

# Analysis of Internet Delay Times

H.S. Martin, A.J. McGregor & J.G. Cleary

*Abstract*—A methodology for analysing delay times on the Internet is described. A work in progress, this technique distinguishes physical delays, queueing delays, and processing delays in server and client machines. Measurement is done passively on pre-existing TCP streams with a focus on HTTP traffic and server side delays. Results which break down the three components are presented for traffic measured in New Zealand. There is a discussion of the reliability and potential errors in the measurement and analysis process.

*Keywords*— passive measurement, network delays, server delays, HTTP, TCP

## I. INTRODUCTION

When a user clicks on a link on a web page there is often a significant delay before the newly fetched page is displayed on the screen. There are many potential causes for this delay including network congestion and load on the server or client machines.

This paper describes a technique for disentangling the different causes for the delays. The work starts from the base of passive network measurements taken on high bandwidth links. The traces obtained this way include all IP and TCP headers, although for security reasons, they do not contain packet contents. The measurement technology we are using provides time-stamps, accurate to within 1 microsecond of UTC, for all measurements. [1]

The technique starts by identifying a TCP connection between two machines (typically this is an HTTP flow and contains traffic in both directions). The stream is then analysed to identify the different message types (for example data, ACK, SYN and FIN) and the state of the TCP stacks at the time when the messages were sent. For some of the packets that are sent the TCP state is ambiguous (for example it can sometimes be difficult to tell if a timeout has occurred or if a packet has been lost). This initial analysis omits all data where the state of the TCP stack is ambiguous.

Further processing the unambiguous data allows identification of packets where the TCP stack immediately replied to a message (for example an ACK sent in reply to a SYN), those where processing had to occur (for example a data packet sent in reply to an ACK) and those where timeouts or packet loss occurred.

Using this data we can breakdown the delay into contributions from the following causes:

- physical latency caused by speed of light delays and router and switch delays;
- network delays caused by queueing and congestion in routers;
- processing delay on the server or client machines;

H.S. Martin, A.J. McGregor and J.G. Cleary, Department of Computer Science, University of Waikato, Hamilton, New Zealand. {hmartin,tonym,jcleary}@cs.waikato.ac.nz.

- timeouts;
- packet loss;
- TCP protocol delays.

An ability to reliably separate the different causes of delays is important to many Internet users and suppliers. A response to unsatisfactory performance will be quite different depending on the causes of the delays.

Results from measurements taken in New Zealand will be presented. These initial results indicate surprisingly high contributions from end-system processing delays even for heavily loaded trans-oceanic links between New Zealand and the United States.

This paper does not consider the delays arising from packet loss or TCP protocol delays although we intend to report on these in more detailed analyses in the near future.

The following sections describe work in progress to isolate and measure separately, each of these contributions to the delay.

Section II gives a description of the basic technique for extracting valid Round Trip Times.

Section III discusses methods for summarising these times.

Section IV then presents a coherent way of using these results to summarise server performance and results from some NZ servers.

The paper concludes with some overall conclusions and a discussion of where future work is needed.

## II. ROUND TRIP TIMES

The initial focus for the work is on HTTP sessions (which use TCP as their underlying protocol). The reason for this focus is that HTTP is the dominant source of traffic on many Internet links and focusing in this way enables a better understanding and checking of the data. However, our technique can, in principle, be used on any TCP stream and we are working on validating it for other applications.

At this point we have not validated estimates of processing delays on the client side of HTTP transactions (the situation is not symmetric between server and client sides). Thus results that require use of estimates of processing delays will only be for HTTP servers.

Fig. 1 shows how our measurements are taken. A passive measurement box snoops on a link and gathers information about all IP packets that flow past it. Information gathered includes the IP and TCP headers and an accurate time-stamp[1].

For a particular HTTP session all packets flowing between the client and the server can be seen and the timing is sufficiently accurate that the ordering within and between the two streams is accurately recorded. The measurement machine maybe placed at any point between the client or

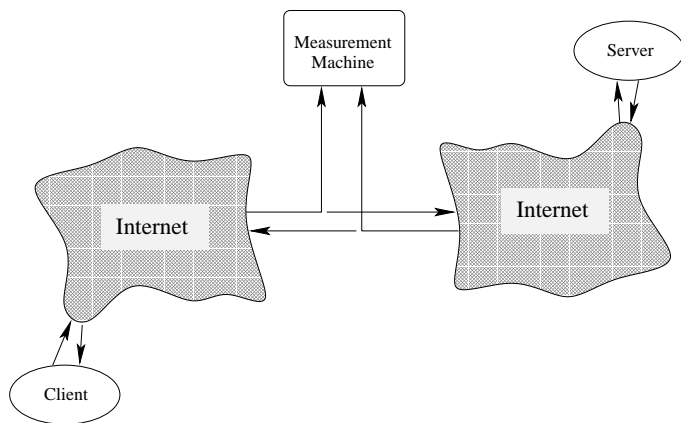


Figure 1. Setup

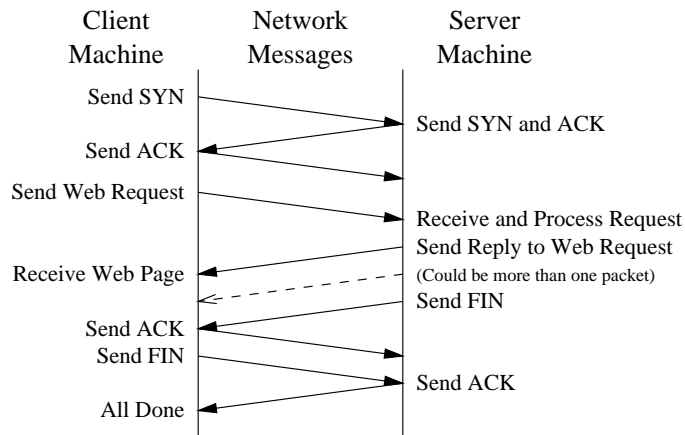


Figure 2. Simple HTTP session

the server, allowing for large variations in delays between servers and clients.

A simple HTTP session is shown in Fig. 2. The flow of packets in such a session is largely determined by the TCP protocol[2]. A typical interaction will have a packet sent by one host (for example a SYN or a data packet) and a reply is sent back (possibly an ACK or a data packet). Such interactions can be unambiguously paired because the sequence number of the original packet will equal that of the reply.

The simplest HTTP session is used later in computing and presenting estimates of delays. As in Fig. 2 there will be a single data packet sent in reply to the request and a FIN immediately following it. The majority of the time taken by a session is spent in traversing the network twice and host processing time at the point where the data is retrieved by the server. The majority of web requests are of exactly this form.

### A. SYN-ACK

The basis for our current analysis is Round Trip Times (RTT) from the measurement point to a host (client or server) and then back to the measurement machine. For such a RTT to be useful it is necessary to pair the outgoing

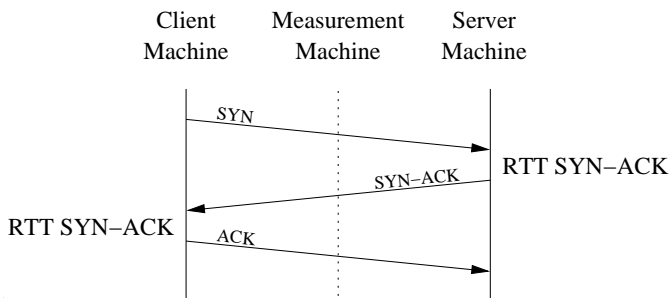


Figure 3. SYN-ACK

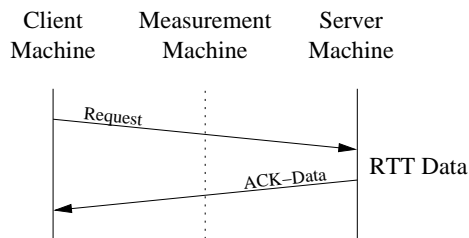


Figure 4. First Data Packet

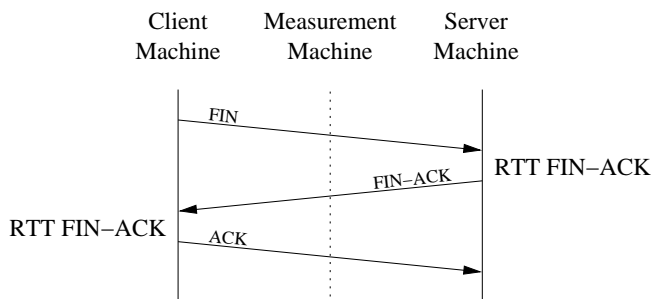


Figure 5. FIN-ACK

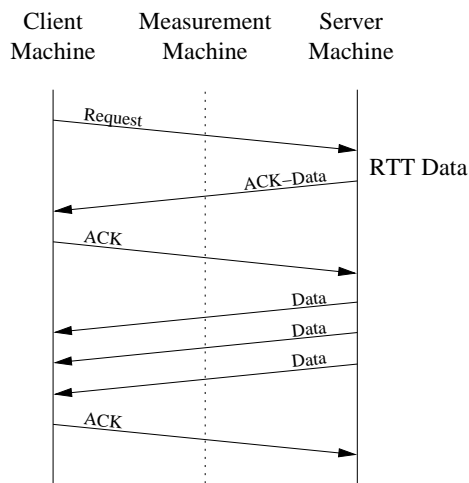


Figure 6. Sequence of Data Packets

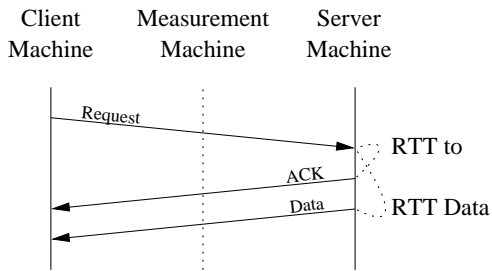


Figure 7. Sequence of Data Packets

message with the response that comes back. For example, this can be done at the start of a TCP session when a SYN packet causes an ACK packet in reply[2]. Thus the SYN and ACK can be unambiguously paired and the time difference between them used as a RTT. This situation is shown in Fig. 3. We use a number of different types of RTTs which we assigned to different classes. We will refer to this type of RTT as SYN-ACKs. Figs. 11 and 12, show example histograms of the SYN-ACK and *data* RTT times for two different hosts.

After the initial SYN/SYN-ACK exchange between the client and the server the SYN-ACK is Acknowledged by the client with an ACK. The RTT of this pair is associated with the client. One important attribute of the SYN-ACK RTTs is that they do not cause any processing on the host outside of the TCP stack.

### B. Data

After the initial exchange of SYN packets, a *request* is sent from the client to the server. The first reply packet to this will contain data that has had to be fetched (possibly from disk) by the server. The RTT, measured from the request to the reply, is classed as *data* and is used later to estimate server load.

Fig. 7 shows a more complex case that can arise as a result of a timeout. In this example, the request packet arrives at the server and the time for the user level code to respond is long enough that a timeout on the server is triggered. The timeout causes an ACK packet to be returned from the server. At some later time the reply to the request packet is sent. This leads to two RTTs. One is a new class that we call *timeout* between the request and the ACK. The length of this RTT is determined by internal stack processing time, the settings of the timeout intervals, network delays and physical latency[2]. A second RTT is also calculated between the request and the data which is classed, as before, as *data*.

The returned data may consist of many packets. This leads to a number of different possible classes of RTTs in addition to the *data* RTTs. An example is shown in Fig. 6. There are RTTs between the first returned data packet and its ACK and also between the ACK and the second returned data packet. For the purposes of analysing the causes of delays these RTTs are not so useful as it is difficult to unambiguously assign a cause for the delay. For example the ACK that is returned from the first data packet

might be a timeout from a TCP stack that is using delayed acknowledgements[3] or an immediate reply from a stack that is not. Similarly, in the case of the second data packet, the packet may have been already ready in the TCP stack buffers and so an immediate reply from within the stack occurred. Alternatively, the stack might have had to wait for the user level processing to write the data to the stack. Thus the RTT may contain an unknown component of user level processing on the server.

### C. FIN

At the end of the TCP session there is an exchange of FIN packets. The server first sends a FIN to which the client responds with an ACK. These give rise to a class of FIN-ACK RTTs as illustrated in Fig. 5. These should be very similar in distribution to the SYN-ACK RTTs as they involve purely stack processing.

There is a second FIN sent back from the client to the server and ACKed by the server leading to a class of ACK-FIN RTTs. Unfortunately, this part of the TCP protocol is, in practice, differently implemented by various stacks[4]. In many cases we see that the final FIN is never sent. Sometimes, if it is sent, it is not acknowledged. Also the time distributions that we get from the ACK-FIN differ markedly from the SYN-ACK and other stack only RTTs. Until we understand what is happening in these cases we have not used these RTTs in further processing.

### D. Example

Fig. 8 shows an example TCP session together with the assignment of RTT classes. The first line shows the two host involved in this session and the TCP ports on which the where communicating. For each of the following lines, packets are shown with a time value in seconds at the start. Following that is the direction the packet travelled and the number of bytes of data. The following two numbers give the sequence and acknowledgement numbers. Next is a list of flags set for this packet, which can be URG, ACK, PSH, RST, SYN, FIN[5]. Finally is a packet line number.

RTT times are shown, starting with the packet line to which they are paired and the host to which they are associated, ie  $< -$  means the left host, and  $- >$  means the right host. The RTT is given next, along with the two times from which it was calculated. Lastly the RTT is given a classification as described in section II above.

As well Fig. 9 shows this same session rendered as a diagram in the same form as those earlier.

Figs. 15 and 16 show histograms of the distributions of RTT times for the SYN-ACK and data times. We discuss the details of these histograms later.

### E. Packet Loss

Packets may be lost in transit through the Internet. What is more, they may be lost before or after transiting the measurement point. Thus, a packet may be seen by the measurement point but not by the final destination. If lost before the measurement point then neither the host nor the measurement point will see it. The problem is that

```

951432040.744383 --> 1 4183672438 0 syn 0
0 -> rtt: 0.000971, 951432040.744383 951432040.745354 syn ire
951432040.745354 <-- 1 3186806649 4183672439 syn ack 1
1 <- rtt: 0.000129, 951432040.745354 951432040.745483 syn ire too
951432040.745483 --> 0 4183672439 3186806650 ack 2
1 <- rtt: 0.007217, 951432040.745354 951432040.752571 first
951432040.752571 --> 89 4183672439 3186806650 psh ack 3
3 -> rtt: 0.019126, 951432040.752571 951432040.771697 ire-ws too
951432040.771697 <-- 0 3186806650 4183672528 ack 4
3 -> rtt: 0.094375, 951432040.752571 951432040.846946 data
951432040.846946 <-- 1460 3186806650 4183672528 ack 5
5 <- rtt: 0.000130, 951432040.846946 951432040.847076 ire-ws too
951432040.847076 --> 0 4183672528 3186808110 ack 6
6 -> rtt: 0.001155, 951432040.847076 951432040.848231
951432040.848231 <-- 1460 3186808110 4183672528 ack 7
951432040.850213 <-- 1176 3186809570 4183672528 psh ack 8
951432040.850405 <-- 124 3186810746 4183672528 psh ack 9
9 <- rtt: 0.002642, 951432040.850405 951432040.853047 ire-dd ire too
951432040.853047 --> 0 4183672528 3186810870 ack 10
10 -> rtt: 0.000583, 951432040.853047 951432040.853630 ackfin
951432040.853630 <-- 1 3186810870 4183672528 fin ack 11
11 <- rtt: 0.000098, 951432040.853630 951432040.853728 fin too
951432040.853728 --> 0 4183672528 3186810871 ack 12
11 <- rtt: 0.028388, 951432040.853630 951432040.882018 fin ackfin
951432040.882018 --> 1 4183672528 3186810871 fin ack 13
13 -> rtt: 0.000638, 951432040.882018 951432040.882656 fin too
951432040.882656 <-- 0 3186810871 4183672529 ack 14

```

Figure 8. sample trace

it is necessary to disentangle cases where such lost packets require retransmission from the normal case. This is significantly more complex and we are still working through the various cases that can occur. In the data given below, all RTTs where we are not sure if data loss occurred are ignored and not used in the results.

The data we have been collecting contains about 1% to 4% “duplicate” packets - that is, ones that we have observed to be re-transmitted. At these levels retransmission is probably having a significant effect which we intend to account for in future work. Fig. 10 shows the percentage of duplicate packets detected on various data sets that we have been using. On average half the lost packets will be dropped before the measurement point and half after it. So there will probably be as many lost packets that we have not detected as duplicates as there are recorded. Thus there are roughly 2% to 8% lost packets in our data. These lost packets could significantly effect the total delay time.

### III. DELAY ESTIMATES

Given a set of RTTs as described above we will separate these into orthogonal classes of delays. The analysis below is not complete as it omits the effects of timeouts and packet loss. However, it does give an initial estimate of some important components in the overall delay.

To summarise the results succinctly we consider the benchmark “simplest” TCP session as shown in Fig. 2. This contains two complete traversals of the network and a delay for the server to fetch the returned data. Two other delays occur in this simple session, these are between the first ACK received on the client side and the request being sent, and later the delay from the data reply to the FIN being sent. Both of these will include processing within and outside the TCP stack. However, preliminary examination

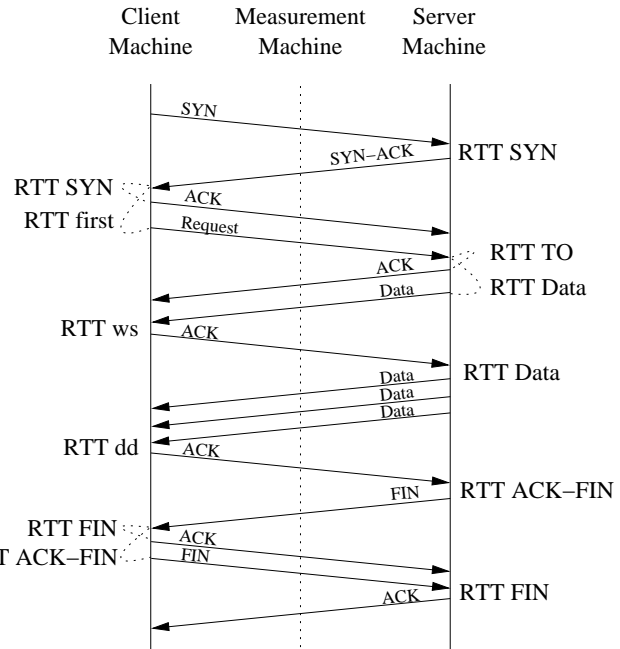


Figure 9. sample trace diagram

trace	NZIX	ISP-trace1	ISP-trace2
TCP packets	234567	268987	13843
duplicates	2316 (3.8%)	3477 (2.9%)	84 (1.7%)
date started	Feb99	Nov99	Dec99
length	10 min	1 hour 20 min	7 min
number of hosts	610	838	92
clients (>10 RTTs)	61	70	10
servers (>10 RTTs)	65	84	7

Figure 10. Trace details

of them shows that in most cases they are small compared with the other delays and we have ignored them for the moment.

#### A. Physical Network Delay

The first component that we extract from the RTTs is the physical network delay. This is composed of the time it takes photons or electrons to traverse the physical links as well as any fixed delays in routers and other computers along the way. We extract this by using the minimum time of all the well defined RTTs. We define the Physical Network Delay as the minimum time a message can take to traverse the network and estimate it from the minimum actually observed.

In this way we obtain estimates of the latency from the measurement point to both the servers and the clients. The results we give below are on a server by server basis. To compute the benchmark latency we first compute a weighted average of all the latencies to the clients. (We are making an assumption here that the usage of servers by different clients is equivalent). Second we compute the latency to the server as described above. The sum of these two values gives a total latency for traversing the whole network end to end. As described earlier, this value is then doubled

because the benchmark involves two complete traversals of the network. This gives us the final latency which is used in the results.

There are some issues about the accuracy of estimating latencies in this way. In the example SYN-ACK histograms of Figs. 11 and 12 the low end of the distribution has a sharp cutoff. This is good because the minimum will be a good measure even with only a small number of RTTs. Although there will always be a tendency to over estimate the minimum, the sharp cutoff means that the over estimation will be small. In cases where the distribution does not have a sharp lower cut off many more points will be needed to prevent this over estimation.

Taking a minimum is very sensitive to errors. A single point in error well below the minimum will completely alter the estimate. At this time we have visually checked the data to see if such cases arise. We hope to add automated checking later.

Another effect that is not allowed for in this analysis is alternate routes when traversing the network. If the route changes during the measurement period then the difference in latency between the two routes will be incorrectly seen as queueing delay. The data sets used here were captured over periods from ten minutes to a few hours. We expect most routes to be stable over these periods.

### B. Queueing Delay

The next component of the delay that we estimate is that due to queueing and congestion in the network between the host and the measurement point. Again we use the fact that SYN-ACK pairs involve little processing on the host. WE assume that the network delay is anything greater than the physical delay time. Thus, the queueing delay is estimated as the mean of the SYN-ACK RTTs less the physical delay.

A SYN-ACK does require some processing at the lowest levels of the host's TCP stack so effectively that part of the processing is counted as part of the network.

Again the queueing delay for the benchmark transaction is computed by adding the average network delay for a particular server to the average delay for the clients. This is then doubled to account for the two traversals of the network.

### C. Host Processing

The third component we estimate is user level processing on the host. To do this we look at RTTs which will require processing and which we know have not involved a timeout or some other non-processing delay. At the moment the only class of RTTs that unambiguously includes such user delays is the *data* class. The average of this RTT, less the average of the SYN-ACK, is used as the estimate of the host processing delay.

### D. Traces

In order to get valid values for the three different components it is necessary to have enough different values that statistical fluctuations are evened out. A SYN-ACK and a

*data* RTT will occur at once, for each host, in each TCP session, so this means that sufficient sessions need to be included for each server to get meaningful results. The situation for the clients is not so demanding as all clients are aggregated together so it is the number of TCP sessions over all clients that is important.

The need for sufficient sessions also limits the time resolution of the technique. If the delay parameters are changing on time scales significantly below the time needed to collect significant numbers of sessions then the changes will not be seen.

The main requirement of timing precision in the traces is that in a bidirectional link the two directions can be unambiguously aligned so that no replies are out of order with respect to their requesting messages. In practice this means that accuracies and reliability of a few milliseconds is sufficient. The total size of most of the delays that we are measuring are tens to hundreds of milliseconds, so again accuracies of a few milliseconds suffice. This means that we can use either hardware techniques or carefully calibrated software techniques[6].

## IV. RESULTS

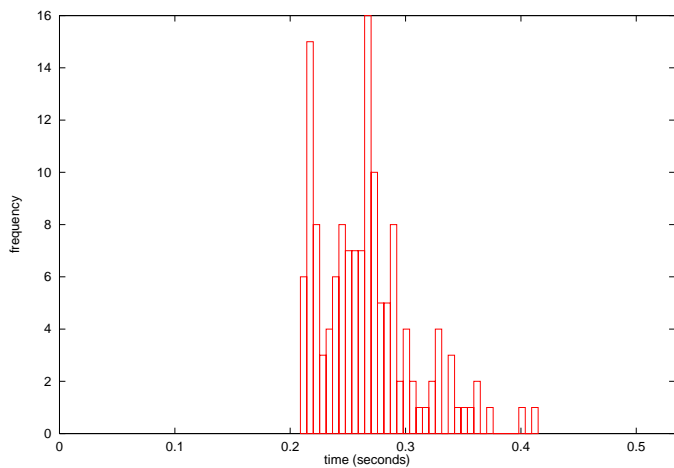
### A. RTTs

Figs. 11, 12 and 13 show histograms of the *SYN-ACK* and *data* RTTs respectively for three different servers. The first two are representative measurements taken from the large number of servers summarised in Figs. 15. The third is an anomalous case from the same data set where the server delay has been computed as being negative.

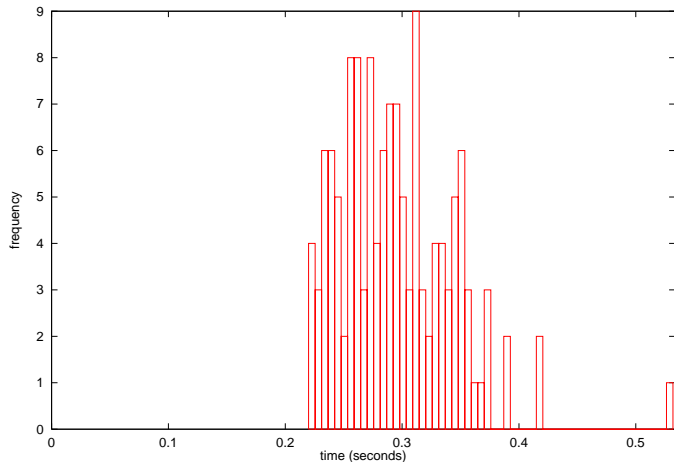
The histograms as shown have distributions that are similar to the majority of the cases that we have examined. The SYN-ACK times have a sharp clearly marked lower bound with the bulk of the peak immediately above that followed by a long tail. This indicates that the estimate of the latency taken from the lower bound of the SYN-ACK RTTs will be well defined and that the rest of the distribution is caused by variable network and stack delays. In both there is some indication of two peaks with a narrow lower one close to the minimum and a higher broader one just above that. It is not clear how to interpret these features. In Fig. 11 the bulk of the SYN-ACK distribution is spread from 100milli-seconds to 250 milliseconds, in Fig. 12 the spread is somewhat wider from 100 milli-seconds to 350 milli-seconds.

As would be expected the *data* histograms are displaced to right indicating substantial user processing delays. In Both Figs. 11 and 12 show data distributions displaced to the right of the SYN-ACK data and with a broader peak and longer tail.

The distributions in Fig. 13 are notable for having a small number of data points (15 in the SYN-ACK and 16 in the data). As well there is a single outlier point in the SYN-ACK distribution at just over 0.9 seconds. Apart from this single outlier the distributions are very similar with the data distribution displaced a little to the right of the SYN-ACK distribution as would be expected. The conclusion is that the incorrect negative server delay is the result

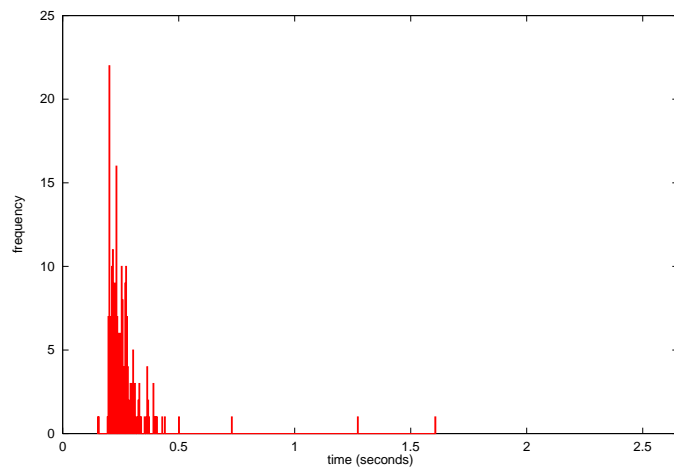


a) SYN-ACK

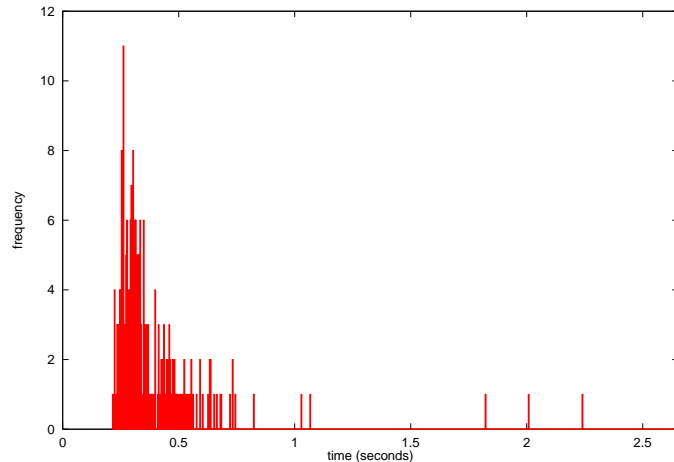


b) data

Figure 11. RTT for a server (128) from ISP-trace1



a) SYN-ACK



b) data

Figure 12. RTT for a server (63) from ISP-trace1

of statistical fluctuations amongst a small number of data points. A couple of other points in the triangle graphs are also slightly negative and manual checking of them shows that they are all similar to the case above with outliers in small SYN-ACK distributions causing a fluctuation below 0.

### B. Delays

Figs. 15, 16 and 14 show a summary of the delay values from a large collection of servers. Fig. 15 and Fig. 16 are taken from data collected at a local NZISP. Fig. 14 is taken from the New Zealand Internet Exchange. See Table 10 for details of the traces. All hosts where the total number of SYN-ACK RTTs and data RTTs both exceeded 10 are included in the results.

The results are displayed by taking each of the latency, network delay and server delay, as described above, and re-normalising them so that they sum to 100%. Thus the diagrams are a display of the relative proportions of the three forms of delay. So as not to lose the symmetry between the three values they are displayed in a triangular format. The vertices of the triangle are the points where one of the delays is equal to 100% and the other two are

0%.

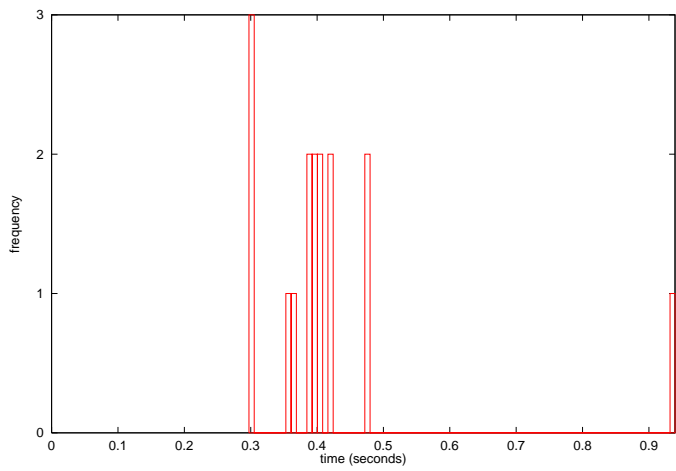
As can be seen in all three plots the results are clearly clustered in a group around the point where queueing delays and latency are the same and server delay is zero with a scatter of points toward the vertex where server delay dominates.

The interpretation is that, network delays of one kind or another are the dominant effect on overall delay. Although in some cases the effect of the server delay can be as great as the network effects.

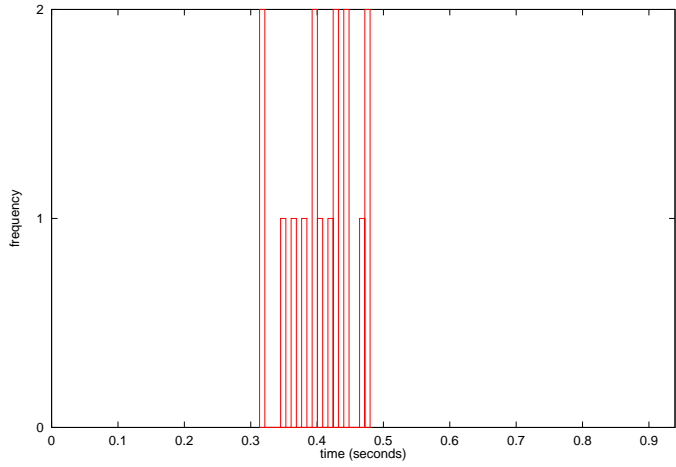
The shape of the grouping of points indicates a rough correlation between latency and queueing delay. This would be expected in a model where the number of queues and links is approximately proportional to the length of the route.

## V. CONCLUSION

We have explored the issues in analysing the cause of Internet delays from passive network traces. Initial results indicate that it is possible to get reliable estimates of different forms of delays occurring in HTTP connections. This in turn can provide explanations of perceived delays between clicking on a link and receiving a page displayed on



a) SYN-ACK



b) data

Figure 13. RTT for a server (146) from ISP-trace1

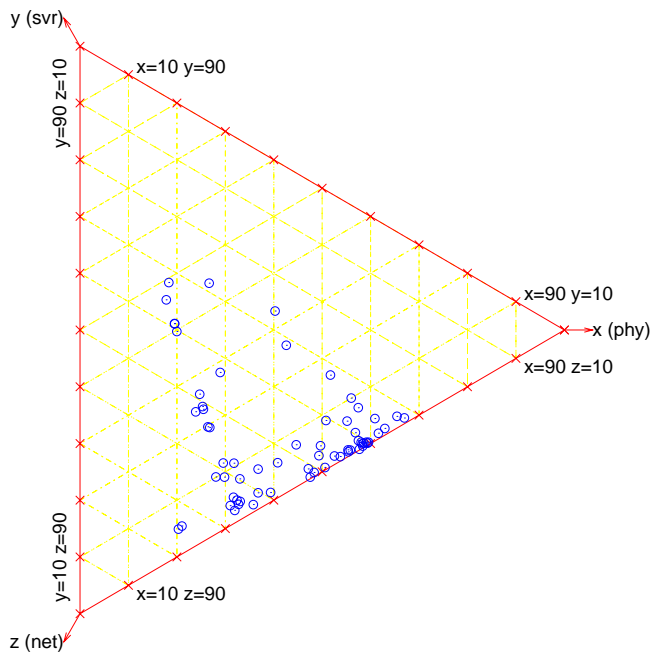


Figure 14. New Zealand Internet Exchange (NZIX)

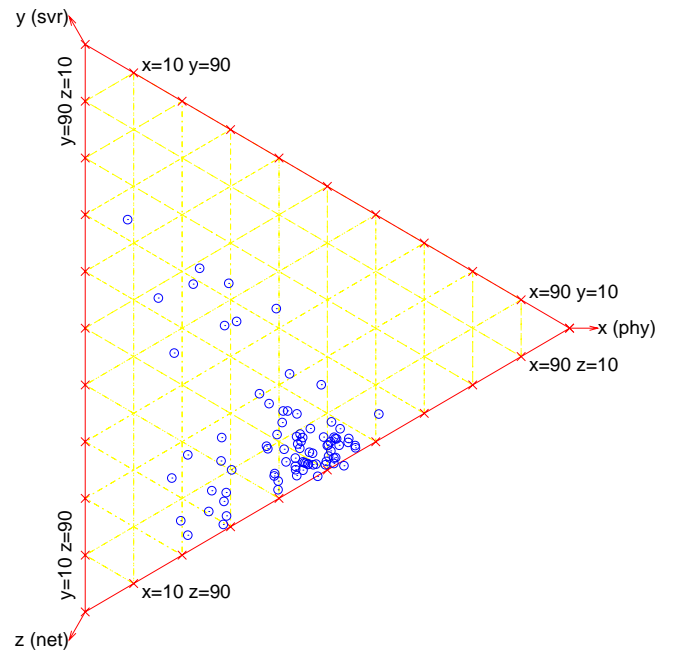


Figure 15. ISP web traffic - trace1

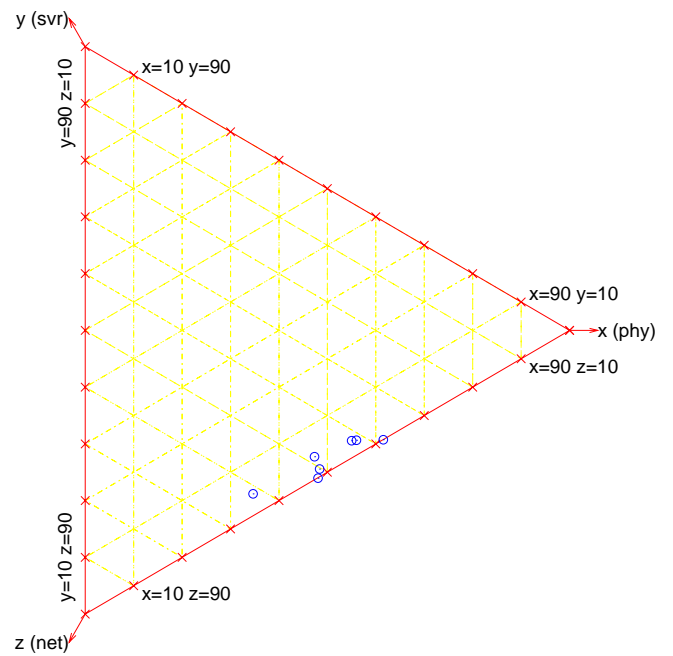


Figure 16. ISP web traffic - trace2

the screen. Having, these different measures should enable more useful responses to the perceived delays. For example, there is not much point in increasing network bandwidth if most of the delays are occurring in the server. The techniques used are passive and can be used on any link where measurements can be taken, meaning that the technique can easily be deployed.

A number of significant issues remain to be resolved. Firstly great care is needed in carefully selecting valid RTTs not contaminated by complex situations caused by lost packets or “unusual” implementations of TCP. Also more work is needed to provide error estimates on the final values. For example, error bounds on the process of taking the minimum times as estimates of the latency should be provided.

Future work will also include validation under laboratory conditions, and accounting for more of the total time in sessions. The ultimate aim is to ascribe the whole time in each TCP session to a well defined cause. Ambiguities in the state of the TCP stacks may make this impossible but it is not clear at this point how far it is possible to go toward this goal in practice. This work provides a good first step in the process.

#### REFERENCES

- [1] Ian D. Graham, Stephen Donnelly, Stele Martin, Jed Martens, and John Cleary, “Nonintrusive and accurate measurement of unidirectional delay and delay variation on the internet,” in *Proceedings, INET'98 Conference*, July 1998.
- [2] W. Richard Stevens, *TCP/IP Illustrated*, vol. 1, The Protocols, Addison-Wesley, 1994.
- [3] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “Tcp selective acknowledgment options,” *Internet RFC 2018*, October 1996.
- [4] R. Braden, “Time-wait assassination hazards in tcp,” *Internet RFC 1337*, May 1992.
- [5] W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms,” *Internet RFC 2001*, January 1997.
- [6] Cleary J.G., Donnelly F.F., Graham I., and McGregor A.J. Pearson M.W., “Design principles for accurate passive measurement,” in *Proceedings, PAM2000*, Hamilton, New Zealand, April 2000.