Load-Sensitive Routing of Long-Lived IP Flows

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

[†] Real-Time Computing Laboratory
Department of EECS
University of Michigan
Ann Arbor, MI 48109-2122
{ashaikh,kgshin}@eecs.umich.edu

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

Abstract

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 signalling. 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routing of IP traffic are ineffective, and can introduce significant route flapping, since paths are selected based on out-of-date link-state information.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]. Several QoS-routing protocols have been proposed in recent years 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. Most notably, 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 [16–18]. 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, 18–21].

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 QoS routing, 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) [17], which triggers the creation of IP shortcut connections across an underlying ATM network. In contrast to recent studies that evaluate the overheads of a single-link MPOA system [20, 21], 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. The benefits of the hybrid approach are illustrated in Section 4, 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 5 concludes the paper with a summary of our results and a discussion of future research directions, and Appendix A describes the specifics of our simulation model.

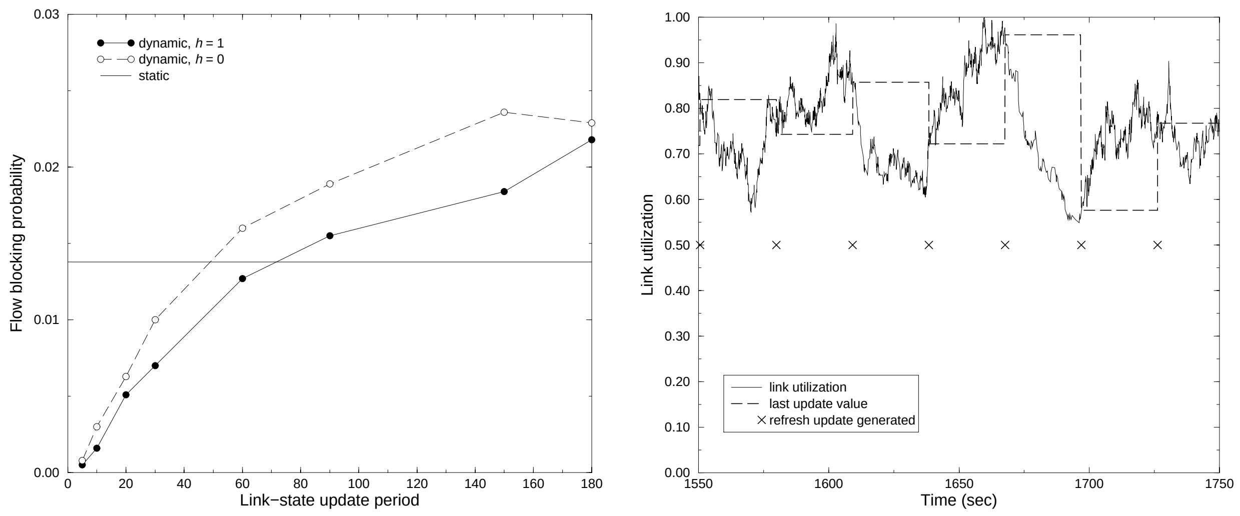
2 Stable Load-Sensitive Routing

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 [22]. 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 [23] 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.



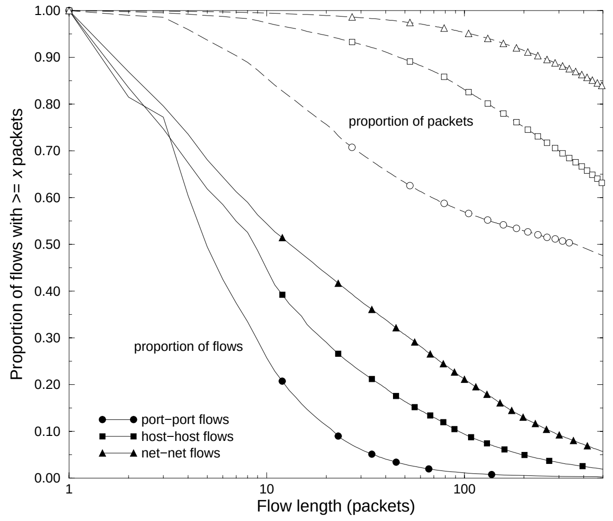
(a) Flow blocking vs. update period (b) Link state vs. time for 30-sec period

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 minimal routing. The right graph illustrates the fluctuation in utilization for a single link over time with an update period of 30 seconds.

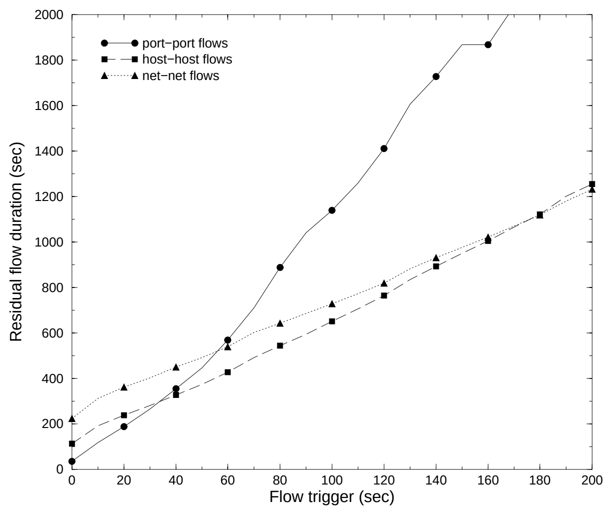
Link-state updates may be distributed periodically, or triggered when the link-state metric changes by some threshold. Triggered updates are typically coupled with a hold-down timer to impose a minimum time between updates to avoid overloading the network with updates during intervals of rapid fluctuation. 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 also 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 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 results in out-of-date information which can cause substantial route flapping and induce a router to select a suboptimal or 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 amongst a set of several paths, rather than targeting the one “best” path. But, ultimately, flapping arises when the timescale of the



(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.

arriving and departing traffic is small relative to the link-state updates. Previous studies [11, 24, 25] 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 Separating 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 only 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.

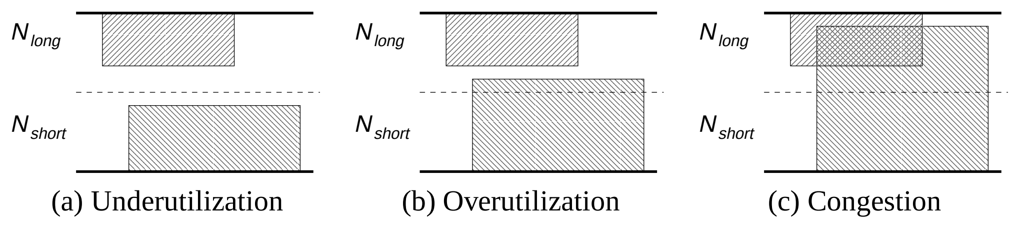


Figure 3: **Dynamic sharing of link bandwidth:** This figure shows three examples of the load on N_{short} and N_{long}. Congestion occurs when the resources required by both sets of traffic exceed the total link capacity.

Exploiting these potential gains requires careful consideration of how long-lived traffic interacts with the many short-lived flows in the network. In particular, the transient load fluctuations of the short-lived traffic should not affect the stability of the dynamic routing of the long-lived flows. Hence, we logically partition the link resources between the two traffic classes, resulting in a logical division of the traffic into N_{short} and N_{long}. That is, each link has capacity c_s and c_l for N_{short} and N_{long}, respectively, for a total link capacity of $c_s + c_l$. Long-lived flows are dynamically routed based on the utilization $u_l \in [0, c_l]$ of the link resources on N_{long}. In addition, the routing algorithm used for long-lived flows should favor paths that enhance the performance of the short-lived flows in the other partition. The division of link capacity into c_s and c_l is based on the distribution f of flow size, where $f(x)$ is the likelihood that a flow consists of no more than x bytes. The probability distribution is known in advance, based on network traffic measurements, and changes much more slowly than the offered load in the network. The policies for flow detection, route selection, and provisioning are discussed in more detail in Section 3.

Despite the 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 possible scenarios are illustrated in Figure 3, and play an important role in our performance evaluation in Section 4. 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). The allocation of bandwidth between N_{long} and N_{short} exists only to control routing, and need not dictate the link-scheduling policies. 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}.

In contrast to previous work on flow switching, the choice to separate long-lived and short-lived flows is not necessarily motivated by the QoS requirements (if any) of the individual flows, or the desire to forward packets in hardware. In fact, N_{short} and N_{long} 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. 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. While N_{long} need not provide per-flow scheduling or buffer management, a particular router 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

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)
Update period	Maximum time between link-state updates (e.g., 90 seconds)
Hold-down timer	Minimum time between link-state updates (e.g., 5 seconds)
Link-state trigger	Minimum proportional change in link state (e.g., 30% change)
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.

Table 1, which serves as the basis of our simulation study in Section 4.

3.1 Detecting Long 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 [26, 27] 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. By default, the router forwards arriving packets on the path(s) selected by the static intra-domain routing policy. 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. The flow classifier continues to track the arriving packets, and signals the termination of the dynamic route after the timeout period expires. Figure 4 summarizes the flow detection and routing procedure.

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 on the long-lived partition of 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 load u_l. The resource requirement b for each flow could be included in the flow classifier at the edge router, or implicitly associated with other parameters in the classifier, such as the port numbers or IP addresses.

Dynamic routing of long-lived flows draws on link-state metrics u_l that represent the load on N_{long}. Link-state advertisements can be flooded throughout the network, as in QOSPF and PNNI, or may be piggybacked 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. The value of b could represent the peak, average, or

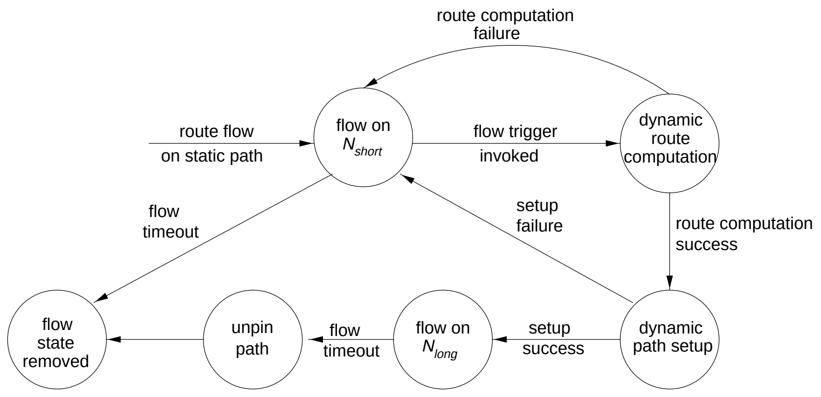


Figure 4: **Flow state machine:** This figure describes the steps in detecting and routing long-lived flows at an edge router.

effective bandwidth of the flow. Since any of the flows on N_{short} and N_{long} 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

During path selection on N_{long}, 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 path in N_{short}. 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 N_{short}. Note that, in contrast to research on QoS routing, we do not reject a blocked flow, but instead continue to forward the flow's packets on N_{short}. Another dynamic routing operation may be performed after the flow trigger is reached again. As an alternate policy, the flow could be accepted on N_{long}, even though the resources on the dynamic path are temporarily over-allocated. With effective provisioning of the long-lived partition, encountering an over-constrained link in N_{long} should be an unlikely event. Once the flow has been accepted on N_{long}, the remaining packets are forwarded along the new path.

By defining the link state u_l as the amount of *unreserved* bandwidth on N_{long}, we are following the same basic approach as previous work on QoS routing. This differs from earlier approaches to dynamic routing in packet networks, which operated on *measured* quantities, such as average utilization, queue lengths, or delay. These measured quantities can fluctuate on a fairly small time scale due to the variability in Internet traffic, and are very sensitive to the selection of the estimation interval. Reserved bandwidth is a much more stable metric.[1] Still, only a subset of the long-lived flows are active at any moment, 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 $f(x)$ show that flows on the long-lived partition 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 on N_{long}, making our scheme robust to inaccuracies in estimating the aggregate bandwidth requirements of

[1]Placing long-lived traffic on a separate partition 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. For now, we assume that the link-state metrics represent the reserved link resources.

the long-lived flows.

A variety of load-sensitive routing algorithms could be used for path selection on N_{long}. Drawing on the insights of previous studies of load-sensitive routing [6, 28], 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 traffic on N_{short}. 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 on the long-lived partition. 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 higher values of h than an approach that performs dynamic routing of all flows. Path selection can be implemented efficiently using the Bellman-Ford algorithm, with relatively simple extensions to support *precomputation* of the path(s) to each destination [29].

3.3 Provisioning Resources for Short-Lived and Long-Lived Flows

The effectiveness of the hybrid routing policy depends on how the network resources are allocated between N_{short} and N_{long}. The resources on N_{short} should be provisioned to reduce the likelihood that the traffic is constrained. Yet, over-allocating the resources for N_{short} increases the likelihood of "blocking" on N_{long}. Such blocking would, in turn, increase the load on N_{short} to carry the "blocked" 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. The simplest provisioning model divides link bandwidth in proportion to the amount of traffic slated for each partition. As an example, we focus on 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. These averages can be derived directly from $f(x)$.

All of the traffic in the short-lived flows, and the first T packets of each long-lived flow, travel on N_{short}. In the absence of blocking, the remainder of the traffic in the long-lived flows travels on N_{long}. This suggests that N_{short} should have a fraction

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

of the link capacity, assuming for the time being that the short-lived and long-lived flows have the same distribution of end points. The value of T should be large enough to control the fluctuations in the link-state metrics on N_{long}, yet small enough to ensure that a large amount of traffic can be dynamically routed. In addition, if T is very 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 on N_{long} 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. Fragmentation of the resources in N_{long} should not be a problem for reasonable flow trigger values. The simulation experiments in Section 4 address how to select the appropriate value of T that balances the various cost-performance trade-offs.

The allocation rule based purely on $f(x)$ is unnecessarily conservative for the short-lived traffic in two key ways. First, the provisioning rule does not account for any blocking of long-lived flows on N_{long}. Although the blocking probability could be simulated, and incorporated into the resource partitioning equation, the target network configuration should have a negligible blocking probability. In the unusual case when blocking occurs, subsequent routing attempts for the long-lived flow are likely to be successful. Second, since long-lived flows have greater routing flexibility, they can more easily adapt to limited link resources,

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 parameters of the simulation model.

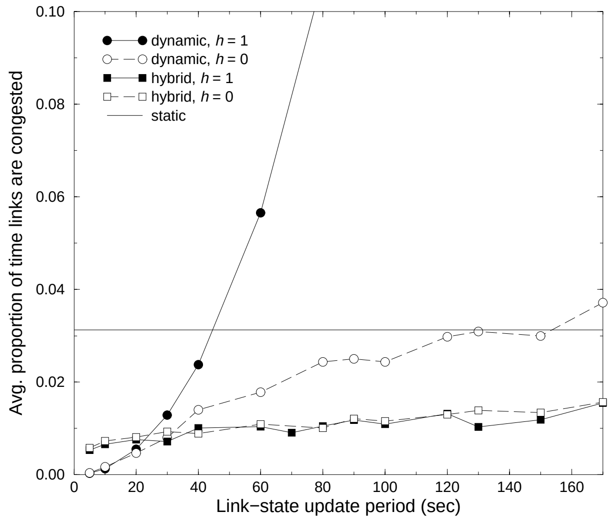
allowing a slight over-provisioning of bandwidth for N_{short}. The logical separation of network resources also helps the short-lived traffic, since N_{short} does not need to guard against the occasional arrival of a long-lived flow that would consume a large amount of bandwidth. This enables the allocation of N_{short} to focus on the average case, rather than accounting for the high variability in aggregate load introduced by a small number of long-lived flows. Finally, and perhaps most importantly, the ability of short-lived flows to exploit the excess capacity on N_{long} also improves the robustness of our scheme, by tolerating transient periods of excess load on N_{short}, as illustrated in the next section.

4 Performance Evaluation

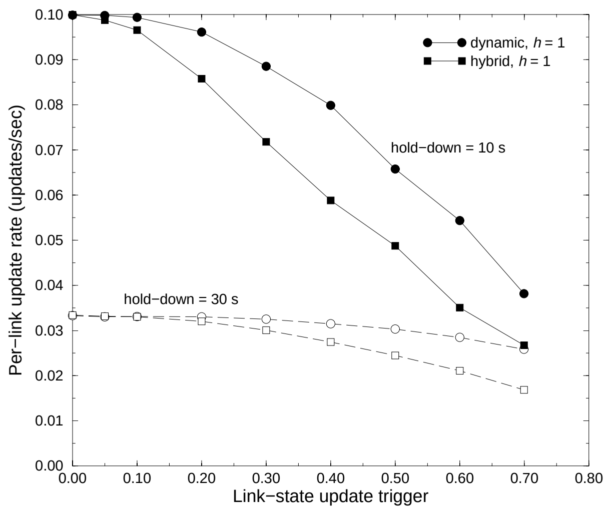
This section evaluates our approach to load-sensitive routing based on detailed simulation experiments, since tractable analytical models for studying the effects of out-of-date link-state information in large networks are elusive. After a brief description of our simulation methodology, we compare our hybrid scheme to traditional static and dynamic routing under a range of link-state update periods and triggers. 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.

4.1 Simulation Model

Our simulation model allows us to study the dynamics of load-sensitive routing, and the effects of out-of-date link-state information, under reasonable simulation overheads, while still capturing much of the Internet's variable nature. In particular, we model flow durations with an empirical distribution drawn from a packet trace from a large ISP network. The event-driven simulator operates at the flow level to efficiently evaluate a variety of routing policies and network configurations. The simulator enables us to vary the network topology, traffic patterns, routing algorithms, and link-state update policies, as well as flow triggering and the partitioning of traffic into N_{short} and N_{long}. Except where noted, our experiments evaluate a network where static routing is well-matched to the volume of traffic between the end-points. 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 4.4 also consider non-uniform traffic to investigate the potential limitations of load balancing from routing short-lived flows on static paths. Table 2 summarizes the parameters in our experiments. A much more detailed description of the simulation model is presented in Appendix A.



(a) Link-state update periods



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

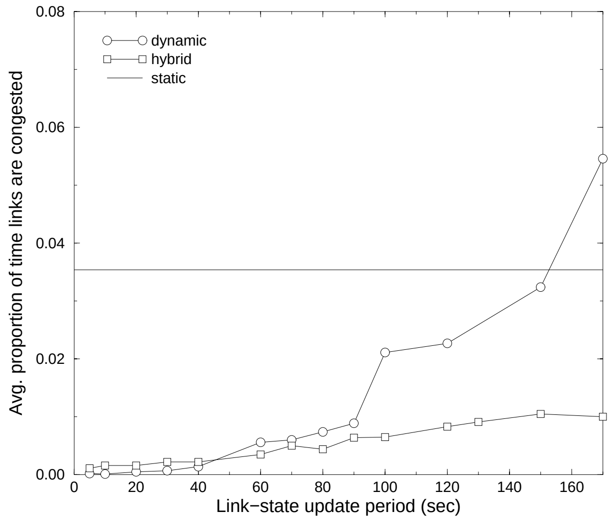
Figure 5: **Effects of link-state staleness**: In both experiments, $b \sim (0.0, 0.01)$, relative to average link capacity, a port-to-port duration distribution with $\ell = 15.07$ sec, and $\lambda = 11.7$ flows/sec, for a load $\rho = 0.80$. The hybrid scheme uses a flow trigger of 20 seconds, which places 15% of the flows and 60% of the traffic on N_{long}, and results in an average residual flow duration of 83 seconds.

4.2 Stale Link-State Information

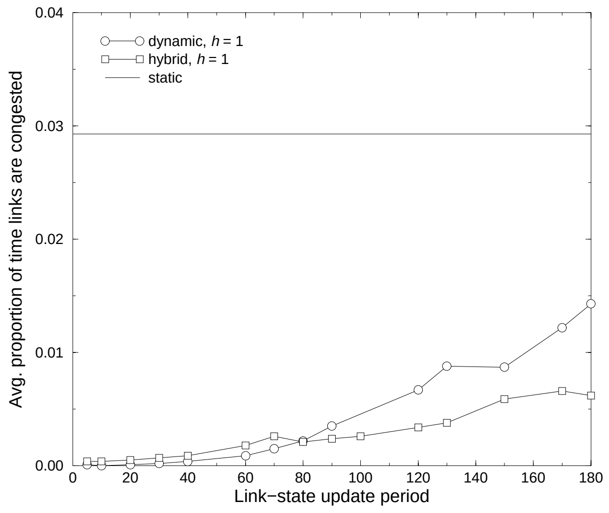
To investigate the effects of stale link-state information, Figure 5(a) plots routing performance as a function of the link-state update period for static routing, dynamic routing, and our hybrid scheme. This graph mirrors the results in Figure 1(a), except that the earlier experiment in Section 2 evaluated a network that blocks a flow after an unsuccessful attempt to reserve resources on the selected route. In contrast to the traditional blocking model in QoS routing, we focus here 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. Hence, rather than plotting a blocking probability, Figure 5(a) plots 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). 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.

By providing the greatest flexibility in path-selection, traditional load-sensitive routing typically outperforms static routing and our hybrid scheme under small link-state update periods, as shown in Figure 5(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 5(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 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



(a) host-to-host flows



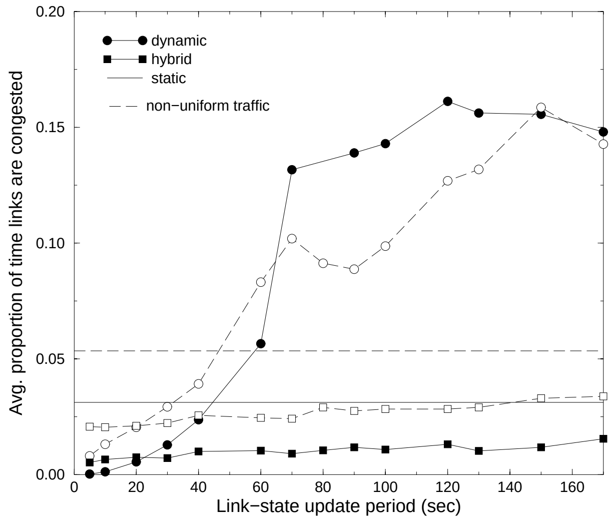
(b) net-to-net flows

Figure 6: **Effects of increased traffic aggregation**: Traffic parameters as in Figure 5 with $h = 1$. With host-level flows, a 20 second flow trigger places 39% of the flows and 82% of the traffic on N_{long}. For net-level aggregation, 53% of the flows and 89% of the traffic are on N_{long}. The average residual flow duration is 141 and 213 seconds for host-level and net-level flows, respectively.

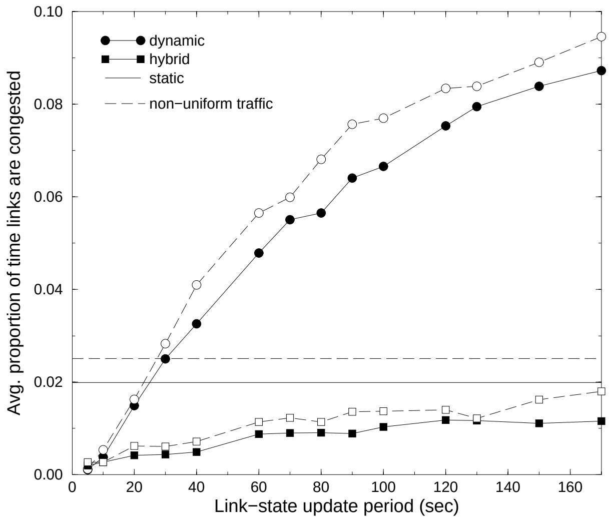
periods, however, the effectiveness of the hybrid scheme becomes clear, as the performance is not degraded much even for periods in the range of a few minutes. By applying dynamic routing only to the long-lived flows, the hybrid routing scheme does not suffer from rapid fluctuations in link state, and 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 further investigate the influence of the link-state update policy, Figure 5(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 5(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 net levels, as shown in Figure 6. As the aggregation increases, the performance advantage of hybrid routing over dynamic routing diminishes somewhat but is still significant, especially in the case of host-to-host aggregation. Observe from Figure 2 that when flow triggers are small, the residual average duration increases with more aggregation. Thus, even when all flows are dynamically routed (i.e., flow trigger of 0), the average flow duration is 57.4 seconds and 116.2 seconds for host- and net-level aggregation, respectively. Since the overall flow duration is longer, dynamic routing is able to find suitable routes more effectively,



(a) Random topology



(b) Uniform topology

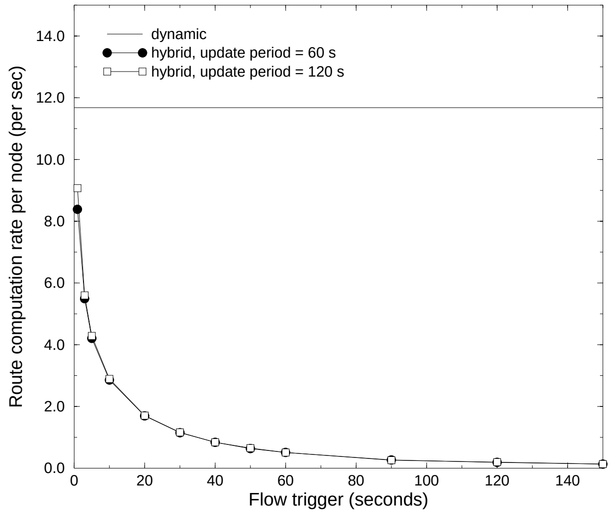
Figure 7: **Interaction of non-uniform traffic and staleness**: In the right graph with the random topology, traffic parameters are as in Figure 5. In (b), $b \sim (0.0, 0.01), \ell = 15.07$ sec, $\lambda = 7.5$ flows/sec and $\rho = 0.75$. The uniform topology has 64 nodes, 384 links, uniform node degree of 6, diameter 4, and mean path length 2.95. In both experiments $h = 1$ with port-to-port flows.

thus improving its performance. For example, there is no crossover between traditional dynamic routing and static routing in the case of net-level aggregation. Notice, however, that hybrid routing still outperforms dynamic routing and static routing for a range of reasonable link-state update periods.

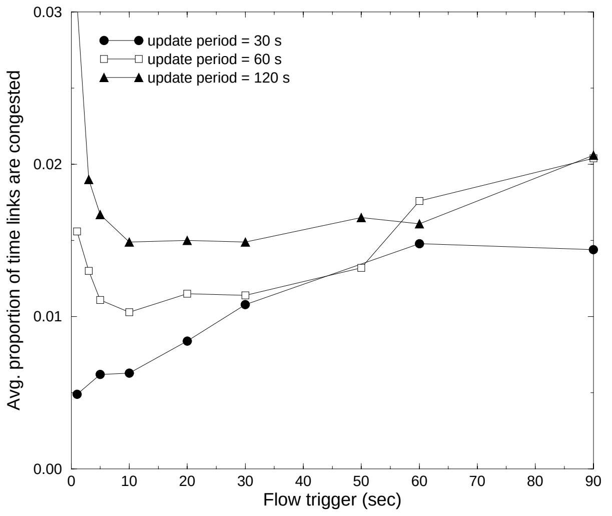
In Figure 7, we examine the performance when the traffic is non-uniform. Each router sends 20% of its traffic to a randomly-selected set of 15 destinations; these "hot" destinations receive 30% more traffic than in the earlier experiments. The rest of the traffic is evenly distributed amongst the remaining destinations. When the traffic is imbalanced, the gains from dynamic routing are more evident, as shown in Figure 7(a). Both traditional dynamic (with accurate information) and hybrid routing perform better relative to static routing when the traffic is non-uniform. But performance is degraded overall since the random graph does not have a high proportion of source-destination pairs with multiple similar length paths, limiting the flexibility of dynamic routing. For instance, only about 50% of the node-pairs have more than one minimum hopcount path between them. Figure 7(b) considers non-uniform traffic in a uniformly connected 64-node topology. In this topology, 90% of the node-pairs have more than one shortest-path. The greater connectivity results in only a minor performance degradation under non-uniform traffic, particularly for hybrid routing. Moreover, the relative performance gains of hybrid routing over static routing are greater in the non-uniform case when the topology offers greater flexibility.

4.3 Detecting Long-Lived Flows

The cost-performance trade-offs of our hybrid scheme relate directly to the selection of the flow trigger, as shown in Figure 8. Figure 8(a) illustrates that our 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



(a) Route computation rate



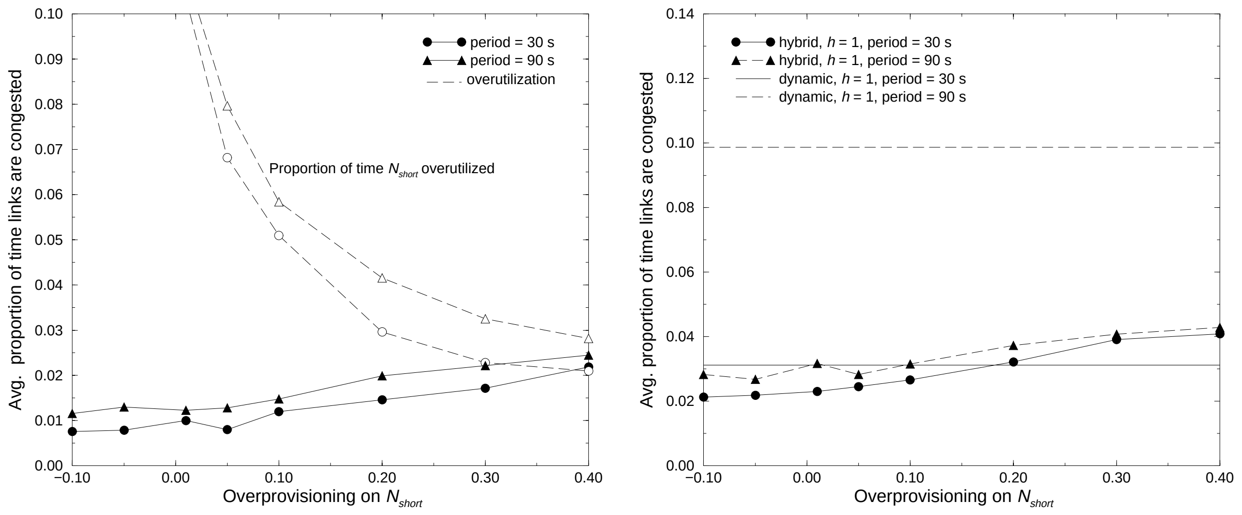
(b) Time in violation

Figure 8: **Choice of flow trigger**: In the left graph, we plot the route computation rate as a function of the flow trigger. The right 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 5.

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 8(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 8(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.



(a) Uniform traffic (b) Non-uniform traffic

Figure 9: **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 5. The hybrid scheme uses a flow trigger of 20 seconds.

4.4 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 9(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 9(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 9(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, 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 the non-uniform traffic pattern, as shown in Figure 9(b).

The non-uniform traffic increases the proportion of time that links are congested, as shown by the higher curve in Figure 9(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 just 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 [28] 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.

5 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.

As part of our ongoing work, we are collecting and analyzing additional Internet traces to study specific aspects of our hybrid routing scheme in greater detail. These 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. In addition, 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. We are also 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. 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. `http://www.acm.org/sigcomm/ccr/archive/1992/wang/final-wang-shortest-path.ps`.

[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. `http://www.bell-labs.com/user/zhwang/papers/qos-routing.ps.Z`.

[5] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick, “A framework for QoS-based routing in the Internet.” Request for Comments (RFC 2386), August 1998. `http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2386.txt`.

[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. `ftp://ftp.atmforum.com/pub/approved-specs/af-pnni-0055.000`.

[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. `http://search.ietf.org/internet-drafts/draft-zhang-qos-ospf-01.txt`.

[9] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, “QoS routing mechanisms and OSPF extensions.” Internet Draft (draft-guerin-qos-routing-ospf-04.txt), December 1998. `http://search.ietf.org/internet-drafts/`
`draft-guerin-qos-routing-ospf-04.txt`.

[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. `http://www.acm.org/sigcomm/sigcomm98/tp/abs_02.html`.

[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. `http://www.eecs.umich.edu/~ashaikh/research/papers/icnp98.ps.Z`.

[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. `http://www.nlanr.net/Flowsresearch/Flowspaper/flows.html`.

[13] M. E. Crovella and A. Bestavros, “Self-similarity in world wide web traffic: Evidence and causes,” in *Proceedings of ACM SIGMETRICS*, pp. 160–169, May 1996.

[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. `http://www.vbns.net/presentations/papers/MCItraffic.ps.gz`.

[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. `http://www.research.att.com/~anja/feldmann/papers/ton98_flow.ps`.

[16] 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.

[17] ATM Forum MPOA Sub-Working Group, *Multi-Protocol over ATM Version 1.0 (AF-MPOA-0087.000)*, July 1997.
`ftp://ftp.atmforum.com/pub/approved-specs/af-mpoa-0087.000.pdf`.

[18] P. Newman, G. Minshall, and T. Lyon, "IP switching: ATM under IP," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 117–129, April 1998.
`http://www.acm.org/pubs/articles/journals/ton/1998-6-2/`
`p117-newman/p117-newman.pdf`.

[19] S. Lin and N. McKeown, "A simulation study of IP switching," in *Proceedings of ACM SIGCOMM*, pp. 15–24, September 1997.
`http://www.acm.org/sigcomm/sigcomm97/papers/p022.html`.

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

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

[22] C. Villamizar, "OSPF optimized multipath (OSPF-OMP)." Internet Draft (draft-ietf-ospf-omp-01), work in progress, October 1998.
`http://search.ietf.org/internet-drafts/draft-ietf-ospf-omp-01.txt`.

[23] D. O. Awduche, J. Malcolm, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS." Internet Draft (draft-awduche-mpls-traffic-eng-00.txt), work in progress, October 1998.
`http://search.ietf.org/internet-drafts/`
`draft-ietf-mpls-traffic-eng-00.txt`.

[24] 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.

[25] 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.

[26] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast scalable algorithms for level four switching," in *Proceedings of ACM SIGCOMM*, September 1998.
`http://www.acm.org/sigcomm/sigcomm98/tp/abs_16.html`.

[27] T. Lakshman and D. Stiliadis, "High speed policy-based packet forwarding using efficient multi-dimensional range matching," in *Proceedings of ACM SIGCOMM*, September 1998.
`http://www.acm.org/sigcomm/sigcomm98/tp/abs_17.html`.

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

[29] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," in *Proc. Global Internet Miniconference*, November 1997.
`http://www.seas.upenn.edu/~guerin/publications/`
`ospf_globecom96.ps.gz`.

[30] 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.

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

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

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

A Simulation Model

In this section, we discuss the traffic model and network configuration used in the simulation experiments in Section 4. 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 [30]. 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 simplifying assumptions, we capture the key parameters that affect the protocol dynamics and avoid biasing the experimental results in favor of our hybrid routing scheme.

A.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 from a single access point of a large ISP network. The trace consisted of $795,446$ port-to-port flows, using a 60-second timeout. 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 distribution at 1000 seconds, which still accounts for 99.8% of the port-to-port flows. 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 ($\rho = 0.8$ 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. 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.

A.2 Provisioning and Capacity Allocation

In evaluating our hybrid routing scheme, we focus on a network provisioned according to the expected traffic load. In a production network this 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].

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 [31, 32]. 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 [33].

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 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 N_{short} and N_{long}. Choosing a particular flow trigger (in seconds) determines the proportion of flows that are routed on N_{short} or N_{long}, as well as the mean duration of flows on either partition. Average residual lifetime of flows after applying a particular trigger (i.e., mean duration of flows on N_{long}) is shown in Figure 2(b). Capacity is then allocated to each partition 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.