Limited Distribution Updates to Reduce Overhead in Adaptive Internetwork Routing[1]

Lee Breslau, Deborah Estrin, Daniel Zappala, and Lixia Zhang[2]

Abstract

The increasing transmission speed and the growing size of computer networks are creating new challenges for network routing. Increased transmission speed enables new, integrated, applications that require service guarantees from the network. In order to provide these guarantees, routes with sufficient resources must be located. The ability of a particular route to support performance-sensitive applications will vary over time as the utilization of the resources along that route changes. Therefore, routing performance can improve with the distribution of dynamic information about the state of the network. At the same time, the increasing size of networks argues against the global distribution of dynamic routing information and against requiring consistent dynamic routing information at all nodes in the network. These conflicting constraints motivate us to consider schemes that allow partial distribution of dynamic routing information.

In this paper we present a protocol for the distribution of routing updates, called Reverse Path Update (RPU). RPU distributes routing updates only along active routes, as an alternative to global flooding. Using RPU, a node receives updates about those areas of the network through which it sends its traffic. Thus, the overhead associated with update distribution is not a function of network size. Rather, distribution overhead depends on the average number of transit nodes through which the source nodes send their traffic. This important characteristic of RPU makes it a promising strategy for the dissemination of routing information in large networks.

We measure the overhead associated with RPU using simulation. When compared to global flooding, we find that RPU significantly reduces the number of dynamic updates distributed in the network. The magnitude of the savings depends on the traffic patterns, with greater savings realized when network traffic, measured by the percentage of communicating source-destination pairs, is sparse.

[1] An earlier version of this work appeared as an extended abstract in [4].

[2] Authors names are listed in alphabetical order. L. Breslau, D. Estrin, and D. Zappala are with the Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782, e-mail: breslau@jerico.usc.edu, estrin@usc.edu, dzappala@jerico.usc.edu. L. Zhang is with Xerox Palo Alto Research Center, Xerox Corporation, Palo Alto, CA 94304, email: lixia@parc.xerox.com.

The work of L. Breslau, D. Estrin and D. Zappala was supported by the National Science Foundation under contract number NCR-9011279, with matching funds from GTE Laboratories.

1 Introduction

The increasing transmission speed of computer networks is enabling new, integrated applications, that will coexist with conventional applications in future networks. Combined with the rapidly increasing size of today's internet, this development presents new challenges for internetwork routing protocols.

Due to the size and decentralized nature of the global internet, we cannot assume that the network will be homogeneously capable of all new services. Therefore, routing must identify routes with specific service capabilities. In addition, we assert that routing for performance sensitive applications, such as real-time voice and video, should adapt to significant changes in network utilization, in addition to topology changes. For example, using a minimum-hop path may not be desirable if the path cannot provide low-delay service due to congestion. Moreover, the global scale of future internets motivates us to minimize reliance on *global* distribution of routing information and mitigate the need to maintain globally consistent routing databases. Therefore, we investigate the use of limited distribution of routing updates.

We base our strategy for limited distribution of routing updates on expectations about internetwork traffic patterns. Traffic patterns have been, and are expected to remain, sparse. By this we mean that each source communicates with only a small subset of destinations, leading us to conjecture that to reach these destinations, a source sends its traffic through only a subset of transit nodes. Therefore, we are interested in routing distribution strategies that provide a source node with information about only those transit nodes through which it sends its traffic, while avoiding the unnecessary overhead of providing nodes with updates they will not use.

In this paper, we present a new distribution protocol called Reverse Path Update (RPU), which exploits communication locality when distributing routing updates. RPU distributes updates along currently active routes as an alternative to global flooding. RPU attempts to provide nodes with information only about those areas of the network relevant to their routing decisions. Also, in contrast to global flooding, which scales with the size of the network, RPU overhead is a function of the average number of nodes the sources use to reach those destinations with which they communicate. This latter feature is important for scaling in a global internet. Simulation results show that RPU can significantly reduce the overhead of update distribution in comparison to global flooding. The magnitude of the overhead reduction is dependent on the traffic patterns in the network.

The remainder of this paper is organized as follows. After presenting our model of the global internet in the next section, we discuss the assumptions about traffic locality that motivate our work (Section 3). In Section 4 we provide a brief overview of a routing architecture within which our distribution strategies can be made to work. The Reverse Path Update protocol is described in Section 5 and simulation experiments are discussed in Section 6. We conclude with brief discussions of related work and future research.

2 Internet Model

The global internet is made up of many domains, where a domain is a collection of networks, routers, links and end systems that is governed by a single authority. Domains can be classified as *stub* domains, which contain the end systems that serve as the sources and destinations for

traffic, and *transit* domains, which forward traffic between the stub domains.[3] We assume that each domain runs an intra-domain routing protocol in order to maintain the externally-advertised connectivity across the domain, and that each domain has one (or more) route server that collects routing information from the outside and computes and distributes inter-domain routes to internal end systems.

For much of our discussion and analysis in this paper, we represent an entire domain as a single node. This abstraction assumes intra-domain routing and forwarding mechanisms that can meet the connectivity and type of service requirements advertised by the domain as a whole. The particular mechanisms used inside a domain to meet a specific type of service are up to that domain. Regardless of the internal mechanisms used, we assume that the domain has a means of representing its routing status (connectivity and load) for the purposes of inter-domain routing advertisements. The overhead associated with the distribution of these domain level advertisements is the major focus of this paper.

3 Scaling in Large Networks

The size of a global internet requires that routing protocols address the issue of scale. In particular, we are concerned with the overhead of routing information distribution. In this section we first review the use of hierarchical aggregation as a solution to the problem of scale in routing. Hierarchy, however, by itself does not present a satisfactory solution. Therefore, we discuss our assumptions about *locality* in traffic patterns, and how we exploit this locality to control the overhead of routing information distribution.

3.1 Hierarchical Routing

In hierarchical routing [17], network nodes are grouped into larger aggregates, or clusters. Detailed information about individual nodes in a cluster is distributed only within the cluster. External advertisements hide details about the individual nodes, and nodes in one cluster cannot distinguish among the elements in other clusters. We propose making use of this form of hierarchy, by aggregating information about the routers, hosts and links for each domain, and distributing domain level information externally.[4] In so doing, the size and number of routing updates distributed between domains is reduced. However, given the size of the internet, global distribution of domain level updates can still introduce excessive overhead if updates are triggered by changes in utilization. Therefore, we are motivated to develop additional mechanisms to cope with scaling in a large internet.

Hierarchy can be used to facilitate scaling further by controlling the type of information advertised in routing updates. For instance, load based information could be distributed within a domain, while inter-domain advertisements contained only dynamic topology (i.e., up/down) information. This would reduce routing overhead by limiting the frequency of inter-domain updates; topology changes occur less frequently than load changes. However, routing for performance-sensitive applications should be based in part on persistent load information. Solutions that only distribute load

[3] Hybrid domains are also possible; for sake of simplicity we do not address them in this paper.

[4] This is also consistent with our belief that administrative domains require control over how their internal resources are used and would prefer to control them by controlling intra-domain routing.

information within a single domain are not helpful in supporting these applications in a wide area, inter-domain environment.

Imposing additional levels of hierarchy on the internet is another possible solution. Domains could be grouped into larger aggregates (e.g., confederations), with domain specific information distributed within the confederation, and aggregated information for the confederation distributed externally. While this would support load-based intra-confederation communication, the larger a confederation, the more likely that flooding within the confederation would introduce excessive overhead. Moreover, this approach is not well-suited to adaptive inter-confederation routing. Additional hierarchy prevents distinctions between different domains in a single confederation, when such distinctions may be relevant to routing decisions. For instance, a confederation routing update will advertise the same status for all of its constituent domains, even if the status of the individual routes to those domains differs. This would prevent external domains from specifying different routes to reach two domains in the same confederation, even when different routes would be needed to meet the requirements of performance sensitive applications.

Imposing additional levels of hierarchy reduces information in a uniform way, and removes the ability to differentiate among elements of a larger grouping. Instead, we recommend a distribution strategy that provides detailed information about certain domains, and less information about others, based on the locality characteristics exhibited by network traffic .

3.2 Exploiting Locality to Achieve Scaling

Studies have shown the matrix for internetwork traffic to be sparse [9, 11, 15, 22, 24]. At any time, each stub domain communicates with only a small percentage of other stub domains. We refer to this as *destination locality*, and expect it to exist in future networks. This leads us to believe that a second kind of locality, which we call *route locality*, also exists and can be exploited to reduce overhead. Route locality refers to the concentration of the routes used by a single domain in a subset of all the transit domains. Since a single source communicates with only a small subset of destinations, the routes it uses to reach those destinations may traverse only a subset of the transit domains. Moreover, factors including policy, geography and topology are likely to further concentrate the routes used by a particular source (to reach all desired destinations), further increasing route locality.

Figure 1 shows an example internet containing 18 domains, and 25 inter-domain links. Stub domains can be identified by the presence of attached hosts, while the domains with no hosts carry transit traffic. As a simple example of route locality, the host attached to domain N1 has only one active route to another domain (N14), denoted by the heavy lines. While there are 10 transit domains in the network, the routes used by domain N1 traverse only 4 of them.

The concentration of the routes used by a single stub domain in a subset of the transit domains implies that a stub domain does not require uniform information about all transit domains. Rather, its routing decisions can be enhanced if it receives accurate and timely information about those transit domains through which it routes its traffic. Furthermore, a stub domain may want dynamic load information about those domains that carry real-time traffic, while up/down status of domains carrying non-real time traffic is sufficient. Receiving information about other transit domains that it does not use is of less importance. Instead of distributing information globally, our information distribution mechanisms will exploit the locality exhibited in network traffic and distribute routing

updates only along active data flows. The Reverse Path Update protocol, which we describe in Section 5, forwards routing updates along currently active source routes, thereby providing routing information where it is needed, while avoiding distribution of the information to places where it may never be used.

Another kind of locality, *topological* locality implies that each domain is more likely to make use of domains that are nearby (in terms of network hops) than those that are far away. One might also try to exploit topological locality by using a hop-count limited flooding scheme. That is, the originating router attaches a hop-count to each routing update and the count is decremented by one at each hop until it reaches zero and is discarded. Both the effectiveness and efficiency of a limited-flooding scheme depend upon the value of the hop-count used to control flooding distribution. If the hop count is too small then the updates will not reach many nodes that need the information to make good routing decisions, and if the hop count is too big then the updates will be distributed to many that do not require the information, leading to excessive overhead. The protocol discussed in the remainder of this paper, RPU, attempts to overcome this tradeoff by distributing dynamic information to those nodes most likely to utilize it.

4 Adaptive Source Routing

We have argued that the limited distribution of routing updates is required to address the scaling problems of routing in large internets. This distribution strategy will result in inconsistent information at different nodes in the network. Such a situation can in general be problematic for routing. In this section we present an overall routing architecture using source routing, within which limited distribution updates can work.

Routing architectures can be characterized by the location of the routing decision. In *hop-by-hop routing*, decisions are made in a distributed manner. Each node maintains a table specifying the next hop used to reach each destination. When a forwarding decisions is made, a node consults its routing table and sends a packet to the next hop toward the destination. The collective decisions made by all the nodes between a source and destination determine the route a packet follows. Using *source routing*, the entire route to a destination is determined at the source. The source node includes the entire route in each packet, or uses a setup procedure to install the forwarding information in the forwarding tables of the transit nodes.

Hop-by-hop routing requires that all nodes along a path make consistent decisions to avoid routing loops. In order to ensure that these decisions are consistent, each node must maintain the same routing information and use the same criteria for selecting the next hop to a destination. Therefore, conventional hop-by-hop routing can not be used in conjunction with the partial distribution strategies we propose. Rather, hop-by-hop routing depends on the global distribution of routing information to maintain consistency across nodes. For this reason, we have developed an architecture using source routing that will work with the limited distribution of updates. Source routing prevents routing loops and allows source nodes to select routes that meet their service criteria.

Our architecture makes use of link-state style updates to provide information about configured and dynamic network state [3, 8]. We assume that the complete inter-domain *configured* topology is distributed globally using event-driven distribution combined with very low frequency periodic updates. This information is relatively stable so we assume that the overhead of distribution and

storage is acceptable. Configuration information alone, however, is not sufficient to support routing in a network of heterogeneous services. Up/down status and long-term utilization measures are needed to meet various service requirements. When performance-sensitive applications require resource reservation or tighter bounds on delay or throughput than a default route provides, up-to-date resource information is needed to select an alternate route.

As stated previously, rather than globally distributing these dynamic updates, we opt for a more limited distribution strategy. The strategy we propose is called Reverse Path Update (RPU). When a node generates an RPU routing update (either because of a link failing or coming into service, or because of a significant change in utilization), the update is forwarded along the active source routes traversing the node. The protocol mechanisms of RPU will be discussed in more detail in Section 5. A hop-count limited flood can distribute updates to domains within a few hops of the triggering domain, and thereby take advantage of topological locality.

A route server in each domain stores both the configured topology of the internet and dynamic updates received from other domains, and uses this information when selecting routes. Specifically, the configured topology can be used to compute a set of candidate paths to each destination. When the route server must provide a route for a new session, it selects from among the candidate paths based on available dynamic information. For example, the route server can avoid routes that it knows to be down or that have congested links, in favor of routes offering better performance.

The information maintained by the route server about other domains will vary, both in terms of level of detail and age. For some domains, the route server will have only configured information, while for others it will also receive dynamic updates describing current status or utilization measures. The age of dynamic updates can also vary. Some updates may have been received over currently active source routes, and can therefore be assumed up-to-date. In other cases, the source route that caused an RPU update to be received by a node may no longer be active. Timely, yet perishable, information of this sort must be timed out eventually.

A route server will also learn about failures when it receives a failure notification after attempting to use a route for the first time. Such a notification temporarily invalidates the route. The route can be revalidated by a subsequent update indicating the route is operational. Even if no such explicit information is received, the dynamic down status will eventually be timed out (since limited distribution of updates does not guarantee the route server will receive an update when the failed link/route becomes operational again).

When selecting a route, the route server must consider both the detail and freshness of its routing information. Fresh information should be assumed to be more valid than older information, and when choosing routes for performance sensitive applications, the route server should first consider those routes for which it has up-to-date dynamic utilization information. The local and global impacts of specific route selection algorithms remain a critical subject for future research (see Section 8).

While our distribution mechanisms strive to provide the route server with the dynamic routing information relevant to its selection decisions, at times it will select routes using incomplete information. For instance, the route server may have to select from among multiple routes that traverse areas of the network work about which it has no current dynamic information. In these cases, the route server's decision will be based on the configured topology information. So while the decisions in these cases might not be ideal (i.e., multiple attempts may be needed to find a viable route), the

architecture still provides a way to select new routes in the absence of dynamic information. Having outlined an architecture that makes use of limited distribution of dynamic routing information, we now describe in more detail our mechanisms for update distribution.

5 Reverse Path Update Protocol

The RPU protocol defines three interrelated components: the triggering event, the contents, and the distribution of the update. RPU updates are triggered by either a significant utilization change or a failure or recovery of a link. Link failure or recovery causes two updates to be generated and propagated, one for each of the nodes adjacent to the link, while a change in load may cause only a single node to initiate an update.

In addition to containing the status of the node that generates the packet, the update contents also include the status of the node's other links and its neighbor nodes in the domain. This extra information is included because of our assumption that the traffic from each source exhibits route locality; a source is likely to use other routes through or near the nodes it already visits. The update is equivalent to a link state advertisements for the entire domain, similar to those distributed between areas in OSPF[20] or between domains in IDPR[26].

Once the node collects the appropriate status information, it distributes it along the existing active routes, using the installed route setup state or a cache of the most recently used source routes. For each link over which the node distributes the update, it copies the status information into a packet and includes a distribution list comprised of the active routes that traverse the link. Nodes that subsequently receive an RPU message continue forwarding the update along these routes, without modifying the packet.[5] When the source node for a particular route receives an RPU, it forwards the message to its route server.

Note that during this distribution, it is likely that an RPU message will be designated for several routes that use the same outgoing link. For example, in Figure 1 if link L18 fails, node N11 will send an RPU message to H4 and to H1. The routes to these two hosts both use outgoing link L15. As described above, the protocol produces only one packet per link, to avoid sending duplicate messages in this instance. Eventually, the packet is replicated if it encounters a node where the routes diverge.[6] In the same example, node N9 will replicate the message and send it to host H1 over link L10 and to host H4 over link L14.

Those updates triggered by load events are sent only to sources of active routes that indicated the need for load related updates at the time of route installation. This avoids the unnecessary overhead of propagating the more frequent updates to sources that are not as sensitive to performance. However, whenever an update triggered by a link failure is sent it may include load information as well, if the cost of including the additional information is not too large.

In some sense, the information provided by RPU is similar to the path congestion information that one can infer from returning acknowledgements (the information is either communicated explicitly or inferred implicitly in the round-trip time or from dropped packets) [21, 23]. However, RPU will provide the source with explicit status information about a broken route, while end-to-end

[5] This has the added feature that the entire RPU packet can be signed for integrity and authentication purposes.
[6] If the routes converge again later, redundant RPU packets are sent on the common links.

acknowledgements will themselves be blocked by a broken route. RPU also provides measures of average utilization, whereas an unmodified ack stream would only provide the source with round-trip-time information. Moreover, RPU may provide information about the domain's other routers, not just the particular router being traversed. Finally, RPU will inform the source of improved, as well as deteriorated, conditions if the source continues to use neighboring regions of the network.

6 Simulation Studies

The limited distribution mechanisms cannot be fully evaluated by simple mathematical analysis[10]. Detailed simulations are needed in order to examine the effects of topology and traffic patterns on the overhead of the mechanisms. We have simulated the RPU protocol and a global flooding mechanism across various topologies with different traffic matrices, and compared the overhead of RPU versus global flooding. We describe the implementation and results of our simulations below.

6.1 Implementation

The simulator we used is a substantially extended version of a packet-based, event-driven simulator originally implemented by Lixia Zhang. The simulator accepts as input a configuration file that specifies nodes, links and hosts. The nodes are packet switches used to represent domains in an internet, and the hosts serve as endpoints for conversations. A set of conversations, and the source routes they use, is statically defined for each simulation.

We have run simulations using the 18-node network shown in Figure 1, and on several 50 and 100 node networks that were created with a network topology generator. The network topology generator automatically creates a network topology that meets certain specified characteristics. These characteristics include the size of the network, the number of nodes at different levels in the network (allowing us to generate networks with varying degrees of hierarchy), and the average number of links between nodes in the same and different levels of the network. Given these topologies we randomly selected host-pairs for conversations and used a Shortest Path First algorithm to compute the routes used by each conversation.

At the start of the simulation, each conversation uses a route setup protocol to install the necessary routing information–route identifier, previous hop, and next hop–in the appropriate nodes. Once all of the routes are installed, a link status event is scheduled. While a link status event could represent either a link failure or a change in load beyond some threshold, in our experiments we simulated link failures. This could have a slight effect on the overhead of the distribution protocols, as updates can not be distributed over links that have failed. The link status event triggers the update distribution mechanism, either the RPU protocol or global flooding, in the nodes adjacent to the link whose status has changed. The update messages are distributed throughout the network, while the nodes track the overhead used. The simulation stops once the update distribution is finished, and the number of update messages processed by each node is reported.

For each network topology, we simulated several different traffic mixtures by varying the number of actively communicating source/destination pairs. Each combination of topology and traffic mixture was characterized by an average *route locality*. The route locality for a host is defined as

$$R = 1 - \frac{n_v}{n_t}$$

where R = route locality, n_v = the number of nodes the routes from a host traverse, without counting duplicates, and n_t = the total number of transit nodes. When route locality is near zero, on average, the routes used by all source nodes are dispersed throughout the network. When the routes used by each source node are concentrated in a small number of transit nodes, route locality approaches 1. The average route locality for a network is the average of the route locality for all hosts. Note that because the communicating host-pairs were generated randomly, the route locality numbers for our simulations are conservative. For each combination of topology and traffic pattern, the sequence of updates generated by all possible link events was investigated, requiring nearly 6000 simulation runs. We report the results of these simulations below. For the purposes of this overhead analysis we treated link status and load events in the same way. We obviously did not attempt to simulate realistic patterns of node/link failures and congestion. We generated a systematic set of link events in order to understand the average behavior for an event anywhere in the network across different network configurations.

6.2 Results

For each topology simulated, the ratio of RPU overhead to global flooding overhead is plotted as a function of route locality. Because the two nodes joined by a link create updates with different status information upon a link status change, we treat the updates as two separate packets that need to be distributed. Thus, the global flood overhead is approximately twice the number of links in the network (we do not include host-node links). If a tree-based global flood is used [6], the overhead of global flooding could be reduced to twice the number of nodes in the network.

We first simulated the 18-node, 25-link network of Figure 1 with 5 different traffic mixtures. The bold lines in the figure indicate the routes used by active conversations in one traffic mixture. Figure 2 shows the results of this simulation as a plot of overhead ratio versus route locality. The results indicate that RPU overhead is significantly less than that of global flooding. For instance, at a route locality of 0.81, meaning that on average each host sends traffic through 19% of the transit nodes, the overhead of RPU is less than one-fifth the overhead of flooding. As would be expected, as the route locality decreases, the relative advantage of RPU decreases; on average, there are more active routes through each transit node, so an RPU update initiated at a transit node will be propagated to more hosts.

To investigate the behavior of RPU in larger networks, we created several 50 and 100-node networks, with both flat and hierarchical topologies. The latter are intended to model the expected structures of future internets. We defined three levels of hierarchy, with the highest level nodes representing backbones, the middle level representing regionals, and the lowest level representing campus or stub networks. The hierarchical topologies can be further distinguished by the presence or absence of lateral links between regional domains. We report the results of simulations on the following 50-node networks below:

- FLAT50a: flat topology with average connectivity of 3 (74 total links)
- FLAT50b: flat topology with average connectivity of 6 (142 total links)

- HIER50: hierarchical topology with no lateral links (56 total links)
- LAT50: hierarchical topology with lateral links between regional domains (66 total links)

We also describe the results for the following 100-node networks:

- HIER100: hierarchical topology with no lateral links (114 total links)
- LAT100: hierarchical topology with lateral links between regional domains (136 total links)

Figure 3 shows a plot of overhead ratio versus route locality for the four 50-node configurations. Figure 4 shows the same plot for the two 100-node configurations. As with the 18 node simulations, RPU reduces the overhead of update distribution, with more substantial savings at higher route localities. The differences between the results for the four 50-node networks can be attributed in part to differences in the connectivity of the networks. More highly connected networks will have shorter paths on average, so an RPU update will traverse fewer network hops to reach the source node. In FLAT50b, the highly connected flat topology, the savings due to RPU are greater than in FLAT50a, the less connected flat topology. Similarly, the presence of lateral links between regional nodes in LAT50 reduces the average path length, and consequently the overhead of RPU, in comparison to HIER50, which has no lateral links. The plots for HIER100 and LAT100 have negligible differences from their 50-node counterparts, showing that the primary factor in RPU overhead is route locality. It remains to be seen whether an even larger simulation, perhaps of 1000 nodes, will significantly impact the overhead ratio.

Further insight can be gained by examining the overhead associated with distributing updates originated at different nodes in the network. Figure 5 shows the overhead for HIER100 broken down between the backbone and regional nodes. The overhead for campus nodes is not shown because any status changes of a regional-campus link isolate these nodes from the network and prevent propagation of a campus node update. This graph shows that the overhead ratio for backbone nodes is higher than the ratio for regional nodes. This is because, on average, there are more active routes through a backbone node than a regional node. Therefore, when a backbone node initiates an RPU update, this update will be propagated along more paths and will reach more source nodes than when a regional node initiates an update. A similar plot for the LAT100 topology is shown in Figure 6. The presence of lateral links between regional nodes enables the use of more routes bypassing the backbone nodes. On average, the routing tables of the regional nodes are larger, and those of the backbone nodes are smaller, when compared to the strictly hierarchical topology. This results in the overhead ratio of the backbone and regional nodes being nearly the same. In addition to contributing to our understanding of this scheme, this insight might motivate the use of alternate strategies for different nodes in the network. For instance, backbone nodes could use global flooding, if the overhead of RPU for these nodes is high, while regional nodes use RPU.

In addition to considering which node initiated an update, we also considered which link caused the update to be generated. For instance, when a backbone node initiated an update, we examine whether the update was initiated by a failure in a link connecting two backbone nodes, or by a failure in a link connecting a backbone and a regional. Similarly, when a regional node initiated an update, we examine whether the link that experienced a change in status connected the regional to another regional, a backbone, or a campus node. The results of this analysis are shown in Figure 7

for the HIER100 topology. These results show that when a backbone node originates an update in response to a change in a Backbone-Regional link, the overhead ratio is higher than when the update is caused by a change in a Backbone-Backbone link. Also, the overhead ratio of an update propagated by a regional node as a result of a change in a Regional-Campus link is higher than the overhead ratio of updates propagated as the result of a change in a Regional-Backbone link.

Our simulation results indicate that RPU can be effective in reducing the overhead of distributing routing information. In all simulated topologies, RPU overhead was much lower than global flooding; the extent of the benefit is dependent upon several factors. First among these is the route locality in the network. We simulated networks at a range of route localities and, as expected, the effectiveness of RPU was a function of that parameter. We expect the matrix of internet traffic to be sparse, and that such a sparse traffic matrix will lead to high route localities. In creating our simulation configurations, we achieved significantly high route localities even when large numbers of host pairs were active. For instance, specifying each host to communicate with approximately 10% of other destinations resulted in route localities between .6 and .8 for all configurations. When considering the world-wide internet, which is the context for this work, 10% is a large number of possible destinations. While there is evidence to support our contention that route localities in actual networks will be high, a study of existing network traffic to determine actual route locality would be useful.

Furthermore, in our simulations we assumed that dynamic updates were distributed along all active source routes. However, it is possible to differentiate those routes carrying real-time traffic from those carrying best effort traffic. Since sources of best effort traffic are not as concerned with adapting to changes in network load, it is sufficient to propagate dynamic updates reporting changes in network load only to the source of real-time traffic, while propagating updates about status changes along all routes. Since the matrix of real-time traffic should be sparser than that for traffic as a whole, limiting the more frequent load-based dynamic updates in this manner would further increase the benefits of RPU.

Finally, our results indicate that the benefit of RPU is also a function of the combination of the position of a link in the internet and the types of nodes that subsequently propagate the update. For a strictly hierarchical network, links that connect two levels, i.e. B-R and R-C links, partition the network unevenly. The update on one side propagates just a short distance, while the other has the potential to reache nearly the whole network, depending on the route locality. The presence of lateral links provides shorter routes and distributes the routes among more nodes, causing RPU overhead to also be distributed more evenly.

7 Related Work

Attempts have been made to adapt to changes in network load in hop-by-hop routing protocols. For example, link costs in the Arpanet SPF algorithm were a measure of average delay [18]. It has been shown analytically [2] and in practice in the Arpanet [16], that load based algorithms can suffer from oscillations caused by all nodes reacting to a routing update in the same way. This is particularly true in hop-by-hop routing in large networks with long paths, where consistent decisions are required by all nodes in order to avoid routing loops.

The problem of adaptive routing has also been considered in the context of circuit switched net-

works. These algorithms, designed for fully connected networks, use a direct one hop path when available, and attempt alternate two hop paths when the direct path is unavailable. In all cases, trunk reservation is used to reduce the chances of alternate routed calls from causing subsequent direct calls to be blocked. Examples of these algorithms include Dynamically Controlled Routing [5], Dynamic Alternative Routing [13] and Fixed Alternate Routing [19]. While these results demonstrate the potential benefit of adaptive routing, as well as the importance of trunk reservation, many of the problems addressed in our work do not arise in fully connected circuit switched backbone networks.

Others have proposed the use of source routing in computer networks. Sunshine [27] and Saltzer, et. al [25] cite the simplicity of gateway implementation and the prevention of routing loops as benefits of source routing. In addition, they note that source routing may be less adaptive than hop-by-hop routing.

Hagouel identified the growing size of networks as a motivation for using source routing[14]. Specifically, as networks grow, the frequency of events triggering network updates increases. More frequent updates will put the network in a transient state more often, leading to a greater probability of routing loops and wasted resources. Source routing increases the header length, so that Hagouel suggests it may be more appropriate to virtual circuit environments, in which connection setup obviates the need for including the entire source route in every packet. The adaptive behavior of source and hop-by-hop routing are compared. It is noted that hop-by-hop routing may adapt faster since source routing must return to the source for a new route. On the other hand, hop-by-hop routing may use suboptimal routes during the period of convergence.

In their simulation study, Gardner et. al. [12] proposed an architecture for separating congestion avoidance from route computation. Multiple routes per destination were computed infrequently, and flow allocation assigned flows to routes in response to congestion signals on a faster time scale. They reported that this reduced update overhead (in terms of bandwidth and computation). Topology changes were reported immediately whereas persistent load changes were reported periodically. They proposed source routing to prevent routing loops and eliminate the requirement for agreement among intermediate nodes while using multiple paths in the Arpanet. The path generation algorithms computed alternate paths according to the type of traffic. They did not consider ways to reduce the overhead of information distribution, other than reducing the frequency of global distribution of updates. They did not address the critical issues of scale discussed here.

The PARIS network architecture developed at IBM uses source routing in a fast packet switching network [7]. Switches maintain a database of current link utilizations. Routes are chosen based on the link utilizations, and available bandwidth is verified during setup to guarantee that the requirements of real-time traffic can be accommodated. The scheme is dependent on fast global distribution of link utilization measures and the switch hardware has been designed to facilitate fast updates along a spanning tree. Nevertheless, the reliance on frequent global flooding prevents the scheme from being used, as is, for global internet routing.

More recently, Bahk and El Zarki published a performance analysis study of adaptive source routing [1]. In order to avoid congestion they move traffic from highly loaded shortest paths to less loaded longer ones. A form of trunk reservation is used to prevent the possible performance degradation when traffic is routed onto alternate paths. Advertised link metrics, which represent a discrete classification of utilization, are distributed globally under the assumption that the network is fairly small. Also, route computation is based on precomputation of routes for all possible combinations

of network metrics in the network, it is also not likely to scale well.

Our work is complementary to these schemes for adaptive source routing. While they present important insight into load measures and route selection techniques (an area not covered by our work to date), they do not address issues of scale. In particular, these schemes assume global distribution of routing information, and do not consider the limited update distribution mechanisms central to our work.

8 Future Research

We have described an architecture utilizing source routing and limited distribution of routing updates to satisfy the requirements of global internet routing. Limited distribution of updates reduces the overhead associated with routing information distribution, while source routing enables loop-free routing in the presence of inconsistent routing databases. The Reverse Path Update (RPU) protocol, a limited distribution update mechanism, forwards routing updates along active source routes, exploiting communication locality and thereby providing updates to those nodes most likely to need them. Simulation results showed that RPU can reduce the number of routing updates propagated in a network, achieving significant savings over global flooding under certain traffic characteristics. We also found that the savings experienced may vary across network links; those links carrying more traffic (e.g., backbone links) realize a smaller savings than those in the periphery of the network that may carry less traffic. This suggests the benefit of a hybrid approach in which routing updates of some nodes are flooded (globally or with a large hop-count), while those of other nodes are forwarded using RPU. While our results indicate that RPU can be an important component of a routing architecture intended to support real-time (performance sensitive) applications in future internets, many unanswered research questions remain, both with the protocol in particular and with our adaptive source routing approach in general.

RPU's ability to provide a reduction in routing overhead is a function of the locality exhibited in internetwork traffic. Studies have shown the matrix of internetwork traffic to exhibit source/destination locality [9, 11, 15, 22, 24]. However, the extent of route locality has yet to be measured systematically. Combining traces of inter-domain traffic, along with information about the routes used by such traffic would provide an actual measurement of route locality and an important basis for further studies of limited distribution mechanisms.

We have investigated the reduction in overhead achieved by the protocol, but we have not thoroughly considered the cost, in terms of added delay, of routing with partial information. As we discussed in Section 4, limited distribution mechanisms will cause some routing decisions to be based on out-of-date information. Therefore, nodes will sometimes attempt to set up routes that will fail, delaying the initiation of a session. While we have designed our distribution mechanisms to exploit traffic locality and reduce the number of failed setups, the impact of RPU on session startup delay, and the tradeoff between overhead reduction and setup delay remains to be studied in more depth.

Beyond the Reverse Path Update protocol, there remain many open research issues regarding our architecture. We have argued that routing decisions for real-time applications should be based in part on load measures. Meaningful measures for average load, as well as algorithms for triggering load related updates must be developed. Also, route selection algorithms based on load related

information and taking into account the freshness or certainty of routing information are required. Algorithms should be evaluated in terms of overhead and startup delay, as well as on total overall throughput and the end-to-end delay experienced by data packets.

A final area that must be addressed by route selection algorithms concerns stability. Load based routing protocols can be subject to two types of instability. First, all nodes reacting to routing advertisements in concert can cause oscillations [2, 16]. Second, instability can be caused by the use of alternate paths that consume more network resources than shorter paths. Under some circumstances, the use of such alternate paths can increase the likelihood of other paths becoming congested, leading to degraded routing performance. Route selection algorithms must address both of these stability related issues. Finally, the interaction of adaptive routing with congestion control must be studied carefully. Investigating the last several open issues requires more extensive simulations in which load-events are not artificially generated, but are generated as a result of the load placed on the system.

9 Acknowledgements

The Network Topology Generator that we used to create the 50 and 100-node networks for our simulations was developed by Steve Hotz and Ron Nagamati.

References

[1] Saewoong Bahk and Magda Elzarki. Dynamic multipath routing and how it compares with other dynamic routing algorithms for ATM networks. Technical report, University Of Pennsylvania, Electrical Engineering Department, June 1991. Submitted To IEEE Infocom '92.

[2] Dimitri P. Bertsekas. Dynamic behavior of shortest path routing algorithms for communication networks. *IEEE Transactions on Automatic Control*, AC-27(1):60–74, February 1982.

[3] Lee Breslau and Deborah Estrin. Design and evaluation of inter-domain policy routing protocols. *Internetworking: Research and Experience*, 2(3), 1991.

[4] Lee Breslau, Deborah Estrin, Daniel Zappala, and Lixia Zhang. Exploiting locality to provide adaptive routing of real-time flows in global internets. In *Multimedia '92: 4th IEEE ComSoc International Workshop on Multimedia Communications*, pages 143–149, April 1992.

[5] W. H. Cameron, J. Regnier, P. Galloy, and A. M. Savoie. Dynamic routing for intercity telephone networks. In *Proceedings of the 10th International Teletraffic Congress*, June 1983.

[6] I. Cidon, I. Gopal, M. Kaplan, and S. Kutten. Distributed control for paris. In *9th Annual Acm Symposium On Principles Of Distributed Computing*, 1990.

[7] Israel Cidon and Inder S. Gopal. PARIS: An approach to integrated high-speed private networks. *International Journal of Digital and Analog Cabled Systems*, 1(2):77–85, 1988.

[8] David D. Clark. Policy routing in internetworks. *Internetworking: Research and Experience*, 1(1):35–52, 1990.

[9] P. B. Danzig, S. Jamin, R. Caceres, D. J. Mitzel, and D. Estrin. An empirical workload model for driving wide-area TCP/IP network simulations. To appear in *Internetworking: Research and Experience*, 1992.

[10] Deborah Estrin, Lee Breslau, and Lixia Zhang. Protocol mechanisms for adaptive routing in global multimedia internets. Technical Report USC-CS-91-499, Computer Science Department, University of Southern California, November 1991.

[11] Deborah Estrin and Danny J. Mitzel. An assessment of state and lookup overhead in routers. To appear in IEEE Infocomm, 1992.

[12] Marianne L. Gardner, Ina S. Loobeek, and Stephen N. Cohn. Type-of-service routing with loadsharing. In *Globecom*, 1987.

[13] R. J. Gibbens, F. P. Kelley, and P. B. Key. Dynamice alternative routing – modelling and behaviour. In *Proceedings of the 12th International Teletraffic Congress*, June 1988.

[14] Jacob Hagouel. Source routing and a distributed algorithm to implement it. In *IEEE Infocomm.* IEEE, 1983.

[15] Raj Jain. Characteristics of destination address locality in computer networks: A comparison of schemes. *Computer Networks and ISDN Systems*, 18:243–254, 1990.

[16] Atul Khanna and John Zinky. The revised ARPANET routing metric. In *Proceedings of ACM Sigcomm*, pages 45–56, September 1989.

[17] Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large networks: Performance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.

[18] John M. McQuillan, Ira Richer, and Eric Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[19] Debasis Mitra and Judith Seery. Comparative evaluations of randomized and dynamic routing strategies for circuit-switched networks. *IEEE Transactions on Communications*, 39(1):102–116, January 1991.

[20] J. Moy. The OSPF specification. RFC 1131, SRI Network Information Center, October 1989.

[21] Don J. Nelson, Khalid Sayood, and Hao Chang. An extended least-hop distributed routing algorithm. *IEEE Transactions on Communications*, 38(4):520–528, April 1990.

[22] Vern Paxson. Measurements and models of wide area TCP conversations. Technical report, Lawrence Berkeley Laboratory, University of California, May 1991.

[23] K. K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 8(2):158–181, May 1990.

[24] Yakov Rekhter and Bilal Chinoy. Injecting inter-autonomous system routing into intra-domain system routing. *Computer Communications Review*, January 1992.

[25] Jerome H. Saltzer, David P. Reed, and David D. Clark. Source routing for campus-wide internet transport. In *Local Networks for Computer Communications*, pages 1–23. IFIP, 1981.

[26] Martha Steenstrup. Inter-Domain Policy Routing protocol specification and usage: Version 1. Technical Report BBN Report No. 7346, Bolt, Beranek and Newman, Inc., July 1990.

[27] Carl A. Sunshine. Source routing in computer networks. *Computer Communications Review*, 7(1):29–33, January 1977.

Figure 1: 18-node topology

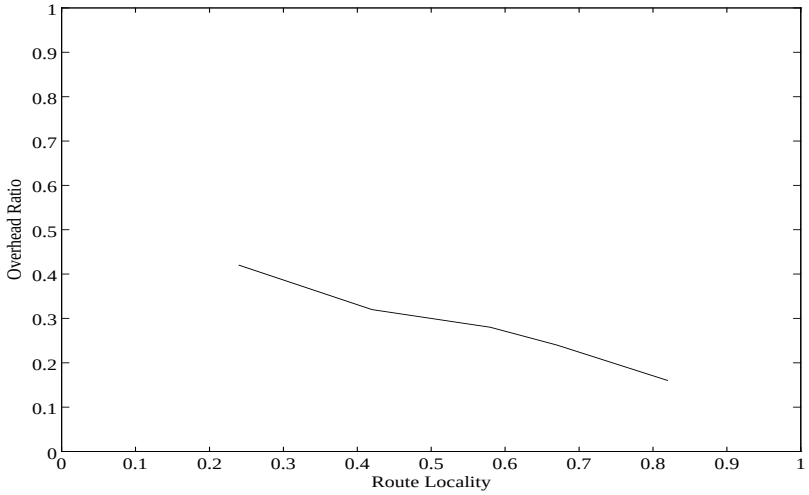

Figure 2: Ratio of RPU overhead to global flood for 18-node network

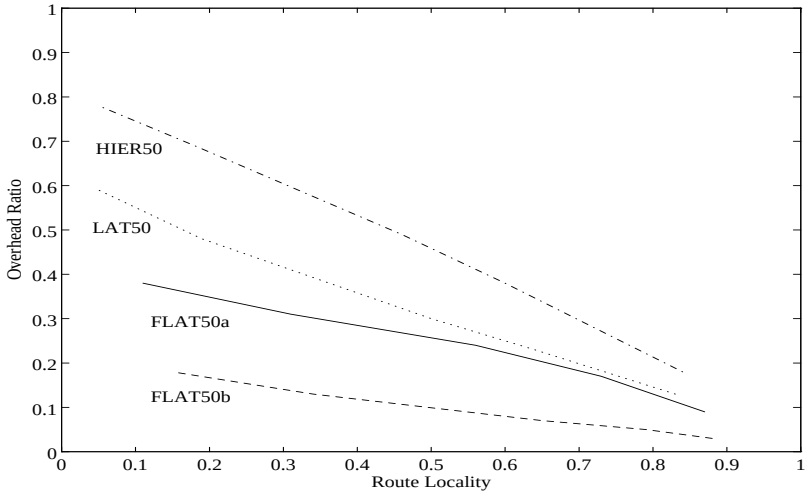

Figure 3: Ratio of RPU overhead to global flood for 50-node networks

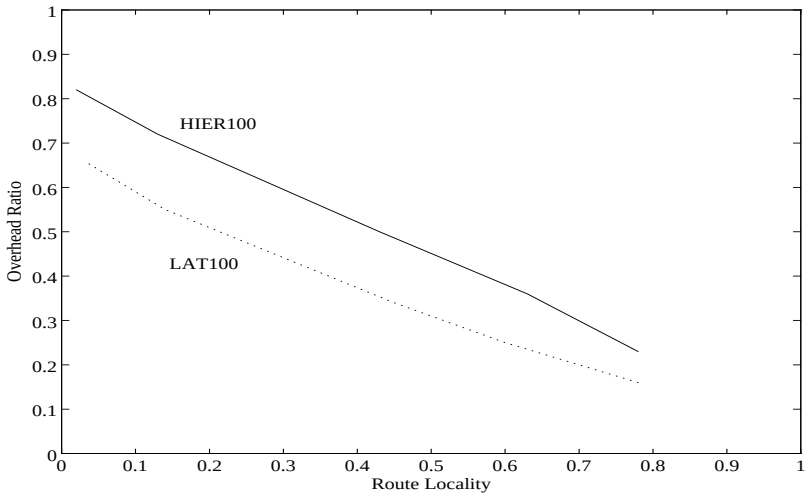

Figure 4: Ratio of RPU overhead to global flood for 100-node networks

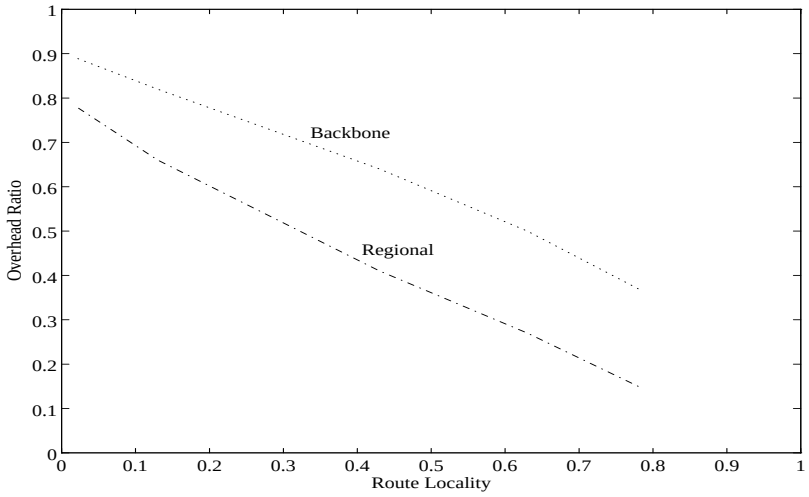

Figure 5: Ratio of RPU overhead to global flood for 100-node hierarchical network (HIER100) by node type

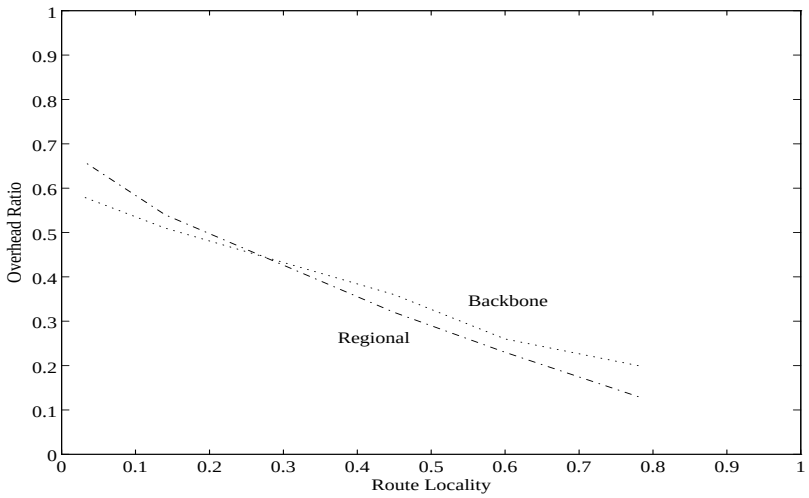

Figure 6: Ratio of RPU overhead to global flood for 100-node hierarchical network with lateral links (LAT100) by node type

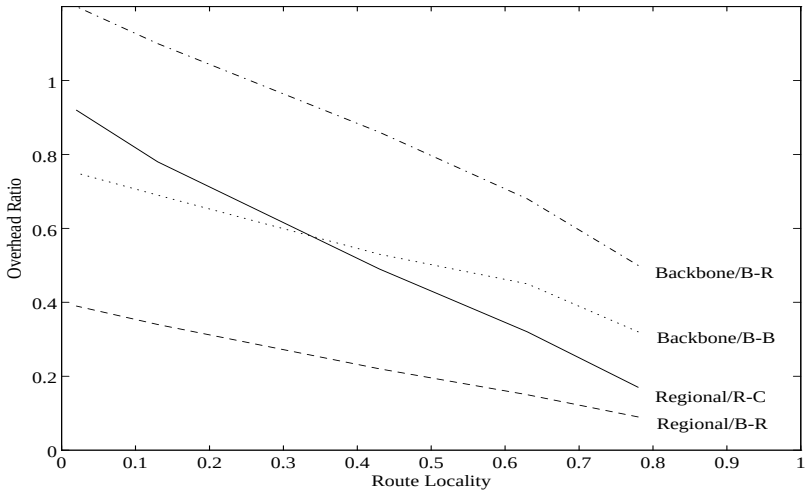

Figure 7: Ratio of RPU overhead to global flood for 100-node hierarchical network (HIER100) by node/link combination