Detour: a Case for Informed Internet Routing and Transport

Stefan Savage Tom Anderson Amit Aggarwal David Becker Neal Cardwell
Andy Collins Eric Hoffman John Snell Amin Vahdat Geoff Voelker John Zahorjan

Department of Computer Science and Engineering
University of Washington, Seattle

Technical Report UW-CSE-98-10-05

Abstract

Despite its obvious success, robustness, and scalability, the Internet suffers from a number of end-to-end performance and availability problems. In this paper, we attempt to quantify the Internet's inefficiencies and then we argue that Internet behavior can be improved by spreading intelligent routers at key access and interchange points to actively manage traffic. Our Detour prototype aims to demonstrate practical benefits to end users, without penalizing non-Detour users, by aggregating traffic information across connections and using more efficient routes to improve Internet performance.

1 Introduction

By any metric, the Internet has scaled remarkably; from 4 nodes in 1969 to an estimated 25 million hosts and 100 million users today. This reflects a sustained growth rate over three decades of roughly 80% per year, all while providing nearly continuous service. As a system, the Internet's growth has been matched only by the major infrastructure projects of the early 1900's: the electric power grid, the automobile, and the telephone network.

The Internet's scalability has been achieved only by the single-minded focus of its designers on robustness and adaptability [Clark 88]. Over the past three decades, the Internet has added support for automatic name translation, hierarchical routing, congestion avoidance, dynamic address assignment, multicast, mobility, and most recently, attempts at real-time support. The future challenges faced by the Internet will require continued evolution. As an example, four billion microprocessors were fabricated in 1997; in the future many of these embedded microprocessors will be Internet-connected, requiring the Internet to continue its rapid scaling well into the future.

Unfortunately, despite the overriding focus of the Internet design on robustness, for all practical purposes the Internet is *the* largest performance and availability bottleneck today for end-to-end applications. While it is possible to build highly available end servers using networks of workstations [Anderson et al. 95] and RAIDs [Patterson et al. 98], as anyone who has used the Web knows, the path to a server can be very slow and often completely unavailable. The result is lost productivity while users wait for Web documents to be transmitted over the network.

The scale, heterogeneity, and dynamic nature of the Internet make it difficult to difficult to determine the exact causes of Internet performance problems [Paxson & Floyd 97]. However, it is clear that a number of assumptions have changed since the Internet protocols were designed in the early 80's. For example, while Internet congestion control was designed to work well with connections that last many round trips – long enough for end-to-end feedback to work – most connections today carry only a small number of packets. Transferring a typical 10KByte Web page requires a minimum of 6-7 round trips as the server probes the network to determine the maximum rate at which it can send. If there is excess capacity in the network, the overhead of these probes will prevent the server from fully utilizing the network. If the network is congested, these short bursty connections will increase the probability that packets are dropped. Internet transport protocols were designed assuming that packet loss rates were under 1% [Clark 88]. Current packet loss rates have been measured as averaging 5-6% [Paxson 97, Balakrishnan et al. 98].

Assumptions about Internet routing have changed as well. The Internet was originally designed to provide universal reachability between networks; all network links were available to carry traffic for any host. Today's Internet restricts the exchange of routing information according to business agreements between service providers. This results in situations

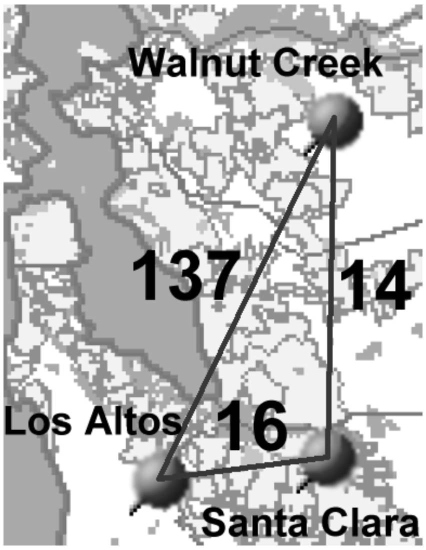

Figure 1: *Round trip time (in milliseconds) of packets sent between three Internet hosts in Northern California.*

where A can reach B and B reach C, but A can't reach C. Further, because current Internet routing ignores performance information, two hosts may be forced to communicate over excessively long or overloaded links. Adding a slow link can actually hurt performance, because packets can be routed over it in preference to faster links.

Finally, the Internet was built by a small community of researchers. In this environment, it was reasonable to assume that end hosts would cooperate in the management of network resources. As the Internet has evolved from a research project into a popular consumer technology, this assumption has lost its validity. For example, there are several commercial Internet "accelerators" that provide better performance for a single user at the expense of other users. In the future, expecting billions of Internet devices to cooperate to prevent network congestion is overly optimistic.

In this paper, we describe inefficiencies in routing and transport protocols in the modern Internet. In Sections 2 and 3, we present initial measurements that try to quantify these effects. Although our results are preliminary, they suggest that there is considerable room for improvement by doing more intelligent routing and congestion control.

We are constructing a prototype, called Detour, to investigate these ideas and to gain experience with potential solutions. Detour is *virtual Internet*, in which routers "tunnel" packets over the commodity Internet in place of using dedicated links. This design allows easy deployment of experimental infrastructure and, unlike dedicated network testbeds, is subject to real Internet traffic loads. We do not envision a tunneled network as a long term solution, but as a vehicle for research.

We describe the Detour prototype in Section 4, along with several of the research challenges we are addressing. Detour can potentially provide much better end-to-end application performance by using tunnels to route around Internet performance and availability bottlenecks, by actively smoothing traffic bursts in the tunnels, and by using measurements to more efficiently feed information about tunnel congestion to upstream routers and hosts. Non-Detour hosts will also benefit as traffic is diverted away from congested Internet links.

2 Routing Inefficiencies

A routing system is responsible for forwarding traffic between nodes of a network. There are a number of ways this system can be inefficient. It can forward packets along routes that are non-optimal or it can spread load unequally, such that some links are over-utilized while others are idle. There is significant anecdotal evidence that the Internet does both. For instance, the picture in Figure 1 depicts measured round trip times (in milliseconds) between three hosts in California's Bay Area. Curiously, we find that the Walnut Creek host can reach the host in Los Altos far faster by sending packets through Santa Clara rather than taking the "direct" route. This is because the "direct" route, chosen by the Internet, is via Chicago. In this section we describe reasons for why Internet routing may be inefficient and then provide data quantifying the magnitude of this effect.

We classify potential sources of routing inefficiencies into four principle categories:

- *Poor Routing Metrics.* Today's backbone, or "default-free", routers generally exchange only connectivity information between each other. In the absence of explicit policy rules, these routers make routing decisions by minimizing the number of independent Autonomous Systems (AS) traversed in getting to the destination. This metric correlates poorly with performance characteristics such as latency or drop rate; it does not change as the performance changes. This is not surprising when one considers that AS's generally correspond to organizational domains and can have enormous scope. For instance, all of MCI's Internet backbone is represented by a single AS number.

- *Restrictive Routing Policies.* Policy routing allows each AS to define its own rules for where to send traffic, which routes to advertise, and what traffic to transit. These policies are constructed to support the interests of individual service providers and can negatively affect overall reachability and performance. For instance, the common *early-exit* policy attempts to dispatch a packet bound for a host on a foreign network as soon as possible, even if this means sending it in the opposite geographical direction from where it is going. This is suboptimal but, for lack of alternative mechanisms, it is used to limit the amount of traffic one network carries for another. For similar reasons, large providers have established private peering relationships to exchange routing information and traffic, while smaller providers are left at the congested public exchange points. Consequently, packets sent from or destined to smaller networks have less diversity in their choice of routes and poorer connectivity as a result. Finally, some government-funded networks have legal limitations on how they may be used, resulting in policies that only carry traffic meeting some *acceptable use* criteria.

- *Manual Load Balancing.* Internet Service Providers and multi-homed organizations generally must pay a fixed fee for the links they use to connect their routers. Consequently, they are interested in balancing the amount of load on their links to take the best advantage of their fixed cost. There is no mechanism for doing this automatically so operators balance load by adding and removing policy rules on a daily basis in response to measured link utilization. While this may keep link utilization high, it does not make for the best routing decisions. In fact, it is extremely likely that there is an alternative assignment of routes to links that would achieve both equal utilization and better overall performance.

- *Single Path Routing.* Current Internet routers select a single path to reach a given destination. Alternate paths to the same destination may have underutilized links. This capacity can only be exploited by routing traffic along multiple paths to each destination.

While it is clear that each of these factors contribute to making a less efficient routing system, the magnitude of the overall problem is not obvious. We have undertaken a study to estimate the degree of routing inefficiency in the Internet. Using 43 publically available servers running the traceroute program, over the course of 35 days, we performed repeated traceroute queries between each pair of hosts. The time intervals between these requests were randomly distributed with a mean of 15 minutes. For the purpose of these experiments, we examined only the last record of the traceroute output, thereby avoiding well-known biases resulting from Internet routers that are slow to respond to ICMP messages. We also filtered our data to eliminate hosts that rate limit ICMP responses. For each sample between a pair of hosts, A and B, we considered its round trip time to be the average of the three samples returned by the last record of traceroute, and its drop rate based on the number of these samples (one, two, or three) that successfully completed a round-trip. We accumulated these samples, and calculated the median round trip time and mean drop rate for each path. After collecting our data, we sorted it to answer the following questions. For each path, AB, is it possible to pick a third point C, such that the round trip times or drop rates of $AC + CB < AB$?

The graph in Figure 2 illustrates our latency results. For roughly half of the paths measured, there is an alternative route that is faster. For 15 percent of the paths there is better than a 25 percent improvement in latency. The absolute benefits are also significant. For more than 15 percent of the paths, our alternate choice of route will shave 25ms over the round-trip of our connection. The results are similar when we look at packet loss rates. Figure 3 graphs the average drop rate seen on the default route for each path, compared to the packet loss rate observed when taking the best alternate route. For almost 80% of the paths there is an alternate route with a lower probability of having its packets dropped. In almost 50% of the paths the improvement is a factor of six or better. An ongoing part of this work is determining the relationship between these measurements and the underlying causes. As we continue our study we hope to quantify the individual effects of common routing policies, limited metrics, and single path routing.

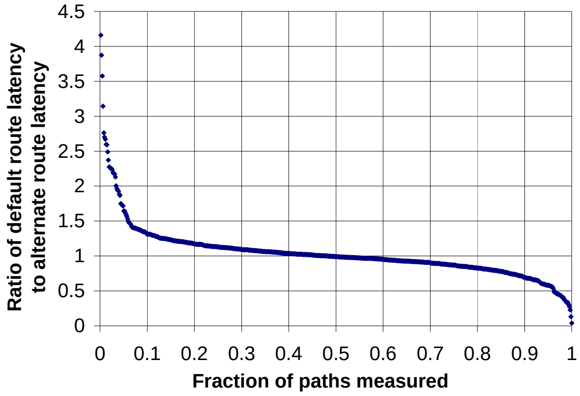

Figure 2: *Ratio of best alternate route latency to default route latency.*

There are several reasons to believe that our measurements underestimate the routing inefficiency present in the Internet. First, we only considered a small number of hosts, so our choices for alternate routes are relatively limited. Second, we only considered a single intermediate host, ignoring alternate routes with two or more intermediate hosts. Third, our sample hosts are not routers and hence any packet traversing the path ABC would undoubtedly traverse B's access links twice; once from A to B and again from B to C. Finally, our study concerns long-term averages and does not reflect the benefit of selecting an alternate path to avoid short-term hot spots. We observed anecdotally that some hot spots did in fact change during our measurement period.

However, it is important to remember that our measurements are of routing inefficiency, not of alternative routing policies. Both latency and packet loss are dependent on traffic, and without additional information about capacity and load we cannot predict the effects of re-routing traffic. One motivation for building Detour is to experimentally evaluate the impact of alternative routing policies.

3 Transport Inefficiencies

The behavior of the Internet infrastructure, as described in the previous section, has a direct impact on the performance experienced by users. However, it is not immediately clear how large this impact can be. In this section, we attempt to quantify the effects of latency and packet loss, and show that for today's transport protocols the effect on throughput can be quite dramatic. As an example, we'll show that the delivered bandwidth of a Web page transfer over a 10Mbps link can be as small as 75Kbps.

To understand these effects we must first recall that, in the Internet architecture, the network is a black box and provides no guarantees. Consequently, when sending a message a host starts with no information and must learn about the resource limitations of the receiver and the intermediate nodes in the network. Network latency is important for good performance because it controls how fast the sender may learn about changes in this environment. Packet loss is important because, in today's wired networks, transmission error rates are exceedingly low, so packet loss frequently signals congestion. Consequently, losing a packet is an implicit indication that the sender may be sending too fast and should slow down. As our results in Section 2 shows, many Internet paths suffer high background drop rates even before a connection starts sending; a connection has no way of telling whether a drop is caused by its own behavior or the burstiness of other flows. In this paper, we'll focus particularly on how latency and packet drop rate affect the popular Transmission Control Protocol (TCP). Other kinds of traffic, such as real-time traffic, face a somewhat different set of tradeoffs.

TCP is the dominant transport protocol in use today and underlies protocols for Web access, E-mail, file transfer, and news distribution. It is a reliable, connection oriented protocol and uses a sliding window mechanism for explicit flow control. TCP has a number of mechanisms for learning about and adapting to network resource limitations, first introduced in [Jacobson 88] and detailed in [Stevens 94]. They have proved to be immensely successful at preventing

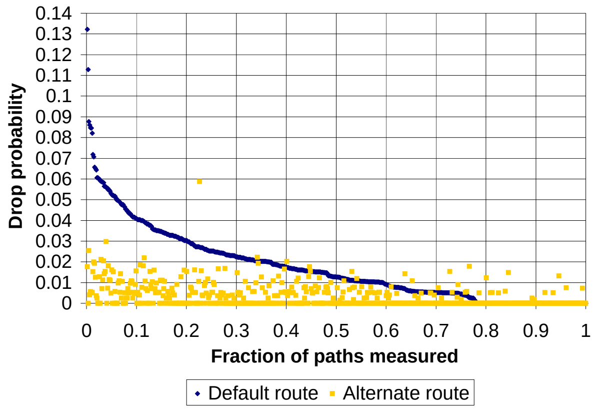

Figure 3: *The line represents the probability that a packet may be dropped while traversing the default route between two hosts. The dots represent the same probability assuming that the hosts use the best alternate route. Most dots are at the zero on the y axis.*

the "congestion collapse" events experienced in the late 1980's. Briefly:

- *Slow start.* When TCP opens a connection it "learns" the bottleneck bandwidth by exponentially expanding the size of the sender's window, starting from a single packet, until there is a loss. After a loss, the last window size for which there was no loss is taken as the estimate of capacity.

- *Congestion avoidance.* After finishing slow start, TCP continues to probe the network to see if the amount of capacity has changed. In the absence of a loss, the allowable window is increased additively by one packet for each round trip time. After a loss, the window is decreased multiplicatively by half.

- *Timeouts and fast retransmit.* There are two mechanisms for detecting a loss. The first is the expiration of the timeout timer set when a packet is sent. The timeout value is chosen somewhat conservatively to accommodate changes in measured round trip time caused by increased load on the network. The second loss indication is the arrival of three duplicate acknowledgments. Since the receiver sends an acknowledgment for the last in-sequence packet for every out-of-sequence packet it receives, duplicate acknowledgments are an implicit indication that packets have been reordered or, more likely, dropped. During this latter case, TCP assumes the missing packet was dropped and the next in-sequence packet is retransmitted immediately, hence the name for the algorithm, *fast retransmit*.

In the remainder of this section we'll explain how TCP's delivered bandwidth is affected by latency and packet loss. We'll start with an idealized network connection and iteratively refine the model to incorporate more detail. To illustrate, we'll use an example consistent with downloading a Web page over a completely unloaded cross country link into a typical LAN environment: a 10KByte transfer over a link with 10Mbps bandwidth, a 70ms round trip time, and a 536 byte maximum segment size (MSS).

3.1 Ideal Connection

In the absence of packet drops or resource limitations in the host or network, the maximum reliable throughput achievable by TCP is:

$$BW < \frac{WIN}{RTT}$$

where WIN is the maximum window size advertised by the receiver, and RTT is the round-trip time.

This is because TCP is a sliding window protocol and can send at most a full window of packets before receiving an acknowledgment from the receiver. TCP can advertise a window of up to 64Kbytes (larger with window scaling options, but these rarely matter in practice for reasons we will see shortly). Therefore, the maximum bandwidth TCP will achieve over a 10Mbps link is a little under 7.5Mbps. For our 10KByte document, the window size cannot exceed 10KBytes, so the maximum bandwidth to send this document and receive a reply is a bit under 1.2Mbps.

3.2 Long Flows

In reality packets are dropped because of network congestion, and the obtainable bandwidth suffers as a result. It is also the case that the Internet cannot accommodate connections that start by sending a full maximum window immediately, since this would increase burstiness at intermediate routers and increase the packet drop rate. For now we will limit our discussion to sufficiently long network flows such that startup effects have negligible impact on performance.

For sufficiently long network flows and no timeouts, a simple model for the average bandwidth delivered by TCP in the presence of loss is:

$$BW < (\frac{MSS}{RTT})\frac{1}{\sqrt(p)}$$

where p is the probability that a packet is dropped [Mathis et al. 97].

The rough intuition behind this model is that $\frac{1}{\sqrt(p)}$ corresponds to the average window size in packets when using the additive increase/multiplicative decrease algorithm. With larger drop rates, fewer packets can be sent before the window is decreased. Assuming a uniform packet drop probability of 5%, then the average bandwidth for our example transfer will be less than 275Kbps, more than four times less than our previous estimate. Sometimes fast retransmit is not effective and the sender must wait for a timeout, further reducing the achievable bandwidth. Incorporating these cases (see [Padhye et al. 98]), brings the average bandwidth for our transfer down to 228Kbps.

3.3 Short Flows

Most network flows are short and consequently the situation is usually even worse. There are a number of protocol choices and implementation artifacts that make startup behavior particularly poor and penalize short flows disproportionately. While new protocols, such as HTTP/1.1 [Nielsen et al. 97], promise to have some impact on increasing the average flow size, we expect short flows to be an important part of the traffic mix for some time.

The first and most obvious problem is connection setup. TCP is a connection oriented protocol and requires a three-way handshake during which the sender and receiver announce and acknowledge each other's connection request. In a short flow, the time for this connection setup is disproportionately large. Moreover, if the sender's request or the response of the receiver is dropped, then the sender will wait for a period and retransmit, each time increasing the timeout exponentially. Since the sending host has no information about the round trip time to the destination, most implementations set the initial timer to a conservative number (six seconds in BSD derived implementations). Consequently, if the drop probability in each direction is 5%, then the probability of losing one of these connection packets is $1 - (1 - .05)^2$, or 10%. That means that 10% of the attempts to open a connection would result in a wait of six seconds or more. Since most TCP implementations give up completely after three attempts to connect, one in a thousand connections would be denied even though the server is operating.

Having made a connection, the next problem is *slow start*. Since each change in the window size takes at least one round trip time, TCP may never reach the bottleneck bandwidth for short flows. Without drops, TCP's bandwidth will be roughly limited by:

$$BW < \frac{TransferSize}{RTT \cdot \lceil \log_{1.5}\left(\frac{TransferSize}{2 \cdot MSS} + 1\right) \rceil}$$

The intuition for this bound is that the number of round trip times necessary to send $TransferSize$ bytes is related to the log of $TransferSize$ because of slow start's exponential growth. The extra factors deal with details of commonly used acknowledgment policies.

Also, many TCP implementations poorly manage the interaction of slow start and the receiver's delayed acknowledgment algorithm. To reduce traffic on the network, receivers do not typically send an acknowledgment immediately, but instead wait to see if additional data arrives to allow acknowledgments to be combined. If no data arrives before the delayed acknowledgment timer fires (200ms in BSD derived implementations) then an acknowledgment is sent. However, during slow start many TCP implementations start with a window size of one and therefore must wait an average of 100ms for the receiver's delayed acknowledgment timer to fire.

Incorporating these effects, we find that our transfer takes a minimum of seven round trips times; one for connection setup, and six to send the data during slow start. Additionally, we will wait 100ms on average for the first delayed acknowledgment. Under these conditions the average bandwidth will be less than 140Kbps.

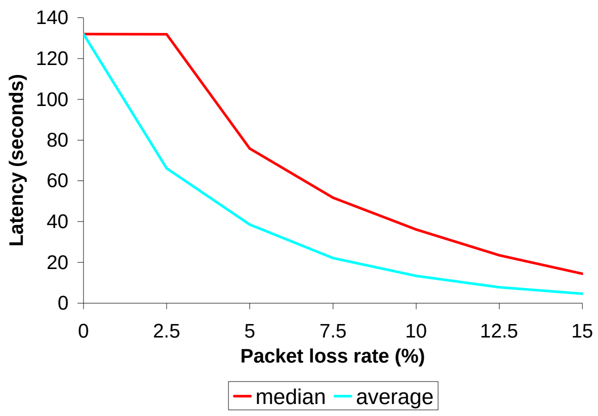

Figure 4: *Median and average bandwidth delivered transferring 10Kbytes over a link with a 70ms RTT, a 536byte MSS, and a 5% drop probability. Simulated using ns2.1.*

Of course, we do experience losses during connection setup and slow-start, which reduces the average bandwidth still further. These losses can be particularly expensive because the window is too small to trigger fast retransmit and the timeout value has not had enough time to converge to the round trip time. Figure 4 shows the results of a complete simulation of our 10KByte TCP transfer for various error rates. With 5% packet loss, the median bandwidth for a transfer such as ours will be about 75Kbps.

Summarizing, for our example Web transfer we observe an order of magnitude less bandwidth (75Kbps) than theoretically possible with a sliding window algorithm (1.2Mbps), and two orders of magnitude less bandwidth than was available (10Mbps).

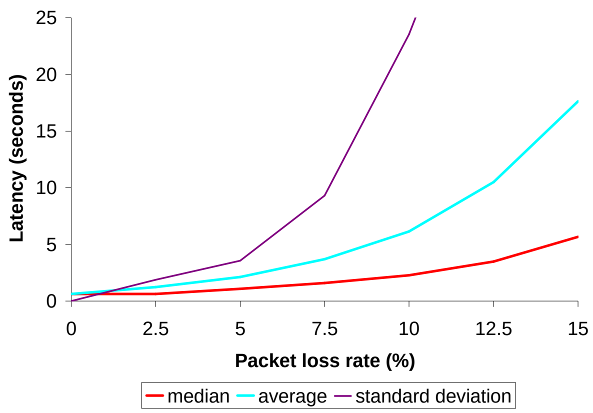

Figure 5: *Median, average, and standard deviation of the time required to transfer 10Kbytes over a link with a 70ms RTT and a 536byte MSS, as a function of packet drop probability. Simulated using ns2.1.*

Finally, note that because TCP uses exponential backoff and long initial timeouts, it has very high response time variance (seen in Figure 5). The consequence is that the Internet has trained many users to short circuit its congestion control – if you are unlucky enough to get a few packet drops you may be stuck in backoff and you can get better

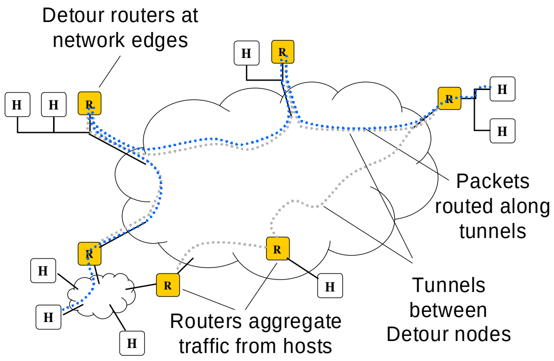

Figure 6: *Architecture of the Detour virtual Internet*

performance by clicking on "Stop" and "Reload". Needless to say, this is suboptimal behavior to encourage from the network perspective.

4 Detour architecture

Addressing these problems in today's Internet is a daunting task. The enormous heterogeneity and scale makes it difficult to anticipate the global effects of any change and also make it impossible to deploy any such change globally. As a consequence, our approach is to prototype a new network *virtually*, on top of the existing Internet. The resulting system, called Detour, will allow us to explore alternative host and network solutions while using real Internet links as the infrastructure and real Internet traffic as our input.

Detour is composed of a set of geographically distributed router nodes interconnected using *tunnels*. A tunnel can be thought of as a virtual point-to-point link. Each packet entering a tunnel is encapsulated into a new IP packet and forwarded through the Internet until it reaches the tunnel's exit point. This same mechanism has previously been used to form the multicast backbone (MBONE) and the experimental IPv6 backbone (6BONE). Tunnels are useful because they allow new routing functionality to be prototyped while using the existing network infrastructure.

A host wishing to use the Detour network will direct its outbound traffic to the nearest Detour router. Its packets will be forwarded along tunnels within the Detour network and will exit at a point close to the destination. In order that responses return in the same fashion, the system must perform network address translation, so the source address of the packet reflects the exit router and not the actual source. This complication is a necessary consequence of using tunnels to superimpose a new routing framework.

The Detour architecture is illustrated in Figure 6. It is important to notice that Detour routers are, generally speaking, edge devices and do not appear in the core of the network. We believe that controlling routing and congestion control at the edge of the network will offer sufficient control to address many of the problems we've raised, while at the same time we avoid potential problems supporting per-flow processing at the very high traffic bandwidths found in the core of the network.

In the remainder of this section, we describe the research agenda for Detour. Following the observations in Sections 2 and 3 we will discuss in turn the opportunities to improve the behavior of the routing system and the behavior of transport protocols.

4.1 Opportunities in Routing

Reviewing the data from Section 2, one opportunity is to use real performance metrics to chose routes within the routing system. Instead of AS numbers, Detour routers can exchange information about the measured latency, drop rate and bandwidth available along their tunnels. The challenge is to use this information to provide a routing service that automatically adapts and routes around emerging hot spots on the Internet, yet is is stable over short time scales. The early ARPANET used measurement-based adaptive routing, but it was abandoned because of instability – fluctuations in load would cause routes to change which in turn would cause the load to fluctuate. However, recent work by [Breslau 95] and others demonstrates that a well designed routing system can be both adaptive and stable.

Another opportunity we plan to explore is dynamic multi-path routing. Routers in the Internet generally send all packets to a particular destination along the same path. This is reasonable if all paths have excess capacity. However, when one path to a destination is congested and an alternate path is not, as demonstrated in Section 2, single-path routing limits the performance and utilization of the network. We hope to automatically load balance our system and avoid congestion before it occurs by randomly assigning flows to good paths and by dynamically varying how traffic is spread across such paths.

Finally, we recognize that there is an opportunity to specialize routing decisions to the needs of different service classes. For example, as described in Section 3.2, long TCP flows are best suited by the route that minimizes $RTT \cdot \sqrt{p}$. However, non-interactive multimedia flows (e.g. RealAudio) are less sensitive to round trip time and may be best served by uniquely minimizing p or minimizing the variance in RTT. We intend to classify traffic extensively in Detour, and select a routing policy best suited to the needs of each traffic class.

4.2 Opportunities in Informed Transport

Improving the latency and packet drop rate will automatically improve transport performance. However, the examination of TCP behavior in Section 3 suggests that there are additional inefficiencies that stem from inadequate information. An individual host is limited by its vantage point because it has a relatively small number of samples from which it must derive the state of network. Inevitably it will either be overly conservative or overly aggressive depending on the assumptions it makes. This behavior will be further exacerbated by short flows because they have little time to discover anything about the network before they complete. Ultimately, we would like to provide a transport protocol that is limited only by network resources and not by the ignorance of the end host. For example, if there is sufficient capacity we would like to transfer a Web page on the order of a single round trip time. We believe that to approach this goal will require abandoning the view that the network is a black box.

One approach is to use the network's expanded vantage point to inform the transport protocol. A Detour router at the edge of the network can observe many different flows, and this it can improve the fairness and accuracy of the underlying transport mechanism by sharing aggregate flow information. To illustrate this point, we use two examples from TCP, connection establishment and slow start.

When a host opens a TCP connection it has no information about round trip time and so defaults to a large number, six seconds in many implementations. As we explained earlier, a 5% drop rate implies that 10% of these connections will wait for six seconds or more. If the request is for a Web page with five or six inline images, then the odds of fetching the complete Web document within six seconds is only about one in two. This delay stems entirely from the uninformed choice of six seconds for the initial timeout. By observing other traffic destined for the same network a Detour router can provide an informed round trip time estimate for subsequent hosts using that path, or, without modifying the end host protocol implementation, the router may choose to retransmit the connection establishment request on the host's behalf.

Similarly, the slow start algorithm exists because when a host starts a connection it has no good way to know what its fair share of the bottleneck capacity is along that path. As a consequence, TCP necessarily sends a burst of up to twice the bottleneck capacity as it overestimates and scales back, causing packets to be unnecessarily dropped. These packet drops and consequent retransmits could be reduced or avoided by informing the host of its fair share of the bottleneck. A Detour router is in an ideal position to make this determination. Minimally, the router may *eagerly* drop packets that exceed the bottleneck bandwidth estimate, as these packets would otherwise uselessly consume network resources on their way to being dropped by downstream routers. More aggressively, the network may communicate a fair share estimate and variance to the host and allow it to initiate slow start with an appropriate window size.

5 Conclusion

The meteoric rise in popularity of the Web has caused the Internet to experience more than a few growing pains. Society is increasingly coming to depend on the Internet for more and more of its everyday operation – from purchasing books and automobiles, to making travel arrangements, to disseminating news and entertainment, to teleconferencing, to controlling embedded devices. As this happens, the Internet must adapt to provide high performance and highly reliable service. Although the Internet is highly scalable in many respects, in several respects it is not. As the diversity increases in Internet link performance and drop rates, we will need to move the Internet towards performance and reliability-sensitive routing. As the number of connections sharing high bandwidth links increases, we will need to implement congestion control strategies that do not require dropping packets on every separate connection through a bottleneck resource. The perennial challenge facing the Internet is fixing scalability and performance problems as they appear, while still providing robust service. The University of Washington Detour project is attacking these issues from the perspective of a building a virtual Internet, with the goal of providing better performance and availability to end users on top of the existing Internet.

References

[Anderson et al. 95] Anderson, T., Culler, D., Patterson, D., and Team, T. N. A Case for Now (Networks of Workstations). *IEEE Micro*, 15(1):54–64, February 1995.

[Balakrishnan et al. 98] Balakrishnan, H., Padmanabhan, V., Seshan, S., Stemm, M., and Katz, R. TCP Behavior of a Busy Internet Server: Analysis and Improvements. In *Proceedings of the IEEE Infocom Conference*, San Francisco, CA, March 1998.

[Breslau 95] Breslau, L. M. *Adaptive Sourcde Routing of Real-Time Traffic in Integrated Services Networks*. PhD dissertation, University of Southern California, December 1995.

[Clark 88] Clark, D. The Design Philosophy of the DARPA Internet Protocols. In *Proceedings of ACM SIGCOMM '88*, pages 106–114, Palo Alto, CA, September 1988.

[Jacobson 88] Jacobson, V. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM '88*, August 1988.

[Mathis et al. 97] Mathis, M., Semke, J., Mahdavi, J., and Ott, T. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm). *ACM Computer Communications Review*, 27(3), July 1997.

[Nielsen et al. 97] Nielsen, H., Gettys, J., Baird-Smith, A., Prud'hommeaux, E., Lie, H., and Lilley, C. Network Performance Effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of ACM SIGCOMM '97*, September 1997.

[Padhye et al. 98] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM SIGCOMM '98*, Vancouver, BC, September 1998.

[Patterson et al. 98] Patterson, D., Gibson, G., and Katz, R. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the International Conference on the Management of Data*, pages 109–116, June 1998.

[Paxson & Floyd 97] Paxson, V. and Floyd, S. Why We Don't Know How To Simulate The Internet. In *Proceedings of the 1997 Winter Simulation Conference*, December 1997.

[Paxson 97] Paxson, V. End-to-End Internet Packet Dynamics. In *Proceedings of ACM SIGCOMM '97*, September 1997.

[Stevens 94] Stevens, W. R. *TCP/IP Illustrated, Volume 1*. Addison-Wesley, 1994.