

2 UDP Request-Response Latency Measurements

The Request-Response tests measure the time taken to send a "Request" to a remote host and obtain a "Response" as a function of the size of the "Response" message. The length of the "Request" message can be varied, but normally, the request will be fixed at 64 bytes long. The time for a series of Request-Responses, typically 10000, is measured using the clock of the sender, and the latency calculated as the time to send the Response message in one direction. Times are measured using the real time clock, via `gettimeofday()` and the Pentium CPU cycle counter using `StopWatch_*` routines modified from the CERN Mesh work for ATLAS.

The Request-Response plots will show the *total round trip* latency as a function of the user data length and one would expect the plots to be described by a linear equation [ref ATLAS note].

UDP packets are used and the requesting node uses a time-out to allow for frames lost on the network. The default UDP port is 14233 or 0x3799, but this can be changed as required.

2.1 Program Description `udp_req_timo` - `udp_resp`

To measure the request-response latency between two systems you start `udp_resp` on the remote system:

```
./udp_resp
```

and `udp_req_timo` on the local system:

```
./udp_req_timo -d194.36.2.1 -p64 -r64 -l1000
```

will make 1000 64 byte requests for 64 byte responses and produce the following output (ignore the line wrap for the headings):

```
response len bytes; Req len; msg_count; loop_count; num_bad; time/frame us;
Mbytes/s; ave time; min time; max time; num timeouts;
 64; 64; 1; 1000; 0; 127.79; 1.00164; 126.270; 120.77; 1193.952; 0
```

`time/frame us;` and `Mbytes/s;` use the total time taken for all the request-response loops.
`ave time;` `min time;` `max time;` use the times for individual request-response measurements.

The command

```
./udp_req_timo -d194.36.2.1 -p64 -r64 -l2000 -i8 -e1500
```

will make a latency scan with 2000 request-responses for each point. It will use 64 byte requests and start by requesting a 64 byte response, when this measurement is finished, it will increment the requested length by 8 bytes and repeat the test until the requested length has reached 1500 bytes.

```
response len bytes; Req len; msg_count; loop_count; num_bad; time/frame us;
Mbytes/s; ave time; min time; max time; num timeouts;
 64; 64; 1; 1000; 0; 127.79; 1.00164; 126.270; 120.77; 1193.952; 0
 72; 64; 1; 1000; 0; 127.812; 1.12665; 126.308; 121.49; 144.857; 0
 80; 64; 1; 1000; 0; 128.723; 1.24298; 127.209; 122.88; 158.274; 0
 88; 64; 1; 1000; 0; 234.72; 0.74983; 233.444; 123.31; 4839.224; 0
...
1496; 64; 1; 1000; 0; 300.382; 9.96065; 298.813; 293.61; 317.982; 0
  Total no. packets sent: 180000 Total no. timeouts: 0
```

2.1.1 Options for `udp_resp`

`-v` Verbose mode - turn on printout
`-u` The UDP port number to use for the tests. Enter using base 10, default is 14233 or 0x3799.

2.1.2 Options for `udp_req_timo_hist`

- d The destination IP address in dot format e.g. a.b.c.d
- p The length in bytes of request packet - not including any headers.
- r The start size of the response message in bytes - not including any headers
- l The number of times to loop making request-responses.
- i The number of bytes to increment the response message
- e The end size of the response message in bytes.
- v Verbose mode - turn on printout
- u The UDP port number to use for the tests. Enter using base 10, default is 14233 or 0x3799.

Control-c displays the number of request-response tests completed, the number requested, the number of timeouts, and the current response length in bytes.

Control-z terminates the program.

2.2 Program Description `udp_req_timo_hist` - `udp_resp`

This pair of programs provides histograms of the request-response latency with 1 us (or larger) bins. The histograms use the ; delimiter to facilitate being imported into Excel.

Running `udp_req_timo_hist` on the local system, with a command similar to:

```
./udp_req_timo_hist -d194.36.2.1 -p64 -r64 -l1000 -m110 -b1  
produce the output similar to:
```

```
response length; packet length; msg_count; loop_count; num_timeout; Time/frame us;  
Data rate MBytes/s  
64; 64; 1; 10000; 0; 123.014; 1.04053; 0  
Hist 0 Time per req-resp us  
counts 9996 mean 120 underflows 0 overflows 4  
110 ; 0  
111 ; 0  
112 ; 0  
113 ; 0  
114 ; 0  
115 ; 0  
116 ; 51  
117 ; 536  
118 ; 957  
119 ; 2036  
120 ; 2924  
121 ; 1804  
122 ; 729  
123 ; 572  
124 ; 229  
125 ; 28  
126 ; 3  
...
```

2.2.1 Options for `udp_req_timo_hist`

- d The destination IP address in dot format e.g. a.b.c.d
- p The length in bytes of request packet - not including any headers.
- r The start size of the response message in bytes - not including any headers
- l The number of times to loop making request-responses.
- i The number of bytes to increment the response message
- e The end size of the response message in bytes.
- v Verbose mode - turn on printout

-u The UDP port number to use for the tests. Enter using base 10, default is 14233 or 0x3799.

-b The histogram bin width - the default is 1 us

-m The low limit for the histogram - the default is 0

Control-c displays the number of request-response tests completed, the number requested, the number of timeouts, and the current response length in bytes.

Control-z terminates the program.

100 Mbit PC to PC Laboratory Tests

Figure 2.1 shows Latency measurements made using UDP frames between two Pentium PCs directly connected with a 100 Mbit Ethernet Link joining the two 3com 3c905 nics.

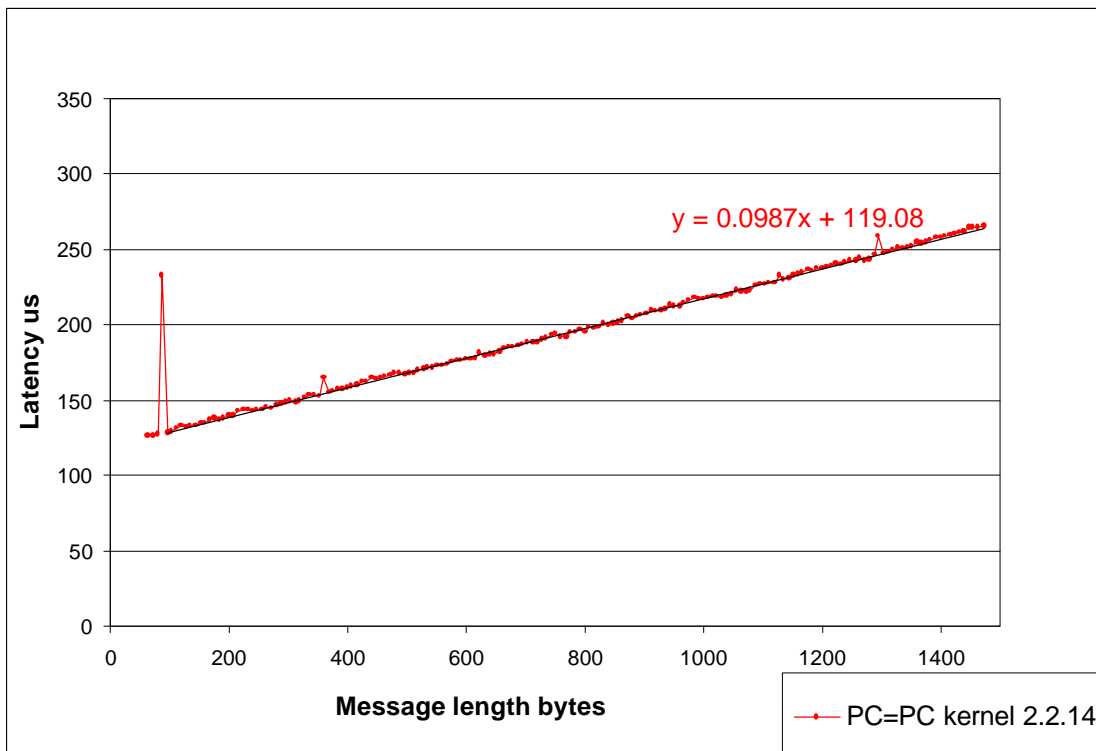


Figure 2.1 UDP Request-Response latency for 2 PCs directly connected with 100 Mbit Ethernet.

The table gives an idea of the contributions to the PC-PC slope, with the 3com interface, each part of the data transfer should be store and forward, as indicated in the middle column. The right hand column allows overlapping of the data moving for transmission.

	μs/byte	μs/byte
the transfer of the frame over the PCI bus of the sending PC	0.0078	
transmission of the packet at 100 Mbit line speed	0.08	0.08
	0.0078	0.0078
Total	0.0956	0.0878
Measured	0.0987	0.0987
Driver and application software (both ends)	0.0031	0.01

For a 100 MHz PC memory bus (64 bits wide) memory-to-memory copy takes an estimated 0.0025 μs/byte. The figures indicate that just one memory-to-memory copy is involved. (A contribution of 0.005 μs/byte from each end of the link would indicate that each end was doing 2 copies.)

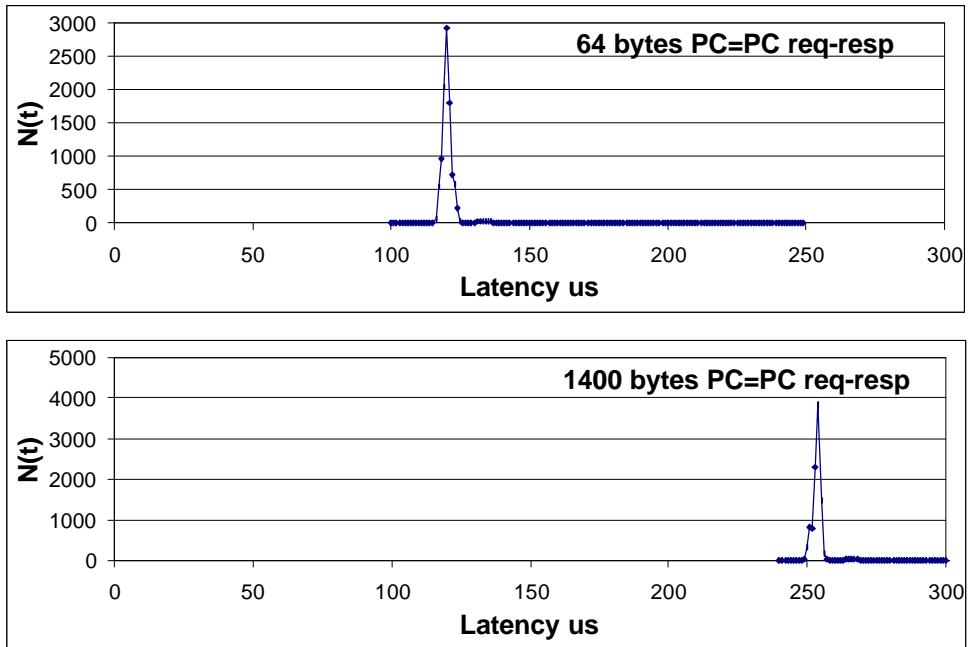


Figure 2.2 Example of histograms of the latency measured between the two directly connected PCs.

3 UDP Bandwidth and Packet Loss Measurements

The bandwidth of the bottleneck, or the network section that limits the bandwidth in the route between the test nodes may be determined by measuring the times taken to send and receive a burst of frames sent from node a to node b.

The test uses UDP/IP frames with the application protocol shown in Figure 3.1. The test starts with the Requesting node sending a “clear statistics” message to the Responder. On reception of the OK acknowledgement, the Requesting node sends a series of “data” packets separated with a given fixed time interval between the packets. At the end, the Requesting node asks for the statistics collected by the Responding node. Packet loss for the control messages are handled by suitable time-outs and re-tries in the Requesting node. The transmit throughput is found using the amount of data sent and the time taken; the receive throughput is calculated from the amount of data received and the time from the first data packet to the last packet received.

Packet loss is measured by the Responding node by checking that sequence numbers in the packets increase correctly, this also detects out-of-order packets. The number of packets seen, the number missed as indicated the sequence number check, and the number out-of-order are reported at the end of each test.

The Responding node also measures the time between successive packets and produces a histogram of these times. This histogram may be requested by the Requesting node at the end of the test.

Queue lengths in the network may be investigated by comparing the extra time taken to receive the burst of packets.

Remember that the available bandwidth and packet loss may be different from node a->b and node b->a.

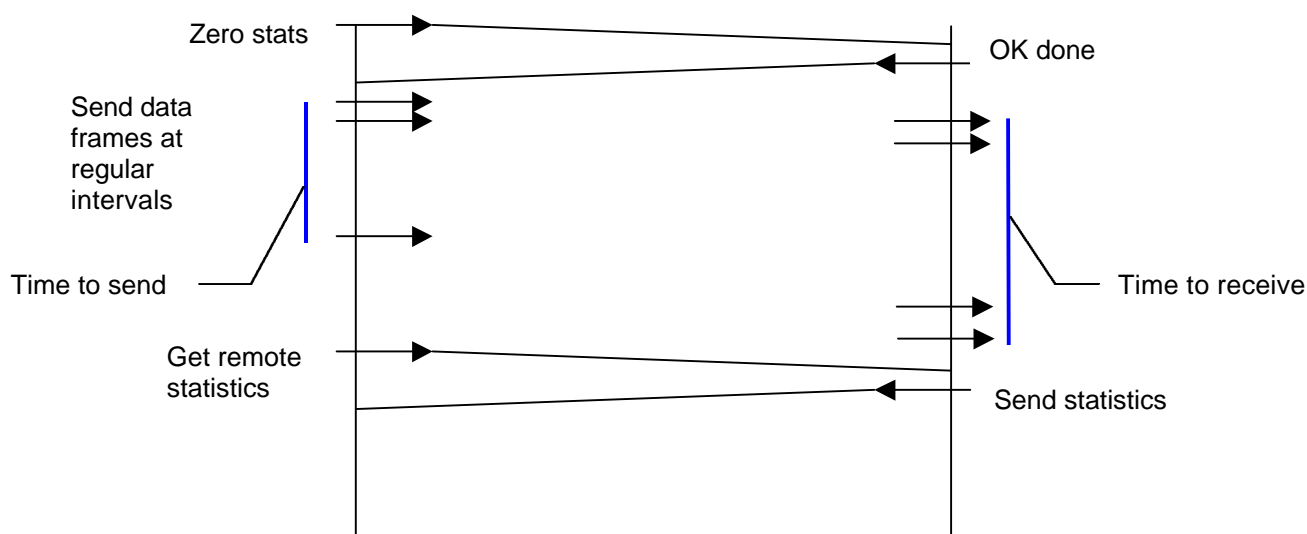


Figure 3.1 Protocol for the Bandwidth and Packet loss measurements

3.1 Program Description `udp_bw_mon` - `udp_bw_resp`

To measure the network bandwidth available between two systems you start `udp_bw_resp` on the remote system:

```
./udp_bw_resp
```

and `udp_bw_mon` on the local system:

```
./udp_bw_mon -d194.36.2.1 -p50 -w20 -l1000
```

will send 1000 50 byte data packets, waiting 20 us between sending each packet; and produce output similar to(ignore the line wrap for the headings):

```

pkt len; loop_count; num_timeout_zero; num_timeout; Time/frame us; wait_time; data_rate; num_rcv; num_lost; num_badorder; time; time/rcv_pkt;
rcv_data_rate
50 ; 1000 ; 0 ; 0 ; 23.7852 ; 20 ; 23785.2 ; 2.10214 ; 568 ; 432 ; 0 ; 23984 ; 42.2254 ; 1.18412

```

Running `udp_req_timo_hist` on the local system, with a command similar to:
`./udp_bw_mon -d194.36.2.1 -p50 -w0 -l1000 -i10 -e100`

will make a scan, increasing the wait time us 10 us each time. 1000 packets will be sent for each point, and it will produce output similar to:

```

pkt len; loop_count; num_timeout_zero; num_timeout; wait_time; Time/frame us; data_rate; num_rcv; num_lost; num_badorder; time; time/rcv_pkt;
rcv_data_rate
50 ; 1000 ; 0 ; 0 ; 21.8196 ; 0 ; 21819.6 ; 2.29152 ; 318 ; 681 ; 0 ; 21962 ; 69.0629 ; 0.723978
50 ; 1000 ; 0 ; 0 ; 21.6953 ; 10 ; 21695.3 ; 2.30465 ; 320 ; 680 ; 0 ; 21845 ; 68.2656 ; 0.732433
50 ; 1000 ; 0 ; 0 ; 23.7852 ; 20 ; 23785.2 ; 2.10214 ; 568 ; 432 ; 0 ; 23984 ; 42.2254 ; 1.18412
50 ; 1000 ; 0 ; 0 ; 30.1143 ; 30 ; 30114.3 ; 1.66034 ; 1000 ; 0 ; 0 ; 29985 ; 29.985 ; 1.6675
50 ; 1000 ; 0 ; 0 ; 40.0933 ; 40 ; 40093.3 ; 1.24709 ; 1000 ; 0 ; 0 ; 39950 ; 39.95 ; 1.25156
50 ; 1000 ; 0 ; 0 ; 50.0877 ; 50 ; 50087.7 ; 0.99825 ; 1000 ; 0 ; 0 ; 49909 ; 49.909 ; 1.00182
50 ; 1000 ; 0 ; 0 ; 60.0712 ; 60 ; 60071.2 ; 0.832345 ; 1000 ; 0 ; 0 ; 59860 ; 59.86 ; 0.835282
50 ; 1000 ; 0 ; 0 ; 70.1032 ; 70 ; 70103.2 ; 0.713234 ; 1000 ; 0 ; 0 ; 69858 ; 69.858 ; 0.715738
50 ; 1000 ; 0 ; 0 ; 80.0843 ; 80 ; 80084.3 ; 0.624342 ; 1000 ; 0 ; 0 ; 79808 ; 79.808 ; 0.626504
50 ; 1000 ; 0 ; 0 ; 90.0852 ; 90 ; 90085.2 ; 0.55503 ; 1000 ; 0 ; 0 ; 89775 ; 89.775 ; 0.556948
50 ; 1000 ; 0 ; 0 ; 100.108 ; 100 ; 100108 ; 0.49946 ; 1000 ; 0 ; 0 ; 99765 ; 99.765 ; 0.501178
...

```

`num_timeout_zero` and `num_timeout` are the number of timeouts seen when sending the “Zero Statistics” and “Get Statistics” commands.
`Time/frame us` is the average time between transmitting data packets, it will be equal to `wait_time` when `wait_time` is greater than the time to send the data.
`Time/frame us` and `data_rate` are calculated from the sending times.
`Time` is the total time to receive the data.
`time/rcv_pkt` and `rcv_data_rate` are calculated from the time to receive the data.

3.1.1 Options for udp_bw_resp

- v Verbose mode - turn on printout
- u The UDP port number to use for the tests. Enter using base 10.

3.1.2 Options for udp_bw_mon

- d The destination IP address in dot format e.g. a.b.c.d
- p The length in bytes of the data packets - not including any headers.
- w The time to wait between sending packets in us.
- l The number of times to loop sending data packets.
- i The number of us to increment the wait time between sending packets.
- e The end size of the wait time in us.
- H If given the histograms made at the receiving node are retrieved and printed out.
- B The remote histogram bin width - the default is 1 us
- M The low limit for the remote histogram - the default is 0
- v Verbose mode - turn on printout
- u The UDP port number to use for the tests. Enter using base 10.

Control-c displays the number of request-response tests completed, the number requested, the number of timeouts, and the current response length in bytes.

Control-z terminates the program.

3.2 100 Mbit PC to PC Laboratory Tests

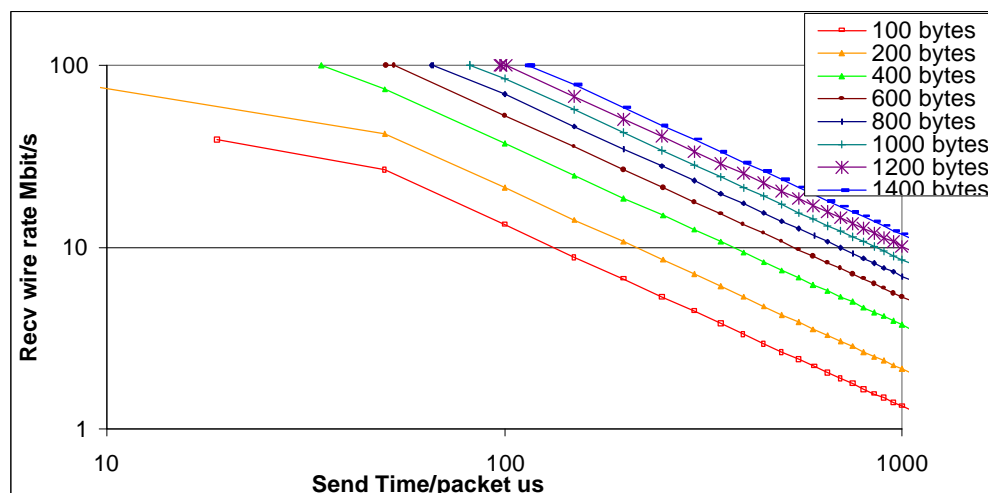
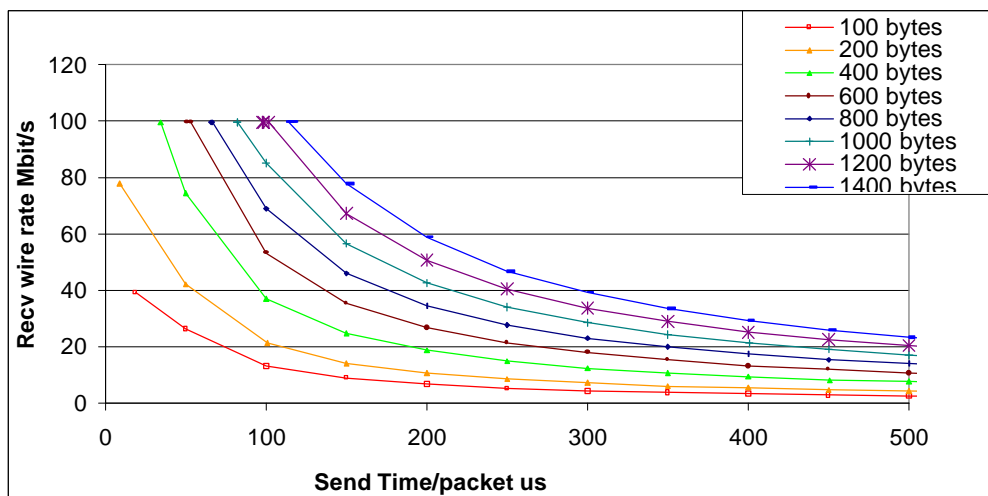


Figure 3.1 Plots of the Received wire rate vs time delay between sending packets.

4 TCP Request-Response Latency Measurements

The concepts are very similar to those discussed for UDP in Section 2. At the moment (Nov2000) the sockets use the default TCP parameters e.g. send and receive buffer size and high water marks. On Linux 2.2.14 the send and receive buffer sizes are set at 65535.

With TCP, no application level timeouts are required.

4.1 Program Description `tcp_req - tcp_resp`

To measure the request-response latency between two systems you start `tcp_resp` on the remote system:

```
./tcp_resp
```

and `tcp_req` on the local system:

```
./tcp_req -d194.36.2.1 -p64 -r64 -l1000
```

will make 1000 64 byte requests for 64 byte responses and produce the following output (ignore the line wrap for the headings):

```
response len bytes; Req len; msg_count; loop_count; num_bad; time/frame us;
Mbytes/s; ave time; min time; max time; num timeouts;
  64; 64; 1; 1000; 0; 152.869; 0.83731; 151.098; 139.01; 1211.76
```

`time/frame us;` and `Mbytes/s;` use the total time taken for all the request-response loops.
`ave time; min time; max time;` use the times for individual request-response measurements.

The command

```
./tcp_req -d194.36.2.1 -p64 -r64 -l2000 -i8 -e1500
```

will make a latency scan with 2000 request-responses for each point. It will use 64 byte requests and start by requesting a 64 byte response, when this measurement is finished, it will increment the requested length by 8 bytes and repeat the test until the requested length has reached 1500 bytes.

```
response len bytes; Req len; msg_count; loop_count; num_bad; time/frame us;
Mbytes/s; ave time; min time; max time; num timeouts;
  64; 64; 1; 1000; 0; 152.869; 0.83731; 151.098; 139.01; 1211.76
  72; 64; 1; 1000; 0; 153.973; 0.93522; 152.212; 147.73; 1193.98
  80; 64; 1; 1000; 0; 154.111; 1.03821; 152.342; 148.32; 178.080
  88; 64; 1; 1000; 0; 155.143; 1.13444; 153.339; 144.07; 337.540
  96; 64; 1; 1000; 0; 155.629; 1.2337; 153.863; 149.60; 181.103
 104; 64; 1; 1000; 0; 162.58; 1.27937; 160.806; 151.11; 3132.08
 112; 64; 1; 1000; 0; 165.323; 1.35492; 163.468; 151.63; 2340.14
```

4.1.1 Options for `tcp_resp`

`-v` Verbose mode - turn on printout
`-u` The TCP port number to use for the tests. Enter using base 10, default is 14233 or 0x3799.

4.1.2 Options for `tcp_req`

`-d` The destination IP address in dot format e.g. a.b.c.d
`-p` The length in bytes of request packet - not including any headers.
`-r` The start size of the response message in bytes - not including any headers
`-l` The number of times to loop making request-responses.
`-i` The number of bytes to increment the response message
`-e` The end size of the response message in bytes.
`-v` Verbose mode - turn on printout
`-u` The UDP port number to use for the tests. Enter using base 10, default is 14233 or 0x3799.

Control-c displays the number of request-response tests completed, the number requested, and the current response length in bytes. Control-z terminates the program.

4.2 Program Description `tcp_req_hist` - `tcp_resp`

This pair of programs provides histograms of the request-response latency with 1 us (or larger) bins. The histograms use the ; delimiter to facilitate being imported into Excel.

Running `tcp_req_hist` on the local system, with a command similar to:

```
./tcp_req_hist -d194.36.2.1 -p64 -r64 -l1000 -m110 -bl
produce the output similar to:
```

4.3 100 Mbit PC to PC Laboratory Tests

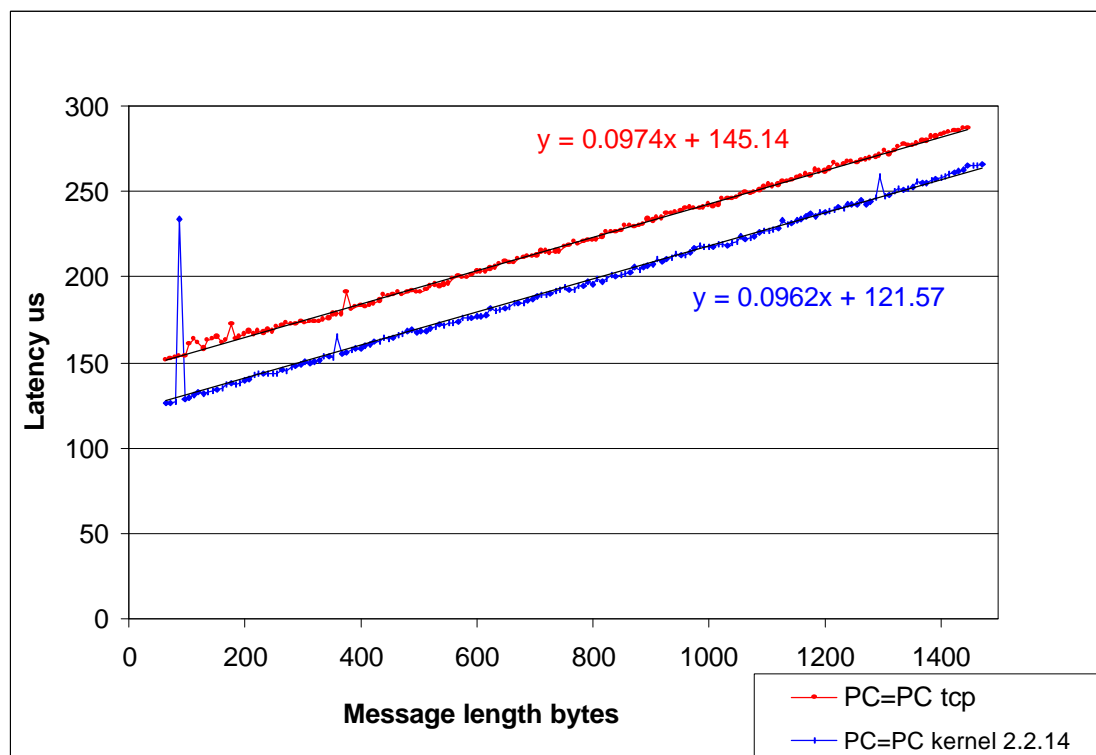


Figure 4.1 TCP and UDP Request-Response latency for 2 PCs directly connected with 100 Mbit Ethernet.

5 TCP Throughput Measurements

These test programs use a similar application level protocol to that described in Section 3 and shown in Figure 3.1, however there are no application timeouts and no lost data as TCP will re-try any frames that are lost on the network. There are two things to note:

- As with the UDP tests the times measured for sending are those when the user data is given to the TCP stack NOT when the data is put on the network.
- Unlike UDP there is no direct correspondence between an application message and a packet or series of packets, on the network. For this reason `tcp_bw_resp` records the number of reads the application makes to obtain the data. TCP is a stream protocol and is free to concatenate or split the user data as it sees fit.

`tcp_bw_mon` and `tcp_bw_resp` allow multiple parallel TCP streams to be established for a given test and the output reports the use made of each stream. Stream 0 is use as the “command” stream

between the two test programs, rather similar to ftp. This is required as when the data is received, no effort is made to parse the data sent over the “data” streams into sections or blocks.

They also avoid blocking when reading from or writing to a socket by determining that either there is already data to read in a given socket or that there is space to write the application data to the socket. At the moment (Nov 2000) `select()` is used to determine which sockets can accept data, and then data is written to all these sockets in turn.

5.1 Program Description `tcp_bw_mon` - `tcp_bw_resp`

To measure the throughput available between two systems you start `tcp_bw_resp` on the remote system:

```
./tcp_bw_resp
```

and `tcp_bw_mon` on the local system:

```
./tcp_bw_mon -d194.36.2.1 -p1000 -w100 -l10000 -s2
```

will use 2 TCP streams to send 10000 1000 byte data packets, waiting 100 us between sending each packet; and produce output similar to:

```
pkt len; loop_count; Time/loop us; wait_time; Total send time; data_rate; Total rcv time; link;; rcv_bytes; # reads
1000; 10000; 87.1173; 100 ; 871173; 11.4788; 871749; 0;; 1648; 2; 1;; 5023000; 4995; 2;; 4977000; 4955;
```

For each link used the number of bytes read and the number of reads made by tcp_bw_resp is reported e.g. for link 1:

```
1;; 5023000; 4995;
```

5023000 bytes read in 4995 socket reads.

```
./tcp_bw_mon -d194.36.2.1 -p1000 -w0 -i8 -e600 -l10000 -s3
```

will make a scan using 3 TCP streams, increasing the wait time by 8 us each time. 10000 packets will be sent for each point, and it will produce output similar to:

```
pkt len; loop_count; Time/loop us; wait_time; Total send time; data_rate; Total rcv time; link;; rcv_bytes; # reads
1000; 10000; 86.9651 ; 0 ; 869651 ; 11.4989 ; 871605 ; 0;; 1648; 2; 1;; 3345000; 3319; 2;; 3331000; 3292; 3;; 3324000; 3286;
1000; 10000; 87.7077 ; 8 ; 877077 ; 11.4015 ; 879062 ; 0;; 1648; 2; 1;; 3347000; 3310; 2;; 3334000; 3275; 3;; 3319000; 3278;
1000; 10000; 86.9669 ; 16 ; 869669 ; 11.4986 ; 871582 ; 0;; 1648; 2; 1;; 3347000; 3327; 2;; 3336000; 3298; 3;; 3317000; 3291;
1000; 10000; 86.9651 ; 24 ; 869651 ; 11.4989 ; 871631 ; 0;; 1648; 2; 1;; 3349000; 3310; 2;; 3331000; 3282; 3;; 3320000; 3251;
1000; 10000; 86.9643 ; 32 ; 869643 ; 11.499 ; 871647 ; 0;; 1648; 2; 1;; 3346000; 3319; 2;; 3330000; 3292; 3;; 3324000; 3267;
1000; 10000; 86.9578 ; 40 ; 869578 ; 11.4998 ; 871623 ; 0;; 1648; 2; 1;; 3347000; 3303; 2;; 3338000; 3268; 3;; 3315000; 3243;
1000; 10000; 86.9659 ; 48 ; 869659 ; 11.4988 ; 871630 ; 0;; 1648; 2; 1;; 3349000; 3324; 2;; 3328000; 3280; 3;; 3323000; 3285;
1000; 10000; 87.3058 ; 56 ; 873058 ; 11.454 ; 874945 ; 0;; 1648; 2; 1;; 3349000; 3316; 2;; 3328000; 3266; 3;; 3323000; 3246;
1000; 10000; 86.9694 ; 64 ; 869694 ; 11.4983 ; 871596 ; 0;; 1648; 2; 1;; 3342000; 3316; 2;; 3335000; 3308; 3;; 3323000; 3280;
1000; 10000; 90.7874 ; 72 ; 907874 ; 11.0147 ; 909712 ; 0;; 1648; 2; 1;; 3347000; 3319; 2;; 3332000; 3263; 3;; 3321000; 3252;
1000; 10000; 86.9724 ; 80 ; 869724 ; 11.4979 ; 871621 ; 0;; 1648; 2; 1;; 3341000; 3311; 2;; 3335000; 3290; 3;; 3324000; 3283;
1000; 10000; 86.9652 ; 88 ; 869652 ; 11.4988 ; 871664 ; 0;; 1648; 2; 1;; 3346000; 3309; 2;; 3333000; 3282; 3;; 3321000; 3257;
1000; 10000; 86.9724 ; 96 ; 869724 ; 11.4979 ; 871620 ; 0;; 1648; 2; 1;; 3348000; 3326; 2;; 3326000; 3288; 3;; 3326000; 3296;
```

...

```
./tcp_bw_mon -d194.36.2.1 -p1000 -w96 -l10000 -s20
```

uses 20 TCP streams

```
pkt len; loop_count; Time/loop us; wait_time; Total send time; data_rate; Total rcv time; link;; rcv_bytes; # reads
1000; 10000; 81.4445 ; 96 ; 814445 ; 12.2783 ; 855137 ; 0;; 1648; 2;
1: ; 609000; 362; 2: ; 331000; 169; 3: ; 141000; 65; 4: ; 558000; 320;
5: ; 548000; 319; 6: ; 445000; 248; 7: ; 517000; 299; 8: ; 574000; 329;
9: ; 594000; 395; 10: ; 601000; 371; 11: ; 330000; 182; 12: ; 605000; 368;
13: ; 639000; 454; 14: ; 621000; 405; 15: ; 442000; 251; 16: ; 644000; 471;
17: ; 482000; 274; 18: ; 309000; 170; 19: ; 421000; 230; 20: ; 589000; 349;
```