Load-Sensitive Routing of Long-Lived IP Flows

Anees Shaikh
Security and Networking Systems
IBM T.J. Watson Research Center
Hawthorne, NY 10532-1507
aashaikh@watson.ibm.com

Jennifer Rexford
Networking and Distributed Systems
AT&T Labs – Research
Florham Park, NJ 07932-0971
jrex@research.att.com

Kang G. Shin
Real-Time Computing Laboratory
University of Michigan
Ann Arbor, MI 48109-2122
kgshin@eecs.umich.edu

Abstract

Internet service providers face a daunting challenge in provisioning network resources, due to the rapid growth of the Internet and wide fluctuations in the underlying traffic patterns. The ability of dynamic routing to circumvent congested links and improve application performance makes it a valuable traffic engineering tool. However, deployment of load-sensitive routing is hampered by the overheads imposed by link-state update propagation, path selection, and signaling. Under reasonable protocol and computational overheads, traditional approaches to load-sensitive routing of IP traffic are ineffective, and can introduce significant route flapping, since paths are selected based on out-of-date link-state information. Although stability is improved by performing load-sensitive routing at the flow level, flapping still occurs, because most IP flows have a short duration relative to the desired frequency of link-state updates. To address the efficiency and stability challenges of load-sensitive routing, we introduce a new hybrid approach that performs dynamic routing of long-lived flows, while forwarding short-lived flows on static preprovisioned paths. By relating the detection of long-lived flows to the timescale of link-state update messages in the routing protocol, route stability is considerably improved. Through simulation experiments using a one-week ISP packet trace, we show that our hybrid approach significantly outperforms traditional static and dynamic routing schemes, by reacting to fluctuations in network load without introducing route flapping.

1 Introduction

Traffic engineering of large IP backbone networks has become a critical issue in recent years, due to the unparalleled growth of the Internet and the increasing demand for predictable communication performance. Ideally, an Internet service provider (ISP) optimizes the utilization of network resources by provisioning backbone routes based on the load between the edge routers. However, the volume of traffic between particular points in the network can fluctuate widely over time, due to variations in user demand and changes in the network configuration, including failures or reconfigurations in the networks of other service providers. Currently, network providers must resort to coarse timescale measurements to detect network performance problems, or may even depend on complaints from their customers to realize that the network requires reconfiguration. Detection may be followed by a lengthy diagnosis process to discover what caused the shift in traffic. Finally, providers must manually adjust the network configuration, typically redirecting traffic by altering the underlying routes.

These traffic engineering challenges have spurred renewed interest in dynamic routing as a network-management tool, rather than as a method for providing quality-of-service (QoS) guarantees. By selecting paths that circumvent congested links, dynamic routing can balance network load and improve application performance. Despite these potential benefits, however, most backbone networks still employ static routing (e.g., based on routing protocols such as OSPF and IS-IS) because techniques for load-sensitive routing often lead to route flapping and excessive control traffic overheads. This paper introduces a new routing scheme that maintains the benefits of dynamic routing, while also making it both stable and efficient.

Early attempts in the ARPANET to route based on dynamic link metrics resulted in dramatic fluctuations in link load over time. Routing packets based on out-of-date link-state information caused flapping, where a large amount of traffic would travel to seemingly under-utilized links. These links would become overloaded, causing future packets to route to a different set of links, which would then become overloaded. Improvements in the definition of the link metrics reduced the likelihood of oscillations [1], but designing stable schemes for load-sensitive routing is fundamentally difficult in packet-based networks like the Internet [2]. With the evolution toward integrated services in IP networks, recent research focused on load-sensitive routing of flows or connections, instead of individual packets. For example, a flow could correspond to a single TCP or UDP session, all IP traffic between a particular source-destination pair, or even coarser levels of aggregation. In particular, several QoS-routing schemes were proposed that select paths based on network load, as well as application traffic characteristics and performance requirements [3–6]. In recent years, several protocols for QoS-routing have been proposed for both IP and ATM networks [7–9].

Dynamic routing of flows should be more stable than selecting paths at the packet level, since the load on each link should fluctuate more slowly, relative to the time between updates of link-state information. Also, defining network load in terms of reserved bandwidth and buffer space, rather than measured utilization, should enhance stability. However, QoS-routing protocols impose a significant bandwidth and processing load on the network, since each router must maintain its own view of the available link resources, distribute link-state information to other routers, and compute and establish routes for new flows. The protocol and computational overheads can be significant in large backbone networks [10, 11]. Since most TCP/UDP transfers consist of just a handful of packets, load-sensitive routing of IP flows would require frequent propagation of link-state metrics and recomputation of routes to avoid the same instability problems that arise in dynamic routing at the *packet level*. We address this problem by proposing and evaluating a hybrid routing scheme that exploits the variability of IP flow durations to avoid the undesirable effects of traditional approaches to dynamic routing.

While most Internet flows are short-lived, the majority of the packets and bytes belong to long-lived flows, and this property persists across several levels of aggregation [12–15]. Although this inherent variability of Internet traffic sometimes complicates the provisioning of network bandwidth and buffer resources, heavy-tailed flow-size distributions can be exploited to reduce the overheads of certain control mechanisms. Observations of heavy-tailed lifetime distributions of UNIX processes have been similarly exploited in the context of processor load balancing, where migrating long-lived jobs can significantly reduce migration overhead [16]. Most notably in the networking context, variability in flow duration has been the basis of several techniques that reduce router forwarding overheads by establishing hardware switching paths for long-lived flows [17–19]. These schemes classify arriving packets into flows and apply a trigger (e.g., arrival of some number of packets within a certain time interval) to detect long-lived flows. Then, the router dynamically establishes a shortcut connection that carries the remaining packets of the flow. The shortcut terminates if no packets arrive during a predetermined timeout period (e.g., 60 seconds). Several measurement-based studies have demonstrated that it is possible to limit the set-up rate and the number of simultaneous shortcut connections, while forwarding a large fraction of packets on shortcuts [15, 19–22].

This paper builds on the above observations to study the implications of the variability in flow durations on the stability of load-sensitive routing. In contrast to previous work on flow switching, the choice to separate long-lived and short-lived flows is not necessarily motivated by the desire to forward packets in hardware, or the QoS requirements (if any) of the individual flows. We focus on dynamic routing as a traffic engineering technique that reacts to fluctuations in network load, rather than as a way to provide explicit performance guarantees. The contributions of this paper are:

- **Load-sensitive routing of long-lived flows:** We propose that backbone networks should perform dynamic routing of long-lived flows, while forwarding short-lived flows on preprovisioned static paths. Our approach exploits flow-classification hardware at the network edge and techniques for flow pinning, as well as basic insights from earlier work on QoS routing.

- **Relating traffic timescale to routing stability:** Separating short-lived and long-lived IP flows dramatically improves the stability of dynamic routing. Our hybrid scheme reacts to fluctuations in network load without introducing route flapping, by relating the detection of long-lived flows to the timescale of link-state update messages.

- **Network provisioning rules:** We propose simple and robust rules for allocating network resources for short-lived and long-lived flows, and techniques for sharing excess link capacity between the two traffic classes. The provisioning rules are tailored to measurements of the distribution of IP flow sizes, and the triggering policy for detecting long-lived flows.

Our approach complements and extends recent work on MultiProtocol Over ATM (MPOA) [18], which triggers the creation of IP shortcut connections across an underlying ATM network. In contrast to studies that evaluate the overheads of a single-link MPOA system [21, 22], we concentrate on the dynamics of load-sensitive routing over an entire network. Though our work is applicable to MPOA, we focus more broadly on IP networking rather than techniques for carrying IP traffic over ATM networks.

Section 2 describes the stability challenges of load-sensitive routing, and outlines our approach for separating short-lived and long-lived flows. Section 3 follows with policies for flow detection, path selection, and network provisioning. Section 4 describes the specifics of our simulation model. The benefits of the hybrid approach are illustrated in Section 5, through detailed simulations based on packet traces from a large ISP network. The experiments show that our hybrid scheme outperforms traditional static and dynamic routing, and is robust to inaccuracies in network provisioning and shifts in the offered traffic. Section 6 concludes the paper with a summary of our results and a discussion of future research directions.

2 Stable Load-Sensitive Routing

In this section, we describe how to dynamically route long-lived flows, while forwarding short-lived flows on static, preprovisioned paths. We show that performing load-sensitive routing on *all* IP traffic flows introduces significant route flapping for reasonable ranges of link-state update periods. We argue that stability and efficiency can be achieved by tying the frequency of link-state update messages to the timescale of the long-lived flows.

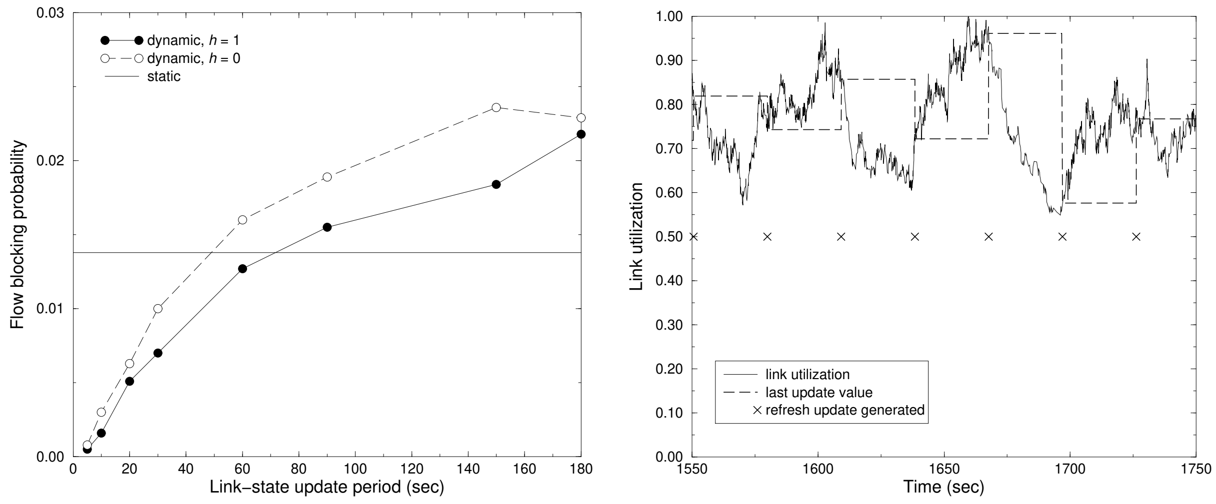
2.1 Stability Challenges in Load-Sensitive Routing

Depending on the network topology and the path-selection algorithm, static routing often cannot select good paths for all source-destination pairs. For example, protocols such as OSPF and IS-IS always forward packets on shortest paths, based on static link weights. As such, they cannot exploit non-minimal routes, and typically have limited control of how traffic is distributed when a source-destination pair has multiple shortest-path routes. Proposed extensions to OSPF and IS-IS support more flexible tie-breaking based on link load, without addressing the other limitations of static shortest paths [23]. With newer routing protocols such as MPLS, network administrators can preconfigure explicit tagged routes between specific source-destination pairs, providing greater control and flexibility in balancing network load for particular traffic patterns. Moving one step further, dynamic routing can circumvent network congestion and balance link load on a smaller time scale by reacting to current traffic demands. This motivation has been the basis of constraint-based routing in MPLS [24] and the proposed QoS extensions to OSPF [8, 9], as well as the ATM Forum's PNNI protocol [7].

Despite the potential benefits of dynamic routing, its deployment remains uncertain due largely to the significant bandwidth and processing requirements imposed by link-state update propagation, route computation, and signaling. Most proposed QoS-routing protocols follow a source-directed link-state approach in which the source router computes paths based on its current view of network resource availability (i.e., its link-state database) and the resource requirements of the flow. Once a path is selected, the router initiates hop-by-hop signaling toward the destination. Each router performs an admission test and reserves resources on behalf of the flow. If one or more of the links cannot support the additional traffic, the flow is "blocked," and perhaps retried later with a different resource request. Once established, the path is typically pinned for the duration of the flow (in the absence of link failure), even if new, better paths become available.

Link-state updates may be distributed periodically, or triggered when the link-state metric changes by some threshold. Triggered updates are typically coupled with a hold-down timer to impose a minimum time between updates to avoid overloading the network with updates during intervals of rapid fluctuation. For example, the update period, trigger threshold, and hold-down timer could be 90 seconds, 40%, and 30 seconds, respectively. Tuning the frequency of link-state update messages introduces an important tension between overhead and performance, and has been the focus of several recent studies on QoS routing [10, 11]. With reliable flooding of link-state update messages, every router receives a copy of every update on every incoming link. This introduces substantial bandwidth and processing overheads in large backbone networks.

For a practical deployment of load-sensitive routing, link-state updates should not occur much more frequently than under traditional static routing protocols. Otherwise, the network provider would have to limit the number of routers and links in individual areas or peer groups, which would significantly limit the advantages of load-sensitive routing. However, routing based on out-of-date information severely degrades the effectiveness of load-sensitive routing, as shown in Figure 1(a). The graph plots the blocking probability for QoS routing across a range of link-state update periods, where flow durations are sampled from our ISP packet trace (with



(a) Flow blocking vs. update period

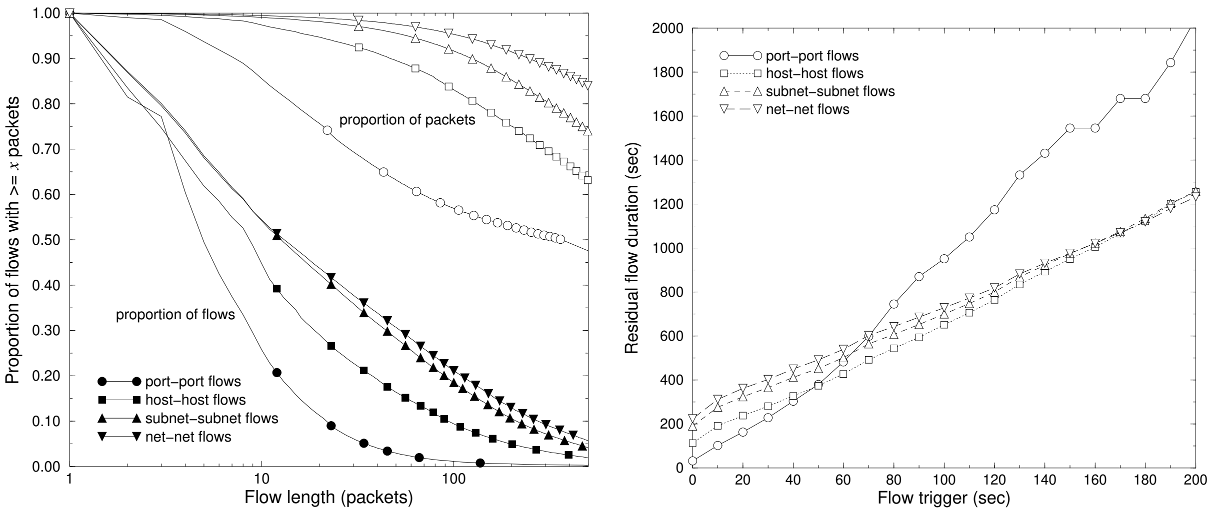
(b) Link state vs. time for 30-sec period

Figure 1: **QoS routing under stale link-state information**: The left graph shows how the performance of QoS routing is affected by increasing the update period. The graph shows minimal ($h = 0$) and non-minimal ($h = 1$) QoS routing, as well as static routing based on minimum-hop paths. The right graph illustrates the fluctuation in utilization for a single link over time with an update period of 30 seconds.

traffic aggregated to the port level). The flow-size distributions are shown in Figure 2(a), and the details of the experimental set-up are described later in Section 4. The graph illustrates both the effectiveness of load-sensitive routing under accurate link-state information, and the degradation in performance for reasonable update periods in the tens of seconds.

Increasing the update period or trigger threshold reduces the link-state update frequency, but it results in out-of-date information which can cause substantial route flapping and induce a router to select a suboptimal or even infeasible path. Under high update periods, flows typically block during the signaling phase, due to out-of-date information about the load on downstream links. This introduces additional set-up overhead in the network for flows that are ultimately blocked. Figure 1(b) illustrates the effects of route flapping. When the link has low utilization, a link-state update causes a rush of new traffic to the link, causing a rapid increase in the utilization until the link is nearly full. Then, many of the new flows block, and utilization remains high until the next update. After the update, new traffic avoids the link and the utilization drops as existing flows terminate, and the cycle repeats.

It is possible to reduce flapping by selecting randomly from a set of several equal paths, if available. Similar observations appear in studies of load balancing among servers with out-of-date information about server load [25]. Another technique is to use coarse-grained link metrics to increase the likelihood of "ties" among several similar (but not necessarily "best") paths. "Safety-based" QoS routing draws on knowledge of the link-state update triggering algorithm and the last advertised value to compute the probability that a link has sufficient resources. Using these probabilities routers can rank paths by their likelihood of accommodating the request [26]. These techniques avoid the problem of targeting the one seemingly best path, but only extend the range of link-state update periods under which conventional load-sensitive routing performs well. Ultimately, flapping still arises when the timescale of the arriving and departing traffic is small relative to the link-state updates. Previ-



(a) Proportion of packets in long flows (b) Average residual lifetime of long flows

Figure 2: **Heavy-tailed nature of IP flows**: The solid lines in (a) show the proportion of flows that have x or more packets for several levels of flow aggregation, plotted on a logarithmic scale. The dashed lines show the proportion of packets on the corresponding flows. For example, for port-to-port flows, only about 20% of the flows have more than 10 packets but these flows carry 85% of the total traffic. The right graph shows the average residual lifetime of flows after applying a x-second flow trigger for several levels of aggregation.

ous studies [11, 27, 28] have also shown that high-bandwidth flows and bursty flow arrivals cause even larger fluctuations in link state, requiring more frequent update messages.

2.2 Routing Short and Long Flows

To address these efficiency and stability challenges, we propose a hybrid routing scheme that performs load-sensitive routing of long-lived traffic flows, while forwarding short-lived flows on static preprovisioned paths. Focusing on the long-lived flows reduces the overheads of load-sensitive routing in three critical ways:

- **Fewer signaling operations**: Limiting load-sensitive routing to long-lived traffic substantially reduces the number of signaling operations for pinning routes, while still carrying the majority of packets and bytes on dynamically selected paths, as shown in Figure 2(a).
- **Fewer link-state update messages**: Dynamic routing of long-lived flows reduces the frequency of link-state update messages, both by reducing the number of flows that are dynamically routed, and by dramatically increasing the average flow duration, as shown in Figure 2(b).
- **Fewer route computations:** The slower changes in link-state information permit the routers to execute the path-selection algorithm less often without significantly degrading the quality of the routes. The routers can exploit efficient techniques for path precomputation rather than computing paths at flow arrival.

In addition, recent measurement studies suggest that long-lived flows have a less bursty arrival process than short-lived flows [15]. Hence, focusing on long-lived flows should reduce the variability in the protocol and computational demands of dynamic routing, and lower the likelihood that a large number of flows route to the same links before new link-state metrics are available.

Exploiting these potential gains requires careful consideration of how long-lived traffic interacts with the many short-lived flows in the network. The interaction between short-lived and long-lived flows depends on how link-state metrics are defined in our hybrid routing scheme. This motivates three key design decisions, discussed in more depth in the next section:

- **Defining link state in terms of allocated resources:** Earlier approaches to dynamic routing in packet networks operated on *measured* quantities, such as average utilization, queue length, or delay. These measured quantities can fluctuate on a fairly small timescale, due to the variability of Internet traffic, and are very sensitive to the selection of the estimation interval. In contrast, we define link state in terms of *allocated* resources on each link, which results in a more stable quantity that changes on the timescale of flow arrival or departure[1].

- **Distributing link state for dynamically-routed traffic:** The link state could conceivably reflect the resources consumed by both the short-lived and long-lived flows. This model is appropriate in an integrated services network that initiates signaling for all flows. However, explicitly allocating resources for each of the many short-lived flows would introduce significant overheads. In addition, the burstiness in the arrivals of short-lived flows would increase the variability of the link-state metric. Instead, we define link state in terms of the resources allocated to the long-lived flows.

- **Allocating resources for statically-routed traffic:** In basing link state only on the dynamically-routed traffic, our hybrid routing protocol runs the risk of directing long-lived flows to links that already carry a significant amount of statically-routed, short-lived traffic. To prevent this situation, we do not permit dynamically-routed flows to be allocated the entire capacity of a link. This is similar in spirit to earlier work on *trunk reservation* [29]. The proportion of link resources that can be allocated for short-lived flows can be tailored to the proportion of traffic carried by these flows.

For example, suppose a link has capacity C. Then, our hybrid routing protocol allows long-lived flows to reserve some portion $c_\ell \leq C$ of these resources. At any time t, these dynamically-routed flows have been allocated some portion $u_\ell(t) \leq c_\ell$ of these resources.

Despite this logical partitioning of network resources, the short-lived flows should be able to consume excess capacity when the long-lived partition is underutilized. This approach is well-suited to best-effort and adaptive Internet applications, which can exploit additional bandwidth when it is available, and reduce the sending rate when the resources are constrained. The allocation of bandwidth c_ℓ exists only to control routing, and need not dictate the link-scheduling policies. This is in contrast to algorithms that separate routing policies and link resources based on traffic QoS requirements [30, 31]. In our scheme, the short-lived and long-lived flows could each select from a variety of link-scheduling and buffer-management techniques, such as class-based queuing and weighted Random Early Detection, to differentiate between flows with different performance requirements.

[1]Separating routing for long-lived traffic may in fact improve the stability of measured quantities like link utilization and queue length, allowing the use of such metrics in path selection. This would be an interesting avenue for future research.

Parameter	Definition
Flow trigger	Minimum bytes, packets or seconds before dynamic routing (e.g., 10 packets)
Flow timeout	Maximum idle period before terminating a flow (e.g., 60 seconds)
Flow aggregation	Level of address aggregation (e.g., port, host, subnet, or net)
Path threshold	Minimum number of hops beyond a shortest path (e.g., 2 hops)
Capacity partition	Proportion of link capacity allocated to long-lived flows (e.g., 55%)

Table 1: **Key parameters**: This table summarizes the key parameters controlling load-sensitive routing of long-lived flows.

For example, a router could direct all incoming traffic receiving assured service to a single queue, irrespective of whether a packet belongs to a short-lived or long-lived flow. The router need not provide per-flow scheduling or buffer management for long-lived flows, though a particular implementation could employ per-flow mechanisms to further improve performance. But, these router mechanisms are not necessary to exploit the traffic engineering benefits of separating long-lived and short-lived flows.

3 Routing and Provisioning

This section describes the details of our approach to separating short-lived and long-lived traffic, including flow detection, path selection, and network provisioning. Like most routing protocols, our approach has a number of tunable parameters that affect the network dynamics. These parameters are summarized in Table 1, which serves as the basis of our simulation study in Section 5.

3.1 Detecting Long-Lived Flows and Pinning Routes

Our hybrid routing scheme requires an effective way for the network to classify flows, and to initiate selection of a dynamic route for the long-lived traffic. Routers at the edge of the network can employ flow classification hardware [32, 33] to associate each packet with a flow, based on bits in the IP and TCP/UDP headers. Depending on the flow definition, the classifier could group packets from the same TCP connection or, more broadly, from the same host end-points or subnets. The hardware can also keep track of the number of bytes or packets that have arrived on each flow, or the length of time that the flow has been active[2]. By default, the router forwards arriving packets on the path(s) selected by the static intradomain routing protocol. For example, in OSPF, the router would forward the incoming packet to an outgoing link along a shortest-path route, based on static link weights. Then, once the accumulated size or duration of the flow has exceeded some threshold (in terms of bytes, packets, or seconds), the router selects a dynamic route for the remaining packets in the flow. Directing the remainder of the flow to an alternate path may cause packets to arrive out-of-order at the destination, but this occurs at most once during the lifetime of a long-lived flow. In addition, effective provisioning and load-balancing ensures that the default paths do not become highly congested, thus reducing the likelihood that packets on the static path arrive later than those on the dynamic path. Once the the flow is rerouted, the classifier continues to track the arriving

[2]To avoid maintaining statistics about each active flow, the ingress router could randomly sample arriving packets to determine which flows should be dynamically routed. A randomly-selected packet is more likely to belong to a long flow than a short flow, though the ingress router would occasionally perform dynamic path selection for a short-lived flow.

packets, and signals the termination of the dynamic route after the timeout period expires.

The dynamic route could be established by creating a label-switched path in MPLS, which populates the forwarding tables in the routers along the flow's new path. In this scenario, the edge router selects an explicit route and signals the path through the network, as in a traditional application of MPLS. If the network consists of ATM switches, the dynamic route would involve path selection and connection signaling similar in spirit to MPOA. The selected route could be cached or placed in a routing table, to ensure that future packets are forwarded along the selected route, until the flow timeout is reached. The dynamic path is selected based on the router's current view u_l' of the load from the dynamically-routed, long-lived traffic on each link, as well as the resources b requested for the flow. Since the router may have out-of-date link-state information, the value of u_l' may differ from the actual utilization u_l. The resource requirement b for each flow could be included in the flow classifier at the edge router (e.g., the parameters of the flow conditioner under differentiated services). Alternatively, the value of b could be implicitly associated with other parameters in the classifier, such as the port numbers or IP addresses (e.g., dedicating a fixed bandwidth to Web transfers, or to the set of users with IP addresses in the range assigned to a modem bank).

Dynamic routing of long-lived flows draws on metrics u_l that represent the reserved resources on each link. Link-state advertisements can be flooded throughout the network, as in QOSPF and PNNI, or may be piggy-backed on the messages used for the default intradomain routing protocol. Our study focuses on link bandwidth as the primary network resource, since application throughput is a critical performance issue. Although network load may be characterized by several other dynamic parameters, including delay and loss, initial deployments of load-sensitive routing are likely to focus on a single simple metric to reduce algorithmic complexity. In addition, bandwidth is an additive metric, which simplifies the computation of the link-state metric and ensures that the core routers need only store aggregate information about the resource requirements of the set of flows on each link (e.g., upon arrival of a long-lived flow, the router could update $u_l = u_l + b$). The value of b could represent the peak, average, or effective bandwidth of the flow. Since any of the short-lived or long-lived flows can consume excess link capacity, inaccuracies in estimating the resource requirements of any particular flow need not waste network bandwidth.

3.2 Selecting Paths for Long-Lived Flows

When dynamically routing a long-lived flow, the edge router can prune links that do not appear to satisfy the bandwidth requirement of the flow (i.e., $u_l' + b > c_l$). After selecting the path, the flow undergoes hop-by-hop signaling to reserve the bandwidth on each link. In the meantime, the flow's packets continue to travel on the default static path. Upon receiving a signaling message, a core router tests that the link can actually support the additional traffic (i.e., $u_l + b \leq c_l$) and updates the link state upon accepting the flow; these resources are released upon flow termination. If any of the links in the path is unable to support the additional traffic (i.e., $u_l + b > c_l$), the flow may be blocked and forced to remain on the static path. Note that, in contrast to conventional QoS-routing schemes, we do not reject a blocked flow, but instead continue to forward the flow's packets on the static, preprovisioned path. Another dynamic routing operation may be performed after the flow trigger is reached again. As an alternate policy, the flow could be accepted on dynamically-routed path, even though the resources are temporarily over-allocated.

With effective provisioning of the resource c_ℓ for long-lived flows, encountering an over-constrained link should be an unlikely event. Once the flow has been accepted, the remaining packets are forwarded along the new path. Only a subset of the long-lived flows are active at any moment, however, and others may not consume their entire allocated bandwidth across time. In particular, no bandwidth is consumed during the flow timeout period, which could be as large as 60 seconds. The presence of inactive flows can be handled by allocating a smaller amount of bandwidth for each individual flow. For example, suppose that measurement of the flow-size distribution $f(x)$ shows that long-lived flows have an average residual lifetime of ℓ_l seconds. Then, each flow could be allocated a bandwidth b that is a proportion $\ell_l/(\ell_l + 60)$ of its estimated resource requirement. In addition, the short-lived flows can capitalize on transient periods of excess capacity ($u_\ell < c_\ell$), making our scheme robust to inaccuracies in estimating the aggregate bandwidth requirements of the long-lived flows.

A variety of load-sensitive routing algorithms could be used for path selection for long-lived flows. Drawing on the insights of previous studies of load-sensitive routing [6, 29], the dynamic algorithm should favor short paths to avoid consuming extra resources in the network. Long routes make it difficult to select feasible paths for subsequent long-lived flows, and also consume excess bandwidth that would otherwise be available to the short-lived traffic. Out-of-date link-state information exacerbates this problem, since stale link metrics may cause a flow to follow a non-minimal path even when a feasible shortest path exists. To further benefit the short-lived flows, the long-lived flows are routed on paths that have the most unreserved capacity ($c_\ell - u_\ell$). In other words, amongst a set of routes of equal length, the algorithm selects the *widest* path. If the widest-shortest path cannot support the bandwidth of the new flow, the algorithm considers routes up to some $h \geq 0$ hops longer than the shortest path. By reducing the effects of stale link-state information, our proposed routing scheme should permit the use of limited nonminimal paths (e.g., $h = 1$), unlike an approach that performs dynamic routing of all flows. Path selection can be implemented efficiently using a variant of the Dijkstra shortest-path algorithm or the Bellman-Ford algorithm [34].

3.3 Trunk Reservation for Short-Lived Flows

The effectiveness of the hybrid routing policy depends on how the link resources are allocated between the short-lived and long-lived flows. In the simplest case, the network could allow dynamically-routed traffic to be allocated the entire capacity C on a link. Since path-selection favors paths with more available bandwidth, the long-lived flows would not typically consume an excessive amount of resources on any one link. However, if a link carries a large amount of short-lived traffic, dynamically routing additional traffic to this link would increase the congestion, particularly under non-uniform traffic. To avoid unexpected congestion, the routing protocol should be aware of the expected load that the statically-routed traffic introduces on each link. The network can preallocate these resources by limiting long-lived flows to some portion c_ℓ of the link capacity C. A larger value of c_ℓ provides greater flexibility to the long-lived flows, while a smaller value of c_ℓ devotes more resources to the statically-routed traffic. Despite the advantages of allocating resources for short-lived flows, selecting c_ℓ too small would increase the likelihood of "blocking" of dynamically-routed, long-lived flows. Such blocking would, in turn, increase the resources consumed by statically-routed traffic to carry the "blocked" long-lived flows. As illustrated in Section 5, the effectiveness of our hybrid routing scheme is not very sensitive to selecting a relatively large value of c_ℓ, since excess short-lived traffic can exploit underutilized resources that were allocated for long-lived flows.

Fortunately, although the traffic load may vary across time, the distribution of the number of packets and bytes in a flow is relatively stable, particularly on the timescale of dynamic path selection; hence, the flow-size distribution can be determined in advance, based on network traffic measurements. We note, though, that the flow-size distribution may not be the same on every link. For example, the distribution of flow sizes for an access link to a video server would differ from the distribution near a DNS server. These links would arguably devote different proportions of resources to short and long flows. In the core of the network, the mixing of traffic from a variety of applications and source-destination pairs should result in relatively similar flow-size distributions across different links. The simplest provisioning model divides link bandwidth in proportion to the amount of statically-routed and dynamically-routed traffic. As an example, we consider the distribution $f(x)$ of the number of packets in a flow. Suppose a flow is dynamically routed after T packets have arrived. That is, a proportion $f(T)$ of the flows are short-lived. Let ℓ_s and ℓ_l be the average number of packets in the short-lived flows (i.e., flows less than T packets long) and the long-lived flows, respectively.

All of the traffic in the short-lived flows, and the first T packets of each long-lived flow, travel on static preprovisioned shortest paths. In the absence of blocking, the remainder of the traffic in the long-lived flows travels on dynamic routes. This suggests that c_ℓ should be a fraction

$$\frac{(1 - f(T))(\ell_l - T)}{f(T)\ell_s + (1 - f(T))\ell_l}$$

of the link capacity C. The value of T should be large enough to control the fluctuations in the link-state metrics for the dynamically-routed traffic, yet small enough to ensure that a large amount of traffic can be dynamically routed. In addition, if T is too large, the capacity c_l dedicated to long-lived flows may not be large relative to the flow bandwidth b; this would introduce additional blocking of long-lived flows due to bandwidth fragmentation. Fortunately, the heavy-tailed flow-size distribution $f(x)$ ensures that c_l is at least half of the link capacity, even for large flow triggers T. Bandwidth fragmentation should not be a problem for reasonable flow trigger values. The simulation experiments in Section 5 address how to select the appropriate value of T that balances the various cost-performance trade-offs.

4 Simulation Model

In this section, we discuss the traffic model and network configuration used in the simulation experiments. The constantly changing and decentralized nature of the Internet presents unique challenges in evaluating routing protocol behavior in a large network setting. The highly variable nature of Internet traffic makes it difficult to define a "typical" scenario [35]. Bursty packet arrivals, heavy-tailed flow durations, complex TCP dynamics, and the large number of flows on each link make it virtually impossible to simulate dynamic routing at the packet level. While we necessarily introduce certain simplifications, we capture the key parameters that affect the protocol dynamics and avoid biasing the experimental results in favor of our hybrid routing scheme. Table 2 summarizes the parameters used in most of the experiments in Section 5.

Parameter	Description
Topology	Random: 100 nodes, Avg. degree 5.6, Avg. hopcount 2.8, Diameter 5
Flow arrivals	Poisson with mean arrival rate $\lambda = 11.67$ flows/sec
Flow bandwidth	Uniformly distributed as $b \sim (0, 2\bar{b})$, $\bar{b} = 1 - 2\%$ of average link capacity
Flow duration	Measured distribution with mean $\ell = 15.07$ sec for port-to-port flows
Link-state updates	Periodic (with skew) or triggered (with hold-down timer)
Routing	Dynamic: widest-shortest path with $h = 1$ Static: shortest-path with random tie-breaking

Table 2: **Simulation model**: This table summarizes the simulation parameters used in the evaluation of hybrid routing.

To evaluate the hybrid routing model, we modified `routesim`, an event-driven routing simulator written in C. `Routesim` operates at the flow level to efficiently evaluate a variety of routing policies, network configurations, and link-state update policies, as well as flow triggering and the separation of traffic into short- and long-lived flows. Additional details on `routesim` appear in [36].

4.1 Traffic Model

In choosing a traffic model, we must balance the need for accuracy in representing Internet traffic flows with practical models that are amenable to simulation of large networks. The inherent variability of flow durations, arrival processes, and flow bandwidths complicates the simulation task by making convergence unlikely within reasonable simulation time. Our approach is to make some simplifying assumptions, while remaining representative of the long-lived flows we wish to study.

Flow durations: To accurately model the heavy-tailed nature of flow durations, we use an empirical distribution from a one-week packet trace collected in June 1997 from a single access point of the AT&T WorldNet network. The trace consisted of $795,446$ port-to-port, $199,638$ host-to-host, $87,336$ subnet-to-subnet, and $51,046$ net-to-net flows, each using a 60-second timeout. Net- and subnet-level flows are classified in the trace by matching the first two or three octets, respectively, of the source and destination IP addresses. As shown in Figure 2(a), the data exhibits a heavy tail both in terms of the flow duration and the traffic volume relative to the number of flows. Such variability in the traffic introduces a fundamental challenge in simulation, requiring extremely long runs to reach convergence while assuring that the distribution is fully sampled and simulated. For this reason, we truncate the distributions at $1,000$, $1,500$, $2,000$, and $2,000$ seconds, which still accounts for 99.8%, 99.3%, 99.3%, and 99.0% of the port-to-port, host-to-host, subnet-to-subnet, and net-to-net flows respectively. Note that this understates the advantages of our hybrid routing scheme, which actually benefits from the very long-lived flows in the tail of the distribution.

Flow arrivals: Each router in the network generates flows according to a Poisson process with rate λ, with a uniform random selection of the destination router. The value of λ, is chosen to fix the offered network load, ρ, at a particular value ($\rho = 0.80$ in most of our experiments). This assumption slightly overstates the performance of the traditional dynamic routing scheme, which would normally have to deal with more bursty arrivals of short-lived flows. Burstiness in the flow-arrival process tends to degrade the performance of load-sensitive routing, particularly under out-of-date link-state information. Long-lived flows typically have a less bursty arrival process [15]. Hence, this assumption slightly biases our results in favor of traditional dynamic routing.

Flow bandwidth: Flow bandwidth is uniformly distributed with a 200% spread about the mean $\bar{b}$ to reflect heterogeneity in the traffic. The value of $\bar{b}$ is chosen to be about $1-5\%$ of the average link capacity. Smaller bandwidths, while perhaps more realistic, inflate simulation time significantly since many more flows must arrive for the links to reach the high utilization regime we are interested in. Higher bandwidth values may also be more representative of aggregated flows, which would consume a larger portion of link capacity. For simplicity, we model flows as consuming a fixed bandwidth for their duration. With a flow arrival rate λ at each of N routers, the offered load, ρ, may be computed as $\lambda N \ell \bar{b} \bar{h} / L$, where ℓ is the mean flow duration, $\bar{h}$ is the average path length between source-destination pairs, and L is the number of network links.

4.2 Provisioning and Capacity Allocation

In evaluating our hybrid routing scheme, we focus on a network where static routing is well-matched to the volume of traffic between the end-points. In a production network this provisioning is typically done on a very coarse timescale by a network administrator who configures link weights (e.g., in OSPF) or tagged routes (e.g., in MPLS) to control the distribution of traffic over the links in the network. We follow a slightly different approach of first selecting a target topology, and then sizing the link capacities so that link utilization is uniform throughout the network, similar to the approach in [10]. This allows us to focus on the effects of out-of-date link-state information in a well-provisioned network, rather than the ability of load-sensitive routing to circumvent hot-spots. The experiments in Section 5.3 also consider non-uniform traffic to investigate the potential limitations of load balancing from routing short-lived flows on static paths.

Network topology: Our evaluation model focuses on backbone networks with relatively high connectivity keeping with the trend towards more highly connected networks. Rather than considering regular graphs, which may hide important effects of heterogeneity and non-uniformity, we consider a 100-node random topology, generated using Waxman's model [37, 38]. The topology has degree 5.6, diameter 5, and an average hopcount between each source-destination pair of 2.8. In assuming that propagation and processing delays are negligible, our model focuses on the primary effects of stale link-state information on path selection. This assumption slightly biases our results in favor of the traditional dynamic routing algorithm, which would suffer additional route flapping under small update periods due to the feedback delays. These effects are much less significant for our hybrid scheme, since propagation delays are negligible relative to the duration of long-lived flows. Each link introduces a small random component to the generation of successive updates to prevent synchronization [39].

Network provisioning: After computing all shortest paths between each pair of routers, we determine the traffic volume on each link, assuming that a source communicates with equal frequency with each destination, and set its capacity proportional to $\lambda \ell \bar{b}$. When multiple shortest paths are present between a node-pair, links on those paths are dimensioned assuming a uniform random selection among the paths (i.e. links are assigned capacity to handle an equal fraction of the total traffic volume between the node-pair). Provisioning in this way essentially produces a load-balanced network with no "hot spots." Note also that considering a well-provisioned network

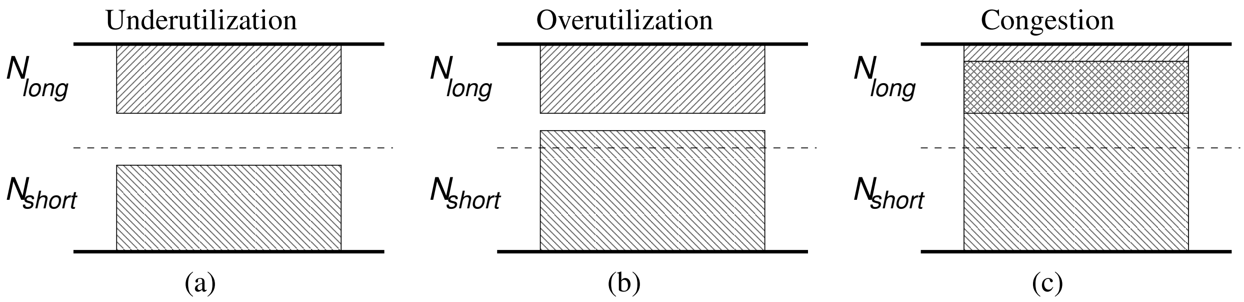


Figure 3: **Dynamic sharing of link bandwidth**: This figure shows three examples of the load on N_{short} and N_{long}. Congestion occurs when the resources required by both sets of traffic exceed the total link capacity.

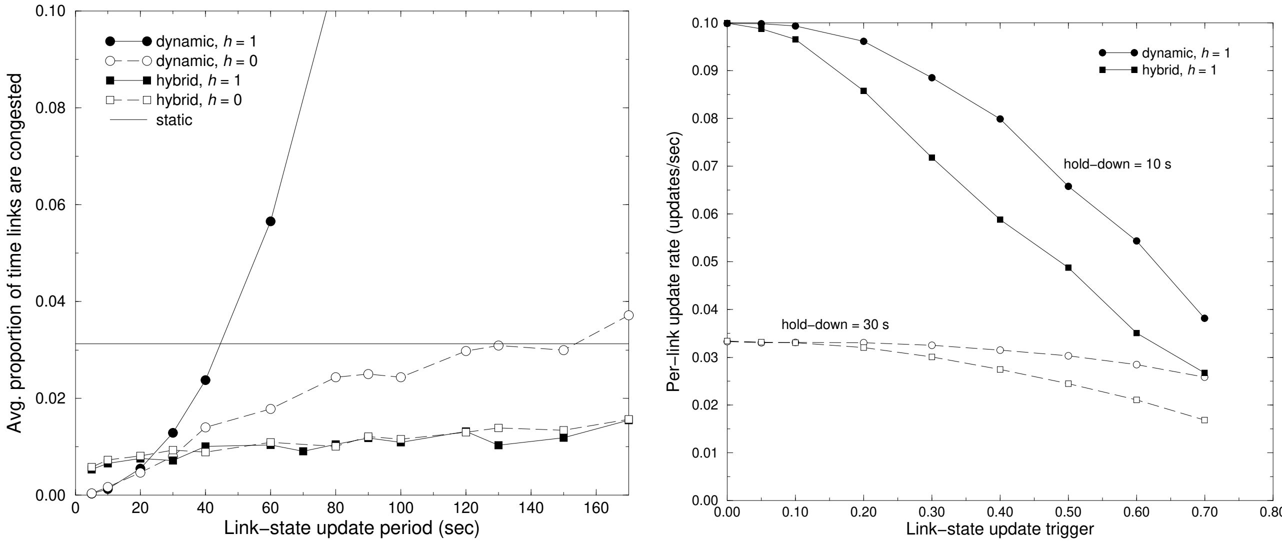
creates a best-case scenario for static routing. This understates the ability of dynamic routing to adapt to shifts in the underlying traffic matrix.

Resource allocation: After sizing the network links, we use the duration distribution to allocate link capacity to short and long flows. Choosing a particular flow trigger (in seconds) determines the proportion of flows that are considered short or long, as well as the mean duration of flows of either type. Average residual lifetime of flows after applying a particular trigger (i.e., mean duration of long flows) is shown in Figure 2(b). Capacity is then allocated to each traffic type as described in Section 3.3, except that the flow trigger is time-based rather than packet-based. Hence, we implicitly assume that the number of bytes in a flow is proportional to its duration. Although this ignores effects that might inflate the lifetime of a flow (e.g., a slow bottleneck link, or TCP retransmissions due to loss), the collected data generally supports this assumption.

4.3 Performance Metrics

While most QoS routing studies consider a traditional blocking model, our focus in evaluating hybrid routing is on a non-blocking model, where a flow defaults to a static shortest-path route when the dynamically-selected path cannot support the additional traffic. This model is more suited to Internet applications that can adapt to changes in available network bandwidth.

In the simulation experiments, we denote the short-lived and long-lived traffic classes as N_{short} and N_{long}, respectively. That is, each link has capacity c_s allocated for N_{short}, and c_l allocated for N_{long}, as described in Section 3.3. The scenarios in Figure 3 illustrate the way link capacity is shared between short and long flows. When a link is not congested, both sets of flows have sufficient resources. Even if the short-lived traffic exceeds the capacity of N_{short}, the traffic is able to consume bandwidth allocated to N_{long}, as shown in Figure 3(b). Congestion only arises in Figure 3(c), when the aggregate traffic exceeds the link capacity. The goal of our routing scheme is to limit the proportion of time that a link is in this state, through appropriate network provisioning and effective routing policies for N_{long}. Therefore, rather than plotting a blocking probability, most of the results in Section 5 show the proportion of time that the aggregate bandwidth requirements of the flows routed to a link exceeds the capacity, as illustrated in Figure 3(c).



(a) Link-state update periods

(b) Link-state triggers (for $h = 1$)

Figure 4: **Effects of link-state staleness**: The left graph measures routing performance as the update period increases. The right graph shows the update rate with triggered updates subject to two different hold-down timers. In both experiments, $b \sim (0.0, 0.01)$, relative to average link capacity, $\rho = .80$, and the hybrid scheme uses a flow trigger of 20 seconds. The port-to-port duration distribution has $\ell = 15.07$ sec, and $\lambda = 11.7$ flows/sec, and load $\rho = 0.80$.

5 Performance Evaluation

This section evaluates our approach to load-sensitive routing based on detailed simulation experiments. We show that, in contrast to traditional dynamic routing, our hybrid scheme performs well under a wide range of link-state update frequencies. Then, we investigate how to further improve the performance of our scheme by tuning the flow trigger to the timescale of link-state update messages. Finally, we show that our approach is robust to inaccuracies in network provisioning under both uniform and non-uniform traffic.

5.1 Stale Link-State Information

To investigate the effects of stale link-state information, Figure 4(a) plots routing performance as a function of the link-state update period for static routing, dynamic routing, and our hybrid scheme. The performance trends as a function of the link-state update period are similar to the earlier results for the blocking model in Figure 1(a) for traditional static and dynamic routing.

Under small link-state update periods, traditional load-sensitive routing typically outperforms static routing and our hybrid scheme, as shown in Figure 4(a). However, performance degrades dramatically under larger link-state update periods. The poor performance of traditional dynamic routing under stale link-state information is also exacerbated by the consideration of nonminimal routes, as shown by the differences between the $h = 0$ and $h = 1$ curves. The $h = 1$ curve in Figure 4(a) eventually flattens at a level of about 0.15. Route flapping increases the likelihood that shortest-path routes are busy, and out-of-date information can also cause the dynamic routing algorithm to select a nonminimal route even when a feasible shortest-path route is available. In fact, for

Aggregation	Proportion of flows	Proportion of traffic	Residual duration
port	.15	.60	83 s
host	.39	.82	141 s
subnet	.51	.87	184 s

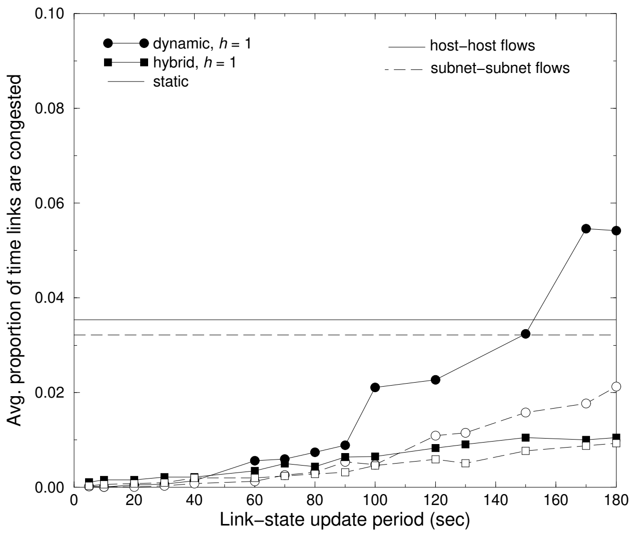
Table 3: **Long flows at different levels of aggregation**: This table summarizes the characteristics of long-lived flows used in hybrid routing experiments with a 20 second flow trigger under various aggregation levels. The values reflect truncation of the duration distribution as described in Section 4.1.

reasonable link-state update periods in the tens of seconds, static routing is preferable to traditional dynamic routing. Even when dynamic routing is restricted to shortest paths ($h = 0$), performance degrades significantly as the link-state update period increases.

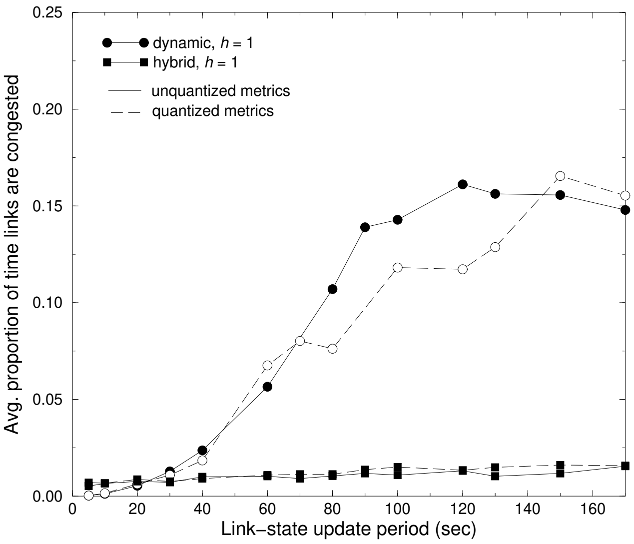
In contrast, using a flow trigger of 20 seconds allows our hybrid scheme to perform well across a wide range of update periods. Under accurate link-state information, the hybrid scheme performs slightly worse than traditional dynamic routing, since the short-lived flows (and the first 20 seconds of the long-lived flows) are forced to travel on static routes. In this regime, link-state updates are distributed so frequently that flapping due to staleness is unlikely under any of the routing schemes. Under more reasonable update periods, however, the effectiveness of the hybrid scheme becomes clear, as the performance does not degrade much even for periods in the range of a few minutes. By applying dynamic routing only to the long-lived flows, the hybrid routing scheme does not suffer from rapid fluctuations in link state, and can better exploit the presence of nonminimal routes. For the uniform traffic load considered in this experiment, the ability to choose nonminimal routes does not significantly improve performance, but the additional routing flexibility is critical to adapting to fluctuations in the underlying traffic pattern.

To investigate the influence of alternate link-state update policies, Figure 4(b) considers a range of link-state update triggers, subject to a hold-down timer. A trigger of 0.0 corresponds to periodic updates, based on the hold-down timer. In this case, the performance matches the results in Figure 4(a) for periods of 10 and 30 seconds, respectively. Consistent with previous work, we find that larger update triggers do not change the likelihood that the routing algorithm finds a feasible route, though staleness does increase the likelihood that failures are discovered during path set-up rather than path selection [11]. Since performance does not vary with the link-state update trigger, we only plot the resulting link-state update rate. As expected, larger hold-down timers and larger triggers both result in fewer link-state updates. The decrease in the number of dynamically-routed flows, and the reduction of route flapping, ensures that the hybrid scheme has a lower link-state update frequency than traditional dynamic routing. Though it might be expected that the hybrid scheme would have a considerably lower triggered update rate (since updates are for only a fraction of the traffic), the partitioning of network bandwidth under the hybrid scheme limits the reduction. For example, a 30% change in the available bandwidth on N_{long} corresponds to less than a 30% change relative to the entire link capacity. Still, despite this effect, the hybrid scheme achieves an appreciably lower link-state update rate than the traditional dynamic scheme.

We also consider the effects of link-state staleness when traffic is further aggregated to the host or subnet levels while keeping the flow trigger at 20 seconds, as shown in Figure 5(a). Characteristics of long-lived flows for different levels of aggregation are shown in Table 3. As the aggregation increases, the performance advantage



(a) Increased traffic aggregation

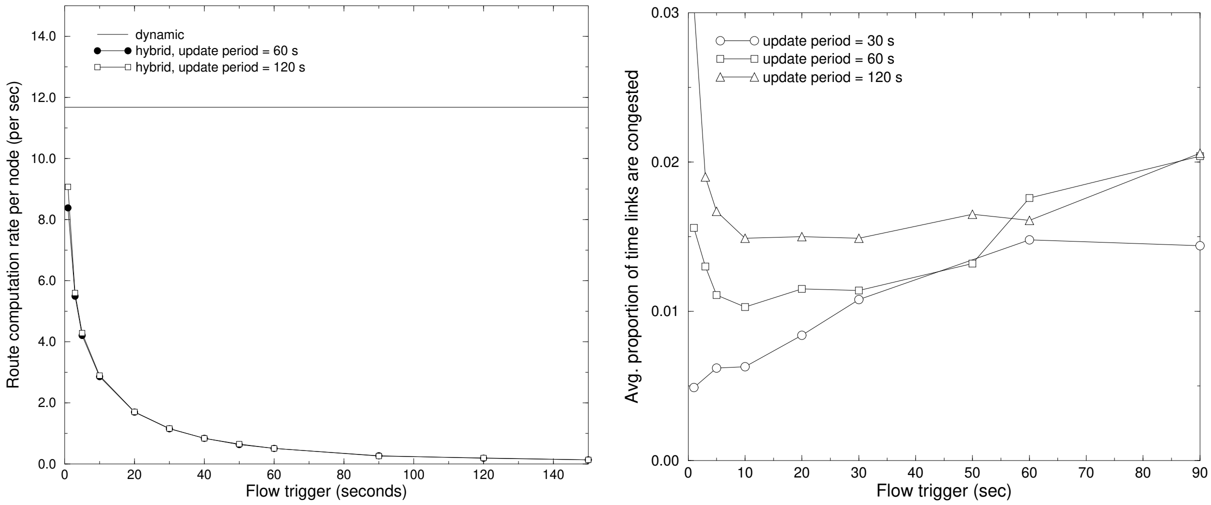


(b) Quantized link metrics

Figure 5: **Aggregation and link metric quantization**: In (a), the average durations ℓ, are 57.4 and 95.4 seconds and λ is 3.07 and 1.85 flows/sec for host- and subnet-level flows, respectively. Other parameters are as in Figure 4(a). Table 3 shows the resulting characteristics of the long-lived flows. The right graph compares performance of traditional dynamic and hybrid routing using coarse and fine-grained link-state metrics, again with simulation parameters the same as Figure 4(a).

of hybrid routing over dynamic routing diminishes somewhat but is still significant, especially in the case of host-to-host flows. Recall from Figure 2(b) that when flow triggers are small, the residual average duration increases with more aggregation. Since the overall flow duration is longer, dynamic routing suffers less from flapping, and its performance is improved. For example, there is no crossover between traditional dynamic routing and static routing when traffic is aggregated to the subnet level. Hybrid routing still outperforms dynamic and static routing, though, for larger update periods.

As mentioned in Section 2.1, previous work shows that coarse-grained or quantized link metrics can improve the ability of dynamic routing to use alternate minimum-hop routes by increasing the likelihood of "ties" between routes[10, 40]. In the experiment results shown in Figure 4, both traditional dynamic routing and hybrid routing use unquantized link metrics to differentiate between shortest-path routes. To investigate the benefits of coarse-grained metrics in improving the relative performance of traditional load-sensitive routing versus hybrid routing, we conducted additional experiments. We use link-state update values that are quantized into equal classes of 50% of the total bandwidth request range (as described in [10]) with $h = 1$. Thus, if flows require bandwidth in the range $(0, 2b)$, the available link bandwidth is advertised as the midpoint of the classes $(0, b), (b, 2b), (2b, 3b), \ldots$ As illustrated in Figure 5(b), quantization reduces route flapping for dynamic routing somewhat, but the performance gains are minor, particularly under update periods which are very long relative to the average flow duration.



(a) Route computation rate

(b) Time in violation

Figure 6: **Choice of flow trigger**: This graph illustrates the tradeoff in choosing the flow trigger for varying degrees of link-state inaccuracy. The traffic parameters are identical to those in Figure 4(a).

5.2 Detecting Long-Lived Flows

The cost-performance trade-offs of our hybrid scheme relate directly to the selection of the flow trigger. Figure 6(a) illustrates that the hybrid scheme substantially reduces the frequency of dynamic route computations, particularly for larger values of the flow trigger. This result stems directly from the reduction in the number of flows that are dynamically routed. The link-state update period has a very small, second-order effect. Under out-of-date link-state information, the hybrid scheme occasionally selects an already-congested route on N_{long}, forcing a subsequent routing attempt after activating the flow trigger a second time. This effect is very small for two reasons. First, since relatively high link-state periods do not appreciably degrade the performance of the hybrid scheme, these recomputations occur very infrequently. Secondly, the flow trigger is large enough to ensure that the second routing attempt will be initiated only after new link-state information is available, particularly when the flow trigger is larger than the link-state update period. As a result, two routing attempts do not typically operate on the same link-state database, in contrast to traditional rerouting or "crankback" operations that are likely to draw on the same information about most of the links. Implicitly introducing delay between the successive routing attempts increases the likelihood of selecting a feasible route.

Despite the advantages of large flow triggers in reducing computational overheads, a smaller flow trigger ensures that more traffic can be routed based on load. The graph in Figure 6(b) investigates how to select the flow trigger to strike the best balance between stability and adaptiveness in the hybrid scheme. Each setting of the flow trigger corresponds to a different division of the network resources between N_{long} and N_{short}, following the provisioning rule in Section 3.3. For a range of link-state update periods and network configurations, the graphs have roughly a cup shape, with worse performance for smaller and larger flow triggers. A small flow trigger allows dynamic routing of a larger proportion of the traffic, at the risk of greater sensitivity to stale link-state information. The flow trigger controls these staleness effects by determining the residual duration distribution for the flows on N_{long}. Hence, the flow trigger should be chosen such that most flows on N_{long} have a residual lifetime that is large relative to the link-state update period. When the update period is small (e.g., 30 seconds), choosing small flow triggers that assign more traffic to dynamic routes improves performance since dynamic routing does not suffer from much flapping in this regime.

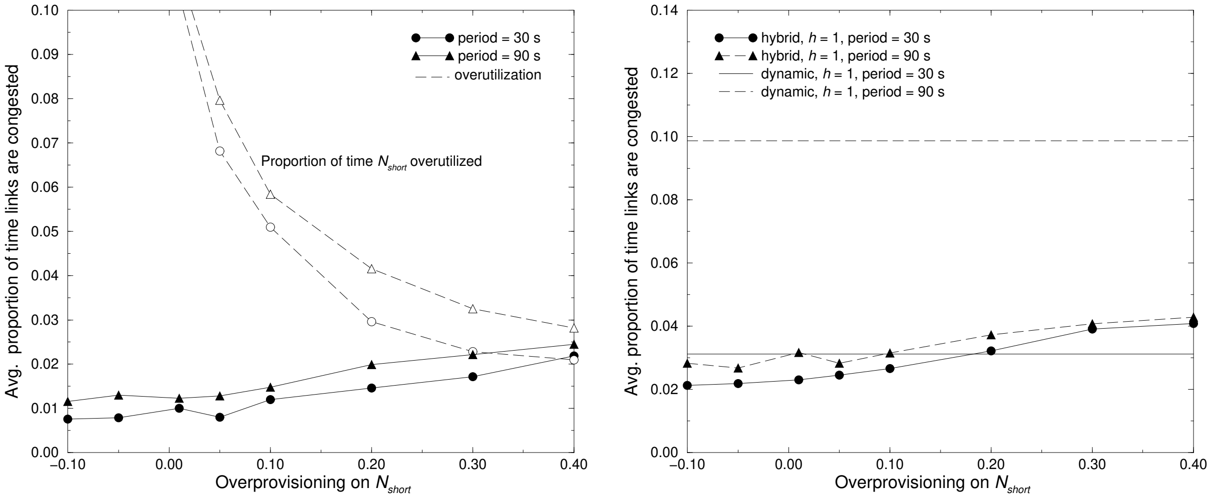
The control over stale link-state information afforded by large flow triggers also comes at a cost. A large flow trigger limits the proportion of traffic that is dynamically routed, which degrades the ability of the hybrid algorithm to respond to fluctuations in the offered traffic load. In addition, large flow triggers effectively allocate fewer resources to N_{long}, which increases the likelihood of bandwidth fragmentation, resulting in more "blocking" of the dynamically-routed flows. These effects are demonstrated by the gradual rise of the curves in Figure 6(b) for larger flow triggers. The degradation in performance is not very significant, since an increase in the flow trigger does not have a very significant impact on the proportion of traffic on N_{long}, due to the heavy tail of the flow-size distribution. For example, flow triggers of 20 seconds and 40 seconds place 60% and 47% of the traffic on N_{long}, respectively. Still, to maximize the hybrid algorithm's ability to react to shifts in traffic load, the flow trigger should be made as small as possible, subject to the link-state update period and the target route computation rate.

5.3 Network Provisioning

The previous subsection illustrated that our hybrid routing scheme is robust across a range of flow triggers. In the next experiment, we evaluate the sensitivity of our scheme to inaccuracies in allocating resources between N_{short} and N_{long}. Figure 7(a) plots the performance of our hybrid scheme with a 20-second flow trigger across a range of over-provisioning factors, for a uniform traffic pattern. An over-provisioning factor of 0 corresponds to the earlier simulation results, using the provisioning rule in Section 3.3 to divide link bandwidth between N_{short} and N_{long}. A larger over-provisioning factor implies additional resources devoted to N_{short}, at the expense of N_{long}, without changing the flow trigger.

The effectiveness of our hybrid scheme remains relatively undiminished as the resource allocation changes, though large over-provisioning factors typically result in slightly worse performance. This occurs because an under-provisioned N_{long} sometimes rejects flows, which are forced to stay on static routes on N_{short}. In contrast, if N_{long} is over-provisioned, the long-lived flows are rarely blocked. In either case, the selection of widest paths on N_{long} helps ensure that the dynamically-routed traffic does not consume the full allocation of each link, allowing the short-lived flows to consume the excess capacity. Links are congested about 1–2% of the time for the uniform traffic pattern in Figure 7(a). During these infrequent periods of congestion, the average amount of excess traffic (not shown) is approximately 8% across a range of over-provisioning factors. This implies that the network experiences only brief periods of minor congestion, even when the bandwidth provisioning rule does not exactly match the proportion of short-lived and long-lived traffic.

Despite the potential advantages of under-provisioning the resources on N_{short}, devoting too much of the link resources to long-lived flows makes the network more vulnerable under changes in the traffic pattern, as suggested by the dashed lines in Figure 7(a), which plots the proportion of time that the short-lived traffic exceeds its allocation on a link in N_{short}. Typically, when the allocation of resources for short-lived flows is overutilized,



(a) Uniform traffic

(b) Non-uniform traffic

Figure 7: **Over-provisioning for short-lived flows**: These experiments evaluate the hybrid scheme under uniform and non-uniform traffic patterns for different link-state update periods. In (b) we also compare to traditional dynamic routing in the presence of hot-spots. The average traffic parameters are equal to those in Figure 4(a). The hybrid scheme uses a flow trigger of 20 seconds.

the corresponding resources for long-lived flows are underutilized, and the link is not actually congested. But, this does not necessarily remain true under shifts in the traffic pattern. To quantify the risk of under-provisioning N_{short}, we experimented with a non-uniform traffic pattern, as shown in Figure 7(b). Each router sends 20% of its traffic to a randomly-selected set of 15 destinations; these "hot" destinations receive 30% more traffic than in the earlier experiments. The rest of the traffic is evenly distributed amongst the remaining destinations.

The non-uniform traffic increases the proportion of time that links are congested, as shown by the higher curve in Figure 7(b). The links are congested 2–4% of the time, across the range of over-provisioning factors. But, the degree of congestion increases when N_{short} is under-provisioned. For example, the congested links have an average of 16% excess traffic for a over-provisioning factor of -0.10, compared to 11% for an over-provisioning factor of 0.30 (not shown). When N_{short} does not have sufficient resources, a significant amount of excess traffic can be dynamically routed to certain links. This occurs because the network continues to route long-lived flows to congested links on N_{long}, even when the link is already busy. A larger over-provisioning factor on N_{short} effectively acts as a form of trunk reservation [29] that controls the proportion of link resources that are devoted to load-sensitive routing. This helps ensure stability under fluctuations in offered traffic. These results suggest that the network resources should be divided in proportion to the amount of short-lived and long-lived flows, with perhaps a slight over-provisioning of N_{short}, as discussed in Section 3.3. More importantly, across a range of provisioning rules, the hybrid scheme significantly outperforms traditional dynamic routing on the non-uniform traffic pattern, particularly under larger link-state update periods.

6 Conclusion

Internet service providers face difficult challenges in engineering large backbone networks, due to wide fluctuations in the underlying traffic and increasing user demands for predictable communication performance. Dynamic routing can play an important role in traffic engineering of ISP networks, if selecting routes based on load can be made both stable and efficient. In this paper, we have introduced a dynamic routing scheme that exploits the extreme variability in IP flow durations by performing dynamic routing of long-lived flows, while forwarding short-lived flows on static preprovisioned paths. Route stability is achieved by relating the detection of long-lived flows to the timescale of the link-state update messages in the routing protocol. Link bandwidth resources are allocated between the two traffic classes based on measurements of the flow-size distribution, and the trigger used for detecting long-lived flows. Our simulation experiments demonstrate that the proposed hybrid scheme significantly outperforms traditional static and dynamic routing algorithms, and can operate effectively under reasonable link-state update periods in the range of 60–180 seconds. In addition, we show that our scheme is robust to inaccuracies in network provisioning and shifts in the offered traffic.

Our results suggest a number of interesting avenues for further research. Collection and analysis of additional Internet traces would allow study of specific aspects of our hybrid routing scheme in greater detail. For example, measuring and analyzing the traffic volume between pairs of routers in the network would provide insight into how much the offered load fluctuates, and on what timescale. Other issues problems include the provisioning rules for short-lived flows and the possibility of routing long-lived flows based on measured link utilization rather than reserved bandwidth, as well as the impact of TCP dynamics on load-sensitive routing. We are also interested in investigating generalizations of our hybrid routing scheme, including dynamic selection of the flow trigger based on the traffic characteristics, and support for precomputation of load-sensitive routes to avoid the processing overheads and set-up delays introduced by on-demand path selection. Finally, the hybrid routing scheme fits well into the MPLS constraint-based routing architecture, and this integration offers an interesting problem in protocol design. These studies can provide insight into how to best exploit our flexible and efficient approach to load-sensitive routing of IP traffic.

References

[1] A. Khanna and J. Zinky, "The revised ARPANET routing metric," in *Proceedings of ACM SIGCOMM*, pp. 45–56, September 1989.

[2] Z. Wang and J. Crowcroft, "Analysis of shortest-path routing algorithms in a dynamic network environment," *ACM Computer Communication Review*, vol. 22, pp. 63–71, April 1992.

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

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

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

[6] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Network Magazine*, pp. 64–79, November/December 1998.

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

[8] 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.

[9] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, A. Przygienda, and D. Williams, "QoS routing mechanisms and OSPF extensions." Internet Request for Comments (RFC 2676), April 1999.

[10] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Quality of service based routing: A performance perspective," in *Proceedings of ACM SIGCOMM*, September 1998.

[11] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the overheads of source-directed quality-of-service routing," in *Proceedings of IEEE International Conference on Network Protocols*, October 1998.

[12] 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.

[13] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 835–846, December 1997.

[14] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network Magazine*, vol. 11, pp. 10–23, November/December 1997.

[15] A. Feldmann, J. Rexford, and R. Caceres, "Efficient policies for carrying Web traffic over flow-switched networks," *IEEE/ACM Transactions on Networking*, pp. 673–685, December 1998.

[16] M. Harchol-Balter and A. Downey, "Exploiting process lifetime distributions for dynamic load balancing," *ACM Transactions on Computer Systems*, vol. 15, pp. 253–285, August 1997.

[17] Y. Katsube, K. Nagami, S. Matsuzawa, and H. Esaki, "Internetworking based on cell switch router - architecture and protocol overview," *Proceedings of the IEEE*, vol. 85, pp. 1998–2006, December 1997.

[18] ATM Forum MPOA Sub-Working Group, *Multi-Protocol over ATM Version 1.0 (AF-MPOA-0087.000)*, July 1997.

[19] P. Newman, G. Minshall, and T. Lyon, "IP switching: ATM under IP," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 117–129, April 1998.

[20] S. Lin and N. McKeown, "A simulation study of IP switching," in *Proceedings of ACM SIGCOMM*, pp. 15–24, September 1997.

[21] I. Widjaja, H. Wang, S. Wright, and A. Chatterjee, "Scalability evaluation of multi-protocol over ATM (MPOA)," in *Proceedings of IEEE INFOCOM*, March 1999.

[22] H. Che and S.-Q. Li, "MPOA flow classification design and analysis," in *Proceedings of IEEE INFOCOM*, March 1999.

[23] C. Villamizar, "OSPF optimized multipath (OSPF-OMP)." Internet Draft (draft-ietf-ospf-omp-02), work in progress, February 1999.

[24] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS." Internet Draft (draft-awduche-mpls-traffic-eng-01.txt), work in progress, June 1999.

[25] M. Mitzenmacher, "How useful is old information?," in *Proceedings of ACM Symposium on Principles of Distributed Computing*, (Santa Barbara, CA), pp. 83–91, August 1997. Extended version to appear in *IEEE Transactions on Parallel and Distributed Systems*.

[26] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, “Improving QoS routing performance under inaccurate link state information,” in *Proceedings of International Teletraffic Congress (ITC-16)*, (Edinburgh, UK), June 1999.

[27] G. Apostolopoulos and S. K. Tripathi, “On reducing the processing cost of on-demand QoS path computation,” in *Proceedings of IEEE International Conference on Network Protocols*, (Austin, TX), pp. 80–89, October 1998.

[28] 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.

[29] R. J. Gibbens, P. J. Hunt, and F. P. Kelly, “Bistability in communication networks,” *Disorder in Physical Systems*, 1990.

[30] Q. Ma and P. Steenkiste, “Supporting dynamic inter-class resource sharing: A multi-class QoS routing algorithm,” in *Proceedings of IEEE INFOCOM*, (New York, NY), March 1999.

[31] I. Matta and A. U. Shankar, “Type-of-service routing in datagram delivery systems,” *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1411–1425, October 1995.

[32] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, “Fast scalable algorithms for level four switching,” in *Proceedings of ACM SIGCOMM*, September 1998.

[33] T. V. Lakshman and D. Stiliadis, “High-speed policy-based packet forwarding using efficient multi-dimensional range matching,” in *Proceedings of ACM SIGCOMM*, September 1998.

[34] R. Guerin, A. Orda, and D. Williams, “QoS routing mechanisms and OSPF extensions,” in *Proc. Global Internet Miniconference*, November 1997.

[35] 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.

[36] A. Shaikh, *Efficient Dynamic Routing in Wide-Area Networks*. PhD thesis, University of Michigan, May 1999.

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

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

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

[40] A. Shaikh, J. Rexford, and K. Shin, “Efficient precomputation of quality-of-service routes,” in *Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video*, July 1998.