Chapter 24:
TCP Futures and Performance

Introduction

- Path MTU discovery mechanism: this time we focus on how it operates with TCP. This often lets TCP use an MTU greater than 536 for nonlocal connections, increasing its throughput .
- Long fat pipes (networks that have a large bandwidth-delay product), and the TCP limits that are encountered on these networks.
  - A window scale option.
  - A timestamp option.
- T/TCP, modifications to TCP for transactions, is a common paradigm for client-server computing.
- The goal of T/TCP is to reduce the number of segments exchanged by the two ends, avoiding the 3-way handshake and the 4 segments to close the connection, so that the client receives the server's reply in one RTT plus the time required to process the request.

Path MTU Discovery

- **Path MTU is the minimum MTU on the any network that is currently in the path between two hosts.**
- **TCP’s path MTU discovery operates as follows:**
  - When a connection is established, TCP uses the minimum of the MTU of the outgoing interface, or the MSS announced by the other as the starting segment size.
  - All IP datagrams sent by TCP on that connection have the DF bit set. If a datagram need to be fragment, it discard the datagram and generates the ICMP “can’t fragment “ error .
  - If ICMP error is received, TCP decrease the segment and retransmits.
  - Since routes can change dynamically, when some time has passed since the last decrease of the path MTU, a larger value can be tried.

Path MTU Discovery

- **An example :** solaris % **sock -I -n1 -w512 slip discard**

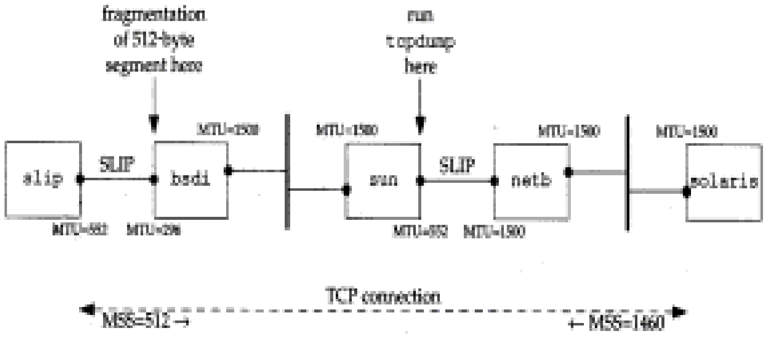
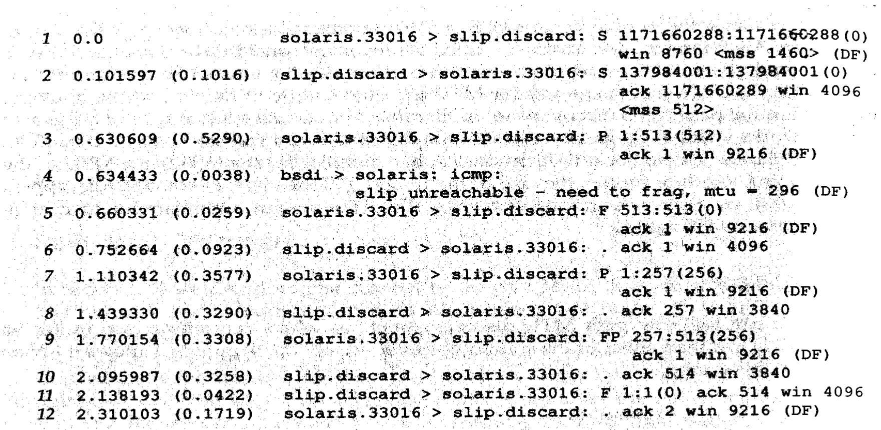


Figure 24.1 Topology for path MTU example.

Page MTU Discovery



```
 1  0.0                     solaris.33016 > slip.discard: S 1171660288:1171660288(0)
                                                          win 8760 <mss 1460> (DF)
 2  0.101597  (0.1016)      slip.discard > solaris.33016: S 137984001:137984001(0)
                                                          ack 1171660289 win 4096
                                                          <mss 512>
 3  0.630609  (0.5290)      solaris.33016 > slip.discard: P 1:513(512)
                                                          ack 1 win 9216 (DF)
 4  0.634433  (0.0038)      bsdi > solaris: icmp:
                                     slip unreachable - need to frag, mtu = 296 (DF)
 5  0.660331  (0.0259)      solaris.33016 > slip.discard: F 513:513(0)
                                                          ack 1 win 9216 (DF)
 6  0.752664  (0.0923)      slip.discard > solaris.33016: . ack 1 win 4096
 7  1.110342  (0.3577)      solaris.33016 > slip.discard: P 1:257(256)
                                                          ack 1 win 9216 (DF)
 8  1.439330  (0.3290)      slip.discard > solaris.33016: . ack 257 win 3840
 9  1.770154  (0.3308)      solaris.33016 > slip.discard: FP 257:513(256)
                                                            ack 1 win 9216 (DF)
10  2.095987  (0.3258)      slip.discard > solaris.33016: . ack 514 win 3840
11  2.138193  (0.0422)      slip.discard > solaris.33016: F 1:1(0) ack 514 win 4096
12  2.310103  (0.1719)      solaris.33016 > slip.discard: . ack 2 win 9216 (DF)
```

Figure 24.2 tcpdump output for path MTU discovery.

Big Packets or Small Packets?

- **Big packets advantages:**
  - The reduced cost is that associated with the network (packet header overhead), routers (routing decisions), and hosts (protocol processing and device interrupts).
- **Not everyone agrees with this, measurements indicate that bigger is not always better.**

Big packets or Small Packets ?

- ❑ (4096+40) x8 / 1544000 = 21.4 , 21.4 x4 = 85.6

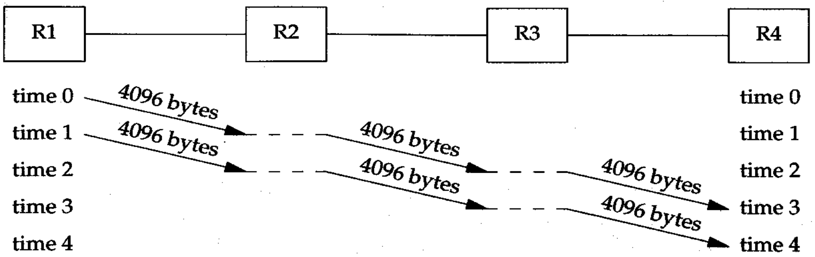


Figure 24.3 Sending two 4096-byte packets through four routers.

Big Packets or Small Packets ?

- ❑ (512+40) x8 /1544000 = 2.9 , 2.9x18= 52.2

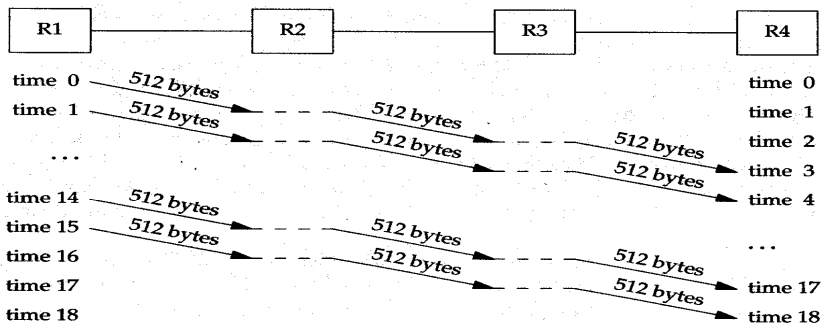
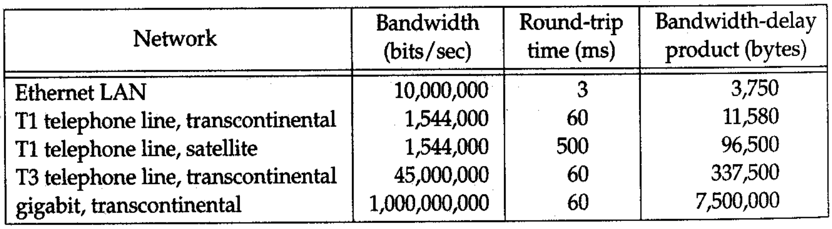


Figure 24.4 Sending sixteen 512-byte packets through four routers.

Long Fat Pipes

- ❑ **Bandwidth-delay product: the size of the pipe between the end points .**
  - ❖ Capacity (bits) = bandwidth (bits/sec) x round-trip time (sec)



Network	Bandwidth (bits/sec)	Round-trip time (ms)	Bandwidth-delay product (bytes)
Ethernet LAN	10,000,000	3	3,750
T1 telephone line, transcontinental	1,544,000	60	11,580
T1 telephone line, satellite	1,544,000	500	96,500
T3 telephone line, transcontinental	45,000,000	60	337,500
gigabit, transcontinental	1,000,000,000	60	7,500,000

Figure 24.5 Bandwidth-delay product for various networks.

Long Fat Pipes (Cont.)

- ❑ **Networks with large bandwidth-delay products are called *long fat networks* (LFN) and a TCP connection operating on an LFN is called *a long fat pipe*.**
- ❑ **Numerous problems are encountered with long fat pipes:**
  - ❖ The TCP window size is a16-bit field in the TCP header, limiting the window to 65535 bytes.
  - ❖ Packets loss in an LFN can reduce throughput drastically. (why?)
  - ❖ Many implementations only measure one round-trip time per window, not measure the RTT of every segment.
  - ❖ TCP identifies each byte of data with a 32-bit unsigned.
    - ➢ This is only a problem if the same sequence number *N* is reused within the MSL period, that is, if the network is so fast that sequence number wrap occurs in less than MSL. (solution: PAWS algorithm)

Long Fat Pipes (Cont.)

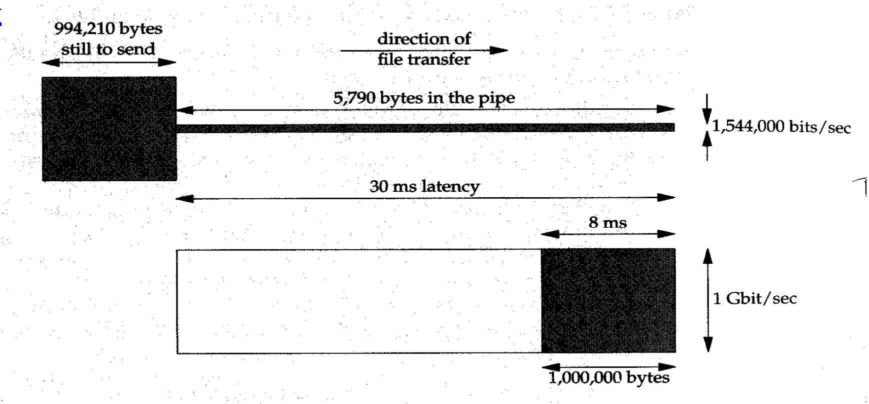


Figure 24.6 Sending a 1-Mbyte file across networks with a 30-ms latency.

Long Fat Pipes (Cont.)

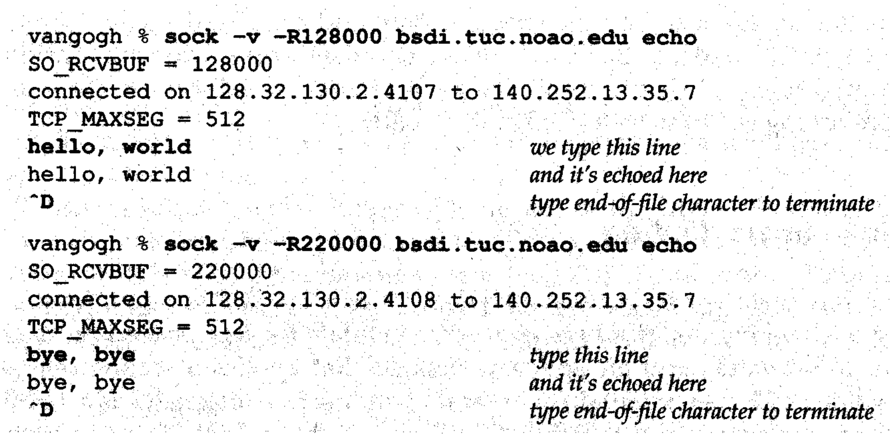
- **The gigabit network the total time to transfer the file (1Mbytes) is :**
  - 0.038 seconds: the 30-ms latency plus the 8 ms for the actual file transfer.
  - 0.034 seconds: the 30-ms latency plus the 4 ms for the actual file transfer (doubling bandwidth decreases the total time by only 10%)
- **The T1 network total time:**
  - 5.211 seconds: the 30 ms latency plus 5181 ms for the actual file transfer.
  - 0.208 seconds (T3): the 30 ms latency plus 178 ms for the actual file transfer.
- **At gigabit speeds are latency limited, not bandwidth limited.**
  - Latency is caused by the speed of light and cannot be decreased.

Window Scale Option

- The window scale option increase the definition of the TCP window from 16 to 32 bits.
- The option can only appear in a SYN segment, therefore the scale factor is fixed in each direction when the connection is established.
- With a shift count of S for sending and a shift count of R for receiving. Receiving from the other end is left shift by R bits. Sending a window advertisement to the other end, take real 32-bit window size and right shift S bits.
- The shift count is automatically chosen by TCP, based on the size of the receive buffer.

Window Scale Option (Cont.)

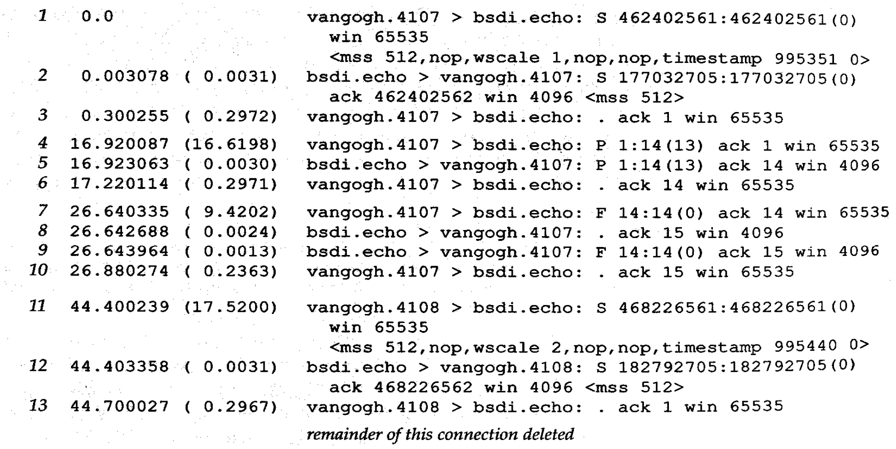
- An example:



```
vangogh % sock -v -R128000 bsdi.tuc.noao.edu echo
SO_RCVBUF = 128000
connected on 128.32.130.2.4107 to 140.252.13.35.7
TCP_MAXSEG = 512
hello, world                        we type this line
hello, world                        and it's echoed here
^D                                  type end-of-file character to terminate

vangogh % sock -v -R220000 bsdi.tuc.noao.edu echo
SO_RCVBUF = 220000
connected on 128.32.130.2.4108 to 140.252.13.35.7
TCP_MAXSEG = 512
bye, bye                            type this line
bye, bye                            and it's echoed here
^D                                  type end-of-file character to terminate
```

Window Scale Option



```
 1   0.0                  vangogh.4107 > bsdi.echo: S 462402561:462402561(0)
                            win 65535
                            <mss 512,nop,wscale 1,nop,nop,timestamp 995351 0>
 2   0.003078 ( 0.0031)   bsdi.echo > vangogh.4107: S 177032705:177032705(0)
                            ack 462402562 win 4096 <mss 512>
 3   0.300255 ( 0.2972)   vangogh.4107 > bsdi.echo: . ack 1 win 65535

 4  16.920087 (16.6198)   vangogh.4107 > bsdi.echo: P 1:14(13) ack 1 win 65535
 5  16.923063 ( 0.0030)   bsdi.echo > vangogh.4107: P 1:14(13) ack 14 win 4096
 6  17.220114 ( 0.2971)   vangogh.4107 > bsdi.echo: . ack 14 win 65535

 7  26.640335 ( 9.4202)   vangogh.4107 > bsdi.echo: F 14:14(0) ack 14 win 65535
 8  26.642688 ( 0.0024)   bsdi.echo > vangogh.4107: . ack 15 win 4096
 9  26.643964 ( 0.0013)   bsdi.echo > vangogh.4107: F 14:14(0) ack 15 win 4096
10  26.880274 ( 0.2363)   vangogh.4107 > bsdi.echo: . ack 15 win 65535

11  44.400239 (17.5200)   vangogh.4108 > bsdi.echo: S 468226561:468226561(0)
                            win 65535
                            <mss 512,nop,wscale 2,nop,nop,timestamp 995440 0>
12  44.403358 ( 0.0031)   bsdi.echo > vangogh.4108: S 182792705:182792705(0)
                            ack 468226562 win 4096 <mss 512>
13  44.700027 ( 0.2967)   vangogh.4108 > bsdi.echo: . ack 1 win 65535
                          remainder of this connection deleted
```

Figure 24.7 Example of window scale option.

Timestamp Option

- The timestamp option lets the sender place a timestamp value in every segment.
- The receiver reflects this value in the acknowledgment, allowing the sender to calculate an RTT for reach received ACK.
- The timestamp is a monotonically increasing value. Since the receiver echoes what it receives, the receiver doesn’t care what the timestamp units are.
- The end doing the active open specifies the option in its SYN. Only if it receives the operation in the SYN from the other end can the option be sent in the future segments.

Timestamp Option (Cont.)

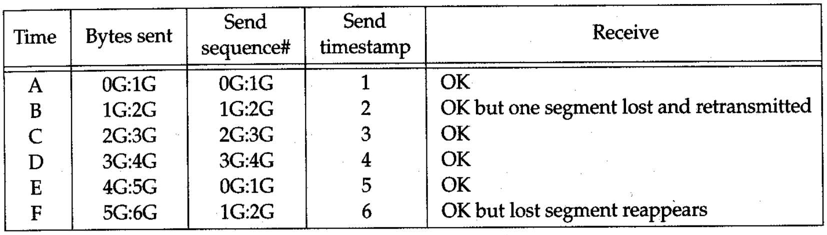
- **To minimize the amount of state maintained by either end, only a single timestamp value is kept per connection. The algorithm to choose when to update this value is simple.**
  - TCP keeps track of the timestamp value to send in the next ACK (a variable named *tsrecent*) and the acknowledgment sequence number from the last ACK that was sent.
  - When a segment arrives, if the segment contains the byte numbered *lastack*, then the timestamp value from the segment is saved in *tsrecent*.
  - Whenever a timestamp option is sent, *tsrecent* is sent as the timestamp echo reply field and the sequence number field is saved in *lastack*.

Timestamp Option (Cont.)

- **Algorithm handles the following two cases:**
  - If ACKs are delayed by the receiver, the timestamp value returned as the echo value will correspond to the *earliest* segment being acknowledged.
  - If a received segment is in-window but out-of-sequence, imply that a previous segment has been lost, when that missing segment is received, its timestamp will be echoed, not the timestamp from the out-of-sequence segment.
- **Timestamp option allows for better RTT calculation, it also provides a way for the receiver to avoid receiving old segments and considering them part of the existing data segment.**

PAWS: Protection Against Wrapped Sequence Number

- **The timestamp option is being used and that the timestamp value assigned by the sender increments by one for each window that is sent. G means a multiple of 1073741824 , j:k means byte j through and including byte k-1.**



Time	Bytes sent	Send sequence#	Send timestamp	Receive
A	0G:1G	0G:1G	1	OK
B	1G:2G	1G:2G	2	OK but one segment lost and retransmitted
C	2G:3G	2G:3G	3	OK
D	3G:4G	3G:4G	4	OK
E	4G:5G	0G:1G	5	OK
F	5G:6G	1G:2G	6	OK but lost segment reappears

Figure 24.8 Transferring 6 gigabytes in six 1-gigabyte windows.

T/TCP: A TCP Extension for Transactions

- **TCP provides a virtual-circuit transport service, there are three distinct phases in the life of a connection: establishment, data transfer, and termination.**
- **A transaction is a client request followed by a server response with the following characteristic:**
  - The overhead of connection establishment and connection termination should be avoid.
  - The latency should be reduced to RTT plus SPT, where RTT is the round-trip time and SPT is the server processing time to handle the request.
  - The server should detect duplicate requests and not reply the transaction when a duplicate request arrives.

T/TCP : A TCP Extension for Transaction (Cont.)

- **The two modifications required for TCP to handle transactions are to avoid the three-way handshake and shorten the TIME_WAIT state.**
- **T/TCP avoids the three-way handshake by using an accelerated open:**
  - It assigns a 32-bit connection count (CC) value to connection it open, either actively or passively.
  - Every segment between two hosts using T/TCP includes a new TCP option named CC.
  - A host maintains a per-host cache of last CC value received in an acceptable SYN segment from that host.
  - When a CC option is received on an initial SYN, the receiver compares the value with the cached value for the sender.

T/TCP: A TCP Extension for Transaction (Cont.)

  - The SYN, ACK segment in response to an initial SYN echoes the received CC value in another new option named CCECHO.
  - The CC value in a non-SYN segment detects and rejects any duplicate segments from previous incarnations of the same connection.
- **The accelerated open avoids the need for a three-way handshake unless either the client or server has crashed and reboot. The cost is that the server must remember the last CC received from each client.**

T/TCP: A TCP Extension for Transaction (Cont.)

- **The TIME_WAIT state is shortened by calculating the TIME_WAIT delay dynamically, based on the measured RTT between the two hosts.**
- **Using these features the minimal transaction sequence is an exchange of three segments:**
  - Client to server, caused by an active open: client-SYN, client-data(the request), client_FIN, and client-CC.
  - Server to client: server -SYN, server-data (reply), server-FIN, ACK of client-FIN, server-CC, and CCECHO of client-CC.
  - Client to server: ACK of server-FIN, which acknowledges the server's SYN, data, and FIN.

T/TCP: A TCP Extension for Transaction (Cont.)

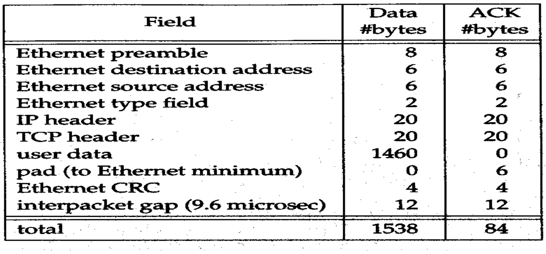
- **Many fine points to the implementation of this TCP option that are covered in the references.**
  - The server's SYN, ACK (the second segment) should be delayed, to allow the reply to piggyback with it.
  - The request can require multiple segments, but the server must handle their possible out-of-order arrival.
  - The API must allow the server process to send data and close the connection in a single operation to allow the FIN in the second segment to piggyback with the reply.
  - The client is sending data in the first segment before receiving an MSS announcement from the server.
  - The client is also sending data to the server without receiving a window advertisement from the server.

T/TCP : A TCP Extension for Transactions (Cont.)

- With the minimal three-segment exchange there is only one RTT that can be measured in each direction. Plus the client’s measured RTT includes the server’s processing time.

- **VMTP, the Versatile Message Transaction Protocol. Unlike T/TCP, which is a small set of extensions to an existing protocol, VMTP is a complete transport layer that uses IP.**
- **VMTP handles error detection, retransmission, and duplicate suppression. It also supports multicast communication.**

TCP Performance



Field	Data #bytes	ACK #bytes
Ethernet preamble	8	8
Ethernet destination address	6	6
Ethernet source address	6	6
Ethernet type field	2	2
IP header	20	20
TCP header	20	20
user data	1460	0
pad (to Ethernet minimum)	0	6
Ethernet CRC	4	4
interpacket gap (9.6 microsec)	12	12
total	1538	84

$$throughput = \frac{2 \times 1460 \text{ bytes}}{2 \times 1538 + 84 \text{ bytes}} \times \frac{10,000,000 \text{ bits/sec}}{8 \text{ bits/byte}} = 1,155,063 \text{ bytes/sec}$$

$$throughput = \frac{22 \times 1460 \text{ bytes}}{22 \times 1538 + 84 \text{ bytes}} \times \frac{10,000,000 \text{ bits/sec}}{8 \text{ bits/byte}} = 1,183,667 \text{ bytes/sec}$$

TCP Performance (Cont.)

- **The following practical limits apply for any real-world scenario.**
  - You can’t run any fast then the speed of the slowest link.
  - You can’t go any faster than the memory bandwidth of the slowest machine.
  - You can’t go any faster than the window size offered by the receiver, divided by the round-trip time. (BW = Window/RTT = 1.073Gbytes/RTT)
- **How fast TCP can run is determined by the size of the TCP window (1.073 Gbytes) and the speed of light.**
- **Many protocol performance problems are implementation deficiencies rather than inherent protocol limits.**

Summary

- **Path MTU discovery allows TCP to use windows larger than the default of 536 for nonlocal connections, when the path MTU is larger.**
- **The window scale option takes the maximum TCP windows size from 65535 bytes to just over 1 gigabytes.**
- **The timestamp option lets more segment be accurately timed and also lets the receiver provide protection against wrapped sequence numbers (PAWS).**
- **T/TCP allow a client-server request-reply sequence to be completed using only three segments in the usual case.**
- **TCP performance is limited only by the maximum 1-gigabyte window and the speed of light.**