DETOUR: INFORMED INTERNET ROUTING AND TRANSPORT

DESPITE ITS OBVIOUS SUCCESS, THE INTERNET SUFFERS FROM END-TO-END PERFORMANCE AND AVAILABILITY PROBLEMS. WE BELIEVE THAT INTELLIGENT ROUTERS AT KEY ACCESS AND INTERCHANGE POINTS COULD IMPROVE INTERNET BEHAVIOR BY ACTIVELY MANAGING TRAFFIC.

Stefan Savage, Thomas Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan
University of Washington, Seattle

By any metric, the Internet has scaled remarkably—from four nodes in 1969 to an estimated 40 million hosts today. This reflects a sustained growth rate over three decades of more than 80% per year, during continuous service. In system growth, the Internet has been matched only by the major infrastructure projects of the early 1900s: the electric power grid, the automobile, and the telephone network.

The Internet's scalability is the result of the single-minded focus of its designers on robustness and adaptability.[1] 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. Future Internet challenges will require continued evolution. For example, consider the fact that 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, although its design has focused overridingly on robustness, for all practical purposes the Internet is the largest performance and availability bottleneck today for end-to-end applications. Indeed, it is possible to build highly available end servers using networks of workstations (NOWs)[2] and redundant arrays of inexpensive disks (RAIDs).[3] However, as anyone who has used the Web knows, the path to a server can be very slow or often completely unavailable. The result is lost productivity while users wait for Web documents to be transmitted over the network.

The Internet's scale, heterogeneity, and dynamic nature make it difficult to determine the exact causes of Internet performance problems.[4] However, it is clear that several of the assumptions made during the design of the Internet protocols have less validity today than they did in the early 1980s—see the box, "The evolving Internet."

In this article, we describe inefficiencies in routing and transport protocols in the modern Internet. We also attempt to quantify these effects. Although our results are preliminary, they suggest that there is considerable room for improvement through 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 a virtual Internet, in which routers "tunnel" packets over the commodity Internet instead of using dedicated links. This design allows easy deployment of an experimental infrastructure

0272-1732/99/$10.00 © 1999 IEEE

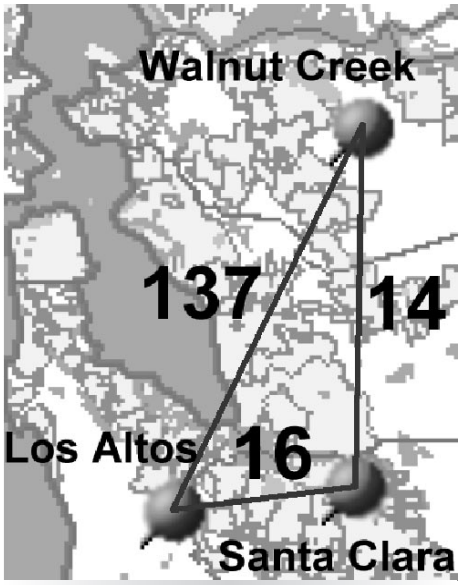

Figure 1. Round-trip time (in ms) of packets sent between three Internet hosts in Northern California.

and, unlike dedicated network testbeds, it 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 along with several of the research challenges we are addressing. Detour can potentially provide better end-to-end application performance by routing around Internet performance and availability bottlenecks and by enhancing congestion control mechanisms using aggregate traffic information.

Routing inefficiencies

A routing system is responsible for forwarding traffic between nodes of a network. There are several ways such a system can be inefficient. It can forward packets along nonoptimal routes, or it can spread load unequally, overutilizing some links while leaving others idle. There is significant anecdotal evidence that the Internet does both. For instance, 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 much 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 that Internet routing may be inefficient; we then provide data quantifying the magnitude of this effect.

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

- *Poor routing metrics.* Today's backbone, or "default-free," routers generally exchange only connectivity information, and not performance information. In the absence of explicit policy rules, these routers make decisions by minimizing the number of independent autonomous systems (ASs) traversed along the way to the destination. This metric correlates poorly with performance characteristics such as latency and drop rate. This is not surprising when one considers that ASs generally correspond to organizational domains and can have enormous scope. For instance, MCI's entire Internet back-

The evolving Internet

The TCP/IP Internet protocol architecture was designed in the early 1980s, at a time when there were many fewer hosts connected to it and typical long-haul links carried only 56 Kbps. Many of the assumptions underlying the Internet's design have changed since then. For example, the designers of Internet congestion control intended it to work well with connections that last many round-trips—long enough for end-to-end feedback to work. Most connections today, however, carry only a small number of packets. Transferring a typical 10-Kbyte Web page requires a minimum of six to seven 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 of dropped packets. The designers of Internet transport protocols assumed that packet loss rates would be less than 1%, yet current packet loss rates have been measured as averaging 5% to 6%.

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 where A can reach B and B can 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 that 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 some of its validity. For example, there are several commercial Internet "accelerators" that provide better performance for a single user at the expense of other users. Expecting billions of Internet devices to cooperate to prevent network congestion in the future is arguably too optimistic.

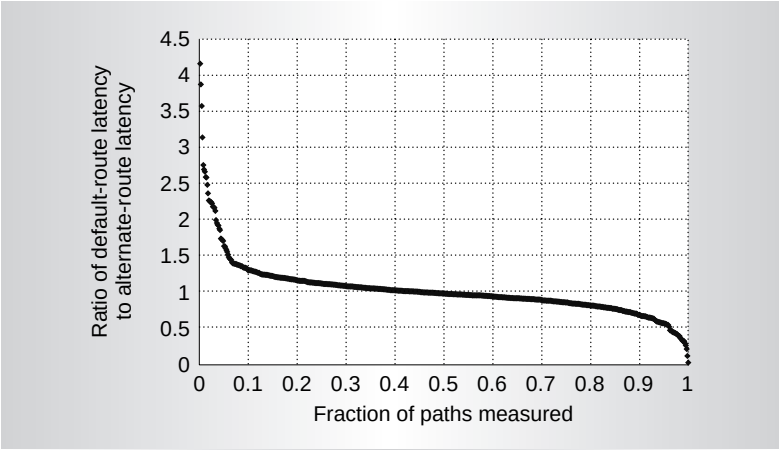

Figure 2. The ratio of best-alternate-route latency to default-route latency.

bone 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 carry. Individual service providers construct these policies to support their own interests, so they can sometimes 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, service providers use it to limit the amount of traffic one network carries for another. For similar reasons, large providers have established private relationships to exchange routing information and traffic; small 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; this results in policies that allow them to carry only traffic meeting their acceptable-use criteria.
- *Manual load balancing.* Internet service providers and multihomed 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 an alternative assignment of routes to links 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.

Although it is clear that each of these factors contributes 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. We call a route between two hosts inefficient when there is some alternate route with superior latency or packet drop rate. Our goal is to measure the fraction of routes for which this occurs and the magnitude of this inefficiency.

Unfortunately, while it is easy to directly measure the performance of the default route between two hosts, it is difficult to obtain the same metrics for alternate routes or even to discover what those alternate routes might be. Instead, we have opted for a conservative approximation based only on pairwise host measurements.

Our methodology was as follows: For a group of hosts, we collected a full set of pairwise latency and drop-rate measurements. Then, for each pair of hosts A and B, we searched our data for some third host C, such that the round-trip times or drop rates of AC + CB were less than those for AB.

We used 43 publicly available servers running the traceroute program, over the course of 35 days, to perform repeated traceroute queries between each pair of hosts. We randomly distributed the time intervals between these

requests, with a mean of 15 minutes per host. 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 Internet control-message protocol messages. We also filtered our data to eliminate hosts that rate-limit ICMP responses.

For each sample between a pair of hosts, we recorded the round-trip time as the average of the three samples returned by the last record of traceroute. Similarly, we recorded the drop rate as the number of these samples that were unable to successfully complete a round-trip. We accumulated these samples, and calculated the median round-trip time and mean drop rate for each path.

The graph in Figure 2 illustrates our latency results. For roughly half of the paths measured, there is a faster route. For 15% of the paths, there is an alternative that offers an improvement in latency better than 25%. The absolute benefits are also significant. For more than 15% of the paths, our alternate route choice will shave at least 25 ms from the round-trip time of our connection.

The results are similar when we look at packet loss rates. Figure 3 graphs the average drop rate of the default route for each path, compared to the packet loss rate observed for the alternate route with the fewest packet losses. For almost 80% of the paths, an alternate route offers a lower probability of dropping packets. 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.

There are several reasons to believe that our measurements underestimate the Internet's routing inefficiency. First, we considered only a small number of hosts, so our choices for alternate routes are relatively limited. Second, we considered only a single intermediate host, ignoring alternate routes with two or more intermediate hosts. Third, our sample hosts are not routers. 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 over short time scales.

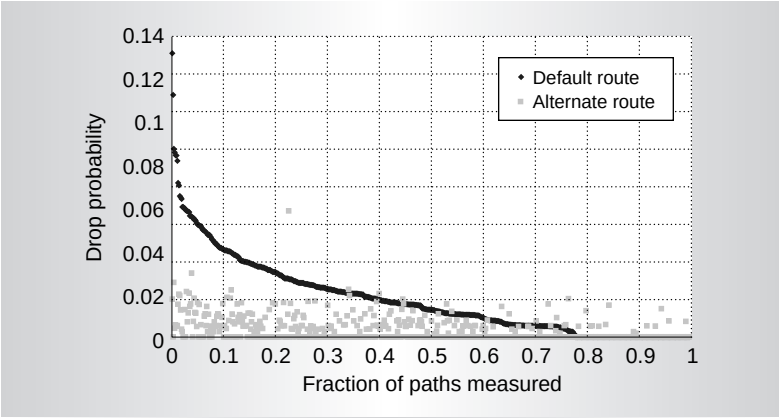

Figure 3. Average drop rates for default routes compared to those for best-alternative routes. The dark line represents the observed probability that a packet is dropped while traversing the default route between two hosts. The light dots represent the same probability assuming that the packet is sent along the best alternate route. Most dots are at zero on the *y* axis.

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

Transport inefficiencies

The behavior of the Internet infrastructure, as we described in the previous section, has a direct impact on the performance that users experience. 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 we 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 10-Mbps link can be as small as 75 Kbps.

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 the resource limitations of the receiver and the intermediate nodes in the network. Roughly speaking, the network's latency limits how quickly a host may learn about changes in the receiver's resources, and the packet drop rate indicates limitations in the resources of intermediate nodes in the network. As latency increases, the host must wait longer to receive acknowledgments indicating that more data can be sent, and consequently will send more slowly. As the packet loss rate increases, the host assumes that congestion is occuring and will also send more slowly.

As shown by the results we presented in the section on routing inefficiencies, many Internet paths suffer high background drop rates. This limits the baseline throughput that a connection can expect. Here, 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 trade-offs.

TCP is the dominant transport protocol in use today; it underlies protocols for Web access, e-mail, file transfer, and news distribution. It is a reliable, connection-oriented protocol that uses a sliding-window mechanism for explicit flow control. TCP has several mechanisms for learning about and adapting to network resource limitations.[5,6] These mechanisms have proved immensely successful at preventing more of the congestion collapse events of the late 1980s:

- *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, TCP resumes sending using the last window size for which there was no loss.
- *Congestion avoidance.* After finishing the slow start mechanism, TCP continues to probe the network to see if the capacity has changed. In the absence of a loss, the protocol increases the allowable window additively by one packet for each round-trip time. After a loss, it decreases the window multiplicatively by half.
- *Time-outs and fast retransmit.* There are two mechanisms for detecting a loss. The first is the expiration of the time-out timer TCP sets when it sends a packet. The protocol chooses the time-out value somewhat conservatively to accommodate changes in measured round-trip time due to increased load on the network. The second loss indication is the arrival of three duplicate acknowledgments. This is because the receiver sends an acknowledgment for the last in-sequence packet for every out-of-sequence packet it receives. Thus, duplicate acknowledgments are an implicit indication that packets have been reordered or, more likely, dropped. In this latter case, TCP assumes the missing packet was dropped and retransmits the next in-sequence packet immediately. This is the source of the algorithm's name, fast retransmit.

Next, we explain how latency and packet loss affect TCP's delivered bandwidth. 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 cross-country link into a typical LAN environment: a 10-Kbyte transfer over a link with 10-Mbps bandwidth, a 70-ms round-trip time, and a 536-byte maximum segment size (MSS).

Ideal connection

Because TCP is a sliding-window protocol, it can send at most a full window of packets before receiving an acknowledgment from the receiver. Therefore, the maximum reliable throughput TCP can achieve is

$$BW < WIN / RTT.$$

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

TCP can advertise a window up to 64 Kbytes (larger with window-scaling options, but these rarely matter in practice, for reasons we will see shortly). Therefore, TCP's maximum bandwidth over a 10-Mbps link is a little under 7.5 Mbps. For our 10-Kbyte document, the window size is not a limitation; however, we are still limited by the round-trip time needed to positively acknowledge the data's arrival. Divid-

ing the transfer size (10 Kbytes) by the sum of the transmission time (about 8 ms) and the round-trip time (70 ms) yields an average throughput of just over 1 Mbps.

Packet losses

In reality, network congestion causes dropped packets, and bandwidth suffers as a result. For sufficiently long network flows that experience no time-outs, a simple model for the average bandwidth that TCP delivers in the presence of loss is

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

where p is the probability that a packet is dropped.[7]

The rough intuition behind this model is that $1/\sqrt{p}$ corresponds to the average window size in packets when using the additive-increase/multiplicative-decrease congestion avoidance algorithm. With larger drop rates, the protocol can send fewer packets before decreasing the window. Assuming a uniform packet-drop probability of 5%, the average bandwidth for our example transfer will be less than 275 Kbps, little more than a quarter of the 1 Mbps we estimated earlier.

Sometimes fast retransmit is not effective, and the sender must wait for a time-out; this further reduces the achievable bandwidth. Incorporating these cases (see Padhye et al.[8]) brings the average bandwidth for our transfer down to 228 Kbps.

Start-up effects

Most network flows are short; consequently the situation is usually even worse. There are several protocol choices and implementation artifacts that make start-up behavior particularly poor, penalizing short flows. While new protocols such as HTTP/1.1[9] promise to increase the average flow size somewhat, 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 that requires a three-way handshake during which the sender and receiver announce and acknowledge each other's connection requests. In a short flow, the time for this connection setup is disproportionately large. Moreover, if the network drops the sender's request or the receiver's response, the sender waits for a period before retransmitting, each time increasing the time-out exponentially. Because the sending host has no information about the round-trip time to the destination, most implementations set the initial time-out to a conservative number (often three seconds). If the drop probability in each direction is 5%, the probability of losing one of these connection packets is $1 - (1 - 0.05)^2$, or 10%. That means that 10% of the attempts to open a connection would result in a wait of three seconds or more. Because 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. Because the round-trip time limits the rate of window growth, TCP may never reach the bottleneck bandwidth for short flows. Without packet drops, TCP's bandwidth will be roughly limited by

$$W < \frac{TransferSize}{(RTT \times \left\lceil \log_{1.5}\left[\left(\frac{TransferSize}{2MSS}\right)+1\right]\right\rceil)}$$

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.

An additional impediment is that many TCP implementations poorly manage the interaction of slow start and the receiver's delayed-acknowledgment algorithm. To reduce network traffic, receivers do not typically send an acknowledgment immediately, but instead wait to see if additional data will arrive to allow it to send combined acknowledgments. If no data arrives before the delayed-acknowledgment timer fires (after 200 ms in implementations derived from BSD—the Berkeley Software Distribution version of Unix), the receiver sends an acknowledgment. However, during slow start, many TCP implementations start with a window size of one and therefore must wait an average of 100 ms for the receiver's delayed-acknowledgment timer to fire.

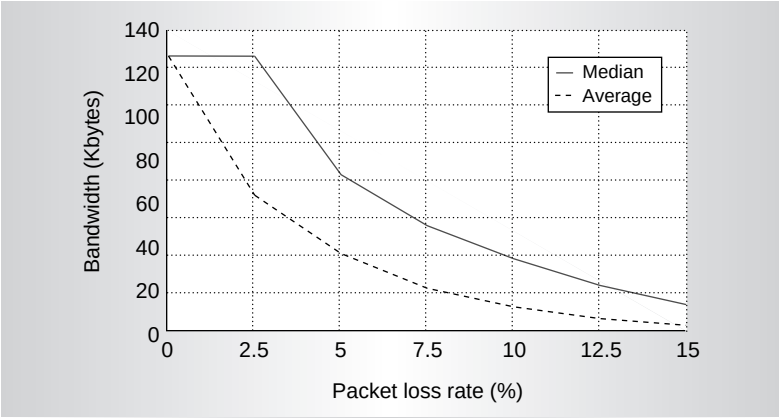

Figure 4. Median and average bandwidth delivered transferring 10 Kbytes over a link with a 70-ms round-trip time and a 536-byte maximum segment size, as a function of packet loss rate.

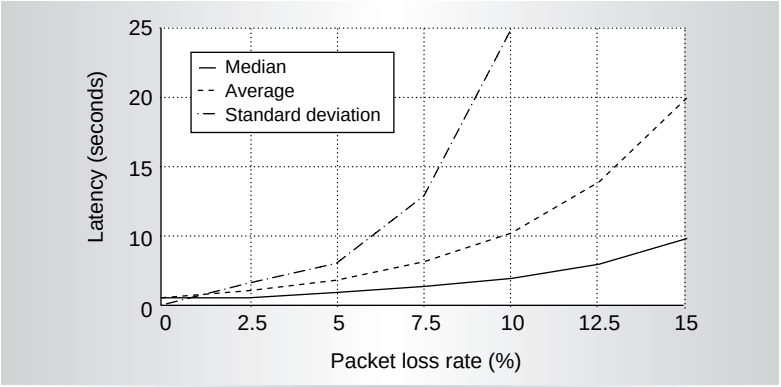

Figure 5. Median, average, and standard deviation of the time required to transfer 10 Kbytes over a link with a 70-ms round-trip time and a 536-byte maximum segment size, as a function of packet loss rate.

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

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 time-out value has not had enough time to converge to the round-trip time.

Figure 4 shows the results of a complete simulation of our 10-Kbyte TCP transfer for various error rates. With 5% packet loss, the median bandwidth for a transfer such as ours is about 75 Kbps.

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

Finally, because TCP uses exponential back-off and long initial time-outs, 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 back-off, and you can get better performance by clicking "Stop" and then "Reload." Needless to say, this behavior should not be encouraged from the network perspective.

Detour architecture

Experimenting with new approaches to these problems in today's Internet is a daunting task. The enormous heterogeneity and scale make it difficult to anticipate the global effects of any change and 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, allows us to explore alternative host and network solutions while using real Internet links as the infrastructure and real Internet traffic as our input.

Figure 6 depicts the Detour architecture. Detour consists of a set of geographically distributed router nodes interconnected using tunnels. We can think of a tunnel 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. Researchers have used this mechanism previously to form the multicast backbone (Mbone) and the experimental IPv6 backbone (6Bone). Tunnels are useful because they allow us to prototype new routing functionality while using the existing network infrastructure.

A host wishing to use the Detour network will direct its outbound traffic to the nearest Detour router. The router will forward these packets along tunnels within the Detour network, and the packets will exit at a point close to the destination. So that responses return in the same fashion, the system must perform

network address translation so that 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.

Note that Detour routers are, generally speaking, edge devices and do not appear in network cores. We believe that controlling routing and congestion control at the edge of the network will be sufficient to address many of the problems we've raised. At the same time, we avoid potential problems supporting per-flow processing at the very high traffic bandwidths in the core of the network.

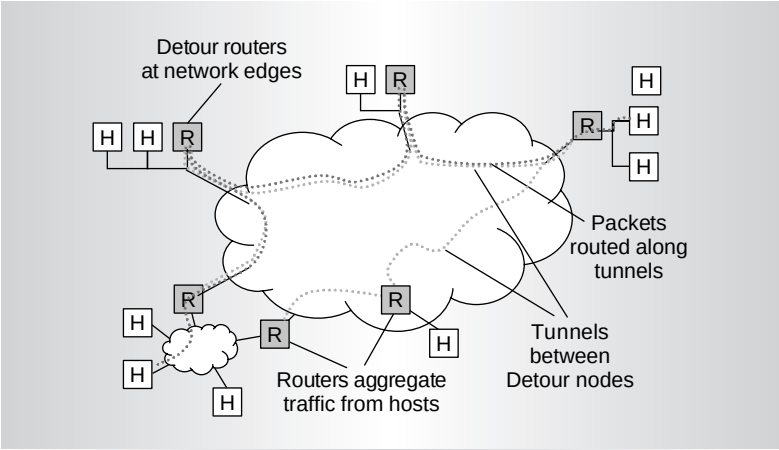

Figure 6. Architecture of the Detour virtual Internet. Nodes labeled R represent Detour routers; those labeled H represent hosts.

Opportunities in routing

Reviewing the data we presented in the section on routing inefficiencies, we see that one opportunity is to use real performance metrics to choose 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 stable over short time scales. The early Arpanet used measurement-based adaptive routing, but it was abandoned because of instability—fluctuations in load caused routes to change, which in turn caused the load to fluctuate. However, recent work by Breslau[10] and others demonstrates that a well-designed routing system can be both adaptive and stable.

Another opportunity we plan to explore is dynamic multipath 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, single-path routing limits the network's performance and utilization. We hope to automatically balance loads in our system and avoid congestion before it occurs by randomly assigning flows to good paths and by dynamically varying how routers spread traffic 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 we described in the section on packet losses, long TCP flows are best suited by the route that minimizes $RTT/\sqrt{p}$. However, streaming multimedia flows (such as 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.

Opportunities in informed transport

Improving latency and packet drop rates automatically improves transport performance. However, our examination of TCP behavior suggests that additional inefficiencies 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 network's state. Inevitably, the host will be either overly conservative or overly aggressive. Short flows exacerbate this behavior because they offer little time for the host to discover anything about the network before they complete.

Ultimately, we would like to provide a transport protocol 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 in a single round-trip time. Approaching this goal will require relaxing 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 thus 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 very conservative number, three seconds in many implementations. As we explained earlier, a 5% drop rate in each direction implies that 10% of these connections will wait for three 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 three seconds is only about one in two. This delay stems entirely from the uninformed choice of three seconds for the initial time-out. 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, hosts use the slow-start algorithm 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 that is up to twice the bottleneck capacity as it overestimates and then scales back, causing unnecessary packet drops. We could reduce or avoid these packet drops and consequent retransmits if the host knew its fair share of the bottleneck.

A Detour router is in an ideal position to make this determination. Minimally, the router may preemptively 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.

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 its everyday operation: Users go online to purchase books and automobiles, make travel arrangements, disseminate news and entertainment, teleconference, and control embedded devices. As this happens, the Internet must adapt to provide high-performance and highly reliable service.

As the diversity of link performance and drop rates increase, we will need to consider routing based on performance and reliability information. As the number of short flows sharing high-bandwidth links increases, we will need to develop proactive congestion control strategies. The University of Washington Detour project is attacking these issues by deploying a virtual network testbed to explore the costs and benefits of such informed routing and transport mechanisms. MICRO

References

1. D. Clark, "The Design Philosophy of the DARPA Internet Protocols," *Proc. ACM SIGCOMM 88,* ACM, New York, 1988, pp. 106-114.
2. T. Anderson et al., "A Case for Now (Networks of Workstations)," *IEEE Micro,* Vol. 15, No. 1, Feb. 1995, pp. 54-64.
3. D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proc. Int'l Conf. Management of Data,* ACM, 1989, pp. 109-116.
4. V. Paxson and S. Floyd, "Why We Don't Know How to Simulate the Internet," *Proc. 1997 Winter Simulation Conf.,* IEEE, Piscataway, N.J., 1997.
5. V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM,* ACM, 1988, pp. 314-329.
6. W.R. Stevens, *TCP/IP Illustrated, Vol. 1,* Addison-Wesley, Reading, Mass., 1994.
7. M. Mathis et al., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Comm. Rev.,* Vol. 27, No. 3, July 1997, pp. 67-82.
8. J. Padhye et al., "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *Proc. ACM SIGCOMM,* ACM, 1998, pp. 303-314.
9. H. Nielsen et al., "Network Performance Effects of HTTP/1.1, CSS1, and PNG," *Proc. ACM SIGCOMM,* ACM, 1997, pp. 155-166.
10. L.M. Breslau, *Adaptive Source Routing of Real-Time Traffic in Integrated Services Networks,* doctoral dissertation, Computer

Science Department, University of Southern California, Los Angeles, 1995.

Stefan Savage is pursuing a PhD in computer science at the University of Washington. His research interests include resource management in wide-area networks, high-performance operating-system design, and automated software-testing tools. He received a BS in applied history from Carnegie Mellon University. Stefan is a member of the IEEE Computer Society and the ACM.

Thomas Anderson is an associate professor at the University of Washington. His projects of the last few years have included the design of Digital's SRC AN2 gigabit ATM switch, language-independent software fault isolation, Berkeley NOW and IRAM, WebOS, and now Detour. He is a member of the Computer Society.

Amit Aggarwal is a graduate student in computer science at the University of Washington. He received a BTech in computer science and engineering from the Indian Institute of Technology, Delhi. His current interests include operating systems, networks, and wide-area distributed systems.

David Becker is a research staff member at the University of Washington. He received a BS from Bethel College and an MS from the University of North Carolina, Chapel Hill. His research interests include kernel design, high-speed networks, and wide-area services.

Neal Cardwell is pursuing a PhD in computer science at the University of Washington. His general research interests include networks, distributed systems, and operating systems, but he is currently involved in tracing, analyzing, and modeling the performance of TCP in today's networks. Cardwell received a BS in computer science from the College of William and Mary. He is a member of the ACM.

Andy Collins is a graduate student in computer science at the University of Washington. He received his BS in electrical engineering and computer science from the University of California, Berkeley. His research interests include wide-area networking, virtual network configuration and management, and systems support for flexible routing and forwarding algorithms.

Eric Hoffman is a research staff member at the University of Washington. His research interests include Internet-scale information flooding, language support for systems programming, and information visualization. He has also worked at Ipsilon Networks, Caida, the Information Sciences Institute of USC, and the US Naval Research Laboratory.

John Snell is finishing a BS in computer science at the University of Washington. His research interests include distributed application design, wide-area networks, and Java programming. Snell is a member of the ACM.

Amin Vahdat is an assistant professor in the Computer Science Department at Duke University. His research focuses on system support for wide-area network services, including security, naming, resource allocation, and cluster support for high-performance scalable services. Vahdat received his PhD in computer science from the University of California, Berkeley.

Geoff Voelker is pursuing a PhD in computer science at the University of Washington. His research interests include distributed systems, binary rewriting systems, and mobile computing. He received a BS in electrical engineering and computer science from the University of California, Berkeley, and a master's degree in computer science and engineering from the University of Washington. Voelker is a member of the IEEE Computer Society and the ACM.

John Zahorjan is a professor at the University of Washington. His current research is in the area of resource management in parallel and distributed systems, with a focus on distributed, real-time rendering. Zahorjan received a PhD in computer science from the University of Toronto.

Direct questions concerning this article to Stefan Savage, Department of Computer Science and Engineering, Box 352350, University of Washington, Seattle, WA 98195; savage@cs.washington.edu.