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

Sharad Jaiswal†, Gianluca Iannaccone§, Christophe Diot§, Jim Kurose†, Don Towsley†

§Sprint ATL
Burlingame, CA
{gianluca,cdiot}@sprintlabs.com

†Computer Science Department
Univ. of Massachusetts, Amherst
{sharad,kurose,towsley}@cs.umass.edu

***Abstract*— We present a measurement study and classification methodology for out-of-sequence packets in TCP connections observed within the Sprint IP backbone. Such out-of-sequence packets can result from many causes 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 they are one indication of the "health" of an end-end TCP connection. Our first contribution is methodological. Because we measure out-of-sequence packets at a *single* point in the backbone (rather than by sending and measuring end-end probe traffic at the sender or receiver), a new methodology is required to infer the causes of a connection's out-of-sequence packets based only on measurements taken in the "middle" of the connection. We thus describe techniques that classify the causes of observed out-of-sequence behavior based only on the previously- and subsequently-observed packets within a connection and knowledge of how TCP behaves. We show that using these simple 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 second contribution is the characterization of the out-of-sequence behavior itself. We analyze numerous several-hour packet-level traces from a set of OC-3 and OC-12 links for several million connections generated in nearly 4,300 unique ASes. Our measurements show a relatively consistent amount of out-of-sequence packets of approximately 5%. We find that few out-of-sequence packets result from pathological problems such as routing loops or in-network duplication/reordering.**

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 magnitude of out-of-sequence packets within Internet TCP connection, and to identify their causes, as the magnitude of out-of-sequence packets is a good indicator of the "health" of a TCP connection and the path that it is traversing.

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 that of a previously observed packet in that connection[1]. Our contributions are twofold. The first contribution is methodological. Because we measure out-of-sequence packets at a *single* point in the backbone (rather than by sending and measuring end-to-end probe traffic at the sender or receiver [1], [7]), a new methodology is required to infer the causes of a connection's out-of-sequence packets based only on measurements taken in 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. While the use of a single measurement point has the advantage of sampling traffic from a very large number of connections, it also poses challenges. Because we are taking measurements in the "middle" of a TCP connection, we do not know whether a data or ACK segment observed at our measurement point is received at the intended destination and/or the action taken at the packet's destination; we can only infer this from the previously- or subsequently-observed packets from that connection and our knowledge of how TCP behaves. We describe techniques and rules to infer and classify the causes of observed out-of-sequence behavior. We show that by using these simple techniques, it is possible to classify almost all out-of-sequence packets in our traces and that we can characterize the uncertainty in our classification.

Several of our classification 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 can again only infer these values from our measurements. Thus, we also describe

This work is supported by the National Science Foundation under grants ANI-9805185 and ITR-0080119, 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. Part of this work was carried out when Sharad Jaiswal was an intern at Sprint ATL.

[1] In this paper, the terms "out-of-sequence" and "reordered" do not have the same meaning. As will be discussed, 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.

techniques for estimating per-connection end-to-end RTT from a single measurement point within the network.

Our second contribution is the characterization of the out-of-sequence behavior itself. We characterize the out-of-sequence behavior of 19 million TCP connections, generated in more than 4,300 unique Autonomous Systems (ASes). 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 5% of packets are of out-of-sequence; we find that out-of-sequence packets result primarily from packet loss, and that few out-of-sequence packets result from pathological problems such as in-network duplication or reordering.

A number of previous efforts (in particular [1], [7]) have examined packet reordering. Our work differs from these earlier works in both the scope of our measurements (millions of connections from thousands of different networks), 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), and in the amount of in-network reordering that we observe. We will discuss our results in relation to these studies in more detail in Section V.

The remainder of this paper is structured as follows. We begin in Section II by describing the rules and rationale that we use to infer and classify the causes of observed out-of-sequence behavior. In Section III we identify sources of uncertainty and possible errors in our inferences. In Section IV, we compare and contrast several approaches for inferring a connection's RTT from a single measurement point, and discuss how this value affects our results. We present our measurement results and classification of observed out-of-sequence packets in Section V. Finally, Section VI concludes this paper.

II. Methodology

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

Given a trace of observed sender-to-receiver data packet headers and receiver-to-sender acknowledgment headers, we process this trace to classify the causes of out-of-sequence measurements. Our first step in processing our 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 we are making our measurements.

Once we have filtered the trace, we can identify and classify the out-of-sequence packets. A packet is out-of-sequence if its sequence number is less than 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 retransmitted packet will have a sequence number that is smaller than previously observed packets 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**. In this case, the network inverts the order of two packets in a connection (for example, because of parallelism within a router [1] or a route change).

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 which result in an out-of-sequence packet.

As detailed below, we will use observed sequence and acknowledgment values in the TCP header, the identification field in the IP datagram (IP ID), and the times at which observations are made to infer the cause of an observed out-of-sequence packet. Below, we will use the 3-tuple notation (W, x, t) to denote a sender-to-receiver data packet with an IP ID value of W and a TCP sequence number of x, that was observed at our measurement point at time t. Figure 1 illustrates this notation. In this example, the sender sends packets $x-1$ through $x+3$. Packet (W, x, t) is observed at the measurement point at time t. Because of the subsequent loss of x between the measurement point and the receiver, the sender will eventually retransmit x. The retransmitted copy of x, (W', x, t'), is observed at the measurement point at time t'. Note that while the sequence number of the retransmitted packet is x, its IP ID may have changed to W'.

Figure 2 summarizes the decision process that implements the rules below to classify out-of-sequence packets. The edges leading to leaf nodes (classifications) in the decision tree in Figure 2 are annotated with the decision rules (R1 through R6, described below) corresponding to those classifications. Let us now consider those classification rules.

A. Retransmissions

Figure 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 with the same sequence number x, (W', x, t'), is observed. Rule R1 below specifies the cases in which this second packet is classified as a retransmission.

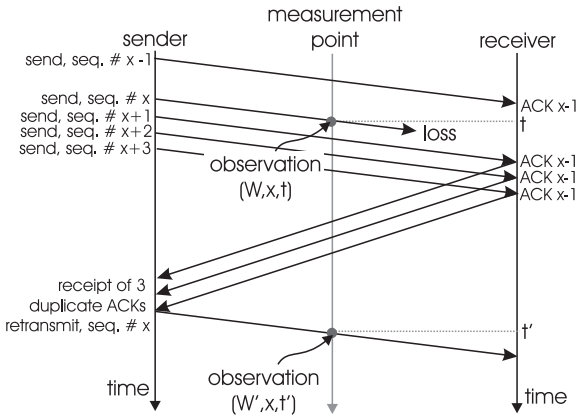

Fig. 1. Retransmission due to a loss after the measurement point

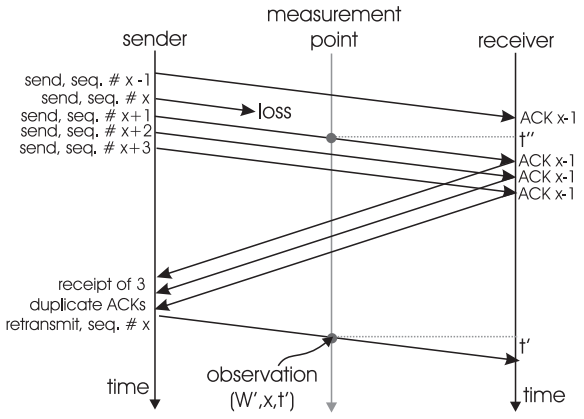

Fig. 3. Retransmission due to a loss before the measurement point

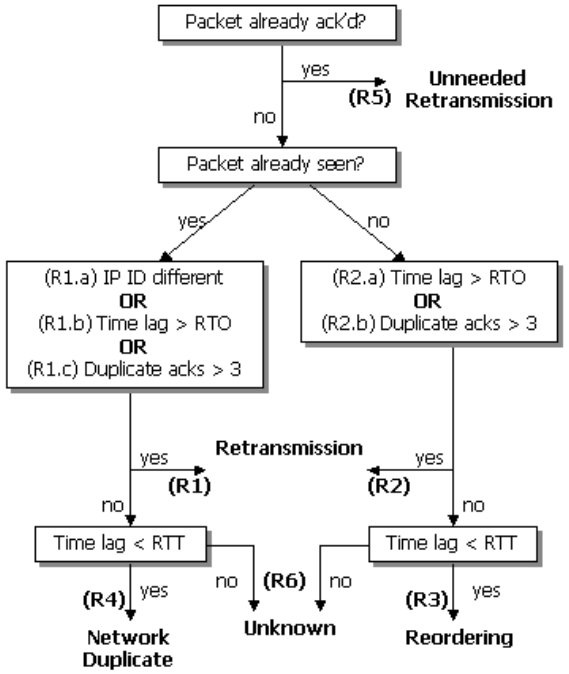

Fig. 2. Decision process for the classification of out-of-sequence packets

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:** $t' - t > RTO$, 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 both RTO estimation as well as for other purposes, as we will see shortly. We address the problem of estimating the sender's RTT in Section IV.
- **R1.c:** the number of duplicate ACKs observed after t but before t' exceeds the sender's duplicate ACK threshold (which we take in this paper to be the typical value of 3).

A closely related scenario to R1 is the case in which an out-of-sequence packet (W', x, t') is observed, but the earlier packet, (W, x, t) has not been observed at the measurement point, as shown in Figure 3. In this case, the original packet (W, x, t) was sent by the sender but lost before the measurement point.

Rule R2*:* Assume (W', x, t') is observed, but an earlier packet, (W, x, t) is not observed at the measurement point. Also assume that no acknowledgment covering x is 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:** $t' - t'' > RTO$, 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, packet (W, x, t) has not been observed at the measurement point.
- **R2.b:** the number of duplicate ACKs observed after t'' but before t' exceeds the sender's duplicate ACK threshold (see rule R1.c).

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 IV 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 R3*:* Assume an out-of-sequence packet (W', x, t') is observed, but an earlier packet, (W, x, t) is not observed at the measurement point. Furthermore assume that none of the conditions R2a - R2b 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. This definition of reordering corresponds in spirit (although not in exact details) to the notion of reordering in [1].

C. Duplicates

Suppose now that we observe a packet, (W, x, t') that has the same IP ID 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 IP ID and sequence number) to a recently observed packet. We classify such a packet as a network-generated duplicate.

Rule R4*:* 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

Figure 4 illustrates the case in which a packet with sequence number x and its acknowledgment (or the acknowledgment of a 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 R5*:* If a packet (W, x, t) and an acknowledgment covering this packet have been observed, and a second packet (W', x, t'), with IP ID 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.

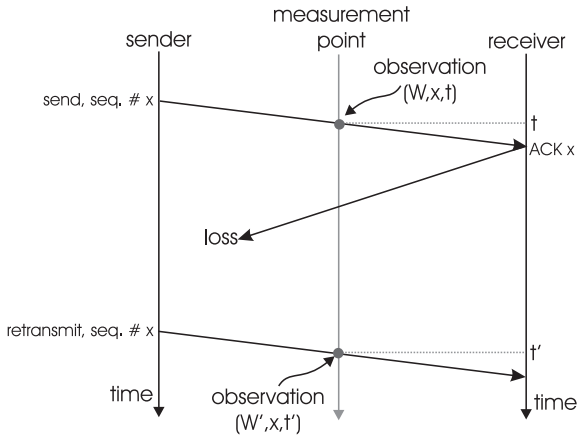

Fig. 4. An unneeded retransmission due to a lost ACK

Rule R6*:* A packet not classified under R1 — R5 is classified as "unknown", i.e. it cannot be unambiguously classified as belonging to any category.

We will shortly show that only a small fraction of the observed out-of-sequence packets are not classifiable under rules R1 through R5.

III. Sources of errors

While the use of a single measurement point within the network (on the path from sender to receiver) has the advantage of sampling traffic from a very large number of source-destination pairs without having to instrument the individual senders and receivers, it also poses several methodological challenges. In this Section we discuss a number of those challenges.

Recall from our discussion in Section II that rules R1, R2, and R4 either directly or indirectly make use of the current value of the sender's current RTT. In Section IV, we evaluate several different approaches towards estimating the RTT. In all cases, however, our estimates are based on measurements made at the observation point, rather than at the sender. Figure 5 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. 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.

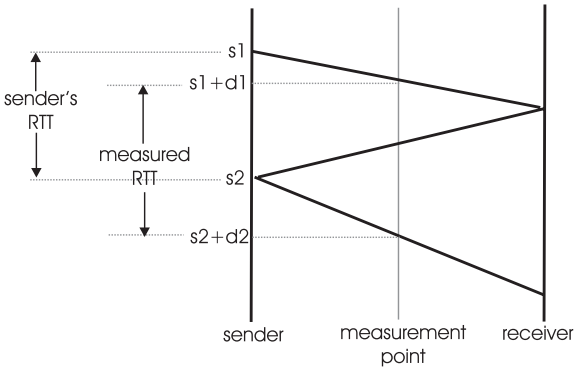

Fig. 5. Differences in RTT estimation

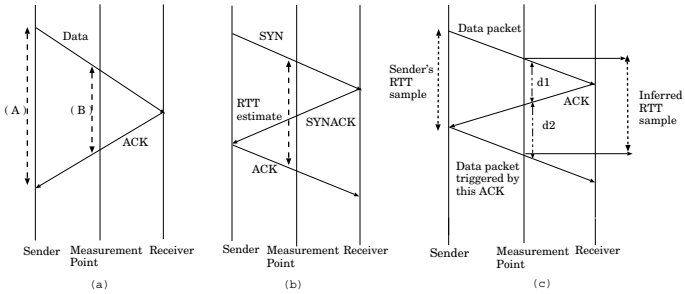

Fig. 6. Techniques for computing RTT (a) imitating the sender, (b) triple-handshake RTT (c) running estimate

A different set of potential errors is introduced as a result of the difference between the information that we observe, and what is actually observed by the sender and receiver:

- If the first packet of the connection (the SYN packet) is lost before our measurement point, we cannot detect it.
- We cannot detect the loss of an *entire* congestion window of packets that occurs between the sender and our measurement point.
- We cannot be sure if an observed ACK at the measurement point is ever received at the sender. While this latter uncertainty does *not* affect classification rules R1 through R6, it would impact our ability to estimate whether a retransmission is due to a retransmission timeout or the fast retransmit algorithm, for example.

IV. RTT Estimation

As discussed earlier, the need to estimate the 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 R3 and R4, respectively).

One possible approach to compute RTT is to mimic the behavior of the TCP sender at an end-point. As shown in Figure 6a, a TCP sender calculates the RTT by computing the time interval between its transmission of a data packet and the receipt of the ACK covering this packet. This interval of time is labeled (A) in Figure 6a. But if the measurement point were to mimic this, the sender-to-measurement-point and measurement-point-to-sender delays would not be accounted for in the RTT estimate (see label (B) in Figure 6a).

Another set of approaches, described in [5], [6] rely on the triple-handshake or slow-start phase of the TCP connection to compute one RTT sample per TCP connection. The triple-handshake technique is illustrated in Figure 6b. In this method, the RTT is computed as the difference between the time at which a SYN packet is observed, and the time at which the last ACK that completes the handshake is observed. In [5], an approach known as *slow-start* estimation searches for the first burst of a congestion window full of packets just after the triple-handshake is complete. It then uses the time difference between this burst of packets and the successive burst as an estimate of the RTT.

The primary drawback of these latter set of techniques is that they measure only one RTT sample per TCP connection. If the TCP connection experiences variable delays during its lifetime, the single sample may be highly unrepresentative of the RTT values for the connection. Moreover, there is no guarantee that this particular sample was itself an accurate RTT estimate at the instant it was computed. For example, servers may respond to an incoming SYN (with a SYNACK) only after some delay. This would lead to an overestimate of the actual RTT if one uses the triple handshake technique.

A. Running RTT estimation

Our approach aims to dynamically make one RTT estimate per window of packets sent by the sender. The basic idea behind this technique is illustrated in Figure 6c. Since we are not able to directly measure the sender's RTT sample shown in the left of Figure 6c, we instead measure *(i)* the round trip from the measurement point to the receiver and then back to the measurement point (labeled d_1 in the figure), and *(ii)*, the round trip delay between the measurement point, the sender and then back to the measurement point (labeled d_2 in the figure). The sum of these two delays $d_1 + d_2$, as shown in Figure 6c, is our estimate of the RTT. We term our method the *running* RTT estimation technique, since it continuously makes RTT estimates, based on the measured values of d_1 and d_2 throughout the TCP connection's lifetime. In the case that the sender's RTT sample, d_1 and d_2 do not change from one packet to the next, our estimate of the RTT is exact. As these quantities vary from packet to packet, our estimate will be approximate. There are other issues which determine the accuracy of our estimation technique. An important requirement of this approach is the ability to determine which data packet transmissions are triggered by the arrival of a particular ACK. This is possible only if we have an accurate estimate of the congestion window (*cwnd*), which allows us to distinguish between packets in successive windows. Another important aspect of our technique is to stop performing RTT estimation as a flow recovers from a loss and restart the sampling scheme once the lost packet has been retransmitted. We are unable to go into the details of these issues due to lack of space, but a more comprehensive

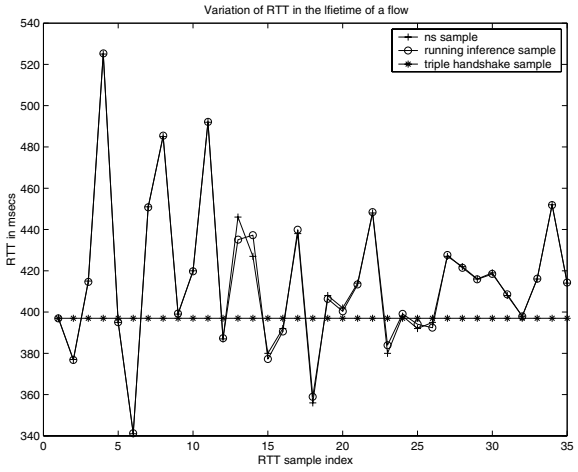

Fig. 7. Variation of RTT of a flow during its lifetime

discussion, along with a detailed pseudo-code describing our approach is presented in [4].

B. Evaluating the running RTT estimation technique

We have evaluated the effectiveness of our running RTT estimation technique using the ns-2 simulator. We shall provide a brief overview of the results here, without going into the details of the simulation framework and the experiments. The reader is referred to [4] for a detailed description. In brief, the simulation topology consists of several end-nodes connected by an end-to-end path several hops long. We use a mixture of both short and long flows, which arrive and leave over the course of the simulation. A shared link is chosen as the bottleneck, and we set the loss and delay characteristics of the network by tuning the parameters of the bottleneck link.

Figure 7 plots the RTT between two end nodes as *(i)* measured at the ns sender, *(ii)* estimated at the single measurement point on a link in the middle, and *(iii)* estimated using the triple handshake approach at the single measurement point. The figure illustrates the variation in the true sender RTT as a function of time. Clearly, the triple-handshake technique does not keep track of such variations, and can thus produce poor a estimate at a given point in time. On the other hand, our running RTT estimation technique dynamically tracks these variations in the RTT with very good precision, even though it is making these estimates based on observations in the "middle"of the end-to-end path.

Figure 7 illustrates the quality of the RTT estimates for just a single sample path, for a relatively short period of time. In order to more broadly assess the accuracy of our RTT estimation technique, we also examine the relative error of our running estimate as compared to the corresponding ns estimate, over all the RTT estimates made.

In our experiments, we observe that the running RTT estimation technique has a significantly smaller relative error than the triple handshake approach. More specifically, we find that 90% of the running RTT estimates have a relative error less than 10%. In contrast, the triple-handshake technique fares relatively poorly, with 90% of connections having a relative error less than 30%.

We have also evaluated our technique over different flavors of TCP (i.e. Tahoe, Reno, Newreno). Our simulations suggest that the running estimation technique consistently provides an accurate estimate of the RTT over a wide range of scenarios.

We thus now have a technique which allows us to dynamically estimate the RTT of a TCP connection from a single measurement point. Armed with this estimation technique and classification rules discussed in the previous section we have a methodology that will allow us to study the causes of observed out-of-sequence packets.

V. Measurement and Classification results

In this section, we describe the measurement infrastructure and network setting used to obtain our packet traces, and then present the results of applying our classification rules to these traces. Our measurements were obtained using infrastructure developed as part of the Sprint IP Monitoring (IPMON) project [3]. 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 a very 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 February 3rd, 2002 and October 9th, 2002. We have observed similar trends to those reported below in many other traces collected on different dates and over different links. Three of the four traces under study are 6 hours long and were collected on access links in Points of Presence (POPs) located in the West and East Coast of the United States. We identify the traces as: "CDN", "Tier-1 ISP" and "Tier-2 ISP" based on the nature of the customer that contributes most to the data traffic in that particular trace. "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[2]. "Tier-1 ISP" and "Tier-2 ISP" are links connecting to other service providers that carry traffic coming from (and destined to) a very large and diverse set of end hosts. The last trace (identified by "OC48") instead was collected on a long-haul OC48 (2.5 Gbps) link.

Table I 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

[2]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.

	CDN	Tier-1 ISP	Tier-2 ISP	OC48
Link speed	OC-12 (622 Mbps)	OC-12 (622 Mbps)	OC-12 (622 Mbps)	OC-48 (2.5 Gbps)
Duration (hours)	6	6	6	1
Unique source ASes	1,587	408	1,196	2,532
TCP connections	4.8M	2.1M	4.7M	6.6M
Percentage of all TCP connx	99.46%	15.68%	12.79%	27.21%
TCP Data packets	91M	39M	245M	153M

TABLE I

SUMMARY OF THE TRACES

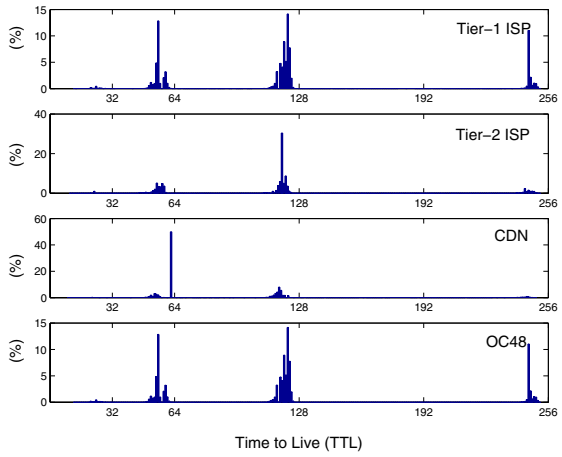

Fig. 8. Distribution of time-to-live values

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 4,353 unique Autonomous Systems[3]. This represents about the 34% of the total number of allocated ASes at the time of the trace collection (we counted 12,822 unique Internet AS numbers as of October 9th, 2002).

Another interesting characteristic of our traces is the relative position of the collection point in the end-to-end path between the source and the destination. In order to explore this further, we used the time-to-live (TTL) values present in the IP packets to determine the distance (in terms of the number of router hops) from the two end hosts to the measurement point. Figure 8 shows the observed distribution of TTL of all connections in the four traces[4]. We note that TTL values tend to aggregate around a few well-known values that are commonly used in the end host systems: 32, 64, 128 and 256. Using this fact, we can easily derive the initial TTL values set by any given source, and then infer the distance between that host and our measurement point.

Table II shows the average distance from the two hosts to the measurement point in number of router hops (d_a and d_b) and autonomous systems. A distance in AS hops of zero means that the traffic is sinked in one of the Sprint's customer network that does not have a separate AS number. We also computed the average value of the ratio $R = min(d_a, d_b)/(d_a + d_b)$ a value close to 0.5 implies that our monitoring point is on average close to the middle point of the path.

As expected, the CDN's servers are located very close to the backbone - they are only two hops away on average! On the other hand, the tier-1 and tier-2 ISP and the OC48 links carry traffic destined to a very diverse set of end hosts; our monitoring point is, on average, very close to the midpoint of the paths of those TCP connections. The average distance between client and server is in the range of 14 to 18 router hops in these traces. For connections destined to the CDN's servers, the average distance reduces to 10 hops[5].

B. Representativeness of symmetric 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-to-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 13% 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 two different ways. First, we compare the distribution of the number of router level hops for the two sets of TCP flows in our traces (all flows and only the symmetric ones). This provides one simple characteristic of the end-to-end paths traversed by all flows. Then, we compare the frequency of the out-of-sequence phenomenon in the two sets of flows (symmetric or not), in order to examine whether they experience similar conditions along their end-end paths.

Figure 9 shows the distribution of the router hops for the connections in all the traces. We note that the distribution of router hops for the connections we examine (i.e., the

[3]The sum of the unique AS numbers from Table I is higher (5,726) because we may have the same AS present in more than one trace.

[4]For the purpose of our analysis we consider the TTL of a connection as the TTL of the first data packet.

[5]Note that even if the TCP connection is symmetrical at the measurement point, we cannot assume that it is symmetrical along the entire path. For this reason, we can only compute the distance in router hops *from* the two end-hosts and in AS hops *to* the two end-hosts. The total length of the path, however, may differ from the sum of the two components.

	CDN	Tier-1 ISP	Tier-2 ISP	OC48
Avg. number of router hops from end-hosts (d_a; d_b)	8.44; 2.00	8.19; 7.66	8.84; 8.92	6.14; 8.44
Average $min(d_a, d_b)/(d_a + d_b)$	0.20	0.43	0.43	0.38
Avg. number of AS hops to the end-hosts	0.80; 1.00	1.38; 0.90	0.99; 1.04	1.05; 1.44

TABLE II

AVERAGE DISTANCES FROM OUR MEASUREMENT POINT

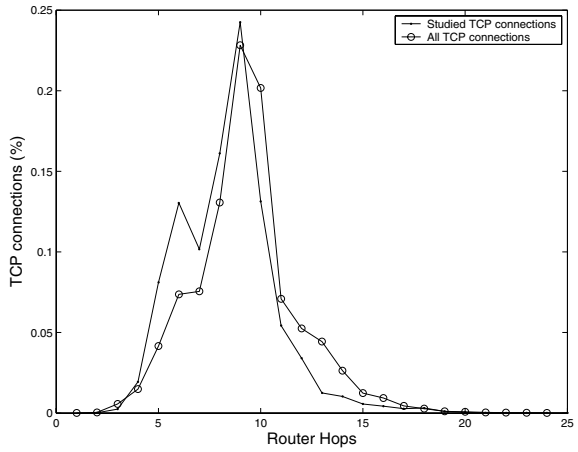

Fig. 9. Distribution of the number of router hops from the end-hosts to our measurement point

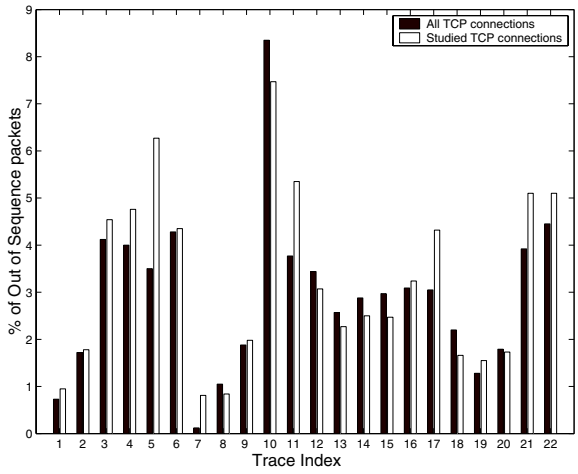

Fig. 10. Percentage of Out-of-Sequence Packets in all TCP flows, and symmetric TCP flows

symmetric ones) is very similar to that of all TCP connections in the traces. We have also studied and observed a similar trend for AS-level hops.

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 non-symmetric flow, (since we lack acknowledgement 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.

Figure 10 presents the percentage of out of sequence packets in symmetric and all TCP flows in our traces.[6]. These numbers are from a larger set of 11 traces, which also include the 4 traces we have described earlier. Associated with each of the 11 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 22 links.

We note that for 10 of the 22 traces examined, the relative difference in the percentage of out of sequence packets, between symmetric flows and all flows, is less that 10%, while 15 traces have a relative error less than 20%. Although there can can exist a discernible (and sometimes significant, as in trace 5) difference between the percentage of out of sequence packets in the two sets of flows, 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. In 15 of the 22 links presented above, symmetric flows experience a higher frequency of out-of-sequence packets as compared to all TCP flows, with the opposite holding true for the other links.

Overall, we note that symmetric and asymmetric flows exhibit similar end-to-end path characteristics, in terms of number of router and AS hops traversed and magnitude of the out-of-sequence phenomenon. However, given that our current methodology can only process symmetric flows, we can not determine whether any differences might exist in the causes of out-of-sequence packets between symmetric and asymmetric flows.

C. Round trip times

We next consider the observed distribution of round trip times of the TCP connections in the three traces under study. Figure 11 plots the cumulative distribution of the average RTT of the TCP flows. We observe that the Tier-1 trace has a lower average RTT as compared to the Tier-2 trace, with a median value of 150 msecs. In contrast, for the Tier-2 trace, the minimum value is around 100 msecs, with the median average RTT at 450 msecs. The difference between the two traces can be explained by the fact that the Tier-1 trace is from an ISP located within the US, while the Tier-2 trace is from an European ISP, and hence has higher propagation delays. The OC48 trace however contains a very diverse mix of traffic

[6]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 packet such flows have transmitted before being captured by our measurement equipment.

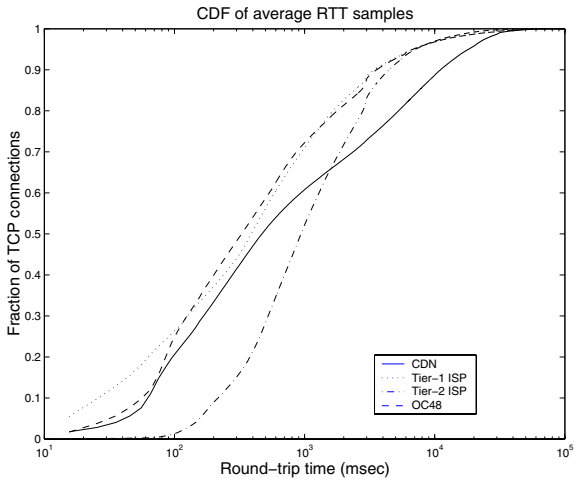

Fig. 11. Distribution of average RTTs from the running estimate

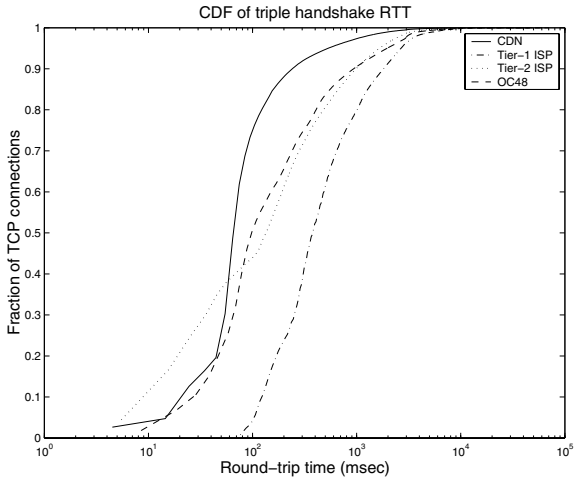

Fig. 12. Distribution of triple handshake RTTs

from many different sources, and it is not easy to extrapolate any general conclusions about its distribution.

More interestingly, even though the connections in the CDN trace cover fewer hops (10) compared to those in the other two traces (around 16), the median of the average RTT (300 msecs) is higher compared to the Tier-1 trace. Although we are still investigating the causes of these observations, we conjecture that a possible cause could be that a large fraction of the CDN traffic is contributed by users with dialup modem connections (which have large, variable delays).

We next compare the average RTT values of the TCP connections with the RTT sample values computed using the triple-handshake technique. Figure 12 plots the distribution of RTTs calculated using this technique. We note that the distribution for the CDN trace has a very steep slope. This implies there are a large number of samples clustered around a particular value. Recall, however, that we have already observed from Figure 11 that flows have a wide range of average RTT values, implying that many flows experience significant variation in delay after the connection has been established. These variations would suggest that the single RTT estimate taken during the triple handshake is not likely to be an accurate estimation of the RTT at another point in time during the connection.

Figure 11 shows also that between 5% and 25% of the flows (depending on the trace) have an average RTT larger than 2 seconds - a value that we believe to be abnormally high. There are several possible explanations. Firstly, we assume that TCP senders in the examined connections are greedy, and when permitted by the protocol, they always have packets to send. In the case this assumption does not hold, our estimate of the RTT could be inflated since it would incorporate significant sender-end delays. Secondly, we note that the CDN trace involves a relatively small set of servers being accessed. If these servers are homogeneous, then any error in estimation that are server-specific would be magnified in these traces (as opposed to the Tier-1 ISP, Tier-2 ISP and OC48 traces). Second, as noted earlier, the Tier-2 trace is from a European ISP and so some of the large RTTs could be the result of large trans-oceanic delays. We plan to examine this further, and hope to separate and quantify the error.

Finally, we note that in the following subsection on classification of out-of-sequence packets, we use the running RTT estimation technique. In order to assess the sensitivity of our classification methodology to errors in the RTT estimation, we also performed the classification using the simple triple-handshake RTT estimation and a constant value of 1 second for the RTO (the smallest prescribed value). We observed that the results showed relatively small differences across the use of different approaches to estimate RTT/RTO, suggesting the low sensitivity of the results to the RTT estimation technique. We note, however, that in order to have confidence in our classification methodology (given its dependence on the RTT estimate) we had to develop multiple RTT estimators and study their relative performance (against each other and as part of our classification process).

D. Out of sequence packets

Table III shows the classification results for the observed out-of-sequence packets in the three traces. The first two rows in Table III indicate the total number of data packets observed, and the absolute number and percentage of these packets that were out-of-sequence. Generally, the number of out-of-sequence packets is limited to about 5% of the total data packets exchanged by the TCP connections.

Overall, we observed that only 13.60% of all the studied TCP connections experienced any out of sequence packets (the percentage varies between 7.20% and 20.10%, depending on the trace). These connections contribute to the bulk of all the data packets (71.03%). 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 and the relative percentage of out-of-sequence packets according to the inferred cause of out-of-sequence behavior. We first note that Rules R1 through

R5 classify the vast majority of these data packets, with only between 1% and 8% of the out-of-sequence packets being classified into the *unknown* category.

Table III 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 out-of-order (retransmitted) packets and earlier packet loss, since a source may retransmit more packets than needed to repair a loss, depending on the TCP flavor (e.g. Tahoe, Reno, New Reno, SACK). However, the number of retransmitted packets does provide a rough upper bound on the total number of packet drops experience by a connection, since every lost packet will result in a retransmission.

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

Another interesting observation that we can derive from Table III is that the connections destined to the CDN customers experience much less "out-of-sequencing" on average than those sourced or sinked in the Tier-1 ISP or Tier-2 ISP. One possible interpretation is that the CDN algorithm used to allocate clients to servers performs well and that paths to the CDN are better provisioned than the paths sourced or sinked in the Tier-1 ISP or Tier-2 ISP. Further investigation is needed to determine the correlation between out-of-sequence packets and other characteristics of the TCP connections, like, for example, the size, the number of router hops or the number of AS hops between the end-hosts.

E. 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 insignificant, but its contribution to out-of-sequence packets varies between the traces. The tier-1 ISP trace shows about two and a half times more reordered packets than the tier-2 ISP trace, even though the length of the paths are very similar (see Table II). Although we have not been able to pinpoint the cause of such a difference, we conjecture that it could be explained (as in the case of [1]) by the specific routers that the two ISPs deploy in their networks.

We also observe that the magnitude of reordering reported by us is much smaller than the results presented in previous works [1], [7]. Our measurements indicate that reordering affects from 0.03 to 0.72% of all the data packets and that between 0.15 to 4.9% of the connections (with the average being 1.84%) experience reordering. Both these figures are substantially less than those in [1], [7]. 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 [1] 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 does not send packets upon detecting congestion, and hence would not sample the network under such conditions. In [7], the author uses long-lived TCP connections while our results are computed over connections of various lengths. Also, [1] uses probes which are sent back-to-back in time. We conjecture that 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 believe that we should see similar results as compared to these studies.

However, we note that in [1] 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 very small percentage of all data packets.

Although the amount of reordering is limited, it is worthwhile to study 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 3 or more packets would trigger the fast retransmit algorithm and force the TCP sender to halve its congestion window. In the four traces we have observed that about 87% 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 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 delay – usually in the 50-100ms 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, 66% of the reordered packets show a time lag of less than 50ms and 60% of all the reordered packets have a time lag less than 50ms 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 minimal and that the impact on the performance of the connections, as perceived by the end-users, is almost negligible.

	CDN		Tier-1 ISP		Tier-2 ISP		OC48	
Data packets	90,905,926		39,403,671		245,535,161		153,143,822	
Out-of-sequence	1,454,772	(1.60%)	1,841,961	(4.67%)	14,131,004	(5.76%)	7,812,412	(5.10%)
Retransmissions	1,263,367	(86.84%)	1,149,450	(62.40%)	9,954,978	(70.45%)	5,648,507	(72.30%)
Unneeded Retransmissions	136,700	(9.40%)	227,279	(12.34%)	2,761,503	(19.54%)	1,108,610	(14.19%)
Network Duplicates	145	(0.01%)	604	(0.04%)	6,748	(0.05%)	1,973	(0.03%)
Reorderings	28,558	(1.96%)	307,615	(16.70%)	943,188	(6.67%)	653,717	(8.37%)
Unknown	26,113	(1.79%)	157,473	(8.55%)	470,571	(3.28%)	402,312	(5.12%)

TABLE III
OUT-OF-SEQUENCE PACKET CLASSIFICATION

VI. CONCLUSIONS

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 34% of the entire Internet.

On the other hand, the location of our monitoring equipment also poses a new set of challenges to perform the classification, namely: i) we have to infer what packets are received by the end hosts; ii) we have to infer what is the round trip time of the connection for each window of data; iii) we have to extrapolate the events that caused a particular reaction (i.e. retransmission, unneeded retransmission, etc.) by the sender.

Our results lead to the following observations:

- About 5% of packets generated by TCP connections are out-of-sequence, most of which are due to retransmission in response to a packet loss;
- Network anomalies such as reordering or duplication of packets represent a very marginal phenomenon and we have shown their minimal impact on the quality of the connection as perceived by the end users;
- The CDN trace exhibits the least amount of out-of-sequence packets. This could be indicative of the advantage CDN-based servers provide to end users.

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 [2] to identify which ASes are responsible for the out-of-sequence packets.

REFERENCES

[1] J. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6), Dec. 1999.

[2] R. Cáceres, N. Duffield, D. Towsley, and J. Horowitz. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, 45(7):2462–2480, Nov. 1999.

[3] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi. Packet-level traffic measurements from a tier-1 IP backbone. Technical Report TR01-ATL-110101, Sprint ATL, Nov. 2001.

[4] 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. Technical Report 02-17, Computer Science Dept., UMass Amherst, May 2002.

[5] H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times. Technical report, July 2001.

[6] H. Martin, A. McGregor, and J. Cleary. Analysis of internet delay times. In *Proceedings of Passive and Active Measurment Workshop (PAM)*, 2000.

[7] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, June 1999.