Exploring the Dynamic Behaviour of Internet Using IP Options

Sneha Kumar Kasera, Jim Kurose and Don Towsley

January 1996
Tech Report Number 96-12
(Under Revision)

1 Introduction

Rapid fluctuations of queueing delays in small time intervals, random packet losses and changes in routes are some of the dynamic characteristics of the Internet. In this paper we study the queueing delays suffered by a packet, during its lifetime, at individual routers, from its source to destination, through experimentation and measurement in the Internet.

Understanding the queueing delay behaviour is very important for designing network policies including link level scheduling policies at hosts and routers. Let us see why this is true. There are several link level scheduling disciplines (for example [2] and [10]) in which a packet is given priority "downstream" after it has suffered delays "upstream." In other words, packets which get delayed at a certain node get priority at later nodes along the path over packets, which have not been delayed as much. Packet scheduling algorithms based on this premise will only improve performance when downstream scheduling can have an impact along a path. *The question of whether or not the occurrences of downstream compensation after suffering a "bottleneck" along a path are common or rare, is extremely important.* To answer this question we need to know the queueing delays experienced by a packet at individual routers along its path. This is the main purpose of our paper.

It is quite hard to theoretically analyze the packet delays at individual nodes without making simplifying assumptions. We have to model the source characteristics of different kinds of sources, the cross-traffic or interfering traffic at intermediate switches and also the routing decisions. Simple but impractical models may not tell us much. Instead, we believe that carrying out experiments on an actual complex network like the Internet would give us real figures for the packet queueing delays. It should be noted that finding the queueing delay distributions at individual routers it not our concern. We are concerned only about finding queueing delays suffered by each packet, at each router, along its path to the destination. We have built a tool using IP options, described in section 4, for our measurements. During the process of our experimentation and measurements we made several other interesting observations related to the Internet. We have listed some of these in the paper.

This paper has six sections. In the next section we look at the existing work on the subject and see that it is not enough to solve our problem. In section 3 we describe a general scheme for finding queueing delays at individual nodes with the help of timestamps. In section 4 we describe our experimental delay measurement tool for the Internet that uses IP *timestamp* options and IP *loose source record route* options. Here we also present a log of the hurdles we came across during our research and how we overcame those hurdles or found a way around them. In section 5 we present some of the empirical data we obtained and describe the filtering methodology we used to sift useful data. We also present our observations and discuss them. Finally in section 6 we conclude our paper.

2 Related Work

Most of the existing work on delay measurements in the Internet has to do with round trip or end-end delays. Mills [8] has studied round trip delays over the Internet as a function of packet length. Here the effectiveness of the TCP retransmission timeout algorithm has also been studied. Sanghi etal. [9] have observed a variability in round trip time which cannot be explained on the basis of cross traffic alone. They have also observed duplication and reordering of packets. Bolot [1] has attempted to analyze the end-to-end packet delay and loss behaviour in the Internet using a simple single server queueing model with two input streams, where one stream represents the probe traffic and the other represents the Internet traffic. Round trip times include total queueing delay suffered by a packet in its path from source to destination and back to the source. These round trip times do not provide any insight into the queueing delays at each router on the path.

Management Information Database(MIB)[3], which is used for network management, contains information, at each host and/or routers, on certain items like the number of datagrams received, number of routing failures etc.. These items of information could be exchanged by using the network management protocol called SNMP(Simple Network Management Protocol). One of the items(or variables) of the MIB is ifOutQLen which stores the number of packets in the outbound queue of a network interface. The number of packets is not sufficient to determine the queueing delays as they depend as well on the packet sizes, and the link capacity. Also this MIB variable is being removed from [4] the new MIB standard and will be available only during the transitional period.

To summarize, to the best of our knowledge, there has been no attempt to experimentally study the queueing delays encountered by a given packet as it passes through the various routers between its source and destination, at individual routers. No mechanisms have been proposed, to do this, so far. Hence there is a need for research to study the queueing behaviour at the routers in the Internet, especially for the purpose of understanding the efficacy of link level scheduling policies.

3 Queueing Delays Using Timestamps

In this section we present a scheme for computing the variable queueing delays at each router along a single path from a source to destination. Figure 1 shows an example of a network with sources, routers and destinations. We focus only on one path from a source to destination (example path from $S2$ to $D1$ in Figure 1) and describe how we compute the queueing delays at each router along the path using timestamps. We assume that we have some way of obtaining timestamps from each router. We describe a way to record timestamps in the Internet in the next section.

Consider a single path from source S to Destination D comprising of K hops. For the purpose of measuring queueing delays such a path could be

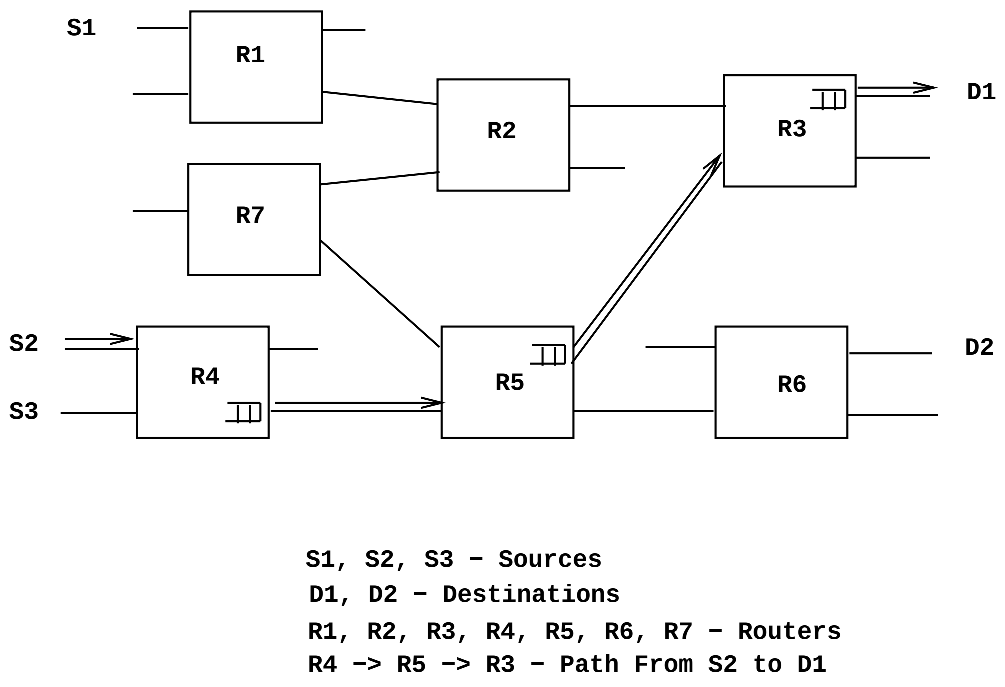

S1
R1
R2
R3
D1
R7
S2
R4
S3
R5
R6
D2
S1, S2, S3 − Sources
D1, D2 − Destinations
R1, R2, R3, R4, R5, R6, R7 − Routers
R4 −> R5 −> R3 − Path From S2 to D1

Figure 1: An Example of a Network

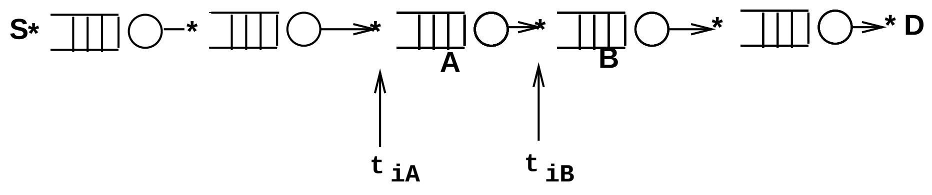


Figure 2: Path from Source to Destination

represented as a system of queues in tandem as shown in Figure 2. Let a sequence of N packets transit through the K hops along the path from S to D. Let each packet be timestamped at the instant it is received at a hop. Let t_{ij} be the timestamp on the i^{th} packet at the j^{th} node. Also let S_i be the time at which the i^{th} packet is transmitted from the source and D_i be the time at which it is received at the destination. The delay experienced by a packet from one hop to the next consists of four parts, transmission delay, processing delay, propagation delay and queueing delay. It is reasonable to assume that the queueing of a packet, at a router, takes place only at the output queue towards the packet's destination. Among the N packets, let packet i_A go from hop A to hop B without suffering any queueing delay. Then $t_{i_A,B} - t_{i_A,A}$ gives a measure of the sum of the transmission delay at hop A , processing delay at hops A and B and propagation delay in traversing the link between A and B. No packet can have a delay smaller than this value. For any other packet j_A, $t_{j_A,B} - t_{j_A,A}$ gives a measure of the three delays plus queueing delay, if any. Considering the other three delays to be constant between the two hops we can obtain the queueing delay, $q_{j,A}$ suffered by a packet at the output link of hop A towards B from the equations

$$q_{j,A} = (t_{j_A,B} - t_{j_A,A}) - (t_{i_A,B} - t_{i_A,A}) \tag{1}$$

Or,

$$q_{j,A} = (t_{j_A,B} - t_{i_A,B}) - (t_{j_A,A} - t_{i_A,A}) \tag{2}$$

An immediate question is how to determine that a packet did not suffer any queueing delay between hops A and B. We do this as follows. For sufficiently large N, for hops A and B, we find the minimum value of $t_{i_A,B} - t_{i_A,A}$ over all N packets. The packet which gives the minimum value is chosen as the base case or the one that has not suffered any queueing delay between hops A and B. Hence by choosing packets which do not suffer queueing delays between two hops (this could be different packets for different hop pairs) as the base cases we could determine the queueing delays suffered by other packets between the two hops. In this way we can determine the queueing delays suffered by a packet at individual routers along the path from its source to destination.

Another difficulty in comparing times obtained from different clocks for measurements is due to clock offsets and clock drifts. We define the clock offset of two clocks as the time difference between them. We define the clock drift between two clocks as change of clock offset over time. The clock offset between two clocks at two neighbouring hops can be non negligible(and actually can be very high) compared to the delay between the nodes. The comparison of timestamps from these clocks could lead to an incorrect value of the delay. The greatest advantage of the scheme we will use to compute queueing delays is that it is free from any problems arising due to clock offset. This is because if we use Equation 2 to determine the queueing delays, we only compare timestamps obtained at the same router. Unfortunately, our scheme is not free from clock drift problems. Our scheme depends upon the choice of base case(i.e., minimum delays) between two hops which are found by comparing the timestamps from different clocks. If the clocks of the two hops drift from each other then we may end up choosing an incorrect base case. To avoid this problem we make our measurements over short intervals of time such that clock drift has no appreciable effect.

Last, it is important to ensure that the frequency of transmission of test packets should not be high enough to affect the network traffic and hence the queueing delays.

4 Measurement Tool

In this section we describe our general research methodology which determines queueing delays from timestamps. We send probes at a certain rate from a source to a destination. We will shortly see that each probe consists of multiple packets. These probes get timestamped at the intermediate routers, arrive at the destination, and are returned back to the host. The returned packets contain the timestamps which are used to compute queueing delays.

In the Internet, probes move as IP datagrams between routers. To measure individual delays in the Internet we decided, or rather were forced, to use IP timestamp options. We start with a brief overview of IP options and specifically the IP timestamp option.

4.1 IP Options

IP Options are an optional part of an IP datagram header. They are included in an IP datagram header primarily for network control functions, such as testing or debugging. However, options processing is an integral part of the

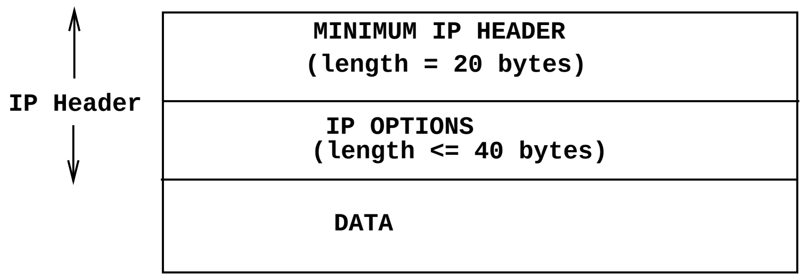
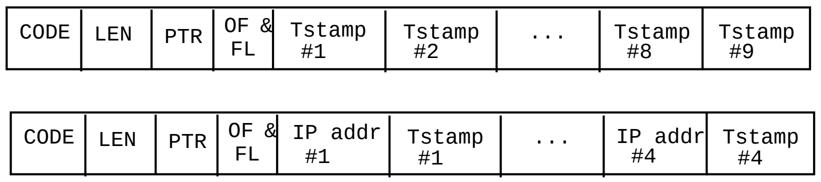


Figure 3: IP Datagram with Options



CODE = 68 (for Timestamp options)
Len = 40 bytes (or 36 bytes)
PTR = Offset to the next available entry
OF = Overflow
FL = Flags (set to 0,1 or 2)

Figure 4: IP Timestamp Option

IP protocol so all standard IP implementations must include it. The length of the IP Options field varies, depending on which options are selected. An IP datagram with Options is shown in Figure 3.

The IP options include provisions for timestamps. The timestamp option contains an initially empty list, and each gateway along the path from source to destination fills in one item in the list. Each item in the list could be a 32 bit timestamp or a combination of a 32 bit IP address and a 32 bit timestamp. Figure 4 shows the timestamp option field. Since the length of the timestamp option cannot exceed 40 bytes we can have at most nine timestamps or a combination of 4 IP addresses and timestamps. The IP addresses could be prespecified so that only those nodes whose IP address is specified in the option field fill in the timestamp. There are several problems in using IP timestamp options. We list the problems below and also present the mechanisms we developed in solving the problems or work around them.

Datagrams with IP options are handled differently at the routers from

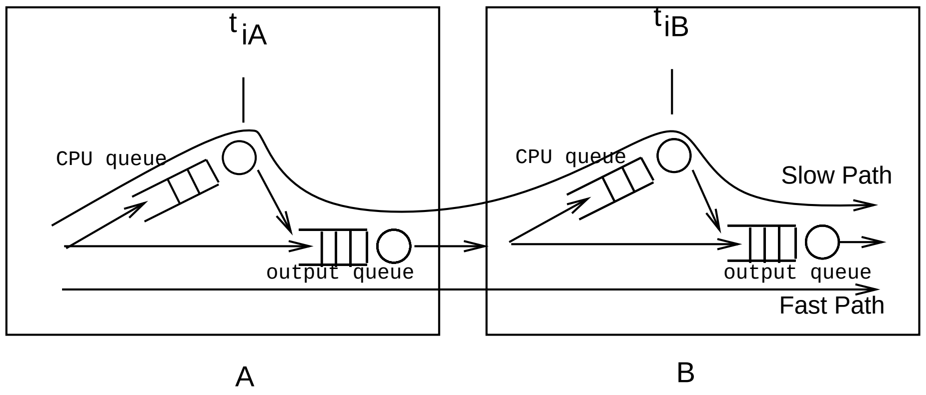


Figure 5: The Fast and Slow Paths

packets without options. Option processing is not done in the fast network hardware but in the host CPU of the router. Hence option datagrams may queue up at the host CPU and experience queueing delays which depend on the processing load at the host CPU. These delays would not be experienced by "normal" IP packets, without options. Figure 5 shows the *slow* path taken by a datagram with IP options. Contrast this with the *fast* path taken by an IP packet without options (also in Figure 2). Thus the timestamps obtained through IP options also include the queueing delay at the host CPU. To handle this problem we include two packets in each probe instead of a single packet. These two packets will be referred to as *packet pair* in rest of the paper. One of these two packets is sent with IP options and the other without IP options. They are sent back to back from the source with the hope that they experience the same network conditions. If the packet with options does not suffer any CPU queueing delays at the routers and the two packets in the pair see the same queueing delays in the network then they will arrive at the destination almost at the same time. Then the timestamps obtained from the packets with options would allow us to obtain a true measure of the network queueing (or output queuing) delays. To see whether the packets forming a packet pair arrived at the same time or not, at the destination, we need to compute the end-end delays for both the packets in the packet pair.

Another limitation of IP timestamp options is that only nine timestamps can be recorded in an IP datagram from its source to destination. Thus a single packet going more than nine hops would not be able to record timestamps at all the hops. We wish to probe longer paths for the purpose of recording timestamps at individual routers for measuring queueing delays. To handle this problem we send multiple packets, with an IP timestamp option, to cover the entire path once, in the following way. The first probe packet is sent with IP addresses of hops 1,2,3,4 prespecified in the option field. The second probe packet contains the IP address of hops 5,6,7,8 prespecified in the option field and so on. In this way the number of packets we send is one fourth of the length of the path. These packets are in addition to the packet pair mentioned above. Again, all these packets are sent out at the same time with the hope that they see, almost, the same network conditions. Such a scheme could lead to a problem if packets from the same set take different routes. For example if the packet covering hops 5,6,7 and 8 takes a less congested path compared to the packet covering hops 1,2,3 and 4 then we might end up choosing a incorrect base case when computing minimum delays across hops 4 and 5. This would make the queueing delays between nodes 4 and 5 appear higher than the true values. To overcome this, we need to throw away packets which are suspected to have taken different routes.

We use the *traceroute* utility to find the route and the IP addresses of all the hops in the route from our source to a destination. These addresses are needed for prespecification in IP timestamp option field.

4.2 ICMP Probes

So far we have discussed the use of IP options. We now discuss the transport mechanism for generating and receiving the probes. Probes could be transmitted as UDP packets to a destination IP address and port number. It is possible to let a user process, listening on that port on the destination machine, receive all the probes and postprocess them. This would be a serious limitation as we will need an account on all the destination hosts we wish to probe. In addition super user privileges are needed to look at IP options field in IP datagrams. One of the goals of our measurement tool is to be able to probe paths to as many destinations as possible. To meet this goal we use ICMP packets instead of UDP packets. ICMP messages such as ICMP echo request could be sent from a source to a destination ICMP layer which then responds by sending back an ICMP echo reply. This does not require the presence of a user process on the destination. The ICMP messages are encapsulated in IP datagrams. The ICMP echo reply is received at the source as an IP packet that contains the entire IP header of the original datagram, including the timestamps gathered on the forward route. Such a scheme has been used very successfully in the *ping* program.

UDP packets sent to a destination port on which no one is listening can lead to the generation of ICMP error messages from the destination host to the source. These messages also carry the entire IP header. This scheme is being used in the *traceroute* program. We use this scheme to determine the IP addresses of the outgoing IP interfaces of routers as described in the next section.

4.3 Unexpected Problems and Solutions

The problems stated in the previous sections indicate the limitations of the timestamping mechanism and the use of IP Options. We now present a log of the unexpected problems that we encountered in our research.

- We found that many hosts in their ICMP replies to probe packets, ICMP echo requests or UDP packets sent to an unused port, strip off the IP options from the IP datagram header. This was unexpected because the ICMP replies are supposed to contain the entire IP header according to [13]. Fortunately, we found that the routers do not strip off IP options so we decided to send probes to routers, and not to end hosts, as destinations.

- To compute end-end delay for packets without options, these packets must be be timestamped at the destination(end-end delays are required in deciding which packet pairs to use in drawing conclusions). ICMP timestamp messages could be used for this purpose. The source could send an ICMP timestamp request and expect a timestamp reply, from the destination, which contains the time at which the request packet was received at the destination. We found however that the routers do not respond to ICMP timestamp messages. Hence routers cannot be used as destinations. We also found that most of the end hosts respond to the ICMP timestamp requests. In the previous item we saw that end hosts cannot be used as destinations for probes gathering timestamp data because many of them strip off IP options. This implies that no single router or host could be used as destination for both IP timestamp probes and ICMP echo request packets.

Note that ICMP timestamps are different from the IP timestamp options. In the first case we get timestamps only at the end points at the ICMP protocol layer. In the latter case we can record timestamps at the intermediate nodes at the IP protocol layer.

- We used the *traceroute* utility to determine the route from a source to destination on the Internet. We wanted to fill in the IP addresses of the hops, obtained by using *traceroute*, in the IP timestamp option field to get the timestamps. The expectation was that a hop on seeing one of its IP addresses (a router can have multiple IP interfaces and hence multiple IP addresses) in the option field would put the time at which the packet was received at that hop. This did not happen but when we used the IP addresses obtained by using the *ping* utility with -R, record route option, we found that the hops were filling up the timestamps. The possible explanation is that *ping* records the IP addresses of the outgoing IP interfaces towards a destination whereas *traceroute* returns the addresses of the incoming interfaces of the routers. It it possible that the routers are timestamping a packet only on seeing the IP address of its outgoing interface towards the destination. Unfortunately due to limitations on the size of the IP option field, *ping* records only nine IP addresses.

To overcome the problems mentioned in the first two items we decided to use two destinations instead of one. All our packets which record timestamps using IP options are send to the last router of the path to an end host. All the ICMP timestamp requests are sent to the end host. These packets, form a single set of probe packets, which are sent back to back with minimal delay between them.

For finding the outgoing interface IP addresses of any intermediate router, we first determine the incoming interface addresses using *traceroute*. Then we used the *loose source record route* IP option in the following way. Refer to Figure 6. Source routing allows the sender to dictate the path to be taken to a destination. This can be done by filling the source routing option field in IP datagram headers with IP addresses of intermediate routers. Loose source routing allows multiple(unspecified) hops to exit between successive IP addresses specified in the option field. The *loose source record route*(LSSR) option requires that the routers along the path to overwrite items in the prespecified address list with the IP address of the outgoing interface towards

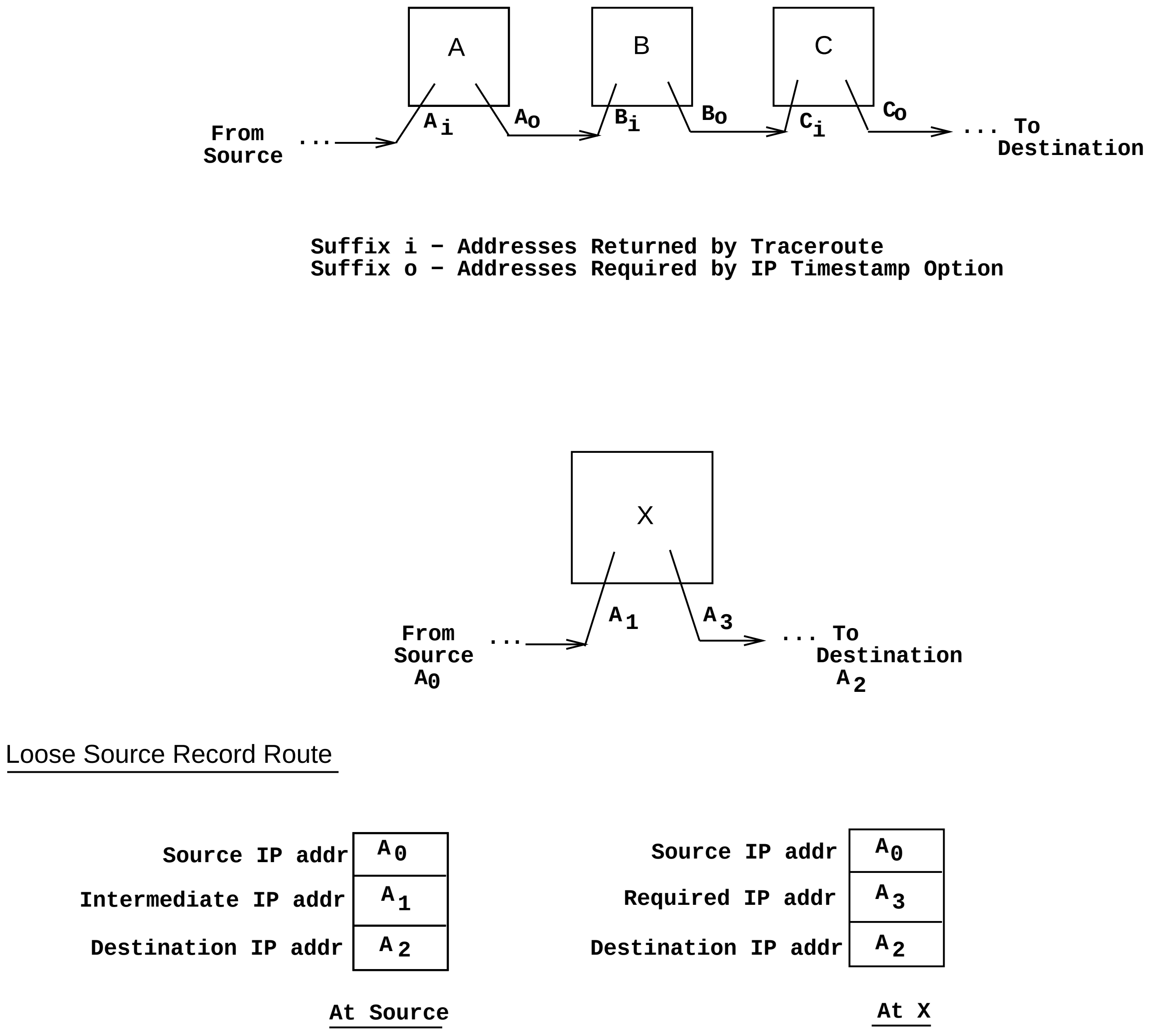


Figure 6: Determining IP Address of Outgoing Interface

the destination. Now suppose we wish to find the outgoing IP address(A_3) of the router X whose incoming IP address (for packets from source A_0), as determined by *traceroute*, is A_1, towards the destination, whose IP address is A_2. We prespecify the address A_1 in the LSRR option field of an IP packet sent to $A2$. The packet is routed to X through multiple hops and when it arrives at X the address A_1 in the option field is replaced by the address A_3. When the packet is received at the destination (or echoed back to the source as in our case) it contains the required outgoing interface's IP address. Repeating this process for all the intermediate routers we determine the necessary IP addresses which are to be prespecified in the IP timestamp option field to record timestamps.

4.4 Summarizing the Steps

We now present a summary of the steps in our measurement process. These are based on the above discussion. Refer to Figure 7.

- Choose an end host. Make sure that it responds to both ICMP timestamps packets with and without options.
- Find the route to the end host using *traceroute*.
- Use the IP addresses obtained from *traceroute* and LSRR to find the addresses of outgoing interfaces of the routers.
- Send multiple ICMP echo request probes with IP timestamp options to the last router before the end host. The IP addresses obtained in the previous step are used, four at a time, to prespecify IP addresses in the timestamp option field. The number of these requests sent depends on the number of hops in the path. A sixteen hop path would require four such echo requests to cover the entire path, as shown in Figure 7.
- Two ICMP timestamp requests are sent, one with and the other without options, to the end host. Both the packets on return contain ICMP timestamps which are used to compute end-end delay. It is to be noted that all probe packets are also timestamped at the source. The packet with IP options is processed in the "slow" path at each and every router from source to destination, even if no timestamps are actually recorded.

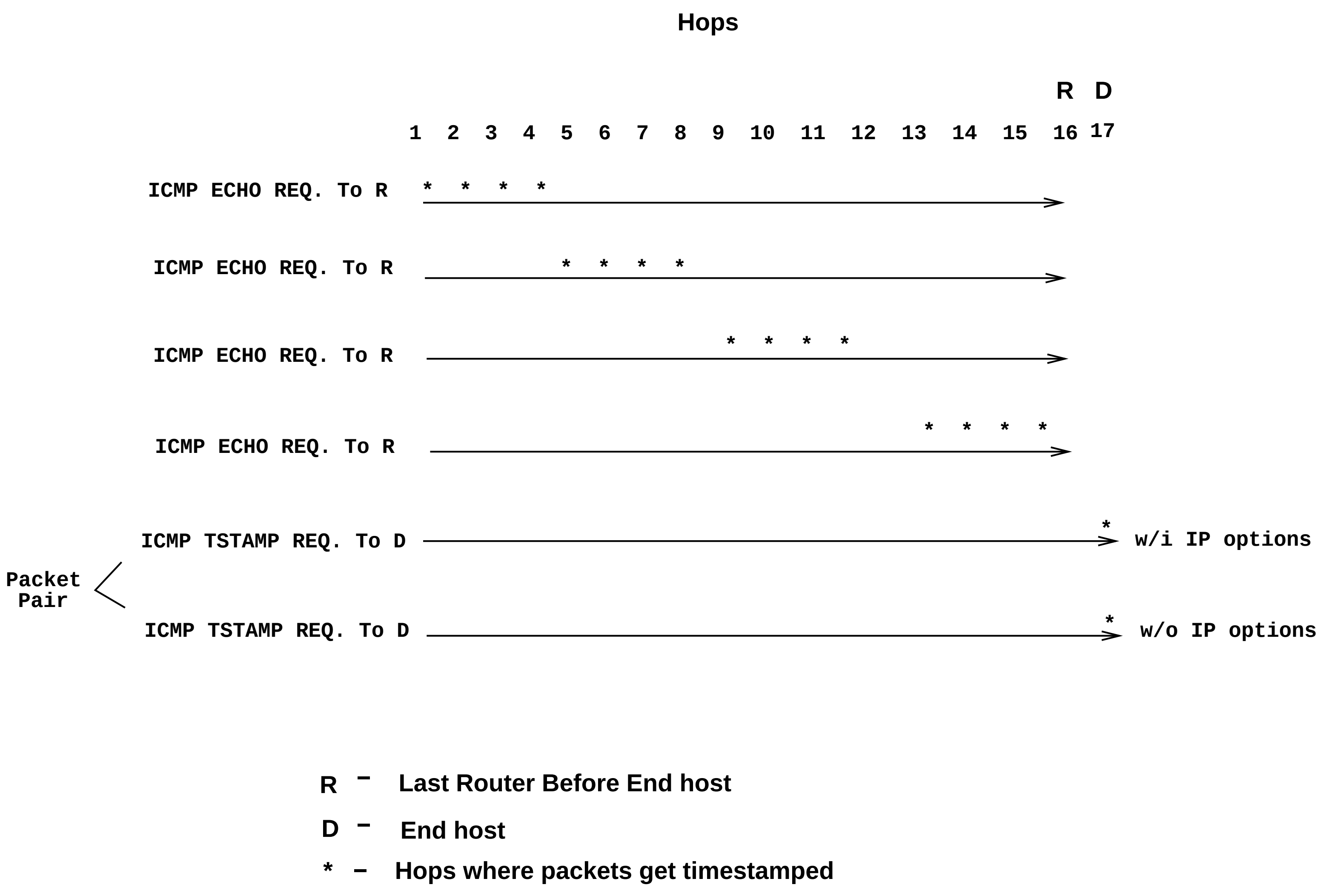


Figure 7: Probe Packets

We end this section with a brief discussion on the choice of time interval between successive probes, where each probe is comprised of multiple packets(all packets belonging to a single probe are sent out at the same time). We do not wish to affect the network with the traffic generated by the probes. So we cannot let the time interval between successive probes to be too small. If our goal was to find the queueing delay distribution at a node, probing the node frequently would have been a good idea. Our goal is to find the queueing delays experienced by a packet at individual nodes. Hence if we choose a large time interval between probes it should not affect our results. Unfortunately this is not true because of the clock drift observed in the Internet. We wish to collect sample data in a time period during which the clock drift is negligible. We have to try to get enough data in such a time period. Thus we have to choose the time interval between successive probes such that we do not change the network conditions by our probe traffic and at the same time are able to collect enough data without any clock drift. Another important point is that periodic transmission of probes could capture some periodic malfunctioning occurring in the Internet at the routers(e.g., periodic spikes in end-end delay were observed in [9]). This is undesirable to our experimentation. So, keeping the above in mind, we chose to send our probes(after some trial runs) with time intervals uniformly distributed between 100 and 500ms. We observed that at this rate, we could send about 450 probe sets, with each probe set having multiple packets, before noticing any appreciable clock drift.

5 Empirical Data

We ran several data collection experiments from our laboratory by sending probes to host and routers in United States and in Europe. The collected data comprised of timestamps at the routers and at the source and destinations. We examined one specific route in detail as an example. This route links trantor.cs.umass.edu at UMass, Amherst to cs.ucl.ac.uk in UCL, London. The route is comprised of nineteen routers before the end host. For collecting timestamps we ignored the first three routers. We will see from the data, these routers did not contribute significantly to the queueing delay. The data presented in this section for this route is from an experiment in which 40000 probes (comprising of 40000 packet pairs and 160000 packets for collecting

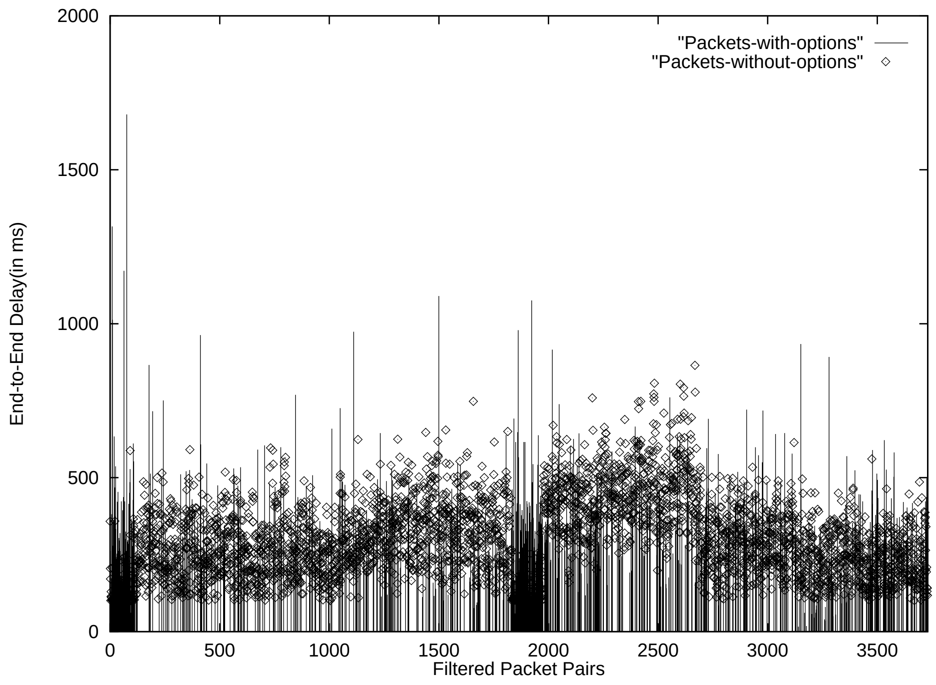


Figure 8: Filtered Data at Level I with Threshold = 100ms

timestamps, at intermediate routers) were sent. The time interval between successive probes was uniformly distributed between 100 and 500ms.

In this section we first present our two level filtering methodology which allows us to focus on relevant data. We present some plots of end-end queueing delays of filtered data and also the table of queueing delays at routers, measured from timestamps, from probes to UCL, London. We then present our observations based on our experimental data and discuss them.

5.1 Filtering Methodology

As mentioned before, the purpose of our study is to see whether or not there are multiple "bottlenecks," or points of high queueing delays, along the path of a packet. Thus it is enough to study the individual queueing delays of those packets which suffer a high end-end queueing delay. Packets that do not experience any substantial end-end queueing delay do not experience any bottlenecks and hence do not contribute to our study. Therefore, it is

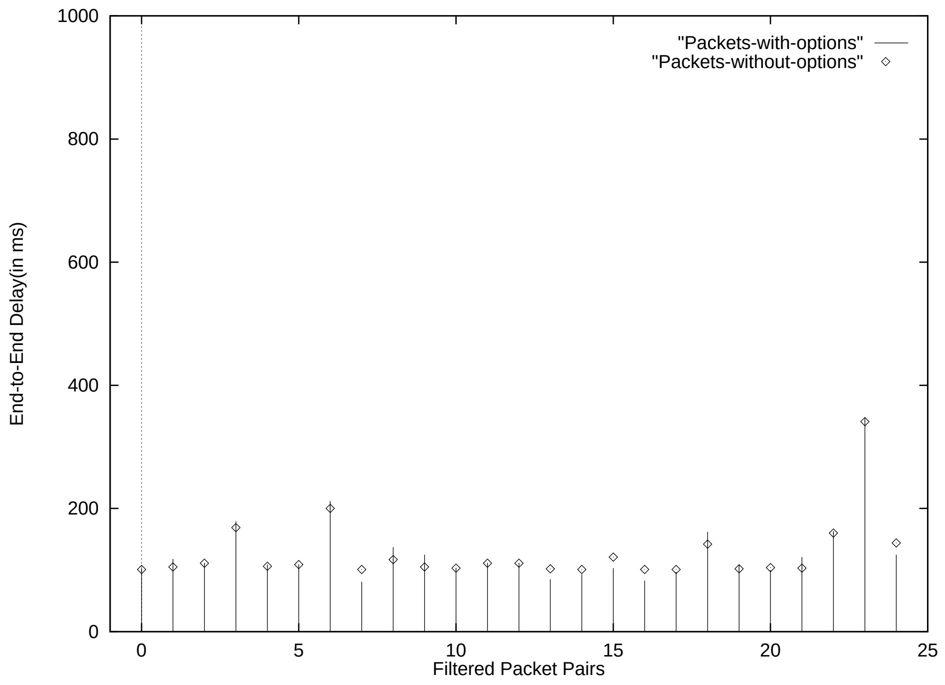


Figure 9: Filtered Data at Level II

a good idea to filter out these packets. Towards this goal we have developed a simple filtering methodology to sift out those packets which suffer high end-end queueing delays. To get the end-end queueing delay we look at the timestamps obtained from the source and the end host only. Comparing the timestamps, using the scheme described in section 3 we find the end-end queueing delays for packets from a packet pair(i.e. the last two packets shown in Figure 7), with and without options. We now select a threshold for queueing delay and remove all of the packets, without options, whose queueing delay is below this threshold. We also remove the corresponding packets(i.e. all other packets in the probe set) with options from trantor.cs.umass.edu to cs.ucl.ac.uk. Figure 8 shows the filtered data from 40000 probes to UCL, London, for a threshold delay of 100ms. The impulses show delays of packets with options and the points show delays of packets without options.

We are interested in output queueing delays and not the CPU queueing delay at the routers. Hence we would like to study timestamp data from only those probes for which the end-end queueing delays are same, or almost the same within a specified value of the difference,for both the packets in the packet pair. Thus we need to remove all of the data from those probes for which the this is not true. This is our second level of filtering. In addition we remove those probes for which the timestamp data is incomplete(some of the timestamps might be missing due to lost options packets). As can be seen in Figure 9 the second level of filtering in the UCL example would leave us with only 25 packet pairs. Here, the end-end queueing delays of packets with options and without options differ by at most 20ms.

The data obtained after the second level of filtering is used to study the presence of "bottlenecks." It is very important to know whether this data sample, is a true representation of the population. In our case, we assume that the population is comprised of all the measured network queueing delays greater than a threshold. Hence the population here is the data obtained after first level of filtering. In other words, we ask whether the data obtained after second level of filtering is a good representation of data obtained from only the first level of filtering.

For this, we use the Kolmogorov-Smirnov(K-S) test, [7] and [6], to test our null hypothesis, H0 defined as follows.

H0: A given sample of n observations, end-end queueing delays of probes after second level of filtering, is from the distribution function, $F_e(x)$, of all the observed end-end queueing delays greater than a certain threshold (for packets without options).

The K-S test tells us whether we have a good enough reason to reject the hypothesis or not. Note that failure to reject does not mean acceptance. This test should be regarded as a systematic approach for detecting fairly gross differences.

The K-S test is applicable to small sample size(compared to some other tests like the Chi-square test which is valid only in the asymptotic sense). As observed from the UMass-UCL trace we expect our sample sizes to be small. We apply the original form of the K-S test where all the parameters of the hypothesized distribution, $F_e(x)$, are known and the distribution is continuous i.e. the parameters are not being estimated from the sample. In our case we expect the delay distribution to be continuous for a sufficiently large population. To find the hypothesized distribution we remove the sample data from the population. The K-S test is based on the observation that the difference between observed CDF(Cumulative Distribution Function) $F_o(x)$ of the sample and the CDF $F_e(x)$ is small. It defines K^+ and K^- as

$$K^+ = \sqrt{n}\max_x[F_o(x) - F_e(x)]$$

$$K^- = \sqrt{n}\max_x[F_e(x) - F_o(x)]$$

$$K = \max(K^+, K^-)$$

A large value of K indicates a poor fit. The test rejects the null hypothesis H0 if K exceeds some constant $d_{n,1-\alpha}$, determined from the table available for the K-S test, where α is the specified level of the test, which is a measure of incorrectly rejecting a true hypothesis.

We now apply the K-S test to our data from UCL. The sample size is 25. The CDFs $F_o(x)$ and $F_e(x)$ are shown in Figure 10. The CDFs are quite far apart and hence the K-S test should reject H0. The value of K was found to be 4.027625. The value of K was more than all the values in the table for the K-S test for $n = 25$, even for $\alpha = 0.01$. This shows that if we reject H0 we have negligible probability of being wrong.

The data from UCL is the only large set of data we have so far. We need to collect large amounts of data from UCL and other end hosts and apply the K-S test to see whether there is any consistency in the rejection of the null hypothesis. With a larger data set we would have a bigger sample size and also the CDF $F_e(x)$ is likely to be more accurate. We are still investigating the applicability of various hypothesis tests to our data.

5.2 Observations and Discussions

We now look at the sifted UCL data. We are interested in the queueing delays of the probes, at each router along the path from trantor.cs.umass.edu to cs.ucl.ac.uk. The queueing delays were determined from the timestamps using the scheme described in section 3. Table 1 shows the cumulative queueing delays at each router for the 25 probes which survived filtering at two levels. Each row represents the cumulative queueing delays experienced by a packet at each router. Each column represents a router, except for the last one which represents the end host. The number in row i and column j is the cumulative queueing delay experienced by the packet i until router j. This number also includes the CPU queueing delay at router j, if any. It can be

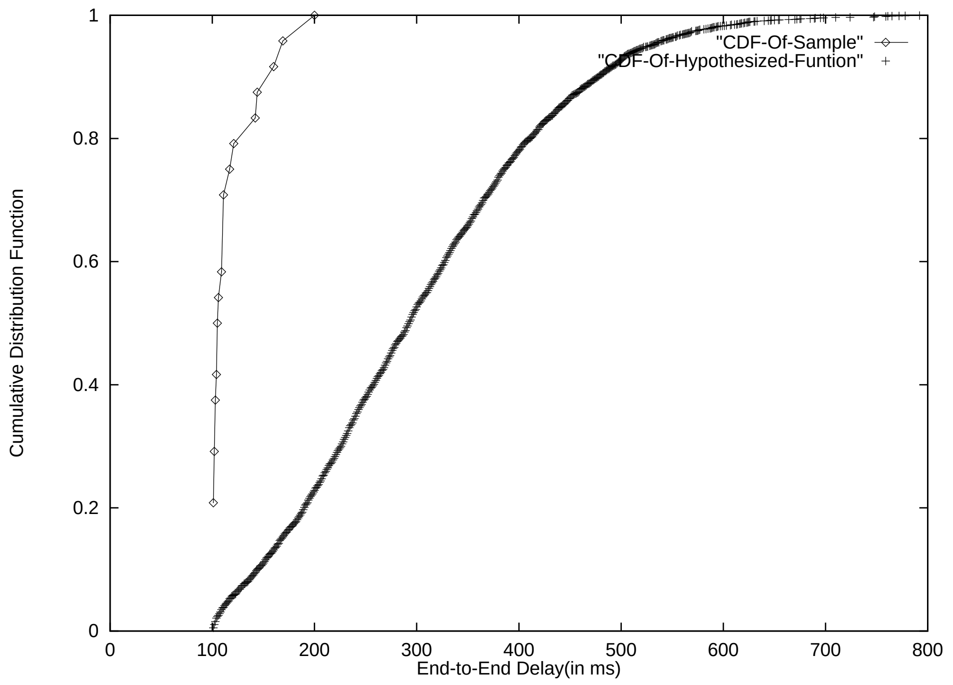


Figure 10: CDFs of the Observed and the Hypothesized Distributions

seen that for majority of the packets (14/25) large queueing delays ($\geq 40ms$) were experienced at more than one router along each packet's path. Another interesting observation is that there is always a large delay between hops 10 and 11. Therefore, if a packet 3 high queueing delay at any link other than the one between hops 10 and 11, then it will see multiple "bottlenecks." While running experiments with other hosts and routes we found high queueing delays at multiple routers. Can we conclusively conclude that there are multiple bottleneck nodes in the fast path of a packet at least for the sample data ? Actually we still cannot. The reason is as follows.

It was observed that packets with options experience very high delays with a large variation. Observing packet pairs we see that packets with options usually suffer higher queueing delays than corresponding packets without options, even when the packets without options do not suffer any significant delays . This means that the extra CPU queueing delay seen by many packets with options is significantly large. Hence even if packets in a packet pair of a probe arrive together we cannot guarantee that they passed through the individual routers as pair, i.e., in lock step. This is because the packet with option, in the packet pair, on experiencing high queueing delay at a router's CPU queue might get separated from the packet without option which does not suffer any CPU queue delays. Therefore, both the packets in a packet pair might see different network conditions and get delayed at different queues and yet experience the same overall end-to-end delays.

In addition to our observations on queueing delays at routers we made several other observations.

- We found that, in the Internet, packet loss seems to be much more dominating than the spikes in the end-end delay. The packet loss behaviour depends on the time of the day. For 40000 probes from trantor.cs.umass.edu to cs.ucl.at.uk at around 10:30 ET we found that 10% packets(without options)had end-end queueing delays greater than 100ms. Whereas the packet loss rate was 50%. For 3000 probes to austria.ac.at, at 11:00 ET on a different day, we found that the about 7% packets experienced network delays greater than 100ms whereas the packet loss rate was about 43%. Some of the losses happen at the time the spikes are observed but even otherwise random packet losses have been observed as in [1].

- Packet loss rate was found to be lower for packets with IP options

H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12	H13	H14	H15	H16	End
2	42	42	46	46	46	50	50	54	54	124	124	128	128	128	132	132
6	6	10	10	34	34	34	38	42	42	120	132	132	132	136	144	144
7	7	11	11	15	19	23	27	27	31	107	111	119	119	123	127	127
9	9	9	13	101	105	109	109	109	113	183	183	183	195	199	199	203
5	53	53	57	57	57	61	61	65	65	133	133	137	137	137	141	141
5	9	9	13	17	25	25	29	37	45	121	125	125	125	129	129	137
1	5	5	5	87	87	87	91	91	151	231	231	231	235	235	235	239
2	2	6	14	14	18	18	22	22	22	98	106	106	110	114	114	118
1	5	9	13	13	53	57	57	61	65	149	149	157	157	161	161	165
5	49	63	67	67	71	79	79	83	83	159	159	163	163	163	163	167
46	50	50	54	54	54	58	62	62	62	122	126	126	130	134	134	138
12	12	20	20	28	28	32	30	34	34	116	116	120	124	128	132	140
17	17	21	21	21	33	45	49	49	53	133	137	137	137	145	145	145
3	3	7	7	11	11	11	67	71	71	143	147	151	155	159	159	159
8	12	12	16	68	82	90	90	94	94	162	166	166	166	170	170	174
6	46	46	46	50	58	66	66	70	70	142	146	146	146	150	150	154
4	8	8	12	16	16	16	20	24	28	100	100	108	108	108	112	112
1	5	5	9	13	17	17	21	25	25	105	105	105	113	113	113	121
79	79	79	83	87	87	91	99	103	103	175	179	183	183	183	187	187
12	12	12	16	20	20	20	24	28	32	112	116	124	124	128	132	132
2	2	2	6	6	62	62	62	66	66	138	138	138	138	142	142	142
4	4	4	4	8	8	12	12	64	64	132	132	136	140	140	140	148
3	7	7	11	11	31	31	35	47	47	155	155	163	163	167	171	179
4	4	8	12	20	20	20	24	32	32	112	308	334	334	338	342	368
1	61	65	65	69	73	73	77	77	77	145	145	145	149	149	153	153

Table 1: Cumulative Queueing Delays(in ms)

compared to packets without IP options for many destinations in Europe (for e.g. sophia.inria.fr , cs.ucl.ac.uk and tuwien.ac.at). For destinations within United States (e.g. gregorio.stanford.edu and alpha.xerox.com) this was not found to be true. In fact the reverse was true,i.e. packets without options suffered a slightly lower loss rates than packets with options. One plausible explanation is that for routers in Europe the queue build up is much higher(higher end-end queueing delays have been observed for destinations in Europe) and hence more packets are lost at the output queues. By including options in the packets we force it go through the CPU queues at the routers. It is possible that the processing routine at the CPU monitors the output queue before sending out packets to it. This leads to an additional buffering mechanism at each router for packets with options and hence they see less losses. For packets without options in the fast path this mechanism is not there and the packets are switched by the router to the appropriate output queue which might not have empty buffer space to accommodate the packets in the queue. These losses are not observed for destinations within the United States because there is less output queue build up. Hence less packets are lost on the fast path.

- Our measurement tool could be used to find changes in route from a source to destination. When we send packets with IP timestamp options we prespecify the IP addresses of the routers that we want to timestamp our packets. If the route changes the prespecified IP addresses will be incorrect for some of the routers along the path and hence we should see zeroes in place of timestamps at these IP addresses. We are not able to capture all the route changes due to loss of test packets and due to reasons explained in section 4.1.

6 Conclusions

We understand that there is a need for determining the queueing delays at individual routers along the path of a packet in real networks. This knowledge will help us in studying the efficacies of certain link level scheduling disciplines. Round trip time measurements do not capture the behaviour of individual nodes. Timestamps could be used to determine the queueing delays at individual nodes. In the Internet IP Timestamps could be used. Routers exhibit partial behaviour in handling packets without options compared to packets with options. Due to high and variable delays seen by packets with IP options because of CPU queueing at each router we have been unable to compute the queueing delays in the output queue accurately at each router and thus have been unable to answer the question, with conviction, whether or not there are multiple "bottlenecks" along the path of a packet.

To overcome the differences in packet delays with and without options we need to implement IP Timestamp options along the fast path or at least need to design the routers so that they take only a small and constant amount of extra time(compared to packet without options) for IP option handling, instead of letting them go through the CPU queue. We feel that such a step should be taken to allow researchers to study the dynamics of the Internet,including individual queueing delays, queueing delay distributions at each router, route changes, packet losses, much clearly, resulting in a better network design.

Our research is useful to any scenario where data is transmitted with IP options. For example if source routing is used in IP options [5] then packets would be sent out with IP source route options and these packets would see the delays and bottlenecks, at the routers,of the kind we have seen in our study.

Finally, in our research, we have identified several problems and inherent limitations of experimentation and measurements in the Internet. Our research should guide the network implementors and designers in providing the necessary hooks in protocols and implementations for exploring the dynamic behaviour in the Internet.

References

[1] J. Bolot, End-to-End Packet Delay and Loss Behaviour in the Internet. Proceeding of ACM SIGCOMM, 1993.

[2] D. Clark, S. Shenker and L. Zhang, *Supporting Real-Time Application in an Integrated Services Packet Network: Architecture and Mechanism.* Proceedings of ACM SIGCOMM, 1992.

[3] D. Comer, *Internetworking with TCP/IP , Volume 1, Principles, Protocols and Architecture.* Second Edition, Prentice Hall Inc., 1991.

[4] S. Feit, *SNMP - A guide to Network Management.* McGraw-Hill Inc., 1995.

[5] C. Graff, *IPv4 Option for Sender Directed Multi-Destination Delivery.* RFC 1770, March 1995.

[6] R. Jain, *The Art of Computer Systems Performance Analysis.* John Wiley & Sons Inc., 1991.

[7] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis.* McGraw-Hill Inc., 1991.

[8] D. Mills, *Internet Delay experiments.* RFC 889, December 1983.

[9] D. Sanghi, A.K. Agrawala, O. Gudmundsson, *Experimental Assessment of End-to-end Behaviour on Internet.* Proceedings of IEEE Infocom, 1993.

[10] H. Schulzrinne, J. Kurose and D. Towsley, *An Evaluation of Scheduling Mechanisms for Providing Best-Effort, Real-Time Communication in Wide-Area Networks.* Proceedings of IEEE Infocom , June 1994.

[11] H. Schulzrinne, *Reducing and Characterizing Packet Loss for High-Speed Computer Networks with Real-Time Services.* Ph.D. thesis, May 1993.

[12] RFC 791, *Internet Protocol.*Darpa Internet Program Protocol Specification, September 1981.

[13] RFC 792, *Internet Control Message Protocol.*Darpa Internet Program Protocol Specification, September 1981.