurs when the source selects a seemingly feasible path that ultimately cannot support the new connection. Setup failures incur extra overhead to reserve resources along the path. Our experiments illustrate how the frequency of routing and setup failures depends on periodic and triggered link-state updates.
- **Traffic and topology**: The performance of QoS routing depends on the traffic patterns and the underlying network topology. Drawing on recent work on traffic characterization, we consider how bursty connection arrivals and highly variable connection durations interact with link-state update policies. In addition, we study how the benefits of having a rich network topology vary with the staleness of the link-state information. The study of network topology draws on parameterized models of random graphs and regular graphs, as well as an example of a real backbone topology.
- **Granularity of link-state metrics:** Limiting the granularity of link-state metrics reduces the computational overhead for the routing algorithm, at the risk of higher blocking probabilities. We present a detailed comparison of routing performance under a range of granularities for the link-state metrics. The experiments focus on how out-of-date link-state information impacts the relative performance of coarse-grain link-state metrics. These results demonstrate how to reduce the computational complexity of the routing algorithm without sacrificing performance.

In Section II, 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. 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 III examines the effects of periodic and triggered link-state updates on the performance and overheads of QoS routing. The experiments in Section IV evaluate several different topologies to explore the impact of inaccurate information on the network's ability to exploit the presence of multiple short routes between each pair of switches. Section V studies the impact of stale load information on the choice of link metrics for selecting minimum-cost routes for new connections. Section VI concludes the paper with a summary of our findings and guidelines for tuning link-state update policies and link-cost metrics for QoS routing in large backbone networks.

II. Routing and Signaling 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.

A. 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. Depending on the admission control policy, b may represent a peak or average bandwidth of the connection, or some other estimate of the necessary bandwidth allocation. 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 signaling request. Each link i has capacity R_i and reserved bandwidth $r_i \leq R_i$ that cannot be allocated to new connections. Consequently, a switch's

link-state database stores (possibly stale) information r_i' about the reserved bandwidth for each link i in order to compute suitable routes for new connections. Each link also has a cost c_i (c_i') that depends on the reserved bandwidth.

A path P consists of a set of links from the source switch to the destination switch. 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], [13], [15], [17], [18]. For example, the "widest shortest path" heuristic selects a path with the minimum number of hops, breaking ties by selecting the path with most available bandwidth (i.e., the shortest path with the maximum value of $\min_{i \in P}\{R_i - r_i\}$). Favoring wide paths 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 $\sum_{i \in P} r_i/R_i$) to balance network utilization. In contrast, nonminimal 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.

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 $r_i' + b > R_i$).
2) Compute shortest (minimum-hop) paths amongst the remaining links.
3) Extract a route with the minimum total cost $\sum_i c_i'$.

This process effectively computes a "cheapest-shortest-feasible" path 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 [19] to compute the shortest 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 [20], the shortest-path computation still incurs significant overhead in large networks. Extracting the route introduces complexity in proportion to the path length.

[1]In practice, a single invocation of Dijkstra's algorithm is sufficient for computing the shortest path (in terms of hop count) with the minimum total 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 $h \leq N$.

B. 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 available bandwidth, or sum of the link delays. Defining the path cost as the sum of link utilization reduces connection blocking probability and results in less route oscillation by adapting slowly to changes in network load [13]. Other studies have shown that assigning each link a cost that is exponential in its current utilization results in optimal blocking probability [21]. For a general model of link cost, we employ a function that grows exponentially in the link utilization ($c_i \propto (r_i/R_i)^\alpha$), 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.

We represent link cost with C discrete values:

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

Small values of C limit the computational and storage requirements of the shortest-path computation [6], [10], [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, though the presence of multiple minimum-cost routes provides efficient opportunities to balance load through alternate routing. Relatively simple algorithms have $O(L + CN)$ complexity [19], while more complicated approaches offer a further reduction in computational complexity [20].

C. Connection Signaling

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 $r_i + b \leq R_i$, although the source determines that $r_i' + b > R_i$). Routing failures do not occur when pruning is disabled. In the absence of a routing failure, the source initiates hop-by-hop signaling to reserve bandwidth b on each link in the route.

As the signaling message traverses the selected path, each switch performs an admission test to check that the link can ac-

tually support the connection. If the link has sufficient resources, the switch reserves bandwidth on behalf of the new connection (i.e., $r_i = r_i + b$) before forwarding the setup 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 *setup failure* occurs if a link does not have enough resources available when the setup message arrives. Stale link-state information contributes to these *setup failures*, since the absence of sufficient resources is discovered only during the call setup process. By blocking connections inside the network, setup failures consume processing resources and delay the establishment of other connections. In addition, a failed connection temporarily holds resources at the upstream links, which may block other connections in the interim. In contrast, routing failures are purely local and do not consume any resources beyond the processing capacity at the source switch.

To deploy QoS routing with reasonable network overheads, the delays for propagating and processing these setup 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 setup 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.

D. 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], [13], 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., $R_i - r_i$). 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 [23]. 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

$$\Delta_i = \frac{|r_i' - r_i|}{R_i - r_i'}.$$

These changes in link state stem from the reservation (release) of link bandwidth during connection establishment (termination). By updating link load information in response to a change in available bandwidth, triggered updates respond to smaller changes in utilization as the link nears capacity, when the link may become incapable of supporting new connections. Similarly, connections terminating on a heavily loaded link introduce a large relative change in available bandwidth, which generates an update message even for very large 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.

E. 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 [24]. In addition, conclusions about algorithm or protocol performance may 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 nonuniformity [25]. 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 performance trends in each configuration.

As our study focuses on backbone networks, we consider topologies with relatively high connectivity, an increasingly common feature of core backbone networks that support a dense traffic matrix (with significant traffic between most pairs of core nodes) and are resilient to link failures. Since our study focuses on intradomain routing, our topology model is meant to represent the nodes and links within a single domain or autonomous system. As such, the model does not include the edge links connecting to neighboring domains. Each node can be viewed as a single switch that sends and receives traffic for one or more sources and carries traffic to and from other switches or routers. Since we focus on a single domain, we do not incorporate recently developed models of interdomain topology [26], [27].

We study a well-known core topology (an early representation of the MCI backbone that has appeared in other routing studies [9], [10], [12]). In addition, we evaluate both random graphs (generated using [28]) and regular topologies in order to

TABLE I
TOPOLOGIES USED IN EXPERIMENTS

Topology	Nodes	Links	Degree	Diam.	Avg. path length
Random	100	492	4.92	6	3.03
MCI	19	64	3.37	4	2.34
Regular	125	750	6	6	3.63

vary important parameters like size and node degree in a controlled fashion. Most of our graphs show results for the MCI and random topologies, though in Section IV 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 at the expense of having more links and, consequently, higher link-state update distribution overhead. Table I lists the pertinent characteristics of the topologies used in our experiments. We further assume that the links have uniform capacity R (i.e., $R_i = R$ for all i). In addition, 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 nonuniform 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 [29]. We model connection durations using a Pareto distribution with shape parameter $a = 2.5$ to capture the long-tailed nature of connection durations[2] [24] while still producing a distribution with finite variance, making it possible to gain sufficient confidence in 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 $\overline{b}$. For instance, call bandwidths may have a mean of 5% of link capacity with a spread of 200%, resulting in $b \sim U(0.0, 0.1R]$. 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 $\overline{b}$ 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 $\rho = \lambda N \ell \overline{b} \overline{h}/LR$, where $\overline{h}$ is the mean distance (in number of hops) between nodes, averaged across all source–destination pairs. Additional details on the simulation methodology are available in [30].

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

III. LINK-STATE UPDATE POLICIES

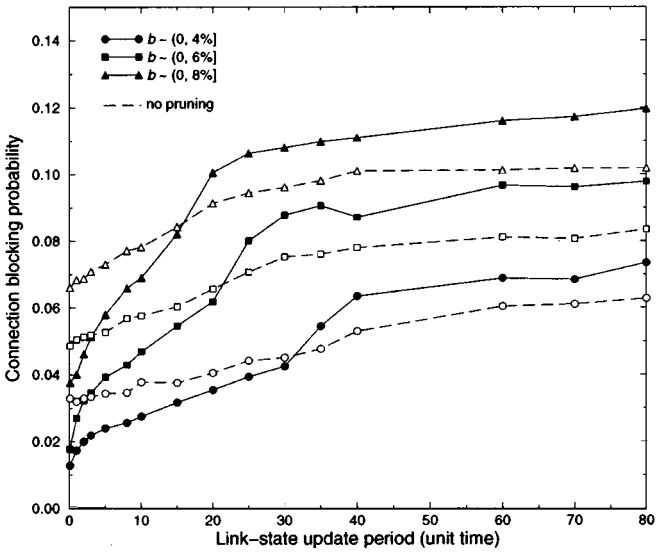
The initial simulation experiments focus on the effects of inaccurate link-state information on the performance and overheads of QoS routing by evaluating periodic and triggered updates in isolation. With a periodic update policy, large periods substantially increase connection blocking, ultimately outweighing 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 setup failures.

A. Periodic Link-State Updates

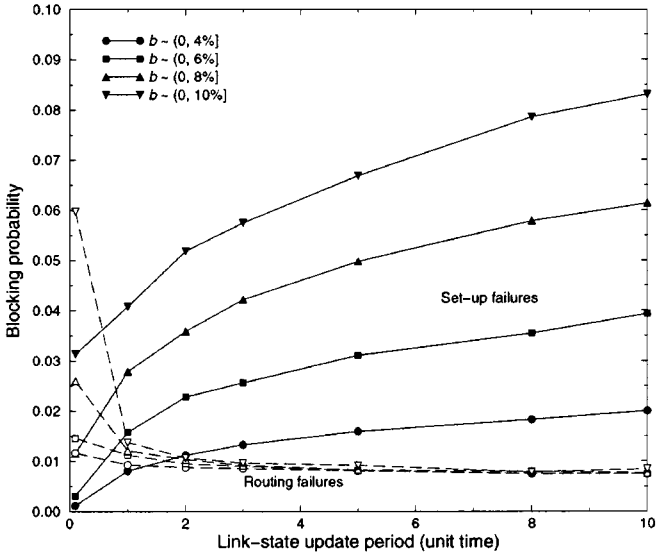
The connection blocking probability increases as a function of the link-state update period, as shown in Fig. 1(a). The experiment evaluates three bandwidth ranges on the random topology with an offered load of $\rho = 0.75$; the connection arrival rate remains fixed at $\lambda = 1$, while the Pareto 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. Due to their higher resource requirements, the high-bandwidth connections experience a larger blocking probability than the low-bandwidth connections across the range of link-state update rates. The blocking probability for high-bandwidth connections, while higher, does not appear to grow more steeply as a function of the update period; instead, the three sets of curves remain roughly equidistant across the range of update periods.

Pruning Versus Not Pruning: In general, pruning improves the effectiveness of QoS routing under small to moderate load by allowing connections to consider nonminimal routes. However, stale link-state information reduces the effectiveness of pruning, as shown in Fig. 1(a). Initially, the "pruning" curves have a lower blocking probability than the "no pruning" curves; however, the curves cross as the link-state update period increases. 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. Even with accurate link-state information, pruning degrades performance under heavy load since nonminimal routes consume extra link capacity at the expense of other connections. 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 be more conservative or more liberal in removing links to balance the tradeoff between minimal and nonminimal routes [31].

Route Flapping: Although QoS routing performs well for small link-state update periods (significantly outperforming static routing [30]), the blocking probability rises relatively quickly before gradually plateauing for large update periods. In Fig. 1(a), even an update period of five time units (five times



(a)



(b)

Fig. 1. **Staleness due to periodic updates:** In (a) we show that the blocking probability grows rapidly as the link-state update period grows for several ranges of requested bandwidth; the dashed lines indicate the performance when pruning is disabled. In (b) we focus on small periods and routing with pruning to show that the blocking is dominated by setup failures after only a small increase in the period. Both graphs show results for the random topology with $\lambda = 1$, $\alpha = 1$, and $\rho = 0.75$. The mean connection durations are 60.9, 40.6, 30.4, and 24.4 for the mean bandwidths 2%, 3%, 4%, and 5%, respectively.

the average connection interarrival time) shows significant performance degradation. By this point, setup 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 Fig. 1(b) which focuses on a small region of the experiments with pruning in Fig. 1(a); when pruning is disabled, routing failures never occur, and setup failures account for all blocked connections. In general, periodic updates do not respond quickly enough to variations in link state, sometimes allowing substantial changes to go unnoticed. This suggests that inaccuracy in the link-state database causes the source switch to mistake infeasible links as feasible; hence, the source selects an infeasible path, even when there are other feasible routes to the destination. We see that routing failures occur only with very accurate information since the source learns about link infeasibility very quickly. When link state can fluctuate significantly between updates the source is virtually certain to find at least one seemingly feasible path, thus avoiding a routing failure.

Under large update periods, relative to the arrival rates and holding times, the links can experience dramatic fluctuations in link state between successive update messages. Such link-state flapping has been observed in packet routing networks [32], 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 interarrival 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 Fig. 1(a) flatten at a level that corresponds to the steady-state blocking probability during the "plateau" interval.

Eventually, QoS routing starts to perform worse than static routing, because the fluctuations in link state begin to exceed the random variations in traffic load. In searching for (seemingly) underutilized links, QoS routing targets a relatively small set of links until new update messages arrive to correct the link-state database. In contrast, under static routing, the source blindly routes to a single group of links, though this set is typically larger than the set identified by QoS routing. Thus, when the update period grows quite large, static routing is more successful at balancing load and reducing connection blocking. The exact crossover point between the two routing algorithms is very sensitive to the distribution of 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 nonuniform load, QoS routing continues to outperform static routing even for large update periods. For example, experiments with the nonhomogeneous 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. We examined this effect by comparing the connection blocking probabilities from Fig. 1(a) to several alternative measures of blocking (not shown). For example, 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 band-

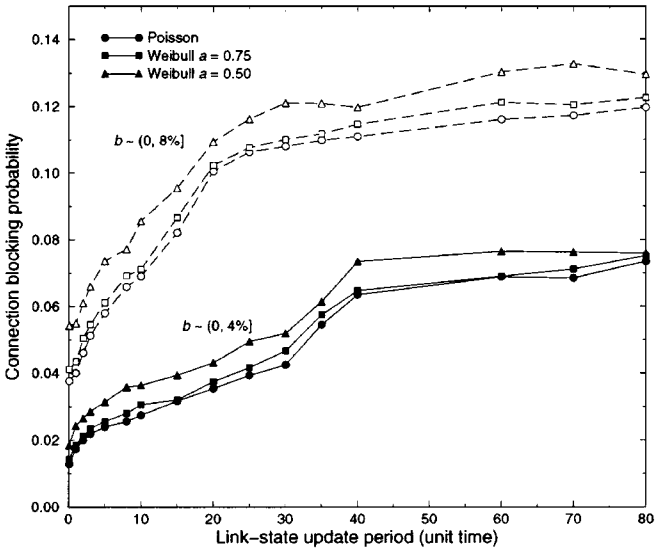


Fig. 2. **Blocking for varying arrival burstiness**: Burstiness is increased with a smaller Weibull shape parameter, a. Simulation parameters are the same as in Fig. 1(a).

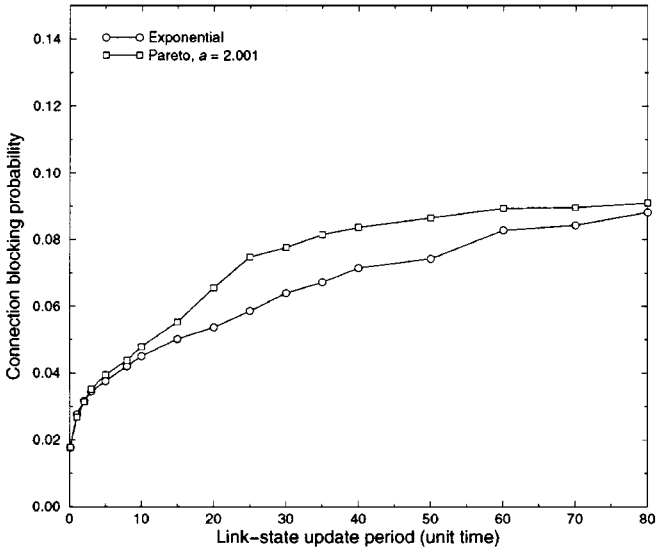


Fig. 3. **Blocking for different duration distributions:** The mean duration is fixed at $\ell = 40$, and we compare exponentially distributed connection durations to a long-tailed Pareto distribution. The arrival rate λ varies to keep offered load fixed at $\rho = 0.75$ in the random topology, with $b \sim (0, 0.06]$, $\alpha = 1$, and pruning enabled.

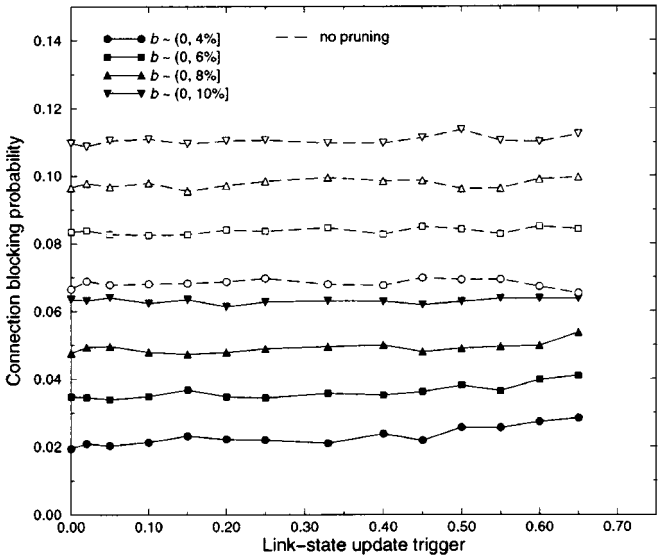
width. Compared to conventional connection blocking, these metrics grow more quickly in the presence of stale information. In general, bandwidth blocking exceeds hop-count blocking, suggesting that high-bandwidth connections are even harder to route than high hop-count connections, though link-state staleness does not seem to affect one metric more than the other [30]. Fig. 2 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 nonbursty 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, our experiments showed that longer durations allow the use of larger link-state update periods to achieve the same blocking probability [30]. Short-lived connections cause link state to fluctuate rapidly, particularly for high-bandwidth requests, and thus require frequent updates to maintain good routing performance. In Fig. 3, we find that exponentially distributed connection durations result in 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 [33].

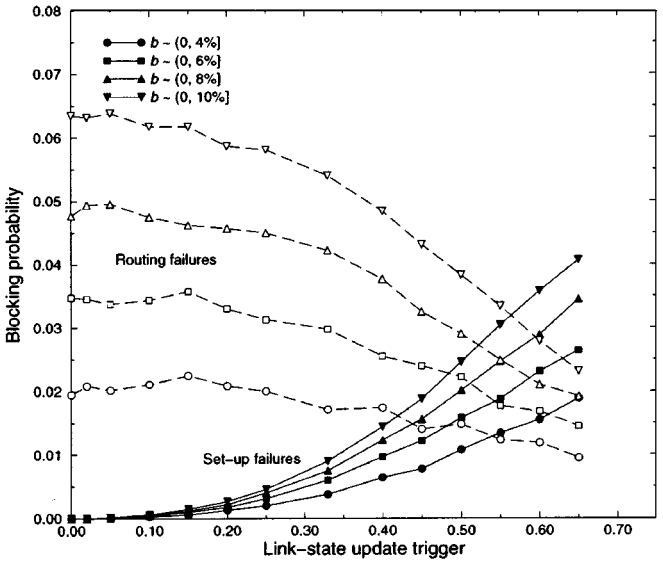
B. Triggered Link-State Updates

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 Fig. 4(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 [30].

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 setup 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 setup failure. Fig. 4(b) highlights this effect by plotting the



(a)



(b)

Fig. 4. **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 $\alpha = 1$. The mean connection durations are 50.4, 33.6, 25.2, and 20.2 for the mean bandwidths 2%, 3%, 4%, and 5%, respectively.

blocking probability for both routing and setup failures. Across the range of trigger values, the increase in setup 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

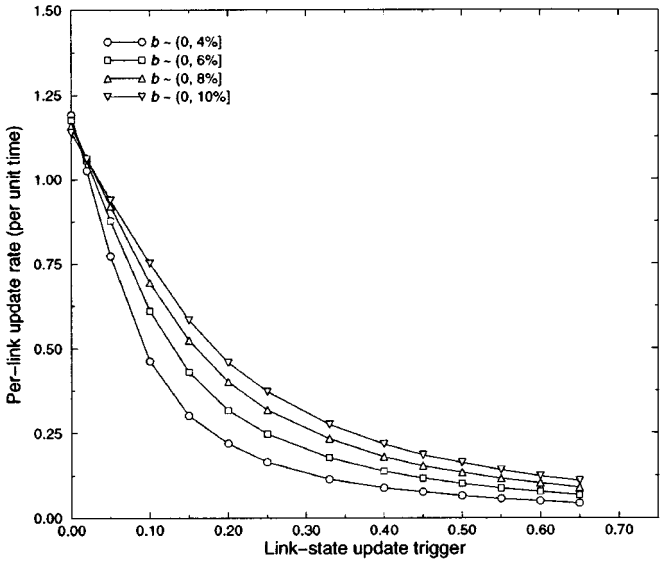


Fig. 5. **Link-state update rate for different trigger values:** This graph shows the link-state update rate for the random topology with $\rho = 0.75$, $\lambda = 1$, $\alpha = 1$, and pruning disabled. Mean connection durations are the same as in Fig. 1.

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 tradeoff between routing and setup failures persists even in the presence of hold-down timers, though the hold-down timer increases the overall blocking probability and rate of setup failures.

Link-State Update Rate: Despite the increase in setup failures, large trigger values substantially reduce the number of update messages for a given blocking probability, as shown in Fig. 5. 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 $2\lambda N\overline{h}/L$ in a network with N switches, L links, and an average path length of $\overline{h}$ hops. Here, the expression reduces to 1.23 link-state update messages per unit time, which is close to the y-intercept in Fig. 5; additional experiments show that the link-state update rate is not sensitive to the connection holding times, consistent with the $2\lambda N\overline{h}/L$ expression. In Fig. 5, 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 signaled 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, setup failures may trigger 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 setup failures for larger triggers slows the reduction in the update rate in Fig. 5 as the trigger grows. The exact effect of setup failures depends

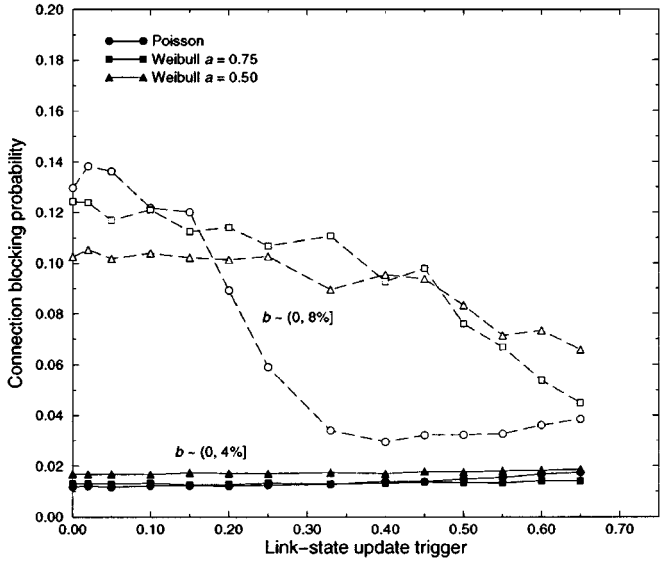


Fig. 6. **Bursty arrivals with triggered updates:** These graphs show the effect on connection blocking of increased burstiness with the same mean arrival rate. Smaller Weibull shape parameters result in more burstiness in the arrival process. Simulation parameters are the same as those in Fig. 5.

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. 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 [30]. Overall, though, modest trigger values are effective at reducing the link-state update rate by about a factor of three to four. Also, for a fixed update rate, triggers are able to significantly reduce the proportion of setup 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 signaling. When using periodic updates at the same frequency, setup failures account for 74% of the blocked connections, and the blocking rate is much higher.

Impact of Non-Poisson Arrivals: Fig. 6 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 blocking rate decreases 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, the source is unable to find (seemingly) feasible routes after pruning, except when using very large triggers, as shown by the gradual decline in blocking probability relative to Poisson arrivals. When pruning is not permitted (results not shown), 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 signaled, results in more link-state fluctuation relative to nonbursty traffic. We investigated this through a comparison of the link-state update rate for bursty and nonbursty arrivals (experiment not shown). 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. When link-state triggers become coarse, however, the update rate for bursty traffic remains high due to a combination of a greater 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 setup failures as in Fig. 4.

Ultimately, the choice of link-state periods and triggers depends on the relative cost of routing failures, setup 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 setup failures. These tradeoffs suggest a hybrid policy with a moderately large trigger value to provide load 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 Fig. 5, 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.

IV. Network Topology

Though the results in Section III were mainly presented in the context of the random topology, we have conducted numerous additional experiments with the MCI and regular topologies (see Table I for topology parameters). In this section, we revisit some of the staleness issues in the previous section to highlight the dependency on the underlying topology. Then, we present a more systematic study of topology based on a parameterized model of regular graphs that allows us to exert direct control over the key topological properties.

A. General Topology Observations

Our experiments with periodic updates in the random topology [Fig. 1(a)] show a strong dependency on the update period regardless of whether link pruning was enabled or disabled. However, in the MCI topology we observe a much weaker dependence on the link-state update period when pruning is disabled. 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 route computation does not react much to changes in link load.

We see a more pronounced effect when considering pruning with triggered updates. Fig. 4 showed that triggered link-state updates generally do not affect the overall blocking rate, with or without applying pruning. However, when pruning in a sparsely connected network, an incorrect pruning decision can cause the

source to erroneously consider nonminimal routes. For example, additional experiments with the random topology (not shown) indicated that it has *higher* blocking rates with more accurate information (i.e., *smaller* trigger values), when trying to route high-bandwidth connections. This effect is also seen with bursty arrivals, as described in Section III-B. The random graph typically does not have multiple equal-length paths between a pair of nodes. As a result, pruning an infeasible link along the shortest path results in the selection of a nonminimal route. In the end, this increases the overall blocking probability, since these nonminimal routes consume extra resources. If, instead, the source chose not to prune this infeasible link (say, due to stale link-state information), then the connection would attempt to signal along the shortest path. Although the connection would block upon encountering the infeasible link, the network would benefit by deciding not to accept the connection. 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.

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 also be limited, however, by explicitly controlling the degree of nonminimality (e.g., at most one extra hop) rather than disabling pruning altogether. Poor performance due to a lack of routing choices in the topology is also worsened when connection requests arrive in bursts. We illustrated in Section III-B that bursty connection arrivals increase blocking by making it harder for the source to find feasible paths. Additional experiments show that this problem is exacerbated in topologies which have few equal-length paths between a source and destination. We find, for example, that routing failures for high-bandwidth connections remain high in the random topology as the update trigger increases, but they decline somewhat in the uniform topology where multiple equal-length routes are available.

We find that topology and traffic together play an important role in defining the range of update frequencies over which QoS routing provides a significant performance advantage over static routing. When the traffic pattern is matched to the topology, static routes (if properly provisioned) can perform quite well. In comparing QoS and static routing in the uniform topology with uniform random requests, we found that when the link-state update period is very large, static routing begins to outperform QoS routing. Experiments with other topologies, however, show that QoS routing is able to capitalize on mismatches between traffic and topology to consistently perform better than static routing. For example, experiments with the nonhomogeneous MCI backbone topology (with uniform traffic) show that QoS routing consistently achieves lower blocking probability than static routing over the entire range of update rates considered. In this case the key point is not the topology configuration, but rather the relationship between the topology and traffic pattern.

B. Parameterized Topology Model

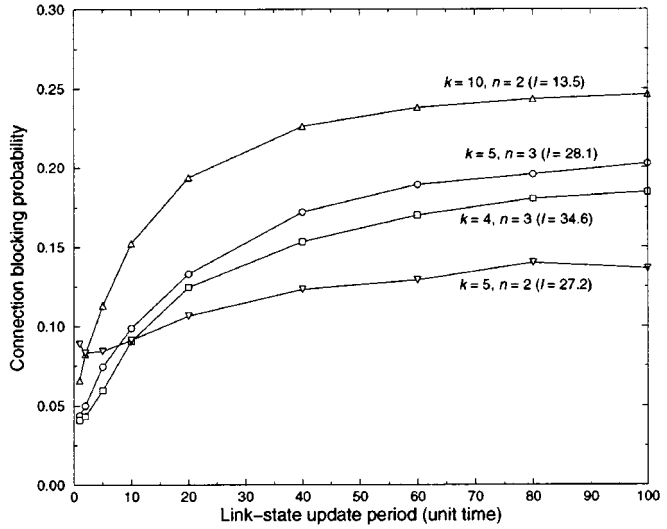
The experiments with the graphs in Table I provide some insights into the impact of topology on the performance of QoS routing in a variety of situations (e.g., pruning/no pruning, bursty arrivals, high-bandwidth connections). However, in order to focus more directly on topology effects, we require a model with greater control over important parameters such as diameter, node degree, mean path length, and the number of similar-length paths between pairs of nodes. Random graph models (e.g., Waxman) offer some control over connectivity but it is difficult to precisely control the other parameters of interest. In addition, the use of random graphs makes it difficult to match the traffic pattern with the underlying topology. For example, applying uniform random selection of destination nodes in a random graph results in nonuniform traffic load. This makes it difficult to draw fair comparisons between experiments with different random graphs.

TABLE II
CHARACTERISTICS OF k-ARY n-CUBES

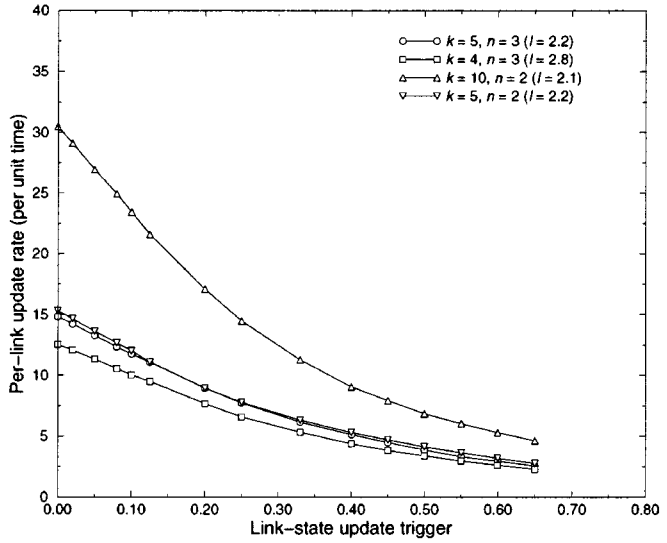
Topology (k, n)	Nodes	Links	Degree	Diam.	Avg. path length
$10, 2$	100	400	4	10	5.05
$5, 3$	125	750	6	6	3.63
$4, 3$	64	384	6	6	2.95
$5, 2$	25	100	4	4	2.50

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 II. 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 Fig. 7(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 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 signaling 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.



(a)



(b)

Fig. 7. **Topology and link-state accuracy:** (a) Richly connected topologies have low blocking probabilities under accurate information, although the benefits of multiple routes degrade under large update periods. (b) With triggered updates, the rate of link-state messages is proportional to k and independent of n. In both experiments, $\rho = 85\%$, $b \sim (0, 0.1]$, and $\alpha = 2$ (with pruning). The arrival rates in both graphs are $\lambda = 1$ and $\lambda = 12.5$, respectively. Load is kept constant across the four topologies by changing ℓ.

Direct comparisons between the four topologies are somewhat difficult, due to differences in the number of nodes and links. For example, the crossover in Fig. 7(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. Fig. 7(b) shows the overheads for triggered link-state updates in the four topologies. Although the 10-ary 2-cube has fewer nodes 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 [30]. Increasing the network dimension n corresponds to growing the underlying network in a manner that increases 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 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 node receives the message on each incoming link. As a result, each node 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 Fig. 7(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.

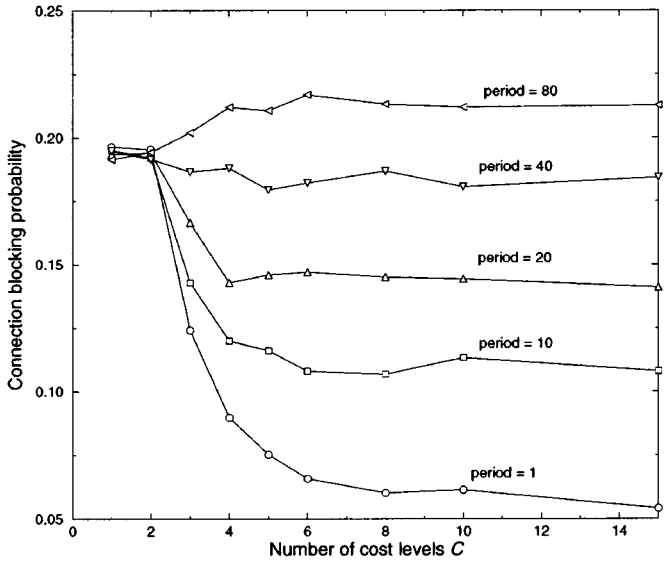
V. Link-Cost Parameters

The link-state update rate also impacts the choice of the link-cost parameters (C and α, Section II-B) 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.

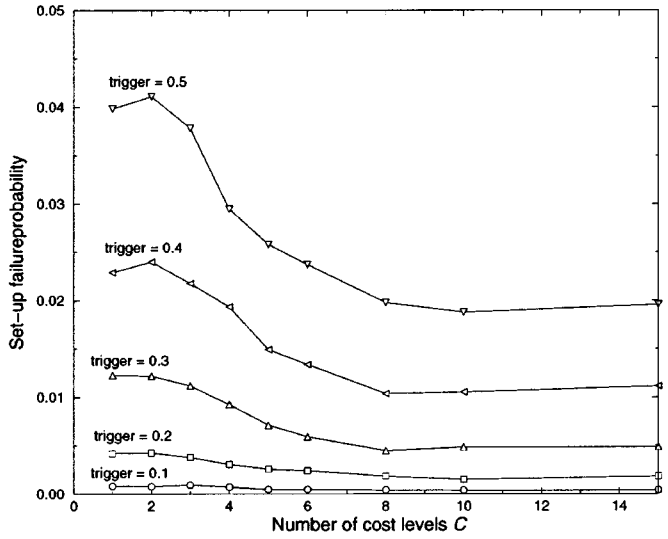
A. Number of Cost Levels (C)

The experiments in Sections III and IV 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. Fig. 8(a) evaluates the routing algorithm over a range of cost granularity and link-state update periods. To isolate the effects of the cost function, the routing algorithm does not attempt to prune (seemingly) infeasible links before invoking the shortest-path computation in this experiment. 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 Fig. 8(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



(a)



(b)

Fig. 8. **Discretized costs with stale link-state information:** (a) 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. (b) Additional cost levels decrease the rate of setup failures when using triggered updates with pruning. 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 $\alpha = 1$.

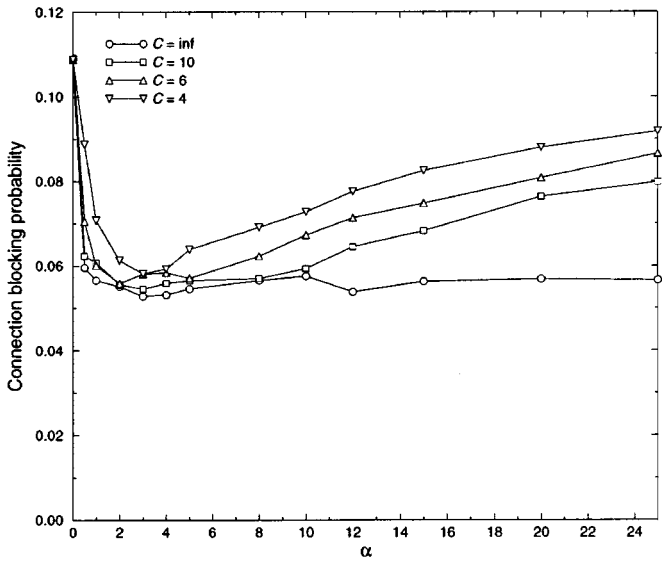


Fig. 9. **Exponent α with stale link-state information:** A larger exponent α decreases the blocking probability, until α grows so large that the cost intervals become too narrow. This experiment evaluates the random topology with update period $= 10$, $\rho = 0.75$, $\lambda = 1$, $b \sim (0.0,\ 0.06]$, $\ell = 40.6$, and pruning disabled.

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 Fig. 8(b). We still find, however, that experiments with a finite number of C values are consistent with the results in Section III-B; that is, the connection blocking probability remains constant over a wide range of triggers. Hence, Fig. 8(b) plots only the setup failures (with pruning). The overall blocking rate curves have roughly the same shape as the setup failure curves, all ranging from about 0.07 with $C = 1$ and flattening to roughly 0.035. 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 setup failures by a factor of two, increasing the number of cost levels eventually offers diminishing returns.

B. 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 Fig. 9. Other experiments (not shown) confirm that these results hold across a range of link-state update periods, from very frequent updates to a period equal to 40 times the mean connection interarrival time. This implies that large values of α do not introduce much extra route flapping. This also

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 Fig. 9 with $C \neq \infty$. 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 Fig. 9. 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.

VI. Conclusion

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. The experiments confirm and extend the findings of other studies, and also lend new insight into the impact of out-of-date link-state information. The results complementing the observations of other recent studies include [9]–[13], [34]:

- **Periodic link-state updates:** The staleness introduced by periodic link-state update messages causes flapping that substantially increases the rate of connection blocking. In extreme cases with large update periods, QoS routing actually performs worse than 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. 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. 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 setup 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.
- **Bandwidth and hop count:** Connections with large bandwidth requirements experience higher blocking probabilities. Likewise, connections traveling a larger number of hops experience higher blocking probabilities. Stale link-state information exacerbates both of these effects. These effects degrade the performance of QoS routing in large networks, unless the topology is designed carefully to limit the worst-case path length.
- **Connection duration:** Longer connection durations change the timescale of the network and allow the use of larger link-state update periods. Our findings suggest that the networks should limit QoS routing to long-lived connections, while carrying short-lived traffic on preprovisioned static routes.

Our experiments also reveal several new insights:

- **Impact of staleness on setup failures:** In addition to increasing the connection blocking probability, out-of-date link-state information increases the fraction of connections that experience setup failures. Larger update periods increase both the blocking probability and the proportion of connections that block in the setup phase. Even though triggered updates do not increase the blocking probability, larger trigger values result in a higher proportion of setup failures. Avoiding heavy signaling loads requires careful selection of the update period and trigger.
- **Pruning under stale information:** 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 allocated to nonminimal routes.
- **Long-tailed connection durations:** Stale information has a more dramatic effect under realistic long-tailed distributions for connection duration. This stems from 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. 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 [33].

- **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 nonbursty 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.
- **Rich network topologies:** The tradeoff between routing and setup 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, as well as techniques for avoiding excessive link-state traffic. For example, the network could broadcast link-state updates in a spanning tree, and piggyback link-state information in signaling 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, as discussed in more detail in [22].

These observations represent an important 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 setup failures and simulation experiments under a wide range of parameters provide valuable insights into the underlying behavior of the network. Future research in this area can exploit these trends to reduce the computational and protocol overheads of QoS routing in large backbone networks.

REFERENCES

[1] Z. Whang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1228–1234, Sept. 1996.

[2] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, "A framework for QoS-based routing in the Internet,", Request for Comments 2386, Aug. 1998.

[3] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Network Mag.*, vol. 12, pp. 64–79, Nov. 1998.

[4] Specification working group, "Private Network–Network Interface Specification Version 1.0," ATM Forum, Mar. 1996.

[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),", Work in progress, Sept. 1997.

[6] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, "QoS routing mechanisms and OSPF extensions,", Request For Comments 2676, Jan. 1998.

[7] R. Guerin and A. Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Trans. Networking*, vol. 7, pp. 350–364, June 1999.

[8] L. Breslau, D. Estrin, and L. Zhang, "A simulation study of adaptive source routing in integrated services networks," Comput. Sci. Dept., Univ. of Southern California, Los Angeles, CA, Tech. Rep. 93-551, 1993.

[9] Q. Ma and P. Steenkiste, "Quality-of-service routing for traffic with performance guarantees," in *Proc. IFIP Int. Workshop Quality of Service*, New York, May 1997, pp. 115–126.

[10] ——, "On path selection for traffic with bandwidth guarantees," in *Proc. IEEE Int. Conf. Network Protocols*, Atlanta, GA, Oct. 1997, pp. 191–202.

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

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

[13] I. Matta and A. U. Shankar, "Dynamic routing of real-time virtual circuits," in *Proc. IEEE Int. Conf. Network Protocols*, Columbus, OH, 1996, pp. 132–139.

[14] S. Rampal and D. Reeves, "Routing and admission control algorithms for multimedia data," *Comput. Commun.*, vol. 18, no. 10, pp. 755–768, Oct. 1995.

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

[16] A. Iwata, R. Izmailov, H. Suzuki, and B. Sengupta, "PNNI routing algorithms for multimedia ATM Internet," *NEC Res. & Develop.*, vol. 38, Jan. 1997.

[17] C. Pornavalai, G. Chakraborty, and N. Shiratori, "QoS-based routing in integrated services packet networks," in *Proc. IEEE Int. Conf. Network Protocols*, Atlanta, GA, Oct. 1997, pp. 167–174.

[18] 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. Copenhagen, Denmark: North-Holland, June 1991, vol. 14, pp. 397–403.

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

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

[21] S. Plotkin, "Competitive routing of virtual circuits in ATM networks," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1128–1136, Aug. 1995.

[22] A. Shaikh, J. Rexford, and K. G. Shin, "Efficient precomputation of quality-of-service routes," in *Proc. Workshop Network and Operating System Support for Digital Audio and Video*, July 1998, pp. 15–27.

[23] S. Floyd and V. Jacobson, "Synchronization of periodic routing messages," *IEEE/ACM Trans. Networking*, vol. 2, pp. 122–136, Apr. 1994.

[24] V. Paxson and S. Floyd, "Difficulties in simulating the Internet," *IEEE/ACM Trans. Networking*, to be published.

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

[26] A. Medina and I. Matta, "BRITE: A flexible generator of Internet topologies," Boston Univ., Boston, MA, Tech. Rep. BU-CS-RT-2000–005, Jan. 2000.

[27] C. Jin, Q. Chen, and S. Jamin, "Inet: Internet topology generator," Univ. of Michigan, Ann Arbor, MI, Tech. Rep. CSE-TR-433-00, Sept. 2000.

[28] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1617–1622, Dec. 1988.

[29] A. Feldmann, "Characteristics of TCP connection arrivals," in *Self-Similar Network Traffic and Performance Evaluation*, K. Park and W. Willinger, Eds. New York: Wiley, 2000.

[30] A. Shaikh, "Efficient dynamic routing in wide-area networks," Ph.D. dissertation, Univ. of Michigan, Ann Arbor, MI, May 1999.

[31] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proc. IEEE GLOBECOM*, Nov. 1997, pp. 1903–1908.

[32] A. Khanna and J. Zinky, "The revised ARPANET routing metric," in *Proc. ACM SIGCOMM*, Austin, TX, 1989, pp. 45–56.

[33] A. Shaikh, J. Rexford, and K. G. Shin, "Load-sensitive routing of long-lived IP flows," in *Proc. ACM SIGCOMM*, Aug./Sept. 1999, pp. 215–226.

[34] I. Matta and M. Krunz, "Packing and least-loaded based routing in multi-rate loss networks," in *Proc. IEEE Int. Conf. Communications*, Montreal, Canada, June 1997, pp. 827–831.

Anees Shaikh (S'91–M'99) received the B.S.E.E. and M.S.E.E. degrees in electrical engineering from the University of Virginia, Charlottesville, in 1994, and the Ph.D. degree in computer science and engineering from the University of Michigan, Ann Arbor, in 1999.

He is currently a Research Staff Member in the Internet Infrastructure Department, IBM T. J. Watson Research Center, Hawthorne, NY. His research interests include network services infrastructure, Internet traffic engineering, and operating systems support for networking.

Jennifer Rexford (S'89–M'96) received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, in 1991, and the M.S.E. and Ph.D. degrees in computer science and electrical engineering from the University of Michigan, Ann Arbor, in 1993 and 1996, respectively.

She is now a Member of the Internet and Networking Systems Center, AT&T Labs–Research, Florham Park, NJ. Her research interests include routing protocols, Internet traffic characterization, and multimedia streaming. She is a member of the editorial board of IEEE/ACM TRANSACTIONS ON NETWORKING and is coauthor of the book *Web Protocols and Practice* (Reading, MA: Addison-Wesley, 2001) with Balachander Krishnamurthy.

Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and the M.S. and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1976 and 1978, respectively.

He is currently a Professor and the founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He has supervised the completion of 41 Ph.D. dissertations, and authored or co-authored over 600 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has co-authored (jointly with C. M. Krishna) the textbook *Real-Time Systems* (New York: McGraw Hill, 1997).

Dr. Shin is a member of the Korean Academy of Engineering, and was the General Chair of the 2000 IEEE Real-Time Technology and Applications Symposium. He also chaired the IEEE Technical Committee on Real-Time Systems during 1991–1993, and was a Distinguished Visitor of the Computer Society of the IEEE.