

User-level Internet Path Diagnosis

Ratul Mahajan Neil Spring David Wetherall Thomas Anderson

University of Washington

ABSTRACT

Diagnosing faults in the Internet is arduous and time-consuming, in part because the network is composed of diverse components spread across many administrative domains. We consider an extreme form of this problem: can end users, with no special privileges, identify and pinpoint faults inside the network that degrade the performance of their applications? To answer this question, we present both an architecture for user-level Internet path diagnosis and a practical tool to diagnose paths in the current Internet. Our architecture requires only a small amount of network support, yet it is nearly as complete as analyzing a packet trace collected at all routers along the path. Our tool, tulip, diagnoses reordering, loss and significant queuing events by leveraging well deployed but little exploited router features that approximate our architecture. Tulip can locate points of reordering and loss to within three hops and queuing to within four hops on most paths that we measured. This granularity is comparable to that of a hypothetical network tomography tool that uses 65 diverse hosts to localize faults on a given path. We conclude by proposing several simple changes to the Internet to further improve its diagnostic capabilities.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Measurement techniques

General Terms

Measurement, performance

Keywords

Path diagnosis, measurement tools

1. INTRODUCTION

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

L. Lamport

Lamport's classic quote is an apt description of the Internet – when it fails to perform as expected it is nearly impossible to tell

Authors' e-mails: {ratul,nspring,djw,tom}@cs.washington.edu

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA.
Copyright 2003 ACM 1-58113-757-5/03/0010 ...\$5.00.

what happened. Consider a user whose Web download is excessively slow due to a problem within the network. There are many potential causes, including data corruption due to “dirty fiber,” small buffers or mistuned RED parameters at routers, packet reordering, or simply, inadequate provisioning by the user's Internet service provider (ISP). All of these problems are hidden inside the “black box” abstraction of the Internet. They typically manifest themselves only as packet loss, even a small amount of which degrades the performance of TCP [29].

In this paper we focus on the problem of locating performance faults such as loss, reordering, and significant queuing at specific links, routers, or middleboxes (e.g., firewalls) along Internet paths. We consider this problem from the point of view of an ordinary user, with no special privileges, in a general setting where paths cross multiple administrative domains. We refer to this as the problem of user-level path diagnosis.

It is important that unprivileged users be able to diagnose their paths. Performance depends on the interaction of the properties of the entire path and the application. Since operators do not share the users' view of the network, they are not always well-placed to even observe the problem. Even when they are, they may be little better off than users. Operators may have no more insight than unprivileged users for problems inside other administrative domains because of the distributed management of the Internet, and most Internet paths cross multiple domains.

Of course, users must also be able to do something about the problems they observe. Often, detailed knowledge is enough. By mapping the faulty component to the ISP that owns it [47, 24], the user can directly contact the responsible ISP leading to faster problem resolution; operators are frequently not even aware of the problem. Further, we believe ISPs will better provision and manage their networks if their users can readily identify faults. For example, users can demand that their ISP provide additional capacity if upstream links are frequently overloaded. In the absence of fault localization, ISPs tend to blame poor performance on factors beyond their control. Finally, recent research has examined various ways to route around network problems, either through manipulating BGP policy choices or via overlay-level source routing [3, 41, 44]. These techniques are more effective and scalable with fault localization than blindly trying all possibilities [37].

Unfortunately, existing diagnosis tools have significant limitations because they are based on round trip measurements to routers. For instance, pathchar measures the queuing at each hop along the path by analyzing round trip times to successive routers [18]. But this approach has the fundamental disadvantage that it confuses the properties of the forward and reverse paths. The asymmetry of most Internet paths, with different paths to and from routers, makes it even harder to draw strong conclusions about per-hop behavior.

We began our work by observing two little exploited features in many Internet routers. These features, ICMP timestamps and IP identifier counters, return remote timing and ordering information. Leveraging this support, we designed and built *tulip*, a tool to diagnose reordering, loss, and queuing along Internet paths. We validate *tulip*'s correctness using a combination of end-to-end and internal consistency measurements. We find that it is able to localize problems even with today's incomplete deployment of ICMP timestamps and IP identifier counters. On most paths we measured, *tulip* narrows the location of reordering and loss to within three hops and queuing to within four hops. Based on these initial tests, we have started using *tulip* to diagnose operational network problems. That work is at an early stage and is not reported here.

To understand how effective *tulip* is at pinpointing faults, we compared it with a promising alternative approach: network tomography that infers the location of faults by comparing observations on overlapping paths taken from multiple vantage points. While to our knowledge there are no widely available tomography tools that can diagnose arbitrary paths, the approach is attractive because it needs no network support and research suggests that such a tool might be feasible [13, 31, 50]. We found that the diagnosis granularity of *tulip* is comparable to that of an idealized tomography tool that coordinates measurements across 65 diverse hosts to localize faults. An implication is that multiple vantage point tomography is not essential to diagnose paths in the current Internet.

We are also interested in understanding how the Internet should evolve to provide support for path diagnosis. There are numerous proposals for adding measurement support into the Internet, e.g., IPMP [23], but to date there has been no theory explaining how powerful each of the options are, either singly or in combination. The router features above, while effective in many contexts, are by themselves incomplete for diagnosing the range of potential problems in the Internet. To explore this question of completeness, we develop an architecture that is nearly as powerful as a complete packet trace at all routers along the path, but is lightweight enough to be implementable using current technology. The features leveraged by *tulip* can be considered approximations of this architecture, and by contrasting the two we can reason about the limitations of *tulip*. We then propose simple, incremental changes to the Internet that would further improve the effectiveness of *tulip*.

We begin the rest of this paper by describing our architecture in Section 2 to help place the features used by *tulip* in context. We next describe *tulip* in Section 3 and evaluate it in Section 4. In Section 5 we discuss the limitations of *tulip* and suggest several simple changes to the Internet to improve its diagnostic capabilities. We discuss related work in Section 6 and conclude in Section 7.

2. DIAGNOSIS ARCHITECTURE

In this section we design an architecture that enables users to diagnose performance faults experienced by their flows. This support must be scalable, work across multiple organizations, and be lightweight enough to be part of packet forwarding. Starting with an ideal support that is complete for our problem but unrealistic, we progressively reduce it to a level that is practical yet retains much of the diagnostic ability of the ideal system. The purpose of this exercise is to document the missing capabilities and so demonstrate the near completeness of the resulting architecture. In Section 3.1, we argue that the Internet already supports many of the features of our architecture, and we show how to leverage this existing support to build a practical path diagnosis tool.

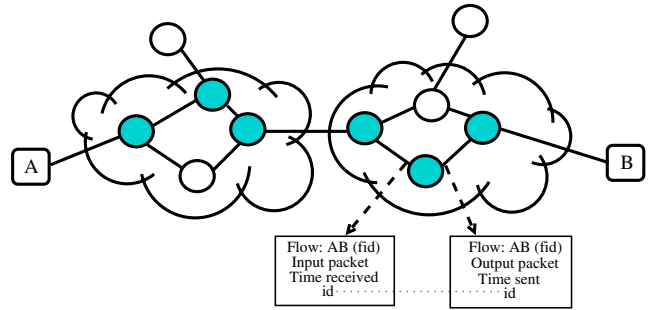


Figure 1: Illustration of the ideal packet trace approach. The clouds represent different administrative domains. The path from *A* to *B* traverses the shaded routers. Routers log every packet at incoming and outgoing interfaces. To diagnose the path, *A* can query for these log items to find properties such as where a packet was lost (the routers before the loss would have the corresponding log entry, while those after the loss would not) and the time taken by each link to forward a packet.

2.1 Problem

Our problem setting is shown in Figure 1. Host *A* is communicating with host *B* by sending packets that travel along a series of routers and links, shown as the shaded path. This path crosses multiple administrative regions, indicated by the clouds. We consider packets exchanged between *A* and *B* that belong to an application such as a Web server and its client. The collection of packets that are exchanged is the flow of interest. Note that with Internet routing a single flow may traverse more than one path, but this is omitted from the figure for simplicity.

Our goal is to allow *A* and *B* to cooperate to find *performance faults* that affect their flow, and measure the magnitude and pinpoint the location of the fault. Intuitively, performance faults are properties, such as significant reordering, loss, and high delay, that adversely impact applications. These are observable properties rather than underlying causes such as Ethernet auto-negotiation failures, congestion, or unreliable links. In our view, once the problem is localized, the domain administrator can be notified and further privileged debugging can isolate the root cause.

We assume that routers and middleboxes along the path are cooperative, and do not experience Byzantine failures such as intentionally misreporting diagnostic information.

2.2 An Ideal Trace-based Solution

We begin by considering a hypothetical trace facility, illustrated in Figure 1. Routers log packet activity and make these traces available to users. Due to privacy concerns, users can only access traces for packets that belong to their flows. The log at each router is recorded for both input and output interfaces. It consists of each input packet and its arrival time, each output packet and its send time, and the linkage between the two packets. The linkage is needed as packets may be transformed inside network devices such as network address translators (NATs).

The architecture above is impractical for deployment, but it is interesting for framing an argument because it provides an ideal level of support for path diagnosis. All manner of flow properties can be estimated for path segments inside the network, including reordering, duplication, loss, corruption, queuing delay, transformations (e.g., network address translation), and so forth. This is important because performance faults depend on the application, and so we do not have a fixed list of properties of interest. We must therefore

ensure that any architectural support is broadly useful and sufficient even for unanticipated properties. With the path trace, any noticeable effect on the flow at some point(s) along the path can be localized by comparing traces taken before and after that point. Without network support, a pair of hosts can only determine *what* happened over the entire path, not *where* it happened.

2.3 Packet-based Solutions

Architectures for collecting system-wide traces similar to that outlined above do exist, such as NetFlow [9], SPIE [46], trajectory sampling [12] and Magpie [5]. Most of these systems use sampling or aggregated trace representation for tractability. However, before we improve the efficiency of our architecture we make a fundamental switch from recording information in logs to recording information in packets as they are forwarded. The reason for this is simple: providing access to traces for many, arbitrary users requires a significant security infrastructure. We work around this problem by moving the information from routers to packets, which are already delivered to the right users. The systems mentioned above are intended for a few privileged users, or they too would face this issue.

2.3.1 Complete Embedding

Still in our idealized world, we ignore packet size and efficiency issues in the switch from an ideal trace to a nearly equivalent embedding of information in packets. Each router along the path records information into each packet that it forwards. As a packet is received, the global address of the router input interface and the time are recorded. As it is sent, the global address of the router output interface, the time, and the entire content of the input packet are recorded. Since the end points of the flow cooperate in the diagnosis, the database consists of the same information and has simply been inverted. Instead of recording the attributes of every packet at each router it traverses, we record the attributes of every traversed router in each packet. This is the reason that an output packet records the whole input packet.

Barring two exceptions, the scheme above is equivalent to the path trace. One exception is when packets are lost, as their trace information is also lost. The second exception is when packets are corrupted such that the diagnostic information contained in them is mutated. Fortunately, most corruption events are turned into loss due to link-level checksums and can be diagnosed similarly (see below). Corruption that does not lead to loss is a transformation, discussed in the next section. While it might seem that packet duplication is also an exception, the recording of time in the packets enables us to discover the responsible link or router. The duplicated packets would have the same timestamps before duplication, and different timestamps after that.

The ability to at least observe the location of loss is valuable for diagnosis. To restore this we require that routers record a flow specific counter in the packet, denoting the number of packets processed for the flow. Differences in counters received from routers reveal loss points. For example, if a packet is lost between the second and third router, the counters in the next successful packet will show that the second router processed one more packet for this flow than the third router.

2.3.2 Reduced Embedding

The complete embedding loses little information compared to the original trace, but to be practical we must make it more efficient. We first remove the step of embedding the complete input packet in the output packet; this is obviously prohibitive, causing the packet to double in size at each hop. The purpose of this em-

bedding was to capture packet transformations. Instead, we assume that the packets in a flow can be uniquely identified even if they are transformed. This can be easily accomplished by including a unique identifier that is not changed by the network. Such an identifier already exists: the IP identifier that is placed in IP packets by the sender to assist with packet fragmentation (a transformation). To locate the transformation point, a sender must also be able to send packets part way along the path and observe the result. Borrowing from the Internet again, this can be accomplished with the time-to-live (TTL) field in packets; TTL is decremented by routers along the path, and when it reaches zero, in our approach routers return an error message to the sender that includes the original packet.¹ The sender can look at the embedded packet to decide if it was transformed before it reached the responding router. Probes with different initial values of TTL can be used to locate the point of transformation. This approach requires that all routers on the return path, including the transformer, forward the error message without modification.

We further reduce the overhead by recording information only at the input interface. Output interface information can often be inferred from the next input interface information. For example, the output timestamp is the next input timestamp minus the fixed propagation delay of the preceding link. The link propagation delay can be estimated using a series of input timestamp measurements at the two ends [18]. Even when we cannot approximate the output information, the net effect is to reduce the resolution of diagnosis by lumping together all components between consecutive input recording points.

At this point routers along the path are recording input interface identifier, arrival time, and flow counter in the packet. The first two of these can be obtained with the IP options for Record Route and Internet Timestamp (with flags 0 or 1) [40]. However, by design these options can return information for only the first few hops along the path.

2.3.3 Constant Space Embedding

The reduced embedding suffers from the problem that the packet size is variable as it depends on the path length. This is undesirable because of packet fragmentation issues [20]. To remedy this problem and further reduce the packet size, we select only one router along the path to record the input information. This is similar to the IP Internet Timestamp option with flag 3, but includes more complete information. In our architecture, the router is selected with another TTL field, the sample TTL, which is set by the sender and counts down until a sample is taken. This moves us to a system in which information is sampled and results are inferred using statistical methods, which is comparable to other traffic monitoring systems [9, 14, 45]. The assumption here is that path properties are stationary enough to be measured [55].

However, sampling paths introduces another complication – paths can change. While we can discover the path between the two hosts (using either TTL field) and then sample it, these steps are not atomic. There is no guarantee that the samples will correspond to the same path. We need to detect path changes to avoid mixing samples from different paths. We solve this problem using a fixed size field to record the path signature. Routers add their identity into the field using a simple operation, e.g., a Bloom filter insertion using an OR operation [52]. With high probability, depending on the number of bits in this field, hosts can now detect route changes.

¹TTL is somewhat of a misnomer, as it reflects hop-count rather than time. TTL-expired responses in the current Internet include the header of the original packet, not the whole packet.

Field	Purpose
Path signature	Records path information
Sample TTL	Selects the sampling router
Timestamp	Time at the sampling router
Counter	Flow counter from the sampling router
Interface Id	Interface address of the sampling router

Table 1: Packet header fields in our architecture.

2.3.4 Real clocks

As a final simplification, we remove the requirement of ideal clocks. Real clocks may not be synchronized with true time. Fortunately, an unsynchronized clock that runs at a stable rate (over short periods of time) is both simple to implement – a glorified cycle counter – and sufficient to measure delay variation, an important diagnostic property because it corresponds to queuing. While it might seem that synchronized clocks would be better, they are not suitable for measurement tasks because the resynchronization process, if invisible to endpoints, can cause large swings in packet timings [34].

Another limitation of real clocks is that they have finite precision, limiting the granularity with which we can obtain packet timings. Finite precision clocks lose ordering information too, if two packets receive the same timestamp, but per-flow counters preserve that information in our architecture.

2.4 An Architecture

At this point, we have a sketch of a simple architecture enabling unprivileged users to diagnose their paths. The fixed-size fields, in addition to those already present in IP, needed in our architecture are shown in Table 1.

Compared to the ideal trace facility presented earlier, this is a vastly simplified architecture which possesses a surprisingly similar diagnostic ability. Briefly, the key limitations are that measurement is statistical and relies on sampling and stationarity of path properties (as do practical trace systems), that only relative rather than absolute one-way delays can be inferred, and that locating transformation points requires a TTL-driven search.

3. DIAGNOSIS TOOLS

We now switch our focus from architectural support to practical tools for performing path diagnosis. In Section 3.1 we show how some key primitives of our architecture can be approximated in the Internet. In the remainder of this section we use these approximate primitives to develop tulip.

3.1 Internet Approximations

We use the following Internet mechanisms that approximate the packet embedding mechanisms sketched in the last section:

1. **Out-of-band measurement probes:** We approximate the in-band sampling of our architecture with out-of-band probes to routers. These probes serve the purpose of both the sample TTL and interface identifier (obtained from the router address in the probe response) of our architecture. In-band probes ensure that the measured behavior matches that of the application; out-of-band probes have the fundamental limitation that if measurement traffic is treated differently than application traffic, our tools may not diagnose network faults properly. We restrict ourselves to probes that traverse the same path as application traffic. This is usually true for TTL-

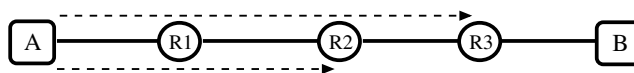


Figure 2: Inferring link properties from path properties. The properties of link $R2 \rightarrow R3$ can be estimated by subtracting the measured properties of path $A \rightarrow R2$ from those of path $A \rightarrow R3$.

limited probes that are addressed to the destination, but some of our probes are addressed directly to the routers along the path. We use traceroute to verify that the path taken by a probe is a prefix of the path to the destination; this may not be the case when directly addressing routers due to policy routing.

2. **ICMP timestamp requests to access time at the router:** We approximate the router timestamps in our architecture using ICMP timestamp requests [39]. Alternatively, we could have used the IP Internet Timestamp option, but packets with options often travel through the network more slowly [7]. In response to an ICMP timestamp request, the router returns its local time with a resolution up to 1 ms. We found this feature to be accessible in over 90% of the routers we measured. This result agrees with that of Anagnostakis *et al.* [2], but we also found that exceptions are systematic. For instance, none of AT&T’s routers support ICMP timestamps.
3. **IP identifiers instead of per-flow counters:** All IP packets include an identifier (IP-ID) field, designed to be unique among all packets sent by a machine to the same destination, to allow IP fragments to be re-assembled. A common implementation on routers uses a 16-bit counter, incremented with every generated packet. Routers generate packets, and thus increment their counter, for routing updates, TTL-expiration replies, and other management traffic; the counter is not incremented during normal packet forwarding. Over 70% of the routers we measured implement the IP-ID using a counter. Because routers do not source much traffic, the counter can allow us to discover the order and number of probe packets that reach the router, at least over a short duration if no other probe or management packets intervene. Thus, the IP-ID field approximates the per-flow counter described in our architecture. In Section 5.3 we discuss a small change to the IP-ID counter implementation to allow for interleaved probes along multiple flows.

These approximations enable the Internet to mimic some of the key features of our architecture. We next describe how tulip leverages these features for path diagnosis in the Internet. In Section 5 we discuss the limitations of these approximations and provide some solutions to address them.

3.2 Building Blocks

We focus on diagnosing packet reordering, loss, and queuing delay. These three properties directly influence the performance of transport protocols and applications. For instance, the performance of TCP is sensitive to even small amounts of loss [29]. It also suffers when more than a few packets are received out of order [49]. A high queuing delay is indicative of congestion, believed to be the source of most losses in the Internet. A variable queuing delay leads to jitter, which can hurt the performance of interactive applications. Obviously, these are not the only metrics worth measuring in a diagnostic tool; they are meant to be both practically useful

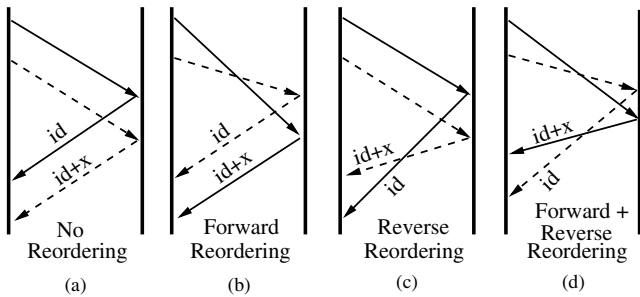


Figure 3: Detecting forward and reverse path reordering. The router must insert sequential IP-IDs in its responses.

and illustrative of the capabilities of our approach. We plan to add support for pinpointing other properties, such as checksum errors and bandwidth limits [19].

For reordering, loss, and queuing delay, locating the fault point requires inferring the properties of individual links. Since we cannot measure these directly, we instead compare measurements of the forward paths to the two ends of the link, as illustrated in Figure 2. Thus, in each case, we need an accurate way to measure the path to each router between the source and destination. The remainder of this section considers how to measure these prefix paths for each metric of interest.

3.2.1 Packet Reordering

Measuring forward path packet reordering requires information about the order in which two packets reach the remote router. This can be obtained using IP-IDs. Our technique, illustrated in Figure 3, sends two probes to the router. Each probe elicits a reply; we can distinguish the replies because they include the header of the probe that triggered them. When there is no reordering in either direction (Figure 3a), we receive the responses in order and the second probe evokes a higher IP-ID. With only forward path reordering (Figure 3b), the second probe evokes a lower IP-ID and its response reaches the source first. With only reverse path reordering (Figure 3c), the first probe evokes a lower IP-ID but its response reaches the source second. Concurrent bidirectional reordering has a signature of its own (Figure 3d).² Our approach is an adaptation of the dual-connection test of sting [6]. Sting targets end hosts running TCP-based servers at well-known ports; our technique can be used with routers as well as hosts.

The extent of reordering, or the reordering rate, is simply the number of reordered probe pairs divided by the number of probe pairs for which both responses are received. We discard probe pairs for which either of the two probes or their replies are lost.

3.2.2 Packet Loss

We next consider the problem of determining whether the forward path to a router is lossy. We use IP-IDs for this measurement as well. Our technique is illustrated in Figure 4. We send three probes to the router: the middle probe is a data packet, and the other two are control packets. The source receives all three responses when there are no losses in either direction. If the data packet is lost on the forward path and the control packets arrive at the router close enough in time to evoke consecutive IP-IDs, the source sees the response pattern in Figure 4b, and knows that the packet was lost in the forward direction.

²A trivial extension of this method can detect forward and reverse path packet duplication with equal IP-ID values.

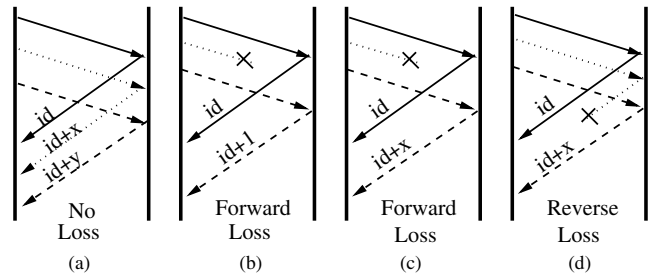


Figure 4: Detecting forward path loss. Consecutive IP-IDs in the control responses, as in (b), implies a forward path loss. Events (c) and (d) are indistinguishable at the source.

Receipt of non-consecutive IP-IDs in the control responses is not a reliable indicator of the direction of the data loss. This behavior can stem from either of the two events – indistinguishable at the source – in Figures 4c and 4d. If it could be known with relative certainty that consecutive IP-IDs were evoked, for example, if the router uses per-flow IP-ID counters, then the separation of the control response IP-IDs by two implies a reverse path loss.

It is not possible to infer the direction of data loss when a control packet or its response is lost.

Our loss detection mechanism has three requirements. First, the control packets should be preserved even when the data packet is lost. Fortunately, this tends to occur, despite possible bursty losses, when the data packets are large (which is usually true except for voice applications) and the control packets are small. This is because routers are more likely to drop large packets, perhaps due to the lack of buffer space. We tested this hypothesis using 40-byte control packets and 1000-byte data packets. We sent 500 probe triplets each to 5000 IP addresses chosen randomly from the Skitter list [17]. In over 60% of the cases when any packet in the triplet was lost, only the data packet was lost. With equal sized packets, the same statistic was 12% for 40-byte triplets and 5% for 1000-byte triplets.

Second, the probes should arrive at the router in order and close enough in time to evoke consecutive IP-IDs. We can reduce reordering while mostly retaining consecutive IP-IDs by introducing a small inter-packet spacing. The reordering probability is higher when a small packet is followed by a large packet (see Section 5.1), as is the case for our second and third probes. So we use an inter-packet spacing of 0.5 ms between the first and second packets, and 3 ms between the second and third packets. Measurements to over 5000 routers that generate sequential IP-IDs show that we can evoke consecutive IP-IDs at least 80% of the time from over 90% of the routers.³ These results are not very sensitive to the exact inter-packet spacing.

Third, rate-limiting of responses at routers should not be interpreted as path losses. The misinterpretation would happen if the response to only the data packet is suppressed due to rate-limiting. To detect such rate-limiting, we experimented with the insertion of a 1000-byte *check* packet after the data packet. The check packet takes the same path to the measured router but goes beyond it. For path losses, there will be a correlation between the data and check packet losses. In a rate-limiting scenario, the check packet will tend

³Some routers appear to increment IP-IDs by two instead of one. Currently, tulip considers such routers as generating non-consecutive IP-IDs. We also found that some routers return an IP-ID of zero for 1000-byte probes while returning sequentially increasing IP-IDs for 40-byte (traceroute-like) probes.

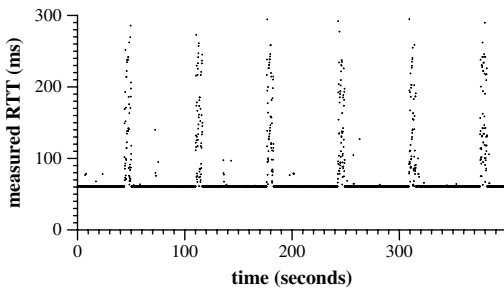


Figure 5: An example of the measured round trip times (RTT) to a router. An ICMP timestamp request was sent every 100 ms for 400 seconds. One-way delays showed a similar effect.

to survive even when the response to the data packet is suppressed. We used this technique early on in our experiments, but found that it is not needed in practice. Responses to closely following control packets are suppressed as well whenever there is rate-limiting. A consequence is that our three-packet methodology is useful even without sequential IP-IDs. It can measure round trip loss to a router by considering only data losses; a single packet mechanism cannot distinguish between rate-limiting and path losses. To minimize network load, we used only the three packet methodology (without the check probe) in the experiments presented in this paper.

We use simple definitions to compute the loss rate. The forward loss rate is the number of triplets with the forward loss IP-ID pattern (Figure 4b) divided by the total number of triplets sent. For the round trip loss rate we ignore the IP-ID values; it is calculated as the number of triplets with only data loss divided by the number of triplets sent. Both these measures underestimate the loss rate when a control loss accompanies the data loss.

3.2.3 Packet Queuing

We next discuss how to measure packet queuing delay on the forward path to a router. Measuring one-way delay requires knowledge of the time at which the packet arrives at the other end. For this purpose, we use ICMP timestamps as recently suggested by Anagnostakis *et al.* [2]. Our methodology is similar but for the two practical problems mentioned below. The response to an ICMP timestamp request contains the local time when the probe arrived at the router.⁴ However, with unsynchronized clocks and an unknown propagation delay, we cannot measure the queuing delay directly. In the absence of route changes, the propagation delay is constant, and any transit time variation is due to queuing. Assuming that some of the probes will observe no queuing, we can obtain the queuing delay for a probe by subtracting the minimum observed transit time from its transit time [18].

The mechanism above assumes that the difference between the two clocks remains constant for the duration of the measurement. This assumption is violated in the presence of relative clock skew (the difference in the rate at which the clocks progress) or clock jumps. These behaviors can be corrected at the source by calibrating the time returned by the router [27, 36, 53]. For this purpose, we use `fixclock` [42], which is robust to congestion and non-linear clock behavior such as jumps and changes in the rate of progress.

⁴A small fraction of routers seem to insert garbage in the timestamp field. It is easy to filter them out as their timestamps do not progress as expected. Some routers insert timestamps in the wrong byte order, which we detect and correct.

Property	Conditions
Reordering and loss	Sequential IP-IDs in TTL-expired responses; or prefix path and sequential IP-IDs in either Port-Unreachable or ICMP echo responses
Queuing	Prefix path and ICMP timestamp support

Table 2: Conditions that must be satisfied by a measurable router on the path. Port-Unreachable responses are generated in reply to probes targeted at a closed port at the router. Prefix path holds when the path for a packet addressed to the router is a prefix of the path to the destination.

Two practical problems remain. First, the time reported by the routers includes the time to generate the ICMP timestamp response. The ICMP specification includes timestamps for both request arrival and response transmission. But in practice routers insert just one timestamp. While other researchers have shown that the time to generate ICMP responses is minimal most of the time [2, 16], we discovered that the deviations from the minimal generation time are systematic. Periodically, the time a router takes to generate an ICMP response increases from a negligible amount to 100-300 ms. These jumps typically occur every sixty seconds, suggesting that the underlying cause of this behavior is periodic maintenance activity at the router [32].⁵ An example of such behavior is shown in Figure 5. We found a similar pattern for many routers, and for all kinds of responses that a router might generate, including ICMP echo, TTL-expired, and Port-Unreachable. This behavior is a measurement artifact – traffic traversing the router is not delayed.

We account for this behavior by spreading our measurements over a period longer than the maintenance window and by using a queuing delay metric that is robust to outliers. Within these constraints, applications are free to pick a delay metric that matters to them. In our experiments we use the median; a high median delay indicates persistent queuing at the router.

The second practical problem is that certain network elements, such as cable modems and wireless links, may introduce large delays due to arbitration for media access. If unaccounted for, these devices will always appear to have a high queuing delay. The difference between such delays and queuing may not matter to certain applications, for instance, those interested only in jitter. Applications interested only in queuing can flag such elements by sending back-to-back probes; if the experienced delay is dominated by queuing, both probes would suffer similar delays. We have not tested this methodology widely across various shared media devices, but in initial tests it was successful with 802.11 wireless links.

3.3 Tulip

In this section we present tulip, which uses the building blocks above to localize performance faults. It takes the destination as input and identifies the faulty segment along the forward path. Equivalently, tulip can be used at the destination to locate problems on the reverse path. Tulip is publicly available [51] and built on top of Scriptroute, a flexible Internet measurement infrastructure [48].

Tulip starts by discovering the path to the destination using traceroute and identifying the measurable routers along it. Measurable routers are those that support the forward path diagnosis primitives summarized in Table 2. Some routers do not produce sequential IP-IDs for the TTL-expired responses, but do so for other responses. We use whichever probe type evokes sequential IP-IDs.

⁵This behavior is colloquially called a “coffee-break” and most likely occurs when routers push forwarding tables to the line cards.

In the next step, based on a user option, tulip conducts a parallel or binary search. Parallel search interleaves measurements to different routers by cycling through them in rounds. A measurement entails executing the building block of the relevant property, for instance, two probes for reordering. A round consists of one measurement to each router. To control the network load, a router is measured after the measurement to the previous router completes. After a configurable number of rounds (M) with a configurable wait time (W) between the start of consecutive rounds, tulip outputs the performance statistics, e.g., the reordering rate, for paths to all measured routers. The faulty path segment will have a non-faulty near end and a faulty far end.

Binary search reduces diagnostic traffic at the cost of increased diagnosis time. It first measures to the destination, and stops if the path is fault-free. Otherwise, it searches the path by measuring to the midpoint, recursively focusing on whichever half exhibits the problem.

Both search modes identify the faulty path segment, surrounded by measurable routers. Optionally, tulip can further localize faults through round trip probing because that does not require sequential IP-IDs or ICMP timestamps, and can rule out faults from some hops. For example, if the round trip to the third router in a four-hop faulty segment is fault-free, the fault must be after that router.

3.3.1 Network Load and Diagnosis Time

The network load and diagnosis time of tulip depends on the search mode and the number of measurements per router. Assume that M measurements of B bytes each are used, wait time is W seconds, and the number of measurable routers is L . Then, the bandwidth requirement is at most $\frac{BL}{W}$ Bps (bytes per second) for parallel search, and $\frac{B}{W}$ Bps for binary search. These values overestimate the bandwidth requirement when the round trip time to routers is large, such that a round cannot be finished in W seconds. Binary search uses less bandwidth but is $\log_2(L)$ times slower than parallel search.

For the experiments in the paper, we used $M=200$, $B=80$, $W=1$ for reordering, $M=500$, $B=1080$, $W=1$ for loss, and $M=1000$, $B=40$, $W=0.5$ for delay. We used more measurements for loss than for reordering because it is rarer. For delay, M has to be large enough to calibrate clocks [42]. The number of measurable routers (L) on most paths is less than 10. These numbers translate to a parallel search bandwidth requirement of 800 Bps for reordering and delay, and 10,800 Bps for loss. For binary search, the requirement is 80 Bps and 1080 Bps respectively. The diagnosis time is approximately 10 and 30 minutes per path.

The bandwidth requirement of parallel search (or equivalently, the time for binary search) can be decreased using some simple modifications. The first few hops can be skipped if they are known to be fault-free; for instance, when they are being used by other flows without problems. Probing to some routers can be aborted early if it can be determined that they are fault-free. Depending on the magnitude of the fault, fewer than M measurements may be needed to rule out a particular router, while more are needed to confirm the identity of the faulty segment.

3.4 Diagnostic Process

We envision tulip being used in two contexts. First, an end user might notice that network performance is particularly slow to some destination. Since the problem could be due to misconfiguration at the end host (e.g., too small a TCP receive window) the user could use existing tools to rule in or out those effects [4, 30]. If network performance is to blame, tulip could be invoked to isolate where along the path attention should be focused.

Some network administrators, particularly those at popular web sites, actively monitor their networks for anomalous behavior by tracing incoming and outgoing packets. For example, packet traces can be used to detect changes in loss rates, reordering, and round trip times between any two hosts [28, 54]. Tulip could then be used to focus in on the source of the problem. This allows for faster detection and repair of problems, especially those that are significant yet too small for any single end user to notice.

Integration with a trace analysis tool would also enable tulip to closely match application behavior. As we point out in Section 5.1, ideally in the future, measurement probes would be automatically carried in application packets, so that the behavior being measured is the same as that being diagnosed. Given that tulip uses out-of-band probes, the basic techniques outlined in Section 3.2 can be adjusted to track the patterns eliciting the anomalous behavior. For instance, more than two probes can be sent for reordering measurements if the application usually generates multi-packet bursts, the probe packet size can be chosen to be the same as the application's, or an ICMP timestamp request can tailgate [21] an application packet to deduce the delay suffered by it. The number of measurements (M) can also be guided by the magnitude of the passively measured fault.

4. EVALUATION

In this section we evaluate tulip's applicability and validate its correctness. While we have begun to use tulip on operational network problems, that work is only just beginning and is not reported here. Instead we argue for tulip's utility through two evaluations. First, in Section 4.1 we measure the diagnosis granularity possible with tulip in the current Internet. Second, in Section 4.2 we verify that tulip's inferences for reordering, loss, and queuing are correct. We are able to find and locate all three properties in the wild.

We show the results from two additional experiments. In Section 4.3 we study the location of loss and queuing delay in aggregate, showing that these properties tend to cluster close to the destinations for the paths we measured. In Section 4.4 we show that network faults are persistent enough to make tulip's binary search a viable strategy. Full characterization of faults and their location in the Internet has been left for future work.

We conducted our experiments from three geographically diverse sources – the University of Washington, the Massachusetts Institute of Technology, and a hosting site in London, United Kingdom. We obtained similar results from all three sources, and present only aggregated results in the paper. The destinations in our experiments were randomly selected from over 400,000 IP addresses in the Skitter list [17]. Skitter is an attempt to map the entire Internet, and the addresses in this list span all parts of the network. While not all of these addresses represent hosts, the vast majority represent entities (routers or hosts) at the edge of the network. Unless otherwise stated, the destinations were chosen independently for each experiment and for each source within the experiment. Of course, our results are valid only for the paths we measured; other source-destination pairs may have different behavior.

4.1 Diagnosis Granularity

In this section we measure tulip's potential for diagnostic precision in the current Internet, by examining how many routers along each path support tulip's forward path measurement probes.

Tulip partitions a path into diagnosable segments by identifying the measurable routers along it. The *diagnosis granularity* of a network path is the weighted average of the lengths of its diagnosable segments, representing the expected precision if a random link in the path were faulty. For example, if a 10-hop path has two diag-

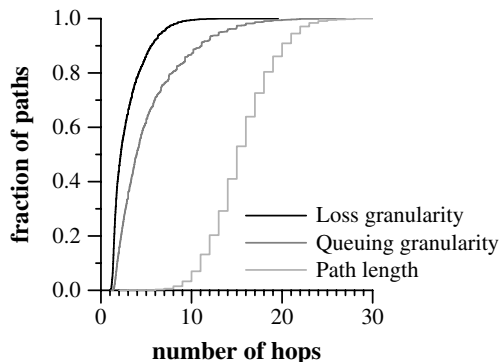


Figure 6: Diagnosis granularity of tulip for the measured paths. The x -axis is the diagnosis granularity or path length. The y -axis is the cumulative fraction of measured paths.

nosable segments of length 4 and 6, the granularity of the path is 5.2. In a perfectly diagnosable path, the length of each diagnosable segment, and the granularity of the path itself, would be one.

Figure 6 shows the diagnosis granularity for loss and queuing, compared to the overall path length, as a cumulative distribution function (CDF) for the paths we measured. This graph was plotted using paths from each source to more than 2000 randomly chosen destinations that responded to our traceroute probes. Since we cannot ascertain the path length for unresponsive destinations, they were not considered in the analysis. Only routers that return consecutive IP-IDs at least 80% of the time were considered measurable for loss. The campus networks of two of our three sources load balance traffic internally, such that the prefix path property holds for some routers when considering only the path beyond the campus but not the part of the path within it. We considered such off-campus routers as measurable in this analysis.

Even with today’s partial deployment of features used by tulip, useful diagnosis granularity is achieved. For loss, 50% of the paths have a granularity of less than three hops, and 75% of them have a granularity of less than four hops. For queuing delay, 35% of the paths have a granularity of less than three hops, and 50% of them have a granularity of less than four hops. The median number of diagnosable segments in a path (not shown) is nine for loss and six for queuing delay. The granularity for reordering (not shown) is slightly better than that for loss since it only requires that a router return monotonically increasing IP-IDs (may not be consecutive).

The diagnosis granularity for queuing delay is coarser than that for loss even though the support for ICMP timestamps (90%) in the current Internet is more than that for sequential IP-IDs (70%). Since ICMP timestamp requests are addressed to the router directly, they cannot be used when the prefix path property does not hold. In contrast, when diagnosing loss, we can use TTL-limited probes for 75% of the routers. These probes usually take a prefix path.

We believe that we can improve the diagnosis granularity of tulip in two ways. First, round trip measurements described in Section 3.3 can be used to rule out the presence of faults in some of the hops in the faulty segment. Second, a segment surrounded by two routers r_1 (near) and r_2 (far), that do not satisfy the prefix path property but support the necessary basic primitive (IP-IDs or timestamps), is diagnosable if the path to r_2 includes the path to r_1 and the part between r_1 and r_2 is part of the end-to-end path [2]. We have not yet quantitatively evaluated these extensions.

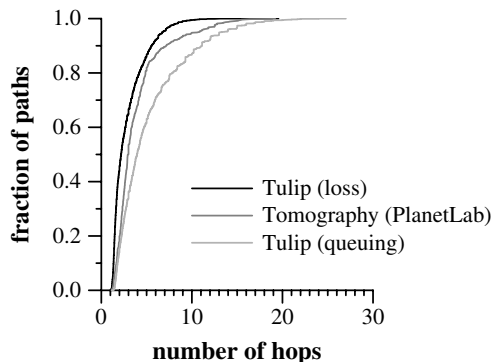


Figure 7: Comparison of diagnosis granularity of tulip with that of an idealized tomography system using PlanetLab. All curves represent the CDFs of the measured paths.

4.1.1 Multiple Vantage Point Approach

We next compare the diagnosis granularity of tulip to that of a hypothetical tomography tool that would infer the fault location by measuring overlapping paths from multiple vantage points. Assuming that faults are rare, problems can be isolated by looking for common elements among those paths that exhibit faults.

We are not aware of any practical tomography tool to diagnose arbitrary paths in the Internet, but research suggests that such an approach might be feasible [13, 31]. Assuming that it is, we computed the diagnosis granularity of a hypothetical tool that uses all of the 65 PlanetLab [38] sites as vantage points to diagnose a given path. We traced from the source to all of the vantage points, and from all of the vantage points to the destination. These traces attached to the path between the source and the destination at various routers that partition the path into diagnosable segments. The diagnosis granularity of the path is then computed as for tulip. This methodology provides an upper bound for the precision of a real tomography tool, since the paths to and from the vantage points also traverse parts of the network not shared by the path being diagnosed. If those other parts of the network also exhibit faults, the ability to localize the fault along the given path could be impaired.

Figure 7 shows the results for the same set of sources and destinations as Figure 6. For the paths we measured, our data shows that the diagnosis granularity of tulip is similar to that of our hypothetical tomography tool, even though the latter uses many hosts to diagnose each path.

An interesting line for future research would be to combine the direct measurement techniques of tulip with tomography data. For example, it may be possible to improve tulip’s granularity by measuring to routers or hosts that are just off the path between the source and destination, chosen so that the paths diverge in the middle of a diagnosable segment (at an otherwise unmeasurable router). Such an approach would need highly accurate topology information [17, 47] to identify candidate measurement targets.

4.2 Validation

In this section we evaluate the accuracy of tulip in measuring reordering, loss, and queuing along Internet paths. Recall from Figure 2 that accurate measurement of forward path to routers is a key building block for tulip; we localize faults by comparing measurements of the near and far ends of each diagnosable segment.

We verified tulip’s correctness in four steps. First, we considered tulip’s accuracy in measuring an end-to-end path where we con-

trolled both end points. In this case, we could precisely measure the amount of reordering, loss, and queuing using packet traces. We then compared these direct measurements with those gathered by tulip using IP-IDs and ICMP timestamps. For this evaluation we used paths from each source to 65 PlanetLab hosts that were spread over different sites, and found that tulip’s results agreed with the packet trace in all cases. This showed that tulip is accurate, at least when measuring the end-to-end paths to those hosts.

We next evaluated tulip’s accuracy while measuring end-to-end reordering and loss on paths to arbitrary hosts. For this, we compared tulip’s results with those of sting [6, 43], a tool to measure one-way reordering and loss. We used sting’s single-connection reordering test for comparison since its methodology is significantly different from that of tulip. Quantitatively comparing the two tools is complicated because they use different patterns of probes; even different types of tests within sting measure different reordering rates [6]. However, we found that sting and tulip generally agreed on the presence or absence of loss and reordering along the one-way paths from our sources to randomly sampled destinations. Due to space constraints, we omit the detailed results of this evaluation.

Third, we evaluated the internal consistency of tulip’s inferences. Without privileged access to routers, we cannot directly verify the accuracy of tulip when working with routers. But if tulip is measuring the property (reordering, loss, or queuing delay) correctly and not getting confused by router idiosyncrasies, the extent of the measured property should not decrease as we move further along the path. The above validations considered only hosts; this evaluation was intended to show that tulip works well with the behaviors of deployed routers.

Finally, we considered tulip’s hop-by-hop measurements on paths to 65 PlanetLab hosts. We found that the hop-by-hop data to intermediate routers along these paths was consistent with the data collected at the end points using packet traces. Due to space constraints, we also omit the detailed results for this portion of the validation; instead, we present detailed results only for the first and third validation steps.

4.2.1 Reordering

In this section we verify the correctness of tulip’s reordering measurements. We measured reordering using 200 probe pairs.

End-to-end correctness We first evaluated tulip using PlanetLab. Concurrently with tulip, we ran tcpdump on the remote end. For every pair of probes, the tcpdump trace provided an authoritative answer as to whether there was reordering. We measured paths from each source to 65 hosts spread over distinct sites. Approximately 20% of the paths had non-zero reordering. For each pair of probes, we compared tulip’s inference with that obtained using tcpdump. We found the two inferences to always be in agreement.

Internal consistency We next evaluated the hop-by-hop consistency of tulip’s reordering measurements. If tulip’s inferences are correct, the reordering rate should not decrease as we move further into the path (at least for the modest amounts of reordering we observe on most paths). We verified this by using tulip in the parallel search mode, which gave us the measured reordering rate at the two ends of each diagnosable segment in the path. We then computed the reordering rate delta for each segment, which is the rate at the far end minus that at the near end. For the tool to be consistent (and correct), these deltas should be non-negative.

Figure 8a shows the CDF for all forward path reordering rate deltas, plotted using data from each source to 2000 destinations. It shows that forward path reordering measurements were consistent, with 85% of the deltas being non-negative. Some negative deltas arose from the statistical nature of the measurement. For example,

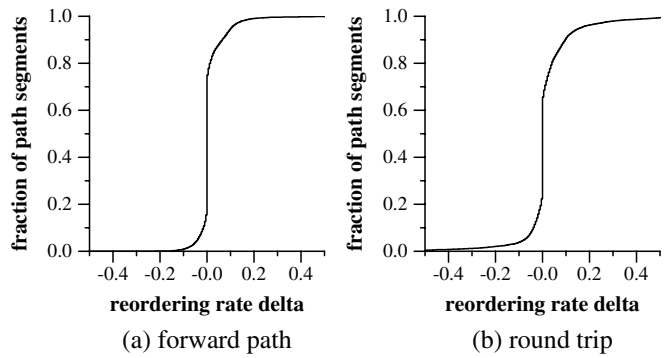


Figure 8: The CDF of reordering rate deltas for forward path and round trip measurements.

even when we measured to the same router using two sets of 200 measurements each, one set had 15 (7.5%) reordered samples and the other had 20 (10%).

We computed the fraction of negative deltas that were not statistically significant. We assumed that the reordering measurements are independent binomial experiments and used the Chi-squared test for proportions [25]. For cases where the reordering rate was too low to use the Chi-squared test (the test has minimum sample size constraints based on the frequency of the underlying process), we used the more conservative exact binomial test. We tested the null hypothesis that the difference can be considered the result of a statistical variation. With a 95% confidence interval 80% of the negative deltas were not statistically significant, and with a 99% confidence interval 90% of them were not statistically significant.

Currently, we do not have an explanation for the remaining statistically significant negative deltas; possibilities include undetected path variations, probes being reordered more than once, and the nature of the reordering process. An example negative delta (15-33, at the fourth segment) rejected by our test was part of a path with 0, 0, 24, 33, 15, 18, and 20 reordered samples to successive routers.

The fraction of paths in which reordering observed at an earlier hop disappeared later in the path was less than 5%, and predominantly consisted of paths with a low overall reordering rate.

For comparison, Figure 8b shows the CDF of round trip reordering rate deltas. The same probes we used for measuring one-way reordering can be used to determine round trip reordering. Note that because of route asymmetry, the return path from adjacent routers may be different. There are visibly more (25%) negative deltas in round trip measurements, presumably caused by uncontrolled variability along the reverse path. This implies that relying on round trip measurements would be less accurate than tulip in locating reordering problems. Roughly half of these negative deltas were statistically significant with a 99% confidence interval.

4.2.2 Loss

In this section we evaluate the accuracy of tulip’s loss measurements. We measured loss using 500 probe triplets.

End-to-end correctness We first evaluated tulip’s loss measurements using PlanetLab. This experiment is similar to that for reordering and used tcpdump. We measured paths from each source to 65 PlanetLab hosts. Approximately 7% of the paths showed non-zero loss. In all instances where tulip reported forward loss, tcpdump showed that loss. PlanetLab hosts have per-flow IP-ID counters (more precisely, one per remote host they converse with), so tulip never confused forward and reverse path loss. They also rate-

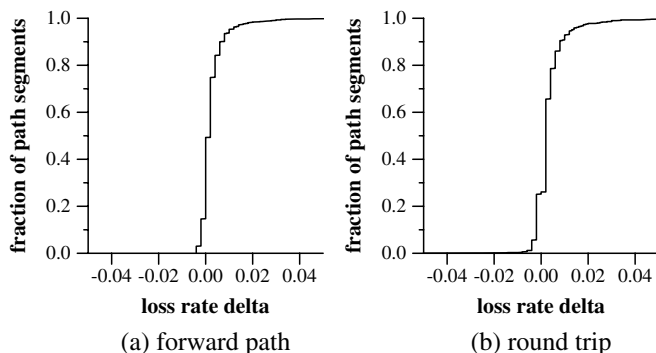


Figure 9: The CDF of loss rate deltas for forward path and round trip measurements.

limit probe responses; we lost more responses due to rate-limiting than due to path losses. Tulip (correctly) did not infer a path loss in any rate-limiting instance. However, tulip missed roughly 38% of the data packet losses seen by tcpdump because a control packet was lost as well. All paths with losses in tcpdump traces had at least one loss that was detected by tulip.

Internal consistency We next evaluated the hop-by-hop consistency of tulip’s loss measurements. If tulip is measuring loss correctly, the measured loss rate should not decrease as we move further into the path. This analysis is similar to the one in the previous section and used loss rate deltas for diagnosable path segments.

Figure 9a shows the CDF for forward path loss rate deltas, plotted using paths from each source to 2000 destinations. Paths that showed no loss were excluded. It shows that the forward loss measurements were consistent, with over 85% of deltas being non-negative. Over 95% of the negative deltas were not statistically significant (computed using the procedure outlined in the last section) with a 95% confidence interval, and all of them were not statistically significant with a 99% confidence interval.

In less than 3% of the paths, forward loss observed along an earlier hop disappeared later along the path. Such paths had a very low loss rate.

Figure 9b shows the CDF of round trip loss rate deltas. Even round trip loss measurements appear internally consistent. Since loss often occurs close to the destination (Section 4.3), round trip loss measurements are confused by path asymmetry to a lesser degree than their reordering counterparts.

4.2.3 Queuing Delay

In this section we evaluate the correctness of tulip’s queuing delay inference. We measured queuing delay using 1000 probes.

End-to-end correctness We first validated the end-to-end correctness of queuing delay measurement using tcpdump on PlanetLab hosts. This experiment evaluated whether there are any inherent limitations in using ICMP timestamps because of their implementation or because ICMP timestamp requests travel through the network more slowly. Concurrent with tulip sending ICMP timestamp requests, we sent UDP probes to the remote machine. We compared one-way queuing delay inferred by tulip with that inferred using the tcpdump trace for the UDP probes. We had to calibrate the timestamps in the tcpdump trace too, even though the machines were synchronized using NTP (with different servers).

We measured to 65 PlanetLab hosts from each source, and found that for all of the paths the two median queuing delays were within 2 ms of each other. The small discrepancy arose from the poorer resolution (1 ms) of the ICMP timestamps.

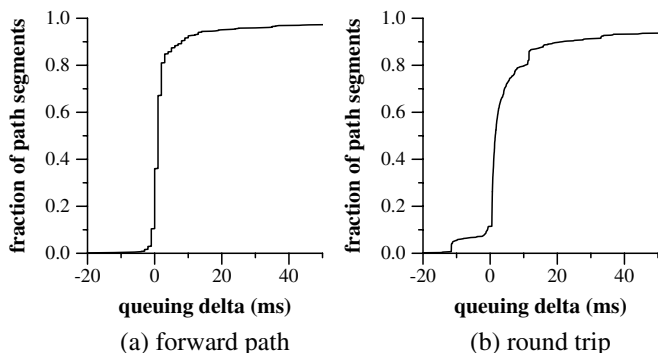


Figure 10: The CDF of median queuing delay deltas for forward path and round trip measurements.

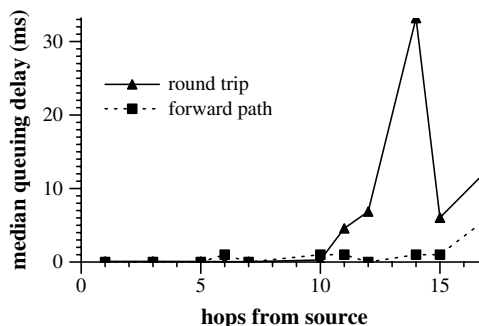


Figure 11: Round trip and forward path median queuing delay for hops along an example path. Using round trip measurements to locate the source of queuing delay would be misleading for this path.

Internal consistency We next validated tulip for its hop-by-hop consistency. If tulip is measuring queuing delay correctly, the delay should not decrease as we move further along the path. We verified this using a methodology similar to the previous two properties and used deltas for median queuing delay.

Figure 10a shows the results for forward path median queuing delays for paths from each source to 2000 destinations. Paths with no queuing delay were excluded. Almost all of the forward median delay deltas were non-negative, pointing to the consistency of tulip.

Figure 10b shows that the round trip median delay deltas were relatively less consistent, reflecting the variability in the return path from the routers. This implies that round trip measurements are less reliable than tulip for locating points of queuing delay in the network. Figure 11 shows an example of a path where round trip measurements would have provided the wrong answer. Round trip measurements indicated that there was significant queuing between hops 10 and 14. The forward path measurements showed that the actual location of the queuing delay was between hops 15 and 17.

4.3 Locating Loss and Delay in the Internet

In this section we examine the average location of loss and delay inferred by tulip along the paths we measured. While the conventional wisdom is that most loss and delay in the Internet occurs close to the edges, it has proven difficult for researchers to verify how frequently this is true [15]. Through its hop-level diagnosis, tulip facilitates answering such questions.

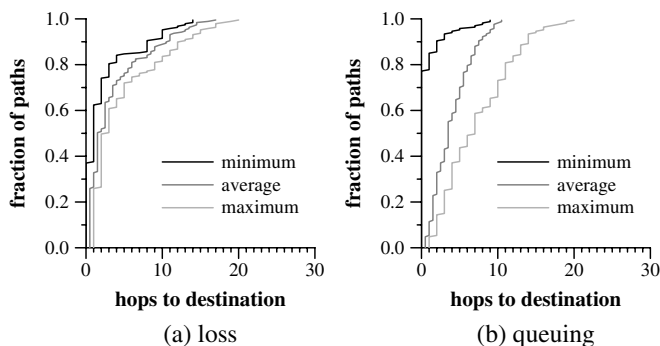


Figure 12: Location of loss and queuing delay along measured paths. The x -axis is the distance from the destination. The y -axis is the cumulative fraction of measured paths.

Figure 12 shows the location of loss and queuing delay along the paths to the same 2000 destinations as in Section 4.2. Since all of our sources are well-connected sites, our results are suggestive rather than representative of the Internet as a whole. The figure shows the distance, to the destination, of the first segment with a non-zero loss rate or more than 5-ms median delay. Since a segment is usually composed of multiple links, we show three curves for the minimum, maximum, and average distance to the destination. For example, if the path is 20 hops long and the first lossy segment is between hops 15 and 17, the minimum, maximum and average distance is 3, 5 and 4 hops.

The figure shows that most loss and queuing delay events occurred close to the destination. In roughly 75% of the cases, for both kinds of properties, the average distance of the problem segment from the destination was less than five hops.

4.4 Persistence of Faults

In this section we evaluate how successful tulip’s binary search will be in diagnosing faults in the Internet. Binary search consumes less network bandwidth, but for it to be effective the fault has to persist for the duration of the search.

In this experiment we first used parallel search to diagnose paths. For paths that contained faults, we counted the number of consecutive runs of parallel search for which the same fault persisted in the same segment. For instance, a fault present in runs 1, 2, 4 and 5 persists for 2 runs. For the purposes of this experiment, a fault is a non-zero loss rate or a median delay of more than 5 ms. Ten parallel search runs were conducted for each faulty path.

Figure 13 shows the results for loss and queuing. Reordering is more persistent as it depends on the network topology itself; loss and delay are primarily functions of load on the network, and thus are more likely to be transient. Each graph was plotted using roughly 100 faulty paths from each source, found from a pool of 1000 destinations. Two curves are shown for loss – paths with any loss and paths with a loss rate more than 0.5%. The difference in the curves does not necessarily imply that low loss rates do not persist. Our definition of persistence is a simple one, based on the presence in consecutive runs; low loss rates found in one run can disappear in the next due to statistical variations. Accurately measuring such low loss rates requires more measurements than we used in this experiment (500).

The figure shows that queuing delays and at least high loss rates tended to persist for the paths we measured. Over 80% of the paths with a high loss rate or queuing delay demonstrated the fault for

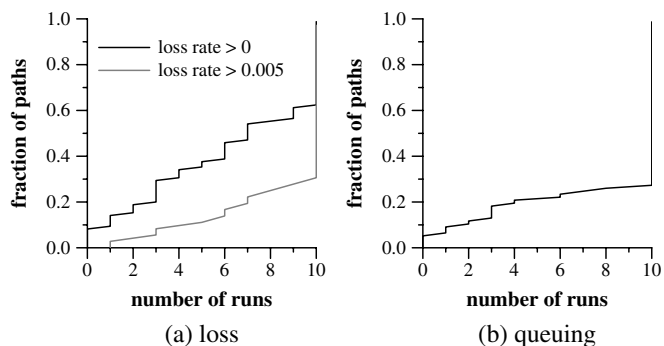


Figure 13: Persistence of faults. The x -axis is the number of parallel search runs for which the fault persisted. Ten parallel search runs were conducted, where each run takes approximately ten minutes. The y -axis is the cumulative fraction of measured paths.

at least six runs, more than the time binary search would take to localize the fault on a typical path.

Given that each parallel search run takes approximately 10 minutes, these fault persistence results are in broad agreement with those of Zhang *et al.* [55].

5. RECOMMENDATIONS

In Section 3.1 we described the Internet router features exploited by tulip as approximations of the more idealized architecture of Section 2. That architecture enables an application to sample key metrics on paths to intermediate routers as part of normal data transfer (in-band). In this section we discuss the limitations imposed by tulip’s use of standard router features, and recommend several, incremental changes to the Internet infrastructure to overcome some of them. The limitations of tulip are:

1. Tulip conducts out-of-band measurements by generating its own traffic to infer path properties. Consequently, its inference might differ from the application experience. We discuss this in Section 5.1.
2. Like all Internet path measurement tools, tulip requires the routing path to be stable. While most Internet routes are stable for long periods [35], route changes during diagnosis should be detected, lest tulip reach an incorrect inference. Currently, we can use TTLs and router addresses in the responses as a coarse verification mechanism. We propose a more robust mechanism in Section 5.2.
3. Tulip’s forward path loss detection relies on the exclusive access to the router’s IP-ID counter for the duration of a probe triplet (not the entire duration of diagnosis). While we found this to be often true today, it may not be in the future, for instance when many users start using tulip. In Section 5.3 we propose a simple mechanism to address this concern.
4. There are limitations in the way ICMP timestamps are currently implemented. We discuss these in Section 5.4.

5.1 In-band vs. Out-of-band Diagnosis

The traffic generated by a measurement tool may not observe the same network behavior as the application it debugs for several reasons. First, different protocols (e.g., ICMP probe packets vs.

TCP) may be treated differently by routers and middleboxes. Second, different connections may traverse different paths due to load-balancing among network paths and servers. Finally, variations in workload (packet size and spacing) may change the observed behavior of the network.

We tested the impact of different protocols by measuring loss rate with three different methods – ICMP echo and TTL-limited UDP and TCP probes. Similar loss rates were inferred by all three methods along the 100 lossy paths we measured. This leads us to believe that most routers at least on these paths do not prioritize these protocols differently.

We show that an application’s experience depends on the workload using two simple experiments with reordering (Section 3.2.2 describes how the loss rate experience differs with the choice of packet size). In the first experiment, we measured the reordering rate on a path that was known to reorder packets using all four combinations of 40- and 1000-byte probes. The reordering rate was 12% with two 40-byte probes, 20% with two 1000-byte probes, 3% with a 40-byte probe followed by a 1000-byte one, and 93% with a 1000-byte probe followed by a 40-byte one.⁶ In the second experiment, we sent three types of packet trains: *i*) six 40-byte probes, *ii*) six 1000-byte probes, and *iii*) a 1000-byte probe followed by five 40-byte probes. We found that the leading packet arrived at the destination later than three or more trailing packets 5%, 7%, and 36% of the time, respectively. The second case represents how frequently TCP’s loss detection strategy based on three duplicate acknowledgements would trigger a false positive if data packets are emitted in a burst.

Potential differences between inference using out-of-band probes and application experience motivated in-band diagnosis in our architecture. These differences are likely to grow as the complexity of the network increases, with differentiated services, traffic shapers, and many kinds of middleboxes becoming more common.

In the long term it seems worthwhile to evolve the Internet towards in-band diagnosis. In the short term, however, our goal is to design our tools to mimic application traffic as closely as possible.

5.2 Path Verification

Tulip would like to detect when the path to the router changes. In our architecture, routers along the path insert their identity in a Bloom filter field to provide an efficient mechanism for end points to verify the path taken by the packet. There is no equivalent facility in the Internet. We propose that one be implemented as an IP option. If routers do not implement IP option processing in the fast path, this path recording packet may travel through the network more slowly. But it would be much cheaper than traceroute in terms of the number of probes. And unlike traceroute, it can measure the entire path atomically, and in the presence of flow-based multi-path routing, probes would take the same network path as application packets. One detail is that routers must insert their interface address rather than their loopback address to be robust against cases where multiple IP-level links exist between two routers. This mechanism is still vulnerable to path variations below the IP-layer, however. It is an open research question how best to measure and diagnose the impact of these changes on application performance.

5.3 IP Identifiers

Forward path loss detection in tulip is less effective in the presence of competing users of the IP-ID counter. The ideal solution

⁶Apparently, when a small packet arrives immediately after a large packet at a router (or link) that reorders packets, the smaller packet is placed in a shorter queue and finishes transmission earlier than the large packet [6].

to this problem is a per-flow counter, but this may be prohibitive for high-speed routers. We propose a light-weight mechanism to approximate a per-flow counter, analogous to the way stochastic fairness queuing [26] approximates fair queuing [11]. Let routers keep a fixed number (N) of counters, where N depends on the router’s ability. The router selects which counter to use for the response IP-ID by hashing the source address in the probe. In the absence of collisions, this is like a per-flow counter for the probing host. But since source addresses are hashed, any collision would be deterministic. To avoid this, both the source address and first 8 bits of the IP-ID in the probe packet can be hashed, allowing the source to choose a different counter if consecutive IP-IDs are not received. Our proposed scheme is completely backward compatible. Routers in the current Internet are a special case with $N = 1$.

Even in the absence of the modification proposed above, it is possible to minimize interference among multiple simultaneous measurements. This involves enhancing tulip with a back-off scheme, similar to that used in media access protocols, to coordinate access to the IP-ID counter. On observing a failure to evoke consecutive IP-IDs, the source waits for a random time interval and tries again. The average waiting time increases exponentially with each failure, and the source stops measuring to the router when the back-off delay goes above a threshold.

5.4 Router Timestamps

In this section we propose three simple changes to the ICMP timestamp implementation in routers. ICMP timestamps are useful in estimating queuing delays, but their poor resolution restricts us to measuring only coarse-grained latency variations. Finer resolution timestamps would not only allow measuring fine-grained queuing behavior but would also simplify inferring other properties such as link capacity [33] and available bandwidth [19].

ICMP timestamp requests have the further problem that the path to the router may not be a prefix of the path to the destination. This can be easily overcome if routers embed a timestamp in each TTL-expired response.

Our final recommendation is simply that ICMP timestamp responses follow the specification [39] and include times for both when the request is received and when the response is transmitted, so that the processing delay can be measured. Currently, routers insert just one timestamp, usually the latter. The presence of both timestamps would enable an accurate inference of one-way delays with fewer measurements as no outlier filtering would be required.

6. RELATED WORK

Our work draws on that of several others. We divide the related work into three categories.

6.1 Diagnosis Approaches

Two basic approaches for performance diagnosis using system support are logging and message-marking. In the former, an activity log is either maintained by the system elements or is passively recorded. These logs are then processed to infer the relevant properties. Examples of this approach are Magpie for performance modeling in clusters [5], Aguilera *et al.*’s trace analysis for performance debugging in distributed systems [1], and SPIE for identifying the source of a packet in the Internet [46]. Summary logs such as those collected using Cisco’s NetFlow [9] have proven useful for network operators to diagnose problems within an administrative domain.

In message-marking approaches, performance parameters are recorded in the messages as they travel through the system. Examples of this approach are ProfileMe for profiling on out-of-order proces-

sors [10] and IP traceback for tracing the packet source [45]. In Section 2.3 we argued that message-marking approaches are more suitable for unprivileged diagnosis in Internet-like environments. Our proposed architecture follows this approach but differs from the systems above in both its goals and its mechanisms.

Tomography is yet another approach to diagnosing performance, but one that requires little system support [8, 13, 31, 50]. It localizes performance faults by correlating information from multiple overlapping paths. To diagnose a given Internet path, it requires support from cooperating hosts deployed across the network. We have shown that router support in the current Internet provides for effective diagnosis, implying that distributed coordinated measurements are not essential to enable users to diagnose their paths. Future research may show synergies from combining tomography with direct measurement techniques such as ours.

6.2 Measurement Primitives

Just as we explore minimal network support required for user-level path diagnosis, Lakshminarayanan *et al.* present overlay primitives to enable a node to efficiently measure delay, loss, and available bandwidth on a path between two arbitrary overlay nodes [22]. They propose two primitives – *path selection* to select the path through the overlay, and *packet replication* to replicate packets inside the overlay. Although the context of our work is considerably different from theirs, our use of TTL-expired responses containing the original packet to localize illegal transformations can be considered an instance of a more general packet replication mechanism.

IPMP is a proposal to measure one-way delays along Internet paths [23]. It is similar to our reduced embedding of Section 2.3.2; routers insert their identity and timestamp as they forward an IPMP packet (out-of-band). If routers were to process such packets in the fast path, IPMP would collect meaningful timestamps (representative of data packets) of all routers in the path using just one packet. We believe that such a mechanism would be even more useful if routers also inserted per-flow counters (Section: 5.3) to facilitate loss inference.

6.3 Measurement Tools

Some diagnosis tools such as ping, traceroute, and pathchar [18] use round trip measurements; these present a design point different from tulip’s. The only support they require from the network is that routers reflect packets back to the source. In theory this allows performance limiting symptoms to be traced to a particular router or link. In practice there are shortcomings. Such tools cannot separate the properties of the forward and reverse path to a router, and the presence of asymmetric routes impairs their diagnostic capability.

Tulip is closely related to tools that measure one-way path properties using only one end point. Sting measures one-way loss and reordering to end points running TCP-based servers [6, 43]. Our reordering measurement methodology is an adaptation of the dual-connection test of sting. To our knowledge cing was the first tool to use ICMP timestamps for measuring queuing delays [2]. We solve two additional practical issues with their use.

7. CONCLUSIONS

In this paper we addressed the problem of letting unprivileged users diagnose communication path problems in a large, heterogeneous, distributed system such as the Internet. We built tulip, a practical tool to diagnose reordering, loss, and significant queuing delay – three properties that impact application performance. Tulip leverages ICMP timestamps and IP identifiers to diagnose paths from the source to arbitrary destinations. We found that even with today’s partial deployment of these features, tulip can localize re-

ordering and loss within three hops and queuing within four hops on most paths we measured. This diagnosis granularity is comparable to that of an ideal, hypothetical network tomography tool that uses 65 diverse hosts to localize faults on a given path.

To explore how the Internet should evolve to provide support for path diagnosis, we presented a practical, in-band diagnosis architecture. This architecture is nearly as powerful as a complete packet trace at all routers along the path but lightweight enough to be implemented using current technology. The features exploited by tulip can be considered approximations of this architecture in the current Internet.

The Internet approximations used by tulip have their limitations. We discussed these limitations and proposed simple changes to the Internet that would facilitate better diagnosis. A key limitation of tulip, common to all active measurement tools, is that out-of-band measurements may not agree with the application’s experience if the network treats measurement packets differently. As complex network elements such as traffic shapers and load balancers are becoming more common, we believe that it would be worthwhile to evolve the Internet towards an in-band diagnosis architecture.

There are several avenues for extending our work. The most important one is gathering operational data with tulip; having made the tool publicly available [51], we hope that users will use it to diagnose their problems. Second, we would like to extend tulip to diagnose other path properties and integrate it with passive measurement techniques. Finally, we intend to explore the synergy between tulip and the multiple vantage point approach: can a practical tool combine these to further improve the efficacy of diagnosis?

8. ACKNOWLEDGEMENTS

We are grateful to David Andersen for access to the RON nodes, Kostas Anagnostakis and Raphael Ryger for fixclock, John Bellardo for sting, and CAIDA for the Skitter data. We also thank the SOSP reviewers for their constructive criticism, and Steve Capell and Tammy VanDeGrift for their feedback on earlier versions of this paper. This work was supported in part by Intel Corporation, Microsoft Research, and DARPA grant F30602-00-2-0565.

9. REFERENCES

- [1] M. Aguilera, J. Mogul, J. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In *SOSP*, Oct. 2003.
- [2] K. Anagnostakis, M. Greenwald, and R. Ryger. cing: Measuring network-internal delays using only existing infrastructure. In *IEEE INFOCOM*, Apr. 2003.
- [3] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, Oct. 2001.
- [4] P. Barford and M. Crovella. Critical path analysis of TCP transactions. *IEEE/ACM Transactions on Networking*, 9(3), June 2001.
- [5] P. Barham, R. Isaacs, R. Mortier, and D. Narayanan. Magpie: Online modelling and performance-aware systems. In *HotOS-IX*, May 2003.
- [6] J. Bellardo and S. Savage. Measuring packet reordering. In *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002.
- [7] A. Broido. Ping Record Route stats. <http://www.caida.org/~broido/png/pingR.html>.
- [8] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic systems. In *Int’l Conference on Dependable Systems and Networks (DSN), IPDS track*, June 2002.

- [9] Cisco IOS NetFlow. <http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml>.
- [10] J. Dean, J. Hicks, C. Waldspurger, W. Weihl, and G. Chrysos. ProfileMe : Hardware support for instruction-level profiling on out-of-order processors. In *IEEE Micro*, Dec. 1997.
- [11] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM*, Aug. 1989.
- [12] N. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking*, 9(3), June 2001.
- [13] N. Duffield, F. Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *IEEE INFOCOM*, Apr. 2001.
- [14] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *ACM SIGCOMM*, Aug. 2002.
- [15] S. Floyd. Questions about the Internet. <http://www.icir.org/floyd/questions.html>, May 2002.
- [16] R. Govindan and V. Paxson. Estimating router ICMP generation delays. In *Passive & Active Measurement (PAM)*, Mar. 2002.
- [17] B. Huffaker, D. Plummer, D. Moore, and k. claffy. Topology discovery by active probing. In *Symp. on Applications and the Internet (SAINT)*, Jan. 2002.
- [18] V. Jacobson. Pathchar. <ftp://ftp.ee.lbl.gov/pathchar>.
- [19] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. In *ACM SIGCOMM*, Aug. 2002.
- [20] C. Kent and J. Mogul. Fragmentation considered harmful. In *ACM SIGCOMM*, Aug. 1987.
- [21] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *USITS*, Mar. 2001.
- [22] K. Lakshminarayanan, I. Stoica, and S. Shenker. Building a flexible and efficient routing infrastructure: Need and challenges. Technical Report CSD-03-1254, University of California, Berkeley, 2003.
- [23] M. Luckie, A. McGregor, and H. Braun. Towards improving packet probing techniques. In *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.
- [24] Z. Mao, J. Rexford, J. Wang, and R. Katz. Towards an accurate AS-level traceroute tool. In *ACM SIGCOMM*, Aug. 2003.
- [25] J. McClave and F. Dietrich. *Statistics*. Macmillan Publishing Company, 6th edition, 1994.
- [26] P. McKenney. Stochastic fairness queuing. In *IEEE INFOCOM*, June 1990.
- [27] S. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *IEEE INFOCOM*, Mar. 1999.
- [28] S. Ostermann. tcptrace. <http://jarok.cs.ohiou.edu/software/tcptrace/index.html>.
- [29] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM*, Aug. 1998.
- [30] J. Padhye and S. Floyd. Identifying the TCP behavior of Web servers. In *ACM SIGCOMM*, Aug. 2001.
- [31] V. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of Internet performance. In *IEEE INFOCOM*, Apr. 2003.
- [32] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot. Analysis of measured single-hop delay from an operational backbone network. In *IEEE INFOCOM*, June 2002.
- [33] A. Pásztor and D. Veitch. Active probing using packet quartets. In *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002.
- [34] A. Pásztor and D. Veitch. PC based precision timing without GPS. In *ACM SIGMETRICS*, June 2002.
- [35] V. Paxson. End-to-end routing behavior in the Internet. In *ACM SIGCOMM*, Sept. 1997.
- [36] V. Paxson. On calibrating measurements of packet transit times. In *ACM SIGMETRICS*, June 1998.
- [37] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT, 1988.
- [38] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the Internet. In *HotNets-I*, Oct. 2002.
- [39] J. Postel. Internet control message protocol. IETF RFC 792, Sept. 1981.
- [40] J. Postel. Internet protocol specification. IETF RFC 791, Sept. 1981.
- [41] RouteScience. <http://www.routescience.com/>.
- [42] R. Ryger. *fixclock*: removing clock artifacts from communication timestamps. Technical Report DCS/TR-1243, Yale University, March 2003.
- [43] S. Savage. Sting: A TCP-based network measurement tool. In *USITS*, Oct. 1999.
- [44] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *ACM SIGCOMM*, Aug. 1999.
- [45] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *ACM SIGCOMM*, Aug. 2000.
- [46] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and T. Strayer. Hash-based IP traceback. In *ACM SIGCOMM*, Aug. 2001.
- [47] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, Aug. 2002.
- [48] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *USITS*, Mar. 2003.
- [49] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. IETF RFC 2001, Jan. 1997.
- [50] Y. Tsang, M. Coates, and R. Nowak. Passive network tomography using EM algorithms. In *IEEE Int'l Conference on Acoustics, Speech, and Signal Processing*, May 2001.
- [51] Tulip. <http://www.cs.washington.edu/research/networking/tulip/>.
- [52] A. Whitaker and D. Wetherall. Forwarding without loops in Icarus. In *IEEE OPENARCH*, June 2002.
- [53] L. Zhang, Z. Liu, and C. Xia. Clock synchronization algorithms for network measurements. In *IEEE INFOCOM*, June 2002.
- [54] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of Internet flow rates. In *ACM SIGCOMM*, 2002.
- [55] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.