Internet Delay Experiments

This memo reports on some measurement experiments and suggests some possible improvements to the TCP retransmission timeout calculation. This memo is both a status report on the measurements and advice to implementers of TCP.

1. Introduction

This memorandum describes two series of experiments designed to explore the transmission characteristics of the Internet system. One series of experiments was designed to determine the network delays with respect to packet length, while the other was designed to assess the effectiveness of the TCP retransmission-timeout algorithm specified in the standards documents. Both sets of experiments were conducted during the October - November 1983 time frame and used many hosts distributed throughout the Internet system.

The objectives of these experiments were first to accumulate experimental data on actual network paths that could be used as a benchmark of Internet system performance, and second to apply these data to refine individual TCP implementations and improve their performance.

The experiments were done using a specially instrumented measurement host called a Fuzzball, which consists of an LSI-11 running IP/TCP and various application-layer protocols including TELNET, FTP and SMTP mail. Among the various measurement packages is the original PING (Packet InterNet Groper) program used over the last six years for numerous tests and measurements of the Internet system and its client nets. This program contains facilities to send various kinds of probe packets, including ICMP Echo messages, process the reply and record elapsed times and other information in a data file, as well as produce real-time snapshot histograms and traces.

Following an experiment run, the data collected in the file were reduced by another set of programs and plotted on a Peritek bit-map display with color monitor. The plots have been found invaluable in the indentification and understanding of the causes of netword glitches and other "zoo" phenomena. Finally, summary data were extracted and presented in this memorandum. The raw data files, including bit-map image files of the various plots, are available to other experimenters upon request.

The Fuzzballs and their local-net architecture, called DCN, have about two-dozen clones scattered worldwide, including one (DCN1) at the Linkabit Corporation offices in McLean, Virginia, and another at the Norwegian Telecommunications Adminstration (NTA) near Oslo, Norway. The DCN1 Fuzzball is connected to the ARPANET at the Mitre IMP by means of 1822 Error Control Units operating over a 56-Kbps line. The NTA Fuzzball is connected to the NTARE Gateway by an 1822 interface and then via VDH/HAP operating over a 9.6-Kbps line to SATNET at the Tanum (Sweden) SIMP. For most experiments described below, these details of the local connectivity can be ignored, since only relatively small delays are involved.

The remote test hosts were selected to represent canonical paths in the

Internet system and were scattered all over the world. They included some on the ARPANET, MILNET, MINET, SATNET, TELENET and numerous local nets reachable via these long-haul nets. As an example of the richness of the Internet system connectivity and the experimental data base, data are included for three different paths from the ARPANET-based measurement host to London hosts, two via different satellite links and one via an undersea cable.

2. Packet Length Versus Delay

This set of experiments was designed to determine whether delays across the Internet are significantly influenced by packet length. In cases where the intrinsic propagation delays are high relative to the time to transmit an individual packet, one would expect that delays would not be strongly affected by packet length. This is the case with satellite nets, including SATNET and WIDEBAND, but also with terrestrial nets where the degree of traffic aggregation is high, so that the measured traffic is a small proportion of the total traffic on the path. However, in cases where the intrinsic propagation delays are low and the measured traffic represents the bulk of the traffic on the path, quite the opposite would be expected.

The objective of the experiments was to assess the degree to which TCP performance could be improved by refining the retransmission-timeout algorithm to include a dependency on packet length. Another objective was to determine the nature of the delay characteristic versus packet length on tandem paths spanning networks of widely varying architectures, including local-nets, terrestrial long-haul nets and satellite nets.

2.1. Experiment Design

There were two sets of experiments to measure delays as a function of packet length. One of these was based at DCN1, while the other was based at NTA. All experiments used ICMP Echo/Reply messages with embedded timestamps. A cycle consisted of sending an ICMP Echo message of specified length, waiting for the corresponding ICMP Reply message to come back and recording the elapsed time (normalized to one-way delay). An experiment run, resulting in one line of the table below, consisted of 512 of these volleys.

The length of each ICMP message was determined by a random-number generator uniformly distributed between zero and 256. Lengths less than 40 were rounded up to 40, which is the minimum datagram size for an ICMP message containing timestamps and just happens to also be the minimum TCP segment size. The maximum length was chosen to avoid complications due to fragmentation and reassembly, since ICMP messages are not ordinarily fragmented or reassembled by the gateways.

The data collected were first plotted as a scatter diagram on a color bit-map display. For all paths involving the ARPANET, this immediately revealed two distinct characteristics, one for short (single-packet) messages less than 126 octets in length and the other for long (multi-packet) messages

longer than this. Linear regression lines were then fitted to each characteristic with the results shown in the following table. (Only one characteristic was assumed for ARPANET-exclusive paths.) The table shows for each host the delays, in milliseconds, for each type of message along with a rate computed on the basis of these delays. The "Host ID" column designates the host at the remote end of the path, with a letter suffix used when

necessary to identify a particular run.

Host ID	Single-packet 40	125	Rate (bps)	Multi-packet 125	256	Rate (bps)	Comments
DCN1 to nearby local-net hosts (calibration)							
DCN5	9			13		366422	DMA 1822
DCN8	14			20		268017	Ethernet
IMP17	22			60		45228	56K 1822/ECU
FORD1	93			274		9540	9600 DDCMP base
UMD1	102			473		4663	4800 synch
DCN6	188			550		4782	4800 DDCMP
FACC	243			770		3282	9600/4800 DDCMP

FOE	608				1917	1320	9600/14.4K stat mux
DCN1 to ARPANET hosts and local nets							
MILARP	61	105	15358	133	171	27769	MILNET gateway
ISID-L	166	263	6989	403	472	15029	low-traffic period
SCORE	184	318	5088	541	608	15745	low-traffic period
RVAX	231	398	4061	651	740	11781	Purdue local net
AJAX	322	578	2664	944	1081	7681	MIT local net
ISID-H	333	520	3643	715	889	6029	high-traffic period
BERK	336	967	1078	1188	1403	4879	UC Berkeley
WASH	498	776	2441	1256	1348	11379	U Washington
DCN1 to MILNET/MINET hosts and local nets							
ISIA-L	460	563	6633	1049	1140	11489	low-traffic period
ISIA-H	564	841	2447	1275	1635	2910	high-traffic period
BRL	560	973	1645	1605	1825	4768	BRL local net
LON	585	835	2724	1775	1998	4696	MINET host (London)
HAWAII	679	980	2257	1817	1931	9238	a long way off
OFFICE3	762	1249	1396	2283	2414	8004	heavily loaded host
KOREA	897	1294	1712	2717	2770	19652	a long, long way off
DCN1 to TELENET hosts via ARPANET							
RICE	1456	2358	754	3086	3543	2297	via VAN gateway
DCN1 to SATNET hosts and local nets via ARPANET							
UCL	1089	1240	4514	1426	1548	8558	UCL zoo
NTA-L	1132	1417	2382	1524	1838	3339	low-traffic period
NTA-H	1247	1504	2640	1681	1811	8078	high-traffic period
NTA to SATNET hosts							
TANUM	107				368	6625	9600 bps Tanum line
ETAM	964				1274	5576	Etam channel echo
GOONY	972				1256	6082	Goonhilly channel echo

2.2 Analysis of Results

The data clearly show a strong correlation between delay and length, with the longest packets showing delays two to three times the shortest. On paths via ARPANET clones the delay characteristic shows a stonger correlation with length for single-packet messages than for multi-packet messages, which is consistent with a design which favors low delays for short messages and high throughputs for longer ones.

Most of the runs were made during off-peak hours. In the few cases where runs were made for a particular host during both on-peak and off-peak hours, comparison shows a greater dependency on packet length than on traffic shift.

TCP implementors should be advised that some dependency on packet length may have to be built into the retransmission-timeout estimation algorithm to insure good performance over lossy nets like SATNET. They should also be

advised that some Internet paths may require stupendous timeout intervals ranging to many seconds for the net alone, not to mention additional delays on host-system queues.

I call to your attention the fact that the delays (at least for the larger packets) from ARPANET hosts (e.g. DCN1) to MILNET hosts (e.g. ISIA) are in the same ballpark as the delays to SATNET hosts (e.g. UCL)! I have also observed that the packet-loss rates on the MILNET path are at present not neglible (18 in 512 for ISIA-2). Presumably, the loss is in the gateways; however, there may well be a host or two out there swamping the gateways with retransmitted data and which have a funny idea of the "normal" timeout interval. The recent discovery of a bug in the TOPS-20 TCP implementation, where spurious ACKs were generated at an alarming rate, would seem to confirm that suspicion.

3. Retransmission-Timeout Algorithm

One of the basic features of TCP which allow it to be used on paths spanning many nets of widely varying delay and packet-loss characteristics is the retranansmission-timeout algorithm, sometimes known as the "RSRE Algorithm" for the original designers. The algorithm operates by recording the time and initial sequence number when a segment is transmitted, then computing the elapsed time for that sequence number to be acknowledged. There are various degrees of sophistication in the implementation of the algorithm, ranging from allowing only one such computation to be in progress at a time to allowing one for each segment outstanding at a time on the connection.

The retransmission-timeout algorithm is basically an estimation process. It maintains an extimate of the current roundtrip delay time and updates it as new delay samples are computed. The algorithm smooths these samples and then establishes a timeout, which if exceeded causes a retransmission. The selection of the parameters of this algorithm are vitally important in order to provide effective data transmission and avoid abuse of the Internet system by excessive retransmissions. I have long been suspicious of the parameters

suggested in the specification and used in some implementations, especially in cases involving long-delay paths involving lossy nets. The experiment was designed to simulate the operation of the algorithm using data collected from real paths involving some pretty leaky Internet plumbing.

3.1. Experiment Design

The experiment data base was constructed of well over a hundred runs using ICMP Echo/Reply messages bounced off hosts scattered all over the world. Most runs, including all those summarized here, consisted of 512 echo/reply cycles lasting from several seconds to twenty minutes or so. Other runs designed to detect network glitches lasted several hours. Some runs used packets of constant length, while others used different lengths distributed from 40 to 256 octets. The maximum length was chosen to avoid complications fragmented or reassembled by the gateways.

The object of the experiment was to simulate the packet delay distribution seen by TCP over the paths measured. Only the network delay is of interest here, not the queueing delays within the hosts themselves, which can be considerable. Also, only a single packet was allowed in flight, so that stress on the network itself was minimal. Some tests were conducted

during busy periods of network activity, while others were conducted during quiet hours.

The 512 data points collected during each run were processed by a program which plotted on a color bit-map display each data point (x,y), where x represents the time since initiation of the experiment the and y the measured delay, normalized to the one-way delay. Then, the simulated retransmission-timeout algorithm was run on these data and its computed timeout plotted in the same way. The display immediately reveals how the algorithm behaves in the face of varying traffic loads, network glitches, lost packets and superfluous retransmissions.

Each experiment run also produced summary statistics, which are summarized in the table below. Each line includes the Host ID, which identifies the run. The suffix -1 indicates 40-octet packets, -2 indicates 256-octet packets and no suffix indicates uniformly distributed lengths between 40 and 256. The Lost Packets columns refer to instances when no ICMP Reply message was received for thirty seconds after transmission of the ICMP Echo message, indicating probable loss of one or both messages. The RTX Packets columns refer to instances when the computed timeout is less than the measured delay, which would result in a superfluous retransmission. For each of these two types of packets the column indicates the number of instances and the Time column indicates the total accumulated time required for the recovery action.

For reference purposes, the Mean column indicates the computed mean delay of the echo/reply cycles, excluding those cycles involving packet loss, while the CoV column indicates the coefficient of variation. Finally, the Eff

column indicates the efficiency, computed as the ratio of the total time accumulated while sending good data to this time plus the lost-packet and rtx-packet time.

Complete sets of runs were made for each of the hosts in the table below for each of several selections of algorithm parameters. The table itself reflects values, selected as described later, believed to be a good compromise for use on existing paths in the Internet system.

Host ID	Total Time	Lost Packets	Time	RTX Packets	Time	Mean	CoV	Eff
DCN1 to nearby local-net hosts (calibration)								
DCN5	5	0	0	0	0	11	.15	1
DCN8	8	0	0	0	0	16	.13	1
IMP17	19	0	0	0	0	38	.33	1
FORD1	86	0	0	1	.2	167	.33	.99
UMD1	135	0	0	2	.5	263	.45	.99
DCN6	177	0	0	0	0	347	.34	1
FACC	368	196	222.1	6	9.2	267	1.1	.37
FOE	670	3	7.5	21	73.3	1150	.69	.87
FOE-1	374	0	0	26	61.9	610	.75	.83
FOE-2	1016	3	16.7	10	47.2	1859	.41	.93
DCN1 to ARPANET hosts and local nets								
MILARP	59	0	0	2	.5	115	.39	.99
ISID	163	0	0	1	1.8	316	.47	.98
ISID-1	84	0	0	2	1	163	.18	.98
ISID-2	281	0	0	3	17	516	.91	.93
ISID *	329	0	0	5	12.9	619	.81	.96
SCORE	208	0	0	1	.8	405	.46	.99
RVAX	256	1	1.3	0	0	499	.42	.99
AJAX	365	0	0	0	0	713	.44	1
WASH	494	0	0	2	2.8	960	.39	.99
WASH-1	271	0	0	5	8	514	.34	.97
WASH-2	749	1	9.8	2	17.5	1411	.4	.96
BERK	528	20	50.1	4	35	865	1.13	.83
DCN1 to MILNET/MINET hosts and local nets								
ISIA	436	4	7.4	2	15.7	807	.68	.94
ISIA-1	197	0	0	0	0	385	.27	1
ISIA-2	615	0	0	2	15	1172	.36	.97
ISIA *	595	18	54.1	6	33.3	992	.77	.85
BRL	644	1	3	1	1.9	1249	.43	.99
BRL-1	318	0	0	4	13.6	596	.68	.95
BRL-2	962	2	8.4	0	0	1864	.12	.99
LON	677	0	0	3	11.7	1300	.51	.98
LON-1	302	0	0	0	0	589	.06	1
LON-2	1047	0	0	0	0	2044	.03	1
HAWAII	709	4	12.9	3	18.5	1325	.55	.95
OFFICE3	856	3	12.9	3	10.3	1627	.54	.97
OFF3-1	432	2	4.2	2	6.9	823	.31	.97
OFF3-2	1277	7	39	3	41.5	2336	.44	.93
KOREA	1048	3	14.5	2	18.7	1982	.48	.96
KOREA-1	506	4	8.6	1	2.2	967	.18	.97

KOREA-2	1493	6	35.5	2	19.3	2810	.19	.96
DCN1 to TELENET hosts via ARPANET								
RICE	677	2	6.8	3	12.1	1286	.41	.97

RICE-1	368	1	.1	3	2.3	715	.11	.99
RICE-2	1002	1	4.4	1	9.5	1930	.19	.98
DCN1 to SATNET hosts and local nets via ARPANET								
UCL	689	9	26.8	0	0	1294	.21	.96
UCL-1	623	39	92.8	2	5.3	1025	.32	.84
UCL-2	818	4	13.5	0	0	1571	.15	.98
NTA	779	12	38.7	1	3.7	1438	.24	.94
NTA-1	616	24	56.6	2	5.3	1083	.25	.89
NTA-2	971	19	71.1	0	0	1757	.2	.92
NTA to SATNET hosts and local nets								
TANUM	110	3	1.6	0	0	213	.41	.98
GOONY	587	19	44.2	1	2.9	1056	.23	.91
ETAM	608	32	76.3	1	3.1	1032	.29	.86
UCL	612	5	12.6	2	8.5	1154	.24	.96

Note: * indicates randomly distributed packets during periods of high ARPANET activity. The same entry without the * indicates randomly distributed packets during periods of low ARPANET activity.

3.2 Discussion of Results

It is immediately obvious from visual inspection of the bit-map display that the delay distribution is more-or-less Poissonly distributed about a relatively narrow range with important exceptions. The exceptions are characterized by occasional spasms where one or more packets can be delayed many times the typical value. Such glitches have been commonly noted before on paths involving ARPANET and SATNET, but the true impact of their occurance on the timeout algorithm is much greater than I expected. What commonly happens is that the algorithm, when confronted with a short burst of long-delay packets after a relatively long interval of well-mannered behavior, takes much too long to adapt to the spasm, thus inviting many superfluous retransmissions and leading to congestion.

The incidence of long-delay bursts, or glitches, varied widely during the experiments. Some of them were glitch-free, but most had at least one glitch in 512 echo/reply volleys. Glitches did not seem to correlate well with increases in baseline delay, which occurs as the result of traffic surges, nor did they correlate well with instances of packet loss. I did not notice any particular periodicity, such as might be expected with regular pinging, for example; however, I did not process the data specially for that.

There was no correction for packet length used in any of these experiments, in spite of the results of the first set of experiments described previously. This may be done in a future set of experiments. The algorithm does cope well in the case of constant-length packets and in the case of randomly distributed packet lengths between 40 and 256 octets, as indicated in the table. Future experiments may involve bursts of short packets followed by bursts of longer ones, so that the speed of adaptation of the algorithm can be directly deterimend.

One particularily interesting experiment involved the FOE host (FORD-FOE), which is located in London and reached via a 14.4-Kbps undersea cable and statistical multiplexor. The multiplexor introduces a moderate mean delay, but with an extremely large delay dispersion. The specified retransmission-timeout algorithm had a hard time with this circuit, as might be expected; however, with the improvments described below, TCP performance was acceptable. It is unlikely that many instances of such ornery circuits will occur in the Internet system, but it is comforting to know that the algorithm can deal effectively with them.

3.3. Improvments to the Algorithm

The specified retransmission-timeout algorithm, really a first-order linear recursive filter, is characterized by two parameters, a weighting factor F and a threshold factor G. For each measured delay sample R the delay estimator E is updated:

$$E = F*E + (1 - F)*R \ .$$

Then, if an interval equal to G*E expires after transmitting a packet, the packet is retransmitted. The current TCP specification suggests values in the range 0.8 to 0.9 for F and 1.5 to 2.0 for G. These values have been believed reasonable up to now over ARPANET and SATNET paths.

I found that a simple change to the algorithm made a worthwhile change in the efficiency. The change amounts to using two values of F, one (F1) when R < E in the expression above and the other (F2) when R >= E, with F1 > F2. The effect is to make the algorithm more responsive to upward-going trends in delay and less respnsive to downward-going trends. After a number of trials I concluded that values of F1 = 15/16 and F2 = 3/4 (with G = 2) gave the best all-around performance. The results on some paths (FOE, ISID, ISIA) were better by some ten percent in efficiency, as compared to the values now used in typical implementations where F = 7/8 and G = 2. The results on most paths were better by five percent, while on a couple (FACC, UCL) the results were worse by a few percent.

There was no clear-cut gain in fiddling with G. The value G = 2 seemed to represent the best overall compromise. Note that increasing G makes superfluous retransmissions less likely, but increases the total delay when packets are lost. Also, note that increasing F2 too much tends to cause overshoot in the case of network glitches and leads to the same result. The table above was constructed using F1 = 15/16, F2 = 3/4 and G = 2.

Readers familiar with signal-detection theory will recognize my suggestion as analogous to an ordinary peak-detector circuit. F1 represents the discharge time-constant, while F2 represents the charge time-constant. G represents a "squelch" threshold, as used in voice-operated switches, for example. Some wag may be even go on to suggest a network glitch should be called a netspurt.

Appendix. Index of Test Hosts

Name	Address	NIC Host Name
DCN1 to nearby local-net hosts (calibration)		

```
DCN5    128.4.0.5       DCN5
DCN8    128.4.0.8       DCN8
IMP17   10.3.0.17       DCN-GATEWAY
FORD1   128.5.0.1       FORD1
UMD1    128.8.0.1       UMD1
DCN6    128.4.0.6       DCN6
FACC    128.5.32.1      FORD-WDL1
FOE     128.5.0.15      FORD-FOE

DCN1 to ARPANET hosts and local nets
MILARP  10.2.0.28       ARPA-MILNET-GW
ISID    10.0.0.27       USC-ISID
SCORE   10.3.0.11       SU-SCORE
RVAX    128.10.0.2      PURDUE-MORDRED
AJAX    18.10.0.64      MIT-AJAX
WASH    10.0.0.91       WASHINGTON
BERK    10.2.0.78       UCB-VAX

DCN1 to MILNET/MINET hosts and local nets
ISIA    26.3.0.103      USC-ISIA
BRL     192.5.21.6      BRL-VGR
LON     24.0.0.7        MINET-LON-EM
HAWAII  26.1.0.36       HAWAII-EMH
OFFICE3 26.2.0.43       OFFICE-3
KOREA   26.0.0.117      KOREA-EMH

DCN1 to TELENET hosts via ARPANET
RICE    14.0.0.12       RICE

DCN1 to SATNET hosts and local nets via ARPANET
UCL     128.16.9.0      UCL-SAM
NTA     128.39.0.2      NTARE1

NTA to SATNET hosts and local nets
TANUM   4.0.0.64        TANUM-ECHO
GOONY   4.0.0.63        GOONHILLY-ECHO
ETAM    4.0.0.62        ETAM-ECHO
```