Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone

Sharad Jaiswal, Gianluca Iannaccone, *Member, IEEE*, Christophe Diot, Jim Kurose, *Fellow, IEEE*, and Don Towsley, *Fellow, IEEE*

***Abstract*—We present a classification methodology and a measurement study for out-of-sequence packets in TCP connections going over the Sprint IP backbone. Out-of-sequence packets can result from many events including loss, looping, reordering, or duplication in the network. It is important to quantify and understand the causes of such out-of-sequence packets since it is an indicator of the performance of a TCP connection, and the quality of its end–end path. Our study is based on *passively* observed packets from a point *inside* a large backbone network—as opposed to actively sending and measuring end–end probe traffic at the sender or receiver. A new methodology is thus required to infer the causes of a connection's out-of-sequence packets using only measurements taken in the "middle" of the connection's end–end path. We describe techniques that classify observed out-of-sequence behavior based only on the previously- and subsequently-observed packets within a connection and knowledge of how TCP behaves. We analyze numerous several-hour packet-level traces from a set of OC-12 and OC-48 links for tens of millions connections generated in nearly 7600 unique ASes. We show that using our techniques, it is possible to classify almost all out-of-sequence packets in our traces and that we can quantify the uncertainty in our classification. Our measurements show a relatively consistent rate of out-of-sequence packets of approximately 4%. We observe that a majority of out-of-sequence packets are retransmissions, with a smaller percentage resulting from in-network reordering.**

***Index Terms*—Internet measurements, out-of-sequence packets, passive measurements, TCP/IP performance.**

I. Introduction

An IMPORTANT characteristic of any TCP connection is the sequencing of packets within that connection. Generally, if sequence numbers are monotonically increasing, then all is well—data flows through that connection without loss, and the network does not introduce pathological problems such as in-network duplication and reordering. Conversely, out-of-sequence packets indicate that the connection suffers from loss, duplication or reordering. It is thus of interest to study the occurrence of out-of-sequence packets within an Internet TCP connection, and to identify their causes, as the magnitude of out-of-sequence packets is a good indicator of the "health" of that TCP connection and the path that it traverses.

In this paper, we present measurements and a classification of out-of-sequence packets in TCP connections within the Sprint IP backbone. Informally, we will say that a packet is out-of-sequence if it has a sequence number that is smaller than or equal to that of a previously observed packet in that connection.[1] Our first contribution is methodological, we passively measure out-of-sequence packets at a *single* point in the backbone (rather than by actively sending and measuring end-to-end probe traffic at the sender or receiver [3], [11]). Therefore, a new methodology is required to infer the causes of a connection's out-of-sequence packets based only on observations from inside the backbone, i.e., in the "middle" of this connection. An advantage of having such a measurement point within the backbone is that we are able to characterize the behavior of flows between a very large number of source-destination pairs, without having to instrument the individual senders and receivers. However, there are also several methodological challenges associated with having a single measurement point in the "middle" of a connection's end–end path. The measurement point can only observe a limited set of events along the end–end path, and does not know the actions taken at the sender/receiver end; it can only infer this from the previously- or subsequently-observed packets from that connection and the knowledge of TCP behavior. We describe techniques and rules to infer and classify the causes of observed out-of-sequence behavior and empirically validate these techniques in a wide-area testbed. We observe, using our techniques, that we can classify most out-of-sequence packets in our traces and that we can quantify the uncertainty in our classification. Several of our techniques require an estimate of the sender's TCP RTO (retransmission timeout interval) and RTT (the current round-trip delay between sender and receiver). In the absence of knowledge of exact per-sender TCP state at our measurement point, we describe techniques for estimating per-connection RTT from a single measurement point within the

Manuscript received April 30, 2004; approved by IEEE/ACM Transactions on Networking Editor E. Zegura. This work was supported by the National Science Foundation under Grants ANI-9805185, ITR-0080119, and ITR-0085848, and by Sprint. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. This work was carried out when S. Jaiswal was a graduate student at the University of Massachusetts at Amherst and partly when S. Jaiswal, G. Iannaccone, and C. Diot were with Sprint ATL, Burlingame, CA.

S. Jaiswal is with Bell Labs Research India, Alcatel-Lucent, Bangalore 560 095, India (e-mail: jsharad@lucent.com).

G. Iannaccone is with Intel Research Cambridge, Cambridge CB3 0FD, U.K. (e-mail: gianluca.iannaccone@intel.com).

C. Diot is with the Thomson Technology Paris Laboratory, 92648 Boulogne cedex, France (e-mail: christophe.diot@thomson.net).

J. Kurose and D. Towsley are with the Computer Science Department, University of Massachusetts at Amherst, Amherst MA 01003 USA (e-mail: kurose@cs.umass.edu; towsley@cs.umass.edu).

Digital Object Identifier 10.1109/TNET.2006.890117

[1]In this paper, the terms "out-of-sequence" and "reordered" do not have the same meaning. As will be discussed shortly, out-of-sequence packets can be caused by sender retransmissions, in-network duplication, and reordering of packets within the network on their end-to-end path. Previous studies have often used the terms "out-of-sequence" and "reordered" interchangeably.

network. Knowledge of a connection's RTT is also of independent interest [8] to compute the amount of buffering required at a link, or for configuring Active Queue Management.

Our second contribution is the characterization of the out-of-sequence behavior itself. We characterize the out-of-sequence behavior of 29 million TCP connections, generated in more than 7600 unique Autonomous Systems (ASes) (in contrast, previous studies [10] collected end–end traces from 30 unique sites). The links at which measurements were performed vary in capacity (OC-12 and OC-48 links), utilization, and location (e.g., intra-POP, inter-POP and peering links). Our data thus represents a diverse mix of end-to-end paths and traffic characteristics. Our measurements show that consistently, approximately 4% of data packets are of out-of-sequence; we find that out-of-sequence packets result mostly from retransmissions in response to packet loss, and that a smaller percentage are due to in-network reordering; we also observe that other phenomena such as in-network duplication are extremely rare.

The reminder of this paper is structured as follows. We begin in Section II by discussing related work. We describe the rules and rationale that we use to infer and classify the causes of observed out-of-sequence behavior in Section III. In Section IV, we introduce our approach for inferring a connection's RTT from a single measurement point, between the sender and the receiver. In Section V, we identify sources of uncertainty and possible errors in our inferences. Section VI describes the empirical validation of our techniques in a wide-area testbed. We present our measurement results and classification of observed out-of-sequence packets in Section VII. We discuss the representativeness of our results in Section VIII, and finally, Section IX concludes the paper.

II. Related Work

A number of previous efforts have examined packet reordering. Bennett *et al.* [3] sent bursts of ICMP probe packets through a network public exchange point and reported on the magnitude of packet reordering induced in these probes by a specific switch in the exchange point. Paxson [11] also examined the extent and effects of packet reordering and replication in TCP bulk data transfers between pairs of measurement sites. Recently, Bellardo *et al.* [1] described active measurement techniques that can estimate packet reordering in TCP data transfers from a remote host, although they did not carry out extensive measurement studies using their techniques. Our work differs from these earlier works in both the scope of the measurements (millions of connections from thousands of different networks) and our methodology (we passively measure traffic at a single point within the network, rather than actively sending probes and taking both source and destination measurements). We will discuss our results in relation to these studies in more detail in Section VII.

There has also been recent work about inferring loss properties of an end–end path based on passive observations of TCP connections. In [2], the authors consider measurements taken at a core or an ingress router, and compute the ratio of packets lost before the measurement point, to those lost after the measurement point. However, the focus of this work is on the estimation of the packet loss ratio rather than on the characterization of the out-of-sequence phenomenon, and achieves its objectives by discarding all packets for which it is not possible to straightforwardly classify a packet as a retransmission. Moreover, their approach relies on OS stacks to generate unique and monotonically increasing IPID values—a property that is not universal and prone to change for various security reasons. Our work instead develops a more general approach based on assumptions about network behavior, knowledge of TCP protocol and information in the packet headers (such as the IPID field) to classify all packets that traverse the measurement point.

III. Methodology

In this section, we describe the methodology and rules used to classify the out-of-sequence packets observed at the measurement point. We capture and record the first 44 bytes of IP and TCP packet headers of all packets passing across a link. We collect two traces, for each monitored link, corresponding to data flowing in either direction. In Section VII we will describe the specific links within the Sprint backbone at which the measurements were taken. For our purposes here, we need only consider these measurements as a recorded sequence of packet headers.

Given traces of observed sender-to-receiver data packet headers and receiver-to-sender acknowledgment headers, in a particular link, we process the traces to classify the causes of out-of-sequence measurements. The first step in processing the traces is to filter them in order to consider only those connections for which sender-to-receiver data packets, and the receiver-to-sender acknowledgment pass through the link at which the measurements are taken. We report on the fraction of connections for which this requirement holds, and the possible sampling bias the removal of several TCP flows introduces in Section VII.

Once the trace is filtered, we can identify and classify the out-of-sequence packets. A packet is out-of-sequence if its sequence number is less than or equal to that of a previously observed sequence number in that connection. An out-of-sequence packet can be caused by three different events:

- **Retransmission.** In this case, a sender infers that a packet has been lost and retransmits the packet. The sequence number of the retransmitted packet is smaller than previously observed packets of the same TCP connection at the measurement point, and hence will be deemed "out-of-sequence."
- **Network duplication.** In this case, a non-sender-retransmitted copy of a packet is observed. This can occur when the measurement point is within a routing loop (and hence the same packet is observed more than once), or if the network itself creates a duplicate copy of the packet.
- **In-network reordering.** The network can invert the order of two packets in a connection (because of parallelism within a router [3], a route change, or a route loop in the end–end path).

As noted earlier, previous studies have used the terms "reordering," "out-of-order," and "out-of-sequence" interchangeably. We emphasize that a reordering event in our classification is just a subset of all possible events that result in an out-of-se-

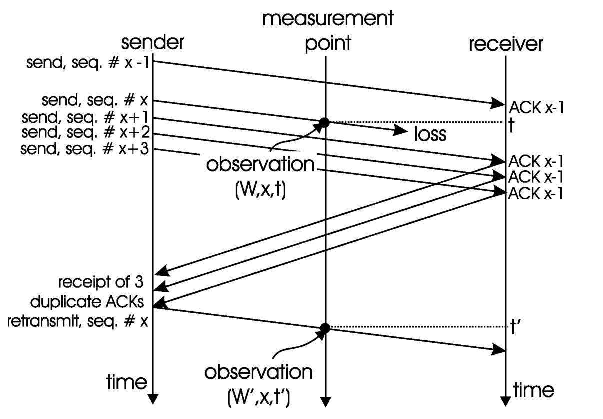


Fig. 1. Retransmission due to a loss after the measurement point.

quence packet. Our classification method will use the knowledge of observed sequence and acknowledgment number values in the TCP header, the identification field in the IP datagram (IPID), and the times at which observations are made to infer the cause of an observed out-of-sequence packet.

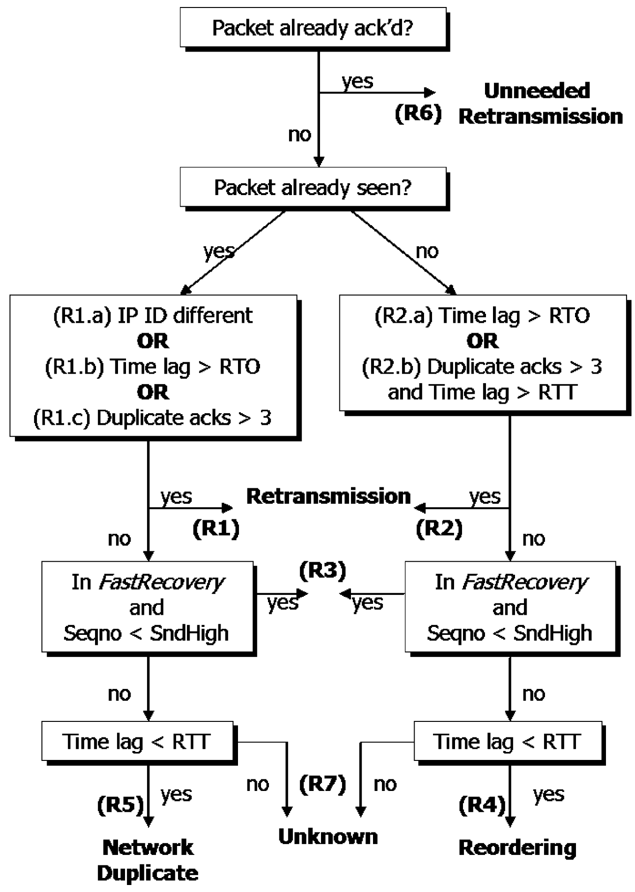
A TCP sender incorporates a sequence number to identify every unique data packet sent into the network. Any packet going through the IP stack in the sender's kernel is then assigned a value in the IPID field in the packet's IP header. In many current TCP/IP implementations this is an unique and monotonically increasing value assigned to every outgoing IP packet. Hence, if a packet is retransmitted by TCP it may have the same sequence number, but a different IPID value. When the measurement point observes a sender-to-receiver data packet with an IPID value W and a TCP sequence number of x at time t, it is referred using the 3-tuple notation (W, x, t). Fig. 1 illustrates this notation. In this example, the sender sends packets $x-1$ through $x+3$. A packet with sequence number x and IPID W is observed at the measurement point at time t (and denoted by (W, x, t)). Because of the subsequent loss of x between the measurement point and the receiver, the sender will eventually retransmit x. The retransmitted copy observed by the measurement point at time t is then denoted by (W', x, t').

We reiterate, to ensure our notation is clear, when we use the 3-tuple notation (W, x, t) to refer to or identify a TCP packet, we are referring to the *observation* of a packet with IPID W and sequence number x, at a time t at the measurement point.

Fig. 2 summarizes the decision process for classifying out-of-sequence packets. The edges leading to leaf nodes (the classifications) in the decision tree in Fig. 2 are annotated with the decision rules R1 through R7, described below) corresponding to those classifications. Let us now consider these classification rules.

A. Retransmissions

Fig. 1 shows the case in which a packet (W, x, t) is observed at the measurement point, and no acknowledgment covering x (i.e., an acknowledgment for a packet with a sequence number greater than or equal to x) is observed before another packet (W', x, t') with the same sequence number is observed. Rule R1 below specifies the cases in which this second packet is classified as a retransmission.



Packet already ack'd?
yes
(R6)
Unneeded Retransmission
no
Packet already seen?
yes
no
(R1.a) IP ID different OR (R1.b) Time lag > RTO OR (R1.c) Duplicate acks > 3
(R2.a) Time lag > RTO OR (R2.b) Duplicate acks > 3 and Time lag > RTT
yes
Retransmission
yes
(R1)
(R2)
no
no
In FastRecovery and Seqno < SndHigh
(R3)
yes
yes
In FastRecovery and Seqno < SndHigh
no
no
Time lag < RTT
Time lag < RTT
(R5)
yes
no
(R7)
no
(R4)
yes
Network Duplicate
Unknown
Reordering

Fig. 2. Decision process for the classification of out-of-sequence packets.

Rule R1: Assume (W, x, t) is observed at the measurement point and no acknowledgment covering x is observed before another packet (W', x, t') with the same sequence number, x. In this case, either (W, x, t) is lost between the measurement point and the receiver, or its ACK is lost between the receiver and the measurement point. Then (W', x, t') is classified as a retransmission if any of the following conditions are true:

- **R1.a:** $W' \neq W$
- **R1.b:** the $timelag(t' - t) > RTO$, i.e., the time between the observation of (W, x, t) and (W', x', t') is greater than RTO, the estimated sender timeout interval. Note that since RTO is a function of the RTT, we will need to estimate the RTT value. We will need RTT for other purposes as well. We address the problem of estimating the sender's RTT in Section IV.
- **R1.c:** the number of duplicate ACKs observed in the period (t, t') exceeds the sender's duplicate ACK threshold (in this paper, we take to be the typical value of 3).

A closely related scenario to those covered in R1 is the case in which an out-of-sequence packet (W', x, t') is observed, but as shown in Fig. 3, the measurement point has not previously observed a packet with sequence number x. In this case, the original packet x was sent by the sender but lost before the measurement point.

Rule R2: Assume (W', x, t') is observed, but no packet with sequence number x has been observed before at the measurement point. Also assume that no acknowledgment covering x is

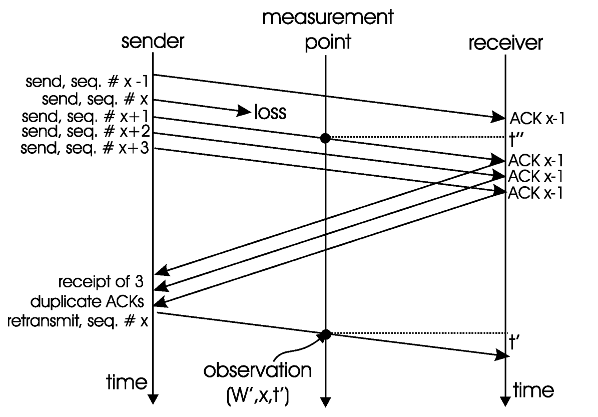


Fig. 3. Retransmission due to a loss before the measurement point.

observed before (W', x, t'). In this case, (W', x, t') is classified as a retransmission if any of the following conditions are true:

- **R2.a:** the $timelag(t' - t'') > RTO$, where t'' is the earliest time at which a packet with a sequence number greater than x is observed. We use t'' here since, as noted above, the measurement point has not previously observed a packet with sequence number x.
- **R2.b:** the number of duplicate ACKs observed after t'' but before t' exceeds the sender's duplicate ACK threshold (see rule R1.c). In addition, we require that $t' - t'' > RTT$ since a sender would not react to any event in the network in less than one RTT.

Rules R1 and R2 rely on the notion that packet retransmissions either have distinct IPIDs, or $timelag > RTO$ or have elicited at least three duplicate ACKs from the receiver. However, not all retransmissions from the sender meet this criteria. After a TCP sender receives three duplicate ACKs, it retransmits the packet for which the duplicate ACKs were intended, and enters the *fast recovery* phase. At this point, the flavor of the sender's congestion control algorithm comes into play. If more than one packet was lost in the window before the sender entered the recovery phase, the sender need not observe three duplicate ACKs for each of the lost packets in order to retransmit these packets. For example, in TCP NewReno, after entering the fast recovery phase, the sender stores the sequence number of the most recently observed data packet, in the variable *sndHigh*. There after, if the TCP sender receives acknowledgments for packets with a sequence number *less* than *sndHigh*,[2] the sender promptly retransmits the packet. The TCP sender eventually exits fast recovery, when it receives an acknowledgment that covers the sequence number stored in *sndHigh*. In order to account for fast recovery retransmissions, that are not triggered by > three duplicate ACKs, we introduce two variables into our classification scheme. The variable *infastRecovery* is a flag associated with a monitored TCP connection, which is set when the TCP sender enters the fast recovery phase; the variable *sndHigh* stores the value of the last sequence number observed before the TCP sender entered fast recovery.

Whenever an out-of-sequence packet is classified as a retransmission by either rules R1.c or R2.b (the presence of three duplicate ACKs), and if the flag *infastRecovery* is not currently set for this connection, then *infastRecovery* is set to 1, also the value of the latest observed sequence number is stored in *sndHigh*.

[2]These ACKs are usually called *partial* ACKs.

The above considerations are embedded in Rule R3 to allow us to identify TCP retransmissions during the fast recovery phase.

Rule R3:

- **R3:** If rules R1 and R2 are not true, but the flag *infastRecovery* is set, and the sequence number $x < sndHigh$ then this out-of-sequence packet is a retransmission, as a result of TCP fast recovery.

B. Reordering

Rule R2.a, requires that $t' - t'' > RTO$ to indicate that sufficient time had passed for the out-of-sequence packet to possibly be a retransmission of an earlier packet. But what if this time lag is too small for a retransmission to have taken place, i.e., that we observe an out-of-sequence packet, but the interval of time between when it would have been observed (if it had been received in order) and when it is observed (out-of-order) is too short a time for the sender to have retransmitted the packet? In this case, the packet cannot be a retransmission, and the packet must have been mis-ordered within the network between the source and the measurement point. But how short a time is "too short a time?" Rather than require that the interval be less than RTO (which, as discussed in Section V is subject to a degree of uncertainty), we take a more conservative approach and require that this interval be less than RTT. Thus, we have:

Rule R4: Assume an out-of-sequence packet (W', x, t') is observed, and again, the measurement point has not previously observed a packet with sequence number x. Furthermore assume that none of the conditions R2a, R2b, and R3 hold. In this case, we know (W', x, t') is not a retransmission. Again, let t'' be the earliest time at which a packet with a sequence number greater than x is observed. If $t' - t'' < RTT$, then the observed packet (W', x, t') is classified as a re-ordered packet—a packet that was transmitted in order by the sender but reordered within the network before it reached the measurement point.

C. Duplicates

Suppose now that we observe a packet, (W, x, t') that has the same IPID and sequence number as an earlier-observed packet (W, x, t). Assume further that condition R1.a does not hold, i.e., that we have not seen enough duplicate ACKs to trigger a retransmission, and that the time interval $t' - t$ is smaller than the RTT. Given these conditions, the observed packet (W, x, t') can not be a sender retransmission, and yet is identical (in IPID and sequence number) to a recently observed packet. We classify such a packet as a network-generated duplicate.

Rule R5: Assume (W, x, t) and (W, x, t'), $t < t'$ are observed at the measurement point. Assume also that the number of duplicate ACKs observed after t but before t' does not exceed the senders duplicate ACK threshold. Finally, assume that $t' - t < RTT$. In this case, we classify the packet as a network-generated duplicate.

D. Unneeded Retransmissions

Fig. 4 illustrates the case in which a packet with sequence number x and its acknowledgment (or the acknowledgment of a

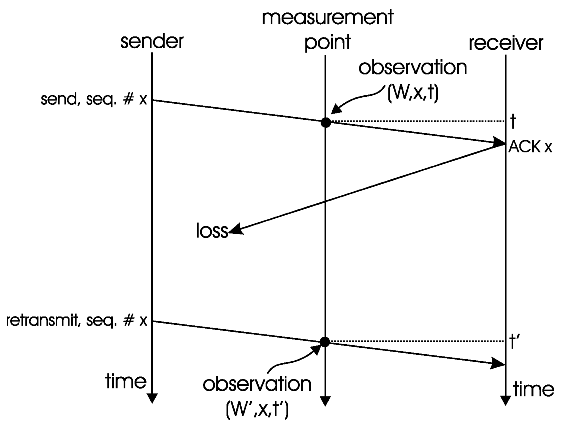


Fig. 4. An unneeded retransmission due to a lost ACK.

packet with a sequence higher than x, i.e., an acknowledgment "covering" this packet) are observed at the measurement point. Although the receiver has clearly received packet x, the sender may still retransmit the packet if either the ACK is lost between the measurement point and the sender, or if the sender timeouts prematurely. In either case, when a second packet is observed with sequence number number x, it is a retransmission—a retransmission that is not needed by the receiver. We thus have:

Rule R6: If a packet (W, x, t) and an acknowledgment covering this packet have been observed, and a second packet (W', x, t'), with IPID different from any other of this connection, is observed, then the second packet is classified as an unneeded retransmission.

In all other cases not satisfying rules R1 through R5, we classify the cause of the out-of-sequence packet as being unknown.

Rule R7: A packet not classified under R1 — R6 is classified as "unknown". We will shortly show that only a very small fraction of the observed out-of-sequence packets are not classifiable under rules R1 through R5.

E. Completeness of Observations

From a single observation point along the end–end path it is not possible to observe all the events as a TCP sender or receiver would. As we described before, using additional information about the behavior of the TCP protocol, one can infer some of those events. However, there is still a set of out-of-sequence events that can go completely undetected given that they are not manifested at the measurement point. In the following we briefly discuss such events:

- If the first packet of the connection (the SYN packet) is lost before our measurement point, this event will go undetected. Likewise, the measurement point cannot detect the loss of an *entire* window of packets that occurs between the sender and the measurement point. In this case, since the measurement point does not observe any packets between the lost packets, and their subsequent retransmissions, it is not possible to identify the retransmissions as out-of-sequence packets.
- According to the rules described before, if a packet is reordered between the measurement point and the receiver, it goes undetected. However, in Section VIII, we attempt to also quantitatively account for reorderings downstream from the measurement point.

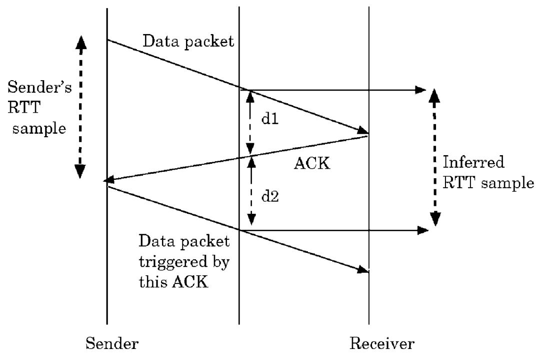


Fig. 5. TCP running sample based RTT estimation.

IV. Estimating RTT

As discussed earlier, the need to estimate the round-trip delay (RTT) of a TCP connection is a crucial ingredient in our classification process—RTT estimation plays a role in computing the *RTO* of a connection (used in rules R1 and R2) and in identifying in-network reordering and duplication (rules R4 and R5, respectively).

In this section, we describe a technique to compute RTT samples throughout the lifetime of a TCP connection. The implementation of this method is simple and in some cases results in as many RTT samples as would be computed by the actual TCP sender. We only provide a brief overview of our technique here; the reader is referred to [7] for a more detailed description.

The basic idea behind our RTT estimation technique is illustrated in Fig. 5. Since we are not able to directly measure the sender's RTT sample shown in the left of Fig. 5, we instead infer the RTT as the sum of 1) the round-trip delay from the measurement point to the receiver and then back to the measurement point (labeled $d1$ in the figure), and 2) the round-trip delay between the measurement point, the sender and then back to the measurement point (labeled $d2$ in the figure). The sum of these two delays $d1 + d2$, as shown in Fig. 5, is our estimate of the RTT. We refer to our method as a *running RTT* estimation technique, since it continuously makes RTT estimates, based on the measured values of $d1$ and $d2$ throughout the TCP connection's lifetime. This approach, can under some circumstances, allow the computation of one RTT sample, for every "round" of packets sent by the sender, which corresponds to the number of samples computed by the TCP sender itself.[3] In Section V we will consider issues which impact the estimation of RTT using our inference technique. We will also compare the estimates taken using our *running* estimation technique with RTT samples computed by the TCP sender itself. We conclude here by mentioning two important aspects of the running RTT estimation technique.

Tracking cwnd: An important requirement of the *running RTT* estimation technique is the ability to determine which data packet transmissions are triggered by the arrival of a particular ACK. This requires an accurate estimate of *cwnd* and we refer

[3]Except, if the sender uses the TCP timestamp option, in which case it will compute a RTT sample for every new data packet transmitted.

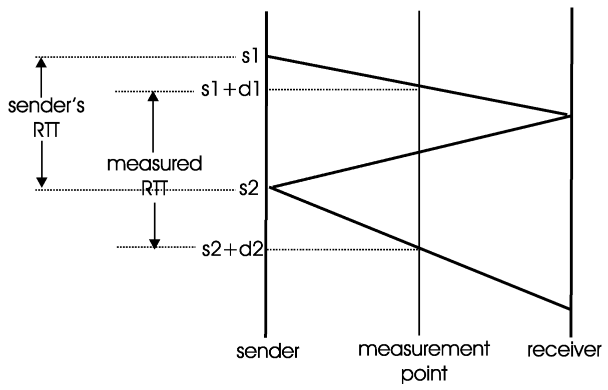


Fig. 6. Differences in RTT estimation.

the reader to [7], which describes in detail how we implement a scheme to keep track of a TCP sender's current congestion window.

Stop estimation during losses: Our technique must also be able to stop (and restart) the RTT estimation as a sender recovers from a loss in order to closely emulate the behavior of the actual TCP sender (which does not compute the RTT during loss recovery). In order to do this, our technique relies on the knowledge of the state of the TCP connection. More details about how this is carried out are explained in [7].

V. Sources of Estimation Inaccuracy

As discussed earlier, given the location of the measurement point in the middle of a connection's end–end path, a set of potential errors is introduced as a result of the difference in information the measurement point observes, and what is actually observed by the sender and the receiver. Moreover, inaccuracy in our estimation techniques are also introduced due to end-host effects which are difficult to account for while making observations only in the middle of the end–end path. Also, recall from our discussion in Section III that rules R1, R2, and R4 either directly or indirectly make use of the current value of the sender's current RTT. Hence, erroneous RTT estimates can affect the accuracy of our classification. Below we discuss instances which introduce errors in our RTT estimates, and hence potentially in our heuristics for classifying out of sequence packets.

Variable delays along the end–end path: Fig. 6 illustrates that the RTT observed at the measurement point and the RTT observed at the sender can differ. The sender transmits a first packet at time s_1, and the packet is observed at the measurement point at time $s_1 + d_1$. Now, suppose the sender sends a second packet (as the result of having received an acknowledgment for the first packet) at time s_2. This second packet arrives at our measurement point at time $s_2 + d_2$. In this case, the measured RTT at the sender is $s_2 - s_1$, while at our measurement point, we compute an RTT of $(s_2 - s_1) + (d_2 - d_1)$. Depending on whether d_2 is greater or smaller than d_1, the observed RTT will either overestimate or underestimate the sender-observed RTT.

End-host delays: A related problem is the fact that we cannot estimate the delays within the end hosts themselves between the receipt of an acknowledgment and the transmission of the data packet triggered as the result of the ACK receipt. However, delays in the receiver's operating system are an inherent component of the TCP sender-measured RTT as well.

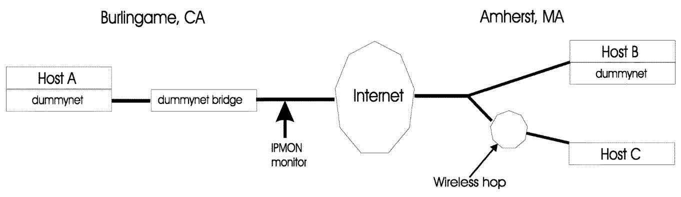


Fig. 7. Validation experiments setup.

Estimate of cwnd: Our RTT estimation technique depends on an accurate estimate of the TCP sender's current value of the congestion window. Under certain packet loss patterns, our estimate of the sender's *cwnd* can be erroneous. The reader is referred to [7] for a detailed discussion of all scenarios which could give an erroneous cwnd estimate.

VI. Experimental Validation

We now validate our heuristics to identify and classify out-of-sequence packets through measurements done in a controlled testbed. Our setup (as illustrated in Fig. 7) consists of PCs running the FreeBSD 4.3 and 4.7 operating systems, with a modified kernel that exports the connection variables to user space using the sysctl facility. The PCs were located at the University of Massachusetts, in Amherst, MA and Sprint ATL, in Burlingame, CA, labeled "Host A" and "Host B" in Fig. 7. Traffic between these two sites passed through an OC-3 access link in Stockton, CA, which was also passively monitored by an IPMON [5] system. The IPMON monitor was our passive measurement site between the sender and the receiver. Our experiments consisted of setting up TCP connections (divided between Reno and NewReno flavors), carrying out bulk data transfers.

In order to study the accuracy of our techniques under *varying* packet loss and reordering rates we used the dummynet emulator [12] at different points in the end–end path to induce loss and reordering events. To generate more packet drops we configured a dummynet "pipe" as a congested bottleneck link. In order to reorder packets we configured two dummynet pipes with variable delays; packets from a flow were probabilistically assigned to traverse either of these two pipes. This emulates multi-path routing which would result in packets getting reordered. We varied the dummynet configuration to create three sets of experiments: one with no packet drops, one with approximately a 3% drop rate, and a third with a 5% drop rate. All of these experiments had between 1%–1.5% of packets sent through the alternate dummynet pipe to induce reordering. We also placed the emulated bottleneck link to be either between the sender and the measurement point, or placed after the measurement point and before the receiver.

We also carried out experiments with actual packet drops along the end–end path between Amherst and Burlingame. We ran a set of experiments at different times of the day, over several days, with bulk transfers lasting between 10 seconds and 5 minutes. However, we observed very low loss rates (rarely exceeding 0.1%) experienced by the connections. We then did

TABLE I
OUT-OF-SEQUENCE PACKETS INFERENCE AND CLASSIFICATION WITH *dummynet* INDUCED LOSSES

Loss rate	~0%				2 - 3%				~5%			
Data packets	15,143				17,411				17,080			
Classification	Inferred		Actual		Inferred		Actual		Inferred		Actual	
Out-of-sequence	204	(1.35%)	204	(1.35%)	618	(3.55%)	628	(3.61%)	1,111	(6.50%)	1,169	(6.84%)
Retransmissions	93	(45.59%)	91	(44.61%)	487	(78.80%)	494	(78.66%)	1013	(91.18%)	1061	(90.76%)
Reorderings	111	(54.41%)	113	(55.39%)	131	(21.20%)	134	(21.34%)	97	(8.73%)	108	(9.24%)
Unknowns		(0.00%)		(0.00%)		(0.00%)		(0.00%)		(0.00%)		(0.00%)

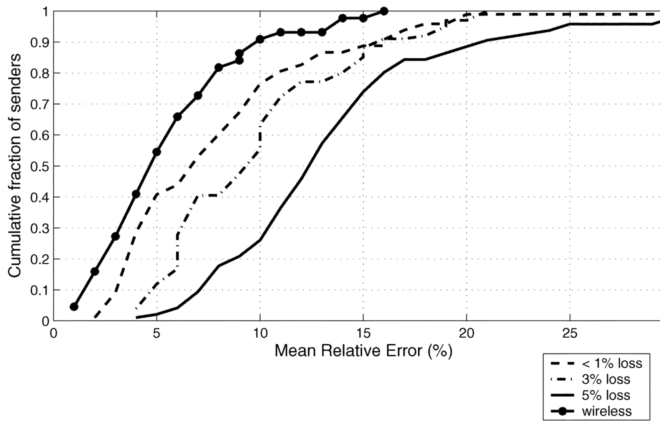


Fig. 8. Cumulative error of RTT inference scheme.

another set of experiments with an end-host behind a wireless hop. This resulted in packet drops of around 4%, and we also include discussions about these experiments in this section.

In order to carry out the validation we collect packet traces using *tcpdump* at the end-host where the TCP sender resides. We process the sender-end *tcpdump* traces using *tcptrace*, a tool which can recreate TCP flow level information from *tcpdump* traces collected at end-hosts.

Before we examine the results of the classification rules, let us first look at the performance of our RTT estimation technique, since several of our rules depend on an accurate estimate of RTT. For each sender we compare the RTT samples estimated using our technique at the IPMON measurement point, with the values computed by the sender. Let p_i be the latest packet observed at the measurement point, and let s_{p_i} and m_{p_i} be the most recent sample (before the arrival of p_i) taken at the TCP sender and computed using the *running* estimation technique at the measurement point respectively.

Given this series of estimated and sender values, we compute the average of the relative errors for each sender as

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\mid s_{p_i} - m_{p_i} \mid}{s_{p_i}}\right)$$

where n is the total number of packets observed at the measurement point.

Fig. 8 plots the cumulative distribution of the mean relative errors for RTT. The relative error in RTT, for drop rates less than 3%, is less than 15% for 90% of the senders, and for a loss rate of around 5%, the error is less than 20% for 90% of the senders. The magnitude of this error is sufficiently small for us to accurately distinguish between retransmissions and reorderings (based on our assumptions about network behavior), since the value of a TCP retransmission timer is usually several times larger than the connection's RTT, and reordering is assumed to occur at timescales smaller than the RTT of the connection. Finally, we observe that our estimates of the RTT in the presence of packet drops in the wireless hop are also very accurate, with the error never exceeding 20%. Even though the packet drop rate in this set of experiments is fairly high—around 4%, there was very little variability in the RTTs along the lifetime of the connections (since there was little congestion along the end–end path), which contributes to the accuracy of our estimates.

Given the fact the we have a fairly accurate estimate of the TCP connection's RTT, we now turn our attention to the accuracy of the out-of-sequence classification methodology. We start with the classification results from the validation experiments using only *dummynet* to induce loss and reordering, as presented in Table I.[4] We observe that we are able to identify nearly all out-of-sequence packets when the packet drop rate is less than 1%. However, at a higher drop rate of around 3%, the measurement point does not observe 10 out of 628 out-of-sequence packets (1.6% of all out-of-sequence packets), and at a drop rate of 5% it does not observe 58 out of the 1169 packets that were missequenced along the end–end path (5% of all out-of-sequence packets). We have examined the traces and confirmed, as discussed in Section V, that this happens when an entire window of packets is lost before the measurement point, or if packet that has been reordered is lost before it is observed at the measurement point.

We now look at the frequency with which we misclassify out-of-sequence packets. We observe that 2 out of 204, 3 out of 628, and 11 out of 1111 out-of-sequence packets are misclassified across the three different loss rates. This occurs if the timelag of a reordered packet is on the order of the connection RTT and the packet crosses the measurement point after three duplicate ACKs (for the reordered packet) were observed at the measurement point. In this case, our methodology considers the packet to be a retransmission.

We now examine the accuracy of the classification scheme in the presence of packet loss along the end–end path, with no *dummynet* in between the end hosts. Almost all packet drops in this experiment were at the wireless last hop. In our first set

[4]Even in the case when there were no packet drops, there exist packet retransmissions. This is due to the reordering induced on the end–end path. If the extent of the reordering is such that it triggers more than three duplicate ACKs from the receiver, it would invoke TCP's fast retransmit mechanism in the sender causing a retransmission.

TABLE II
OUT-OF-SEQUENCE PACKET INFERENCE AND CLASSIFICATION WITH LOSSES AT WIRELESS HOP, WITH *dummynet* INDUCED REORDERING

Loss rate	~4%			
Data packets	11,205			
Classification	Inferred		Actual	
Out-of-sequence	147	(1.31%)	155	(1.38%)
Retransmissions	147	(100%)	155	(100%)

TABLE III
OUT-OF-SEQUENCE PACKET INFERENCE AND CLASSIFICATION WITH LOSSES AT WIRELESS HOP, WITH NO REORDERING

Loss rate	4-5%			
Data packets	17,454			
Classification	Inferred		Actual	
Out-of-sequence	1119	(6.41%)	1135	(6.50%)
Retransmissions	835	(74.62%)	847	(74.63%)
Reorderings	284	(25.38%)	288	(25.37%)
Unknowns		(0.00%)		(0.00%)

of experiments, we observed that out of 155 out-of-sequence packets, 147 were detected at the measurement point, and all were correctly classified to be retransmissions. Once again, we confirmed that the missing out-of-sequence packets were due to the loss of an entire window of packets before the measurement point. Since we observed no reordering along the end–end path in this experiment, we conducted additional experiments in which we induced reordering in about 1% of all packets using *dummynet*. Combining the results from these experiments (as presented in Tables II and III), we observe that we are also able to classify the out-of-sequence packets in this experiment with a high degree of accuracy—only 3 out of 1119 observed out-of-sequence packets were misclassified, and we confirmed this was again because the time-scale of the reordering event was longer than a RTT, and triggered more than three duplicate ACKs from the receiver, as discussed earlier in the section.

To summarize, our classification scheme classifies out-of-sequence packets accurately, with only 1% of all observed out-of-sequence packets misclassfied, across different loss rates. In the next section, we apply our techniques to packet traces collected in the Sprint backbone.

VII. MEASUREMENT AND CLASSIFICATION RESULTS

Our measurements were obtained using infrastructure developed as part of the Sprint IP Monitoring (IPMON) project [5]. The IPMON measurement system provides packet-level traces from OC-3, OC-12, and OC-48 links in several Points-of-Presence (POPs) in the Sprint backbone. The measurement systems themselves are connected via an optical splitter to the links under study so that all packets traversing a link are passed on to the monitoring equipment. A packet capture card copies all the packet headers to disk along with a timestamp generated by an accurate GPS-synchronized clock.

A. Measurement Data

In the following, we present the results of our classification over a set of four packet traces collected on November 21, 2002. We have observed similar trends to those reported below in many other traces collected on different dates and over different links (see [6] for results on a different set of packet traces). We identify the traces under study as "CDN", "Tier-1 ISP", "East Coast", and "Intra-POP" depending on the nature of the customer that contributes most to the data traffic in that particular trace, or the location in the backbone network where the trace was collected. "CDN" refers to a content distribution network hosting web servers that (given the design of the CDN) should receive requests primarily from clients that are relatively "close" to the servers.[5] "Tier-1 ISP" refers to a link connecting to another service provider that carries traffic coming from (and destined to) a very large and diverse set of end hosts. The last two traces trace refer to OC-48 (2.5 Gb/s) links on the East Coast of the U.S. (named *East Coast*) and inside the New York PoP (*Intra-POP*).

TABLE IV
SUMMARY OF THE TRACES, OC-12 = 622 Mb/s, OC-48 = 2.5 Gb/s

	CDN	Tier-1 ISP	Intra-POP	East Coast
Link speed	OC-12	OC-12	OC-48	OC-48
Duration (hrs)	6	6	1	2
Unique src ASes	4,752	520	4,837	1,561
TCP connx	18M	4M	6.3M	1.7M
% of TCP connx	99.67%	12.20%	25.53%	9.20%
TCP Data pkts	335M	53M	74M	40M

Table IV summarizes the characteristics of the four traces. We believe these traces provide a highly representative sample of the Internet traffic, with the connections originating in a significant percentage of the total number of ASes in the Internet. We retrieved the BGP table from Sprint backbone routers during the trace collection, and used the AS path information to derive the source and destination ASes of the TCP connections. Overall, the traces contain TCP connections originating in 7664 unique Autonomous Systems.[6] This represents about the 60% of the total number of allocated ASes at the time of the trace collection (we counted 12 822 unique Internet AS numbers as of November 21, 2002).

Another aspect of our traces is the relative position of the measurement point in the end–end path between the source and the destination. For the CDN trace, we found that the CDN's servers were located just a couple of hops from the backbone (and the measurement point). However, the Tier-1 and the OC48 and East-Coast links carry traffic destined to a very diverse set of end hosts; our monitoring point is, on average, closer to the midpoint of the paths of those TCP connections. For more detailed analysis about the location of the measurement point in the end–end path for our traces, we refer the reader to [6].

B. Out of Sequence Packets

Table V shows the classification results for the observed out-of-sequence packets in the four traces. The first two rows in Table V indicate the total number of data packets observed, and the absolute number and percentage of these packets that are out-of-sequence. Generally, the number of out-of-sequence

[5]The "vicinity" of a client may depend on the number of router hops, on the round-trip time or on the length of the AS path from the source to the destination.

[6]The sum of the unique AS numbers from Table IV is higher (11 670) because we may have the same AS present in more than one trace.

TABLE V
OUT-OF-SEQUENCE PACKET CLASSIFICATION

	CDN		Tier-1 ISP		Intra-POP		East Coast	
Data packets	335,511,737		53,652,349		74,881,505		40,662,075	
Out-of-sequence	8,564,463	(2.55%)	2,635,620	(3.73%)	2,704,839	(3.61%)	1,627,040	(4.00%)
Retransmissions	6,802,611	(79.43%)	1,542,495	(58.52%)	1,709,690	(65.27%)	1,041,307	(64.00%)
Unneeded Retransmissions	1,146,633	(13.39%)	307,468	(11.67%)	395,515	(15.10%)	254,397	(15.64%)
Network Duplicates	6,108	(0.07%)	1,772	(0.07%)	2,851	(0.09%)	2,122	(0.13%)
Reorderings	604,311	(7.04%)	682,557	(25.89%)	108,505	(16.06%)	269,677	(16.57%)
Unknown	7,902	(0.09%)	101,238	(3.84%)	92,856	(3.54%)	60,442	(3.71%)

packets is limited to about 4% of the total data packets exchanged by the TCP connections.

Overall, we observed that only 8.8% of all the studied TCP connections experienced any out of sequence packets (the percentage varies between 6% and 12%, depending on the trace). These connections contribute however to a significant fraction of all the data packets (48%). This is not surprising since, intuitively, longer connections are more likely to experience an out-of-sequence event.

The last five rows of the table break down the absolute number of out-of-sequence packets according to their cause. We first note that our rules identify the cause of the vast majority of the data packets, with only between 1% and 4% of the out-of-sequence packets falling into the *unknown* category.

Table V indicates that the bulk of out-of-sequence packets are due to the retransmission of lost packets. Note that there isn't a one-to-one relationship between a retransmission and an earlier packet loss since a source may retransmit more than one packet to repair a loss. However, the number of retransmitted packets does provide an approximation of the total number of packet drops experienced by a connection since every lost packet will eventually result in a retransmission.

We also observe that unneeded retransmissions make up a significant percentage of all out-of-sequence packets. We should clarify that, from the sender's perspective, these retransmissions are not really "unneeded". If an ACK is lost or delayed, the sender has no choice but to retransmit.

C. Network Anomalies

The four traces do not show a significant number of network-replicated packets, an event that appears to be rare in all the examined traces. The amount of packet reordering as a fraction of all data packets is also small.

We observe that the magnitude of reordering reported by us is smaller than the results presented in a previous work [3]. Our measurements indicate that reordering affects from 0.17 to 0.96% of all the data packets and that between 0.6 to 5.1% of the connections (with the average being 1.58%) experience reordering. Both these figures are substantially less than those in [3]. We would like to point out, however, that it may not be possible to directly compare the two results due to important methodological differences. The study in [3] relied on active measurements based on ICMP probes. Such probes may sample the network differently than packets in a TCP connection (on which our results are based). For example, a TCP source reduces its sending rate upon detecting congestion, and hence would not sample the network under such conditions. Also, [3] uses probes which are sent back-to-back in time. Back-to-back packets may experience a higher probability of reordering when traversing a router or switch characterized by a highly parallel architecture. Given these methodological differences, we have little reason to expect similar results as these other studies.

However, we note that in [3] the authors did their experiments in 1998 by sending probes through the MAE-East Internet exchange point. They isolated the cause of reordering as due to parallelism within the main network switching device in the exchange point, i.e., the DEC/Gigaswitch. They further conjectured that reordering in general would be a significant factor in the future Internet as a result of increased parallelism in network devices. Our results, four years later, instead suggest the contrary: network reordering affects a small percentage of all data packets.

The study in [11] uses long-lived TCP connections and reports that between 0.3% and 2% of all data packets are reordered in the various data sets. In those data sets, 12% and 36% of all TCP connections experience at least one reordering event. Although we report a similar percentage of reordered packets, a smaller fraction of flows in our traces are affected by a reordering event. This difference could be due to the use of bulk data transfers compared to our results computed over connections of various size.

Although the amount of reordering is limited, it is worthwhile studying the impact of reordering events on a particular TCP connection. For this purpose we define two additional metrics for reordered packets: the "packet lag" and the "time lag". Packet lag refers to the number of packets, with a sequence number greater than the reordered packet, that are seen before the reordered packet itself. The time lag, instead, is defined as the difference between the time a hole in a sequence number is discovered and the time the hole is filled by the reordered packet.

Packet lag represents a useful metric to evaluate the impact of reordering on TCP performance: a lag of three or more packets would trigger the fast retransmit algorithm and force the TCP sender to halve its congestion window. In the four traces about 93% of the reordered packets have a packet lag < 3. Thus, in most cases, reordering has only a minimal impact on the throughput of a connection.

The time lag, in addition to the packet lag, permits one to evaluate more precisely the impact of reordered packet on the end hosts. For example, if the time lag is very short (i.e., less than the delayed acknowledgment timeout—usually in the 50–100 ms range) and the packet lag is only 1, we know that there will be no impact on the throughput of a TCP connection. In our traces, 92% of the reordered packets show a time lag of less than 50 ms

TABLE VI
NUMBER OF ORIGINATING ASES, FOR STUDIED TCP CONNECTIONS AND ALL TCP CONNECTIONS IN THE TRACES

	CDN	Tier-1 ISP	Intra POP	East Coast
Studied TCP Connections	4,752	520	4,837	1,561
All TCP Connections	4,767	4,399	10,657	8,106

and 88% of all the reordered packets have a time lag less than 50 ms *and* a packet lag of 1.

In summary, our results indicate that the amount of data traffic affected by network anomalies such as packet replication or reordering is small and that the impact on the performance of the connections, as perceived by the end-users, is almost negligible.

VIII. DISCUSSION OF RESULTS

We have presented measurement results about the frequency and characteristics of out-of-sequence phenomenon as measured in the Sprint IP backbone. We shall now explore some issues regarding the representativeness and completeness of our results.

A. *Representativeness of Sampled Flows*

In our previous analysis, we have only considered flows that are symmetric with respect to our measurement point, i.e., flows for which we see packets in both the data and the ACK direction at the measurement point. Note that our methodology does not require that a flow be symmetric along the *entire* end–end path but only that we can observe data and acknowledgment packets. As noted in Table I, depending on the measurement set, anywhere from only 9.2% up to 99% of the flows meet this criteria. An interesting question is if, after removal of flows that are not symmetric at the measurement point, the remaining flows have characteristics that are representative of the entire set of TCP flows in the trace.

We examine this question in three different ways. First we compare the number of ASes from which symmetric flows originate with that of all flows in our traces. Then, we examine the distribution of the number of the path length (in router level hops) for the two sets of TCP flows (all flows and only the symmetric ones) in our traces. This provides one simple characteristic of the end–end paths traversed by all flows. Thirdly, we compare the frequency of the out-of-sequence phenomenon between symmetric flows and all flows, in order to examine whether they experience similar conditions along their end–end paths.

We infer the origin AS of the source and destination IP addresses of the TCP connections in our traces. The number of originating ASes in our traces is a measure of the geographical diversity of the end–end paths that we study, and Table VI compares the number of number of ASes from which symmetric flows originate, with that for all TCP flows. We find there can be a substantial difference in the number of originating ASes in the flows we study, with respect to the number of originating ASes computed for all TCP flows in a trace. For any given trace, this difference seems to be directly proportional to the percentage of symmetric flows in that trace.[7] Overall, combining all the traces, we find that the flows we examine originate in 7664 unique ASes, while if we look at all TCP flows in our traces, they originate in 11 627 unique ASes.

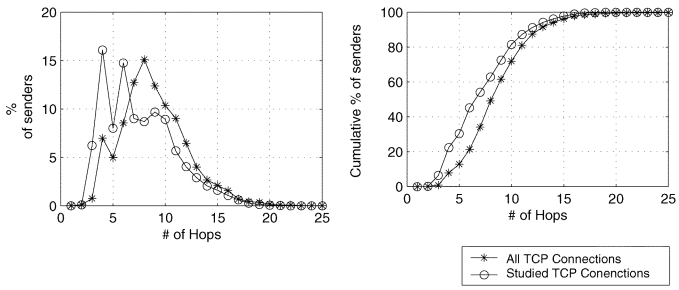


Fig. 9. Distribution of the path length (in router-level hops) from the end-hosts to our measurement point.

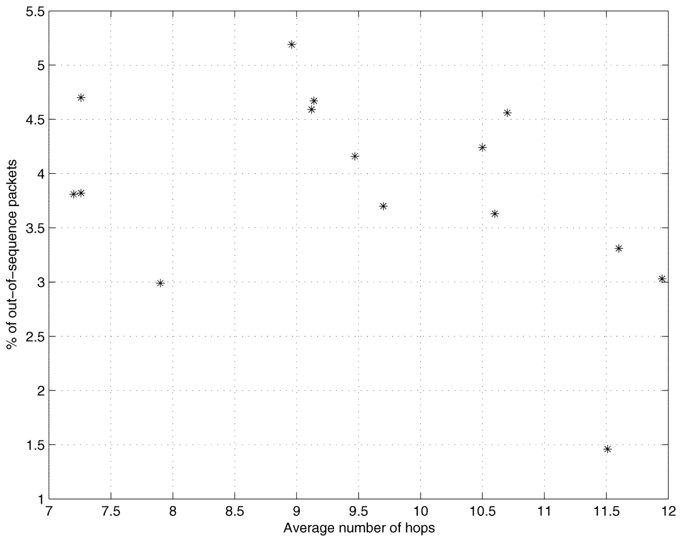


Fig. 10. Scatter plot of the average number of hops from measurement point, and the percentage of OoS packets, in our traces.

Fig. 9 shows the distribution of the path length for the connections in all the traces. We note that the distribution of the length of the path for the connections we examine (i.e., the symmetric ones) and that of all TCP connections in the traces, are somewhat dissimilar. Specifically, a larger fraction of the examined connections are closer to the measurement point, as compared to all flows in the traces. In other words, we observe that closer the measurement point is to one of the end points, there is a higher chance for both the data and ACK paths of a TCP connection to go through the monitoring point.

In order to address the possible impact of this bias in symmetric flows, we examine if there is any correlation between the distance of a sender from the measurement point, and the occurrence of out-of-sequence phenomenon. Fig. 10 is a scatter-plot of the average distance of a sender from the measurement point, and the mean percentage of out-of-sequence packets, in 15 different links. In this plot, we consider *all* flows in these links,

[7]The percentage of symmetric flows is in turn dependant on the nature of the link; access links such as "CDN" have a very high percentage of symmetric flows, while peering and backbone links have a substantially smaller fraction of symmetric flows.

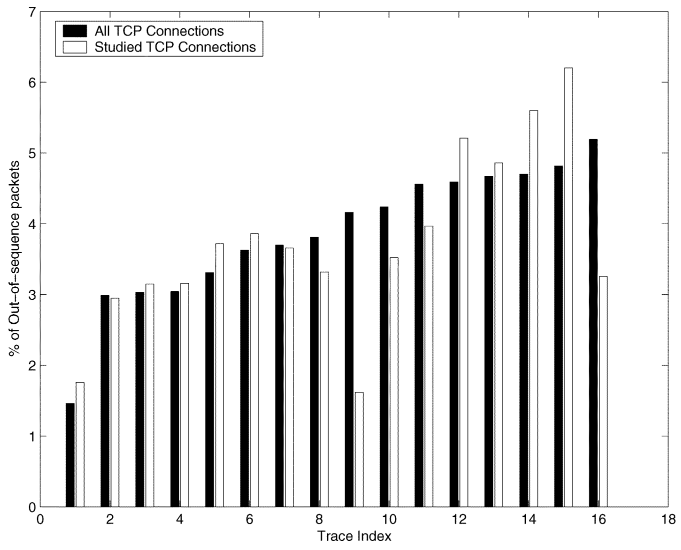


Fig. 11. Percentage of out-of-sequence packets in all TCP flows, and symmetric TCP flows.

and not just those which are symmetric with respect to the measurement point. We observe that there seems to be little correlation, between the distance of the sender from the measurement point, and the percentage of out-of-sequence packets. In a previous work, Padmanabhan *et al.* [9] have also noted the lack of correlation between router-level or AS hop count and end–end loss rate. Thus, we argue, that though the flows we examine in our traces have end-points which are on average closer to the measurement point, than for all flows in the traces, this does not impact (or bias) the magnitude of out-of-sequence phenomenon observed.

A direct comparison can also be made between the percentage of out-of-sequence packets in all flows, and flows that are symmetric at the measurement point. Note that while we cannot identify the causes behind an out-of-sequence packet for a nonsymmetric flow, (since we lack acknowledgment information), we can determine the magnitude (if not the cause) of out-of-sequence packets for all flows, and for symmetric flows, and then compare these two values.

Fig. 11 presents the percentage of out of sequence packets in symmetric and all TCP flows in our traces.[8] These numbers are from a larger set of eight traces, which also include the four traces we have described earlier. Associated with each of the eight traces are two links corresponding to the two directions; we consider each link direction separately and thus show the percentage of out-of-sequence packets for 16 links.

We note that for seven of the 16 traces examined, the relative difference in the percentage of out of sequence packets, between symmetric flows and all flows, is less that 10%, while 12 traces show a difference of less than 20%. Although there can exist a discernible difference between the percentage of out of sequence packets in the two sets of flows (as in trace 9), we do not observe any trend that indicates symmetric flows have a consistently larger or smaller amount of out-of-sequence packets than the entire set of flows.

To summarize, we have explored the representativeness of the studied flows in our traces in two dimensions, the number of the ASes these flows originate from, and the distance of the end hosts from the measurement point. These metrics capture the characteristics of the end–end paths traversed by these flows. We observe that though symmetric flows originate from a smaller number of ASes than all flows, they cover a very substantial percentage (60%) of the number of all ASes in the Internet. We also find that the end-points of the symmetric flows tend to be, on average, closer to the measurement point than all flows. However, since there is little correlation between the occurrence of out-of-sequence phenomenon and the length of a connection's end–end path, this does not bias the magnitude of the out-of-sequence phenomenon observed at the measurement point. We further verify this conjecture by comparing the percentage of out-of-sequence packets in symmetric flows with that in all flows in our traces, and found no evidence that symmetric flows consistently under (or over)-estimate the magnitude of out-of-sequence phenomenon as compared to all flows.

B. *Measuring End-End Reordering/Duplication Frequency*

Our methodology classifies those out-of-sequence events that manifest themselves at the measurement point. The classification rules are only based on the observation of data packets from the TCP sender while the acknowledgments from the receiver are only used to infer the state at the sender (e.g., fast retransmit/recovery state). This is sufficient to capture out-of-sequence packets due to packet drops either before or after the measurement point (except when an entire window of packets is lost before the measurement point) since the consequence of such events (packet retransmissions) are subsequently observed at the measurement point.

However, a closer look at acknowledgments could allow us to identify additional out-of-sequence events that would not manifest in any way in the sequence of data packets observed at the measurement point. A clear example of this is when two previously in order data packets are reordered (or duplicated) after the measurement point, on their way to the receiver. In this case, the receiver immediately responds with a duplicate ACK (an ACK for the packet it was expecting to receive). Thus, looking at duplicate ACKs, one could infer additional reordering/duplication events. However, there are several issues with this approach:

- A duplicate ACK may be generated upon the receipt of any packet that our methodology has already processed and classified to be out-of-sequence, and we would have to eliminate from our analysis duplicate ACKs for packets that have already been classified to be out-of-sequence.
- A receiver sends duplicate ACKs upon the receipt of the unneeded retransmissions of a packet indicating to the sender that it is expecting new data.
- Duplicate ACKs are also used to update the receiver advertised window.
- If the receiver follows a delayed ACK policy, some out-of-sequence events will not trigger a duplicate ACK. For example, in Fig. 12, packets with sequence numbers $x, x+1$

[8]In our calculations, we do not consider "truncated" flows, i.e., flows for which we did not observe the SYN or SYN-ACK packet. Such flows are the ones that were already active before the packet trace collection or that experienced a re-routing event. We discard them because we cannot derive how many packets such flows have transmitted before being captured by our measurement equipment.

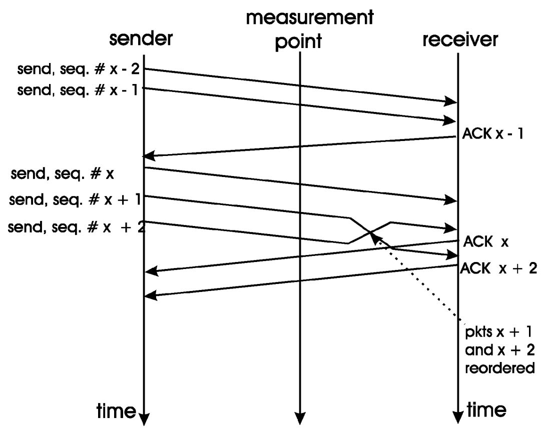


Fig. 12. Packet reordering between the measurement point and the receiver.

TABLE VII
PERCENTAGE OF PACKETS REORDERED OR DUPLICATED AFTER THE MEASUREMENT POINT

	CDN	Tier-1 ISP	Intra POP	East Coast
Reordered/Duplicated	0.96%	0.79%	0.45%	0.63%

and $x + 2$ are sent in a burst. When the receiver observes packet x it refrains from sending an ACK since it adopts the delayed ACK policy. Now, packets $x + 1$ and $x + 2$, are reordered between the measurement point and the receiver, and this triggers an ACK from the receiver, for packet $x+1$. However, this is not a duplicate ACK. In general, if packets are reordered after the measurement point, the resulting packet lag of the reordered packet is 1, and the receiver adopts a delayed-ACK policy, no duplicate ACKs will be observed at the measurement point for this reordering event.

In order to get a rough estimate of the out-of-sequence events (re-ordering and in-network duplication) that take place after the measurement point, we count all the duplicate ACKs and remove those that are clearly associated with the scenarios mentioned above. Table VII lists the percentage of data packets inferred to be reordered or duplicated after the measurement point.

Referring back to Table V we can sum up the figures for reordering before and after the measurement point to get an estimate of the end–end reordering/duplication in these traces, which is 1.13%, 1.75%, 1.02%, and 1.29%, respectively.

IX. CONCLUSION

We have proposed a methodology to classify out-of-sequence packets from passive measurements of backbone traffic. We have developed a set of heuristics to reconstruct the sender's view of a TCP connection by observing the connection in a single measurement point in the middle of the path between the sender and the receiver.

We have shown that the location of our observation point provides a large advantage when compared to traditional end-to-end measurements. We are able to monitor millions of TCP connections originated and destined to about 60% of the entire Internet.

Our results lead to the following observations.

- About 4% of packets generated by TCP connections are out-of-sequence, most of which are due to retransmission in response to a packet loss.
- Packet reordering affects about 1%–1.5% of all data packets, however, they have little impact on the quality of a TCP/IP connection as perceived by the end users.
- Other network anomalies such as duplication of packets represent a very marginal phenomenon in the Internet.

In our opinion, this study represents a fundamental first step for addressing a wide range of research questions, such as the analysis of the correlation between the properties of the path traversed by a connection and other connection-specific metric (e.g., round-trip time, size of the congestion window, packet losses, router hops).

We are also working on identifying the causes behind the packet losses, i.e., congestion, routing or link failures. A first step in that direction would require to study if a TCP connection experiences congestion in a single or multiple bottlenecks along the path. We can also use statistical inference techniques to identify if there is a set of autonomous systems that are responsible for most of the out-of-sequence. To do so, we intend to monitor TCP connections that share portions of the AS path and use a set of tools similar to the ones developed in [4] to identify which ASes are responsible for the out-of-sequence packets.

REFERENCES

[1] J. Bellardo and S. Savage, "Measuring packet reordering," presented at the ACM SIGCOMM Internet Measurements Workshop San Francisco, CA, Nov. 2002.

[2] P. Benko and A. Veres, "A passive method for estimating end-to-end TCP packet loss," presented at the IEEE Globecom Taipei, Taiwan, Nov. 2002.

[3] J. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathologial network behavior," *IEEE/ACM Trans. Networking*, vol. 7, Dec. 1999.

[4] R. Cáceres, N. Duffield, D. Towsley, and J. Horowitz, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. Inf. Theory*, vol. 45, pp. 2462–2480, Nov. 1999.

[5] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the sprint IP backbone," *IEEE Network*, 2003.

[6] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone," presented at the IEEE INFOCOM, San Francisco, CA, 2003.

[7] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," presented at the IEEE INFOCOM, Hong Kong, 2004.

[8] H. Jiang and C. Dovrolis, "Passive estimation of TCP round-trip times," *ACM Comput. Commun. Rev.*, vol. 32, no. 3, Jul. 2002.

[9] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server based inference of internet link lossiness," presented at the IEEE INFOCOM, San Francisco, CA, 2003.

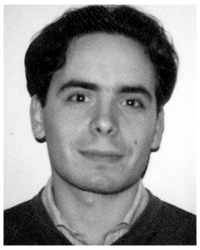
[10] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Trans. Networking*, vol. 5, pp. 601–615, Oct. 1997.

[11] ——, "End-to-end internet packet dynamics," *IEEE/ACM Trans. Networking*, vol. 7, pp. 277–292, Jun. 1999.

[12] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," *ACM Comput. Commun. Rev.*, vol. 27, no. 1, Jan. 1997.

Sharad Jaiswal received the B.E. degree in computer science and engineering from the Regional Engineering College, Trichy, India, the M.S. degree in computer systems engineering from Boston University, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 2005.

He is a Member of Technical Staff at Bell Labs Research India, Bangalore, India. His current interests include network traffic monitoring and modeling and wireless network architecture and design.



Gianluca Iannaccone (M'00) received the B.S. and M.S. degrees in computer engineering in 1998, and the Ph.D. degree in computer engineering in 2002, from the University of Pisa, Italy.

He joined Sprint as a research scientist in October 2001 working on network performance measurements, loss inference methods and survivability of IP networks. In September 2003, he joined Intel Research in Cambridge, U.K. His current interests include system design for fast prototyping of network data mining applications, privacy-preserving network monitoring and routing stability for overlay networks.

Christophe Diot received the Ph.D. degree in computer science from INP Grenoble, France, in 1991.

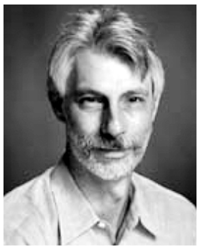
He was with INRIA Sophia-Antipolis, France, from October 1993 to September 1998, Sprint (Burlingame, CA) from October 1998 to April 2003, and Intel Research (Cambridge, U.K.) from May 2003 to September 2005. He joined Thomson in October 2005 to start and manage the Paris Research Laboratory. His research activities focus on communication services and platforms for the future.

Dr. Diot is an ACM Fellow.

Jim Kurose (S'81–M'84–SM'91–F'97) received the Ph.D. degree in computer science from Columbia University, New York, NY.

He is currently Distinguished University Professor (and past chairman) in the Department of Computer Science at the University of Massachusetts, Amherst. He has been a Visiting Scientist at IBM Research, INRIA, Institut EURECOM , University of Paris, LIP6, and Thomson Research. With Keith Ross, he is the coauthor of the textbook *Computer Networking, a Top Down Approach Featuring the Internet (3rd edition)* (Addison-Wesley Longman, 2004). His research interests include network protocols and architecture, network measurement, sensor networks, multimedia communication, and modeling and performance evaluation.

Dr. Kurose has served as Editor-in-Chief of the IEEE TRANSACTIONS ON COMMUNICATIONS and was the founding Editor-in-Chief of the IEEE/ACM TRANSACTIONS ON NETWORKING. He has been active in the program committees for IEEE INFOCOM, ACM SIGCOMM, and ACM SIGMETRICS conferences for many years, and has served as Technical Program Co-Chair for these conferences. He has won many awards for his educational activities, including IEEE Taylor Booth Education Medal. He is a Fellow of the IEEE and the ACM.



Don Towsley (M'78–SM'93–F'95) received the B.A. degree in physics and the Ph.D. degree in computer science from the University of Texas, Austin, in 1971 and 1975, respectively.

From 1976 to 1985, he was a Member of the Faculty of the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, where he is currently a Distinguished Professor in the Department of Computer Science. He has held visiting positions at IBM T. J. Watson Research Center, Yorktown Heights, NY, Laboratoire MASI, Paris, France, INRIA, Sophia-Antipolis, France, AT&T Labs–Research, Florham Park, NJ, and Microsoft Research Lab, Cambridge, U.K. His research interests include networks and performance evaluation.

Dr. Towsley currently serves as Editor-in-Chief of IEEE/ACM TRANSACTIONS ON NETWORKING and on the editorial boards of the *Journal of the ACM* and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and has previously served on numerous other editorial boards. He was Program Co-chair of the joint ACM SIGMETRICS and PERFORMANCE '92 conference and the Performance 2002 conference. He is a member of ACM and ORSA, and Chair of IFIP Working Group 7.3. He has received the 2007 IEEE Koji Kobayashi Award, the 1998 IEEE Communications Society William Bennett Best Paper Award, and numerous best conference/workshop paper awards. He has been elected Fellow of both the ACM and IEEE.