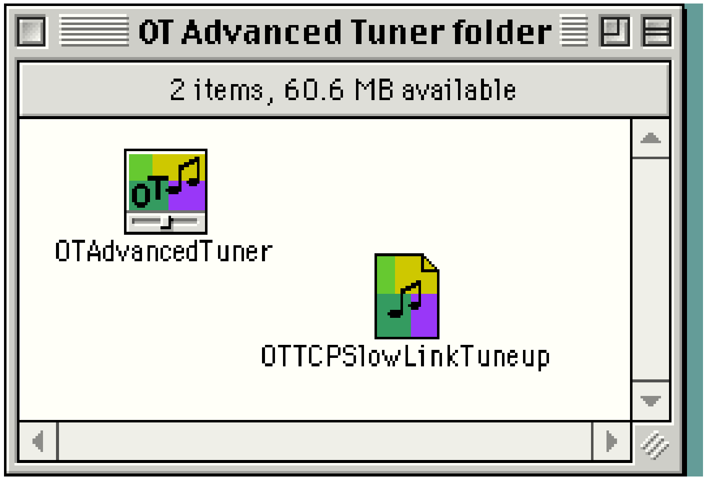
Tuning Open Transport

Session C18

Speaker:
Peter Sichel
Chief Engineer
Sustainable Softworks
psichel@sustworks.com

Thursday, July 9 • 10:00 a.m. - 11:30 a.m.



Introduction

Speaker Background

Macintosh developer specializing in OT for the last two years

- IPNetMonitor Macintosh Internet Tools
- OT Advanced Tuner
- IPNetRouter OT Native Software IP Router
- 14 years prior experience as a software architect at Digital Equipment Corporation. Emphasis on user interface design and communications.

Overview

What is this session about:

- Open Transport and TCP/IP tuning
- Some background on the structure of OT and TCP/IP protocols

Outline:

- Why is it necessary to tune TCP/IP in the first place?
- Open Transport background
- TCP/IP tunable parameters
- Some practical tuning examples
- Tools demo
- Suggested References
- Questions & Answers

Why is it necessary to tune TCP/IP?

TCP/IP is not controlled by any manufacturer and is designed to work on almost any kind of underlying network with widely varying characteristics.

To meet this challenge, the protocol designers made TCP adaptive. TCP is self monitoring and optimizes its own behavior to match the network environment. Adaptation takes time and the default settings cannot optimize for every possible kind of network.

There are many independent implementations, the protocols continue to evolve in response to practical experience, some implementations have peculiar compatibility constraints.

Open Transport background

Based on Mentat Portable Streams and Mentat/TCP.

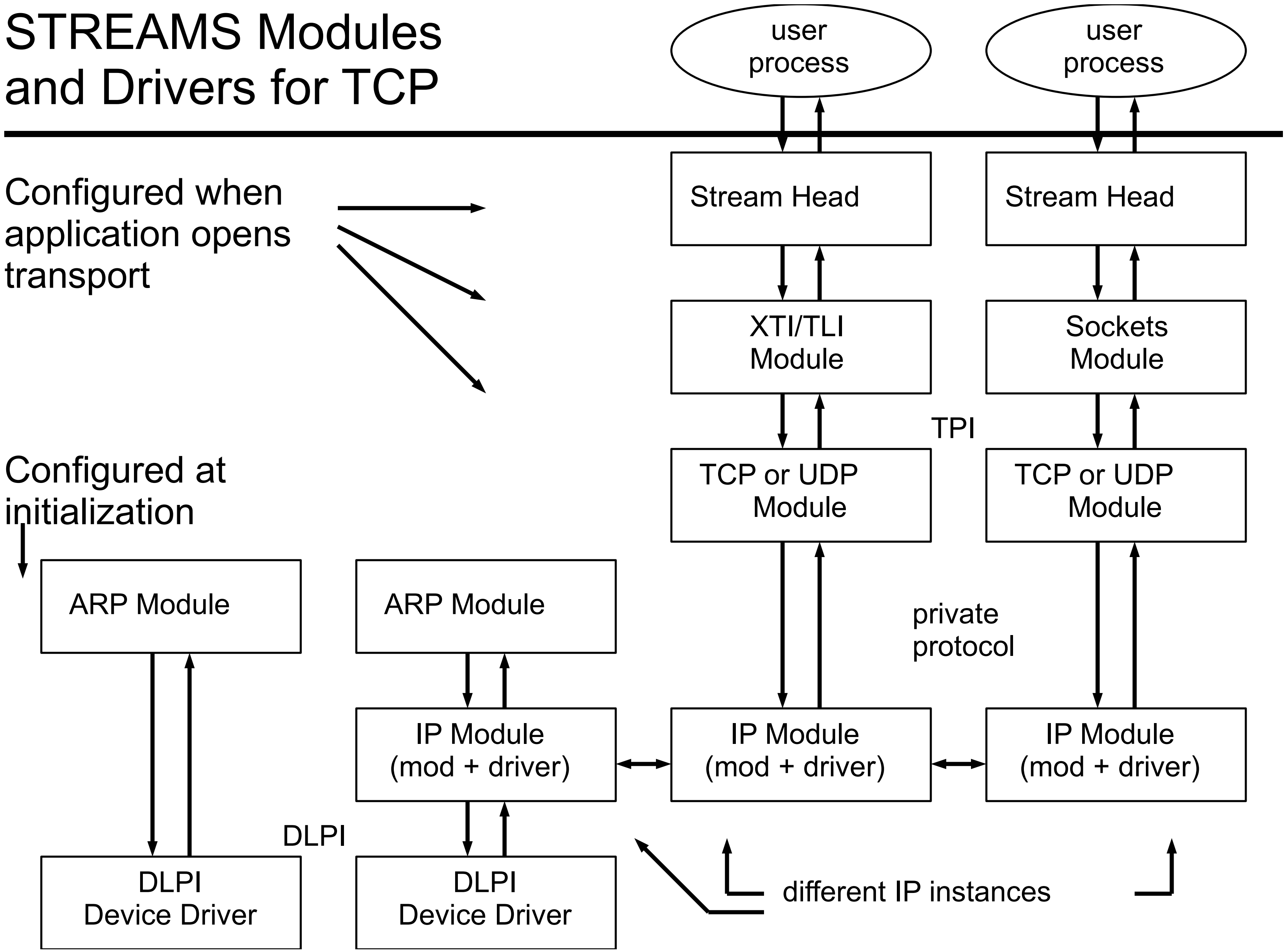
Best-in-class implementation used in Solaris, Digital UNIX, HP-UX 11 and other UNIX workstations.

STREAMS based design provides a modern plug-in architecture for networking. Protocol stacks were designed to be fully tunable on-the-fly.

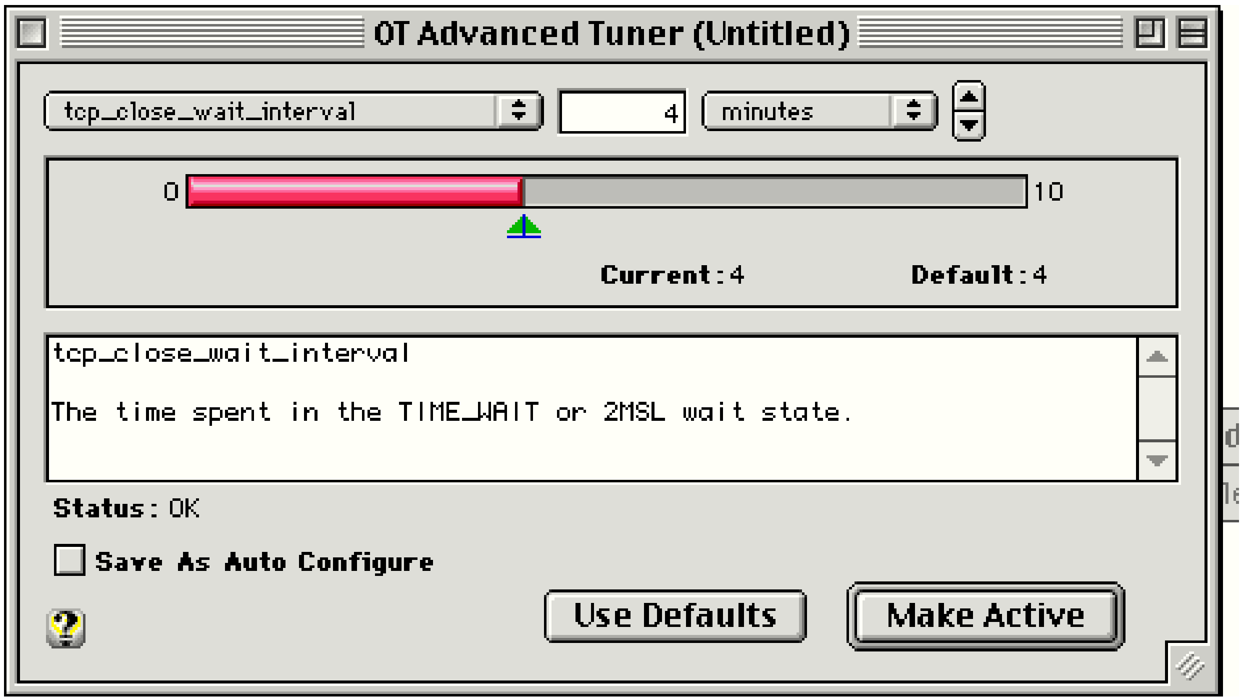
Apple tried to make TCP/IP networking simple.

The OT Advanced Tuner allows you to access the details.

STREAMS Modules and Drivers for TCP

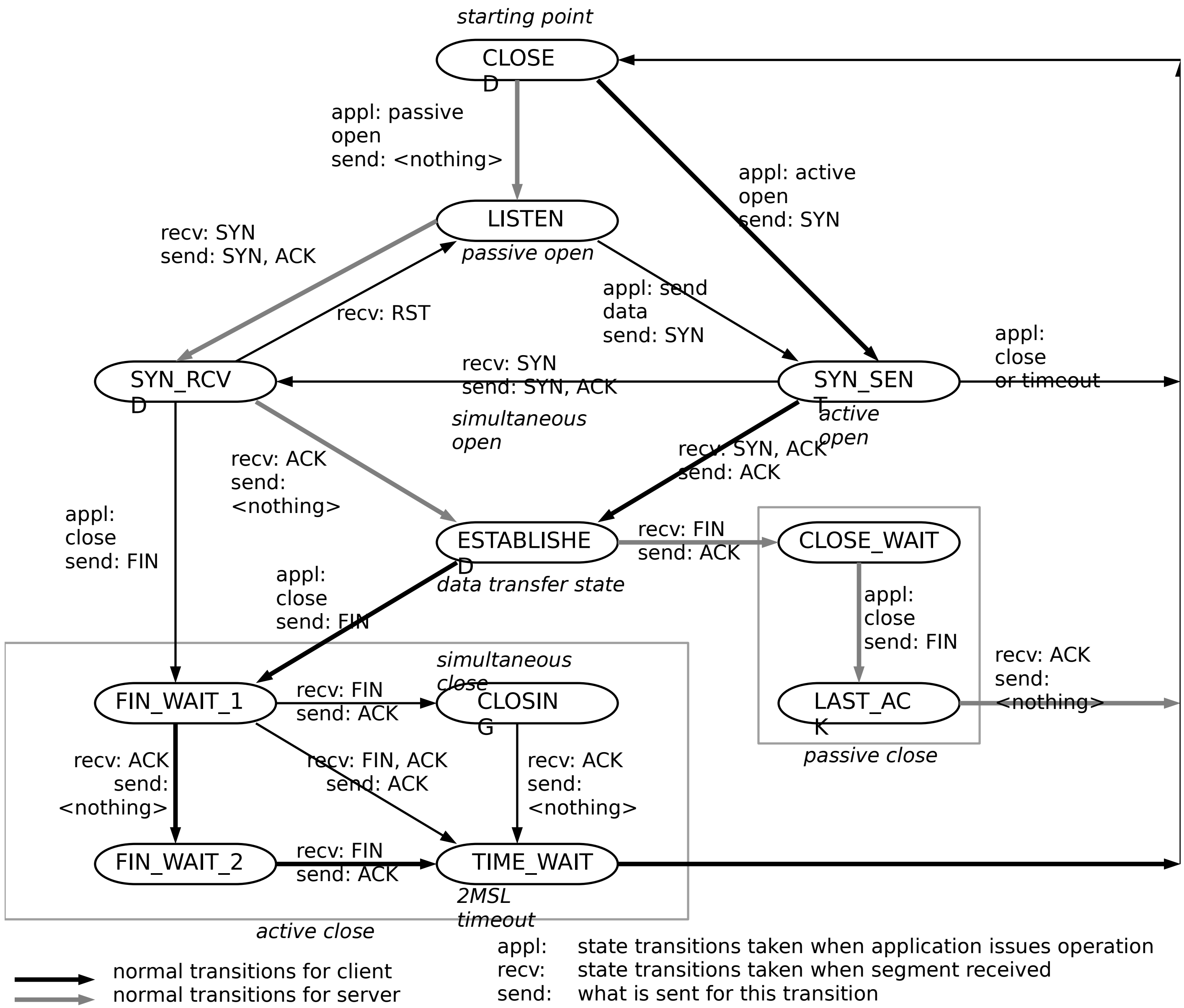


The OT Advanced Tuner allows you to access the details



<http://www.sustworks.com>

Tunable Parameters


starting point
CLOSED
appl: passive open send: <nothing>
appl: active open send: SYN
LISTEN
passive open
recv: SYN send: SYN, ACK
recv: RST
appl: send data send: SYN
SYN_RCVD
recv: SYN send: SYN, ACK
simultaneous open
SYN_SENT
active open
appl: close or timeout
recv: ACK send: <nothing>
recv: SYN, ACK send: ACK
appl: close send: FIN
ESTABLISHED
data transfer state
recv: FIN send: ACK
CLOSE_WAIT
appl: close send: FIN
appl: close send: FIN
simultaneous close
FIN_WAIT_1
recv: FIN send: ACK
CLOSING
LAST_ACK
recv: ACK send: <nothing>
passive close
recv: ACK send: <nothing>
recv: FIN, ACK send: ACK
recv: ACK send: <nothing>
FIN_WAIT_2
recv: FIN send: ACK
TIME_WAIT
2MSL timeout
active close
normal transitions for client
normal transitions for server
appl: state transitions taken when application issues operation
recv: state transitions taken when segment received
send: what is sent for this transition

Some practical tuning examples

Slow Link Tune Up

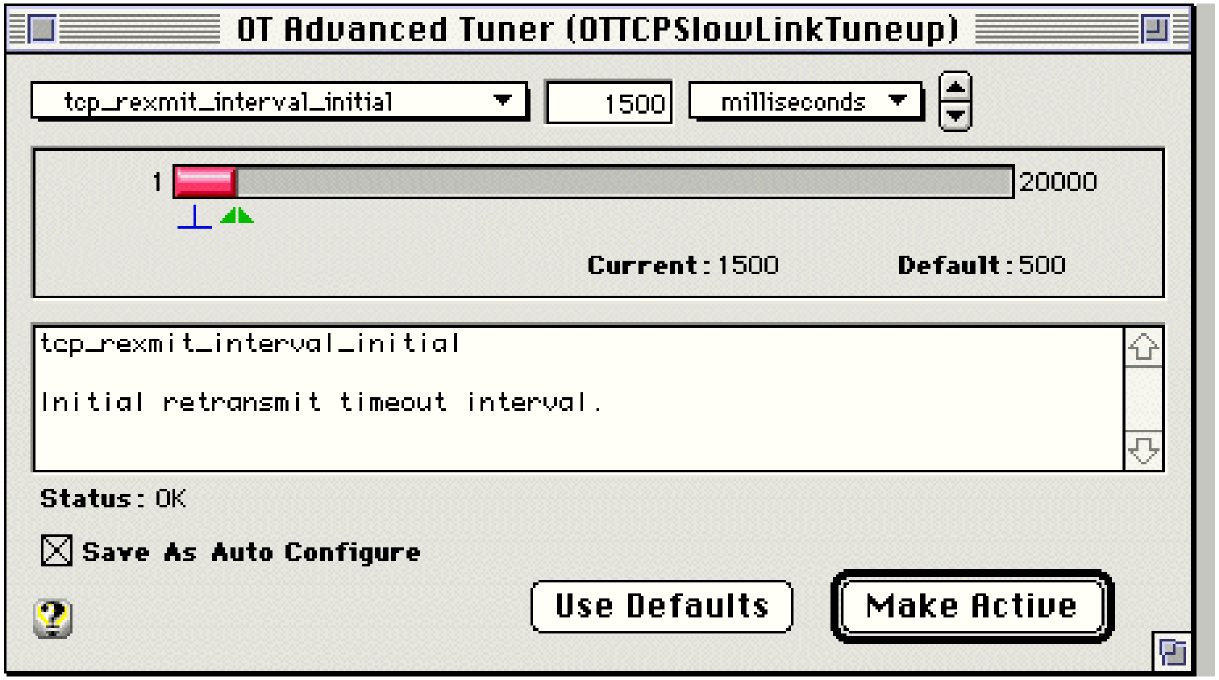
TCP Window Size

How to Download and Surf at the same time

Match the network MTU

Use a monitoring tool

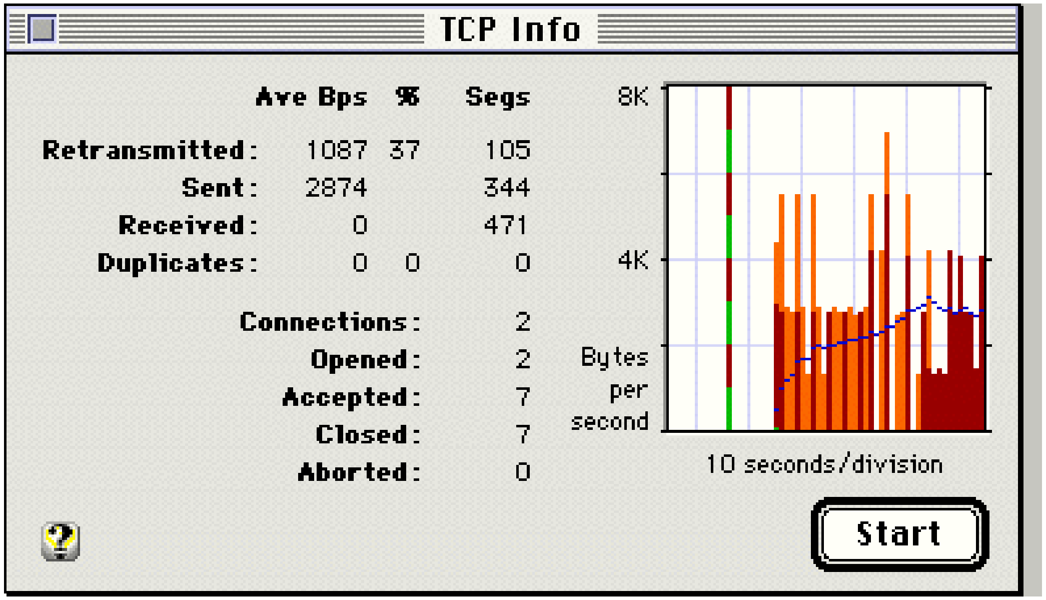
Slow Link Tune Up (1)



OT Advanced Tuner screen showing modified "tcp_rexmit_interval_initial".
One of the settings specified in the OT slow link tuneup settings document.

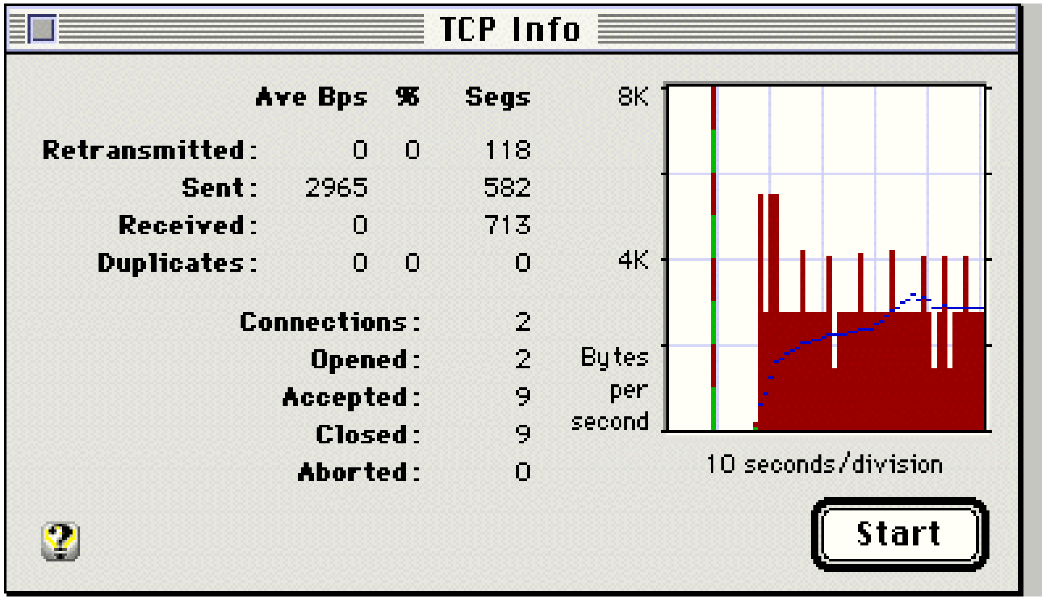
```
#auto
tcp_rexmit_interval_initial=1500
tcp_rexmit_interval_min=1500
tcp_conn_grace_period=1500
#end
```

Slow Link Tune Up (2)

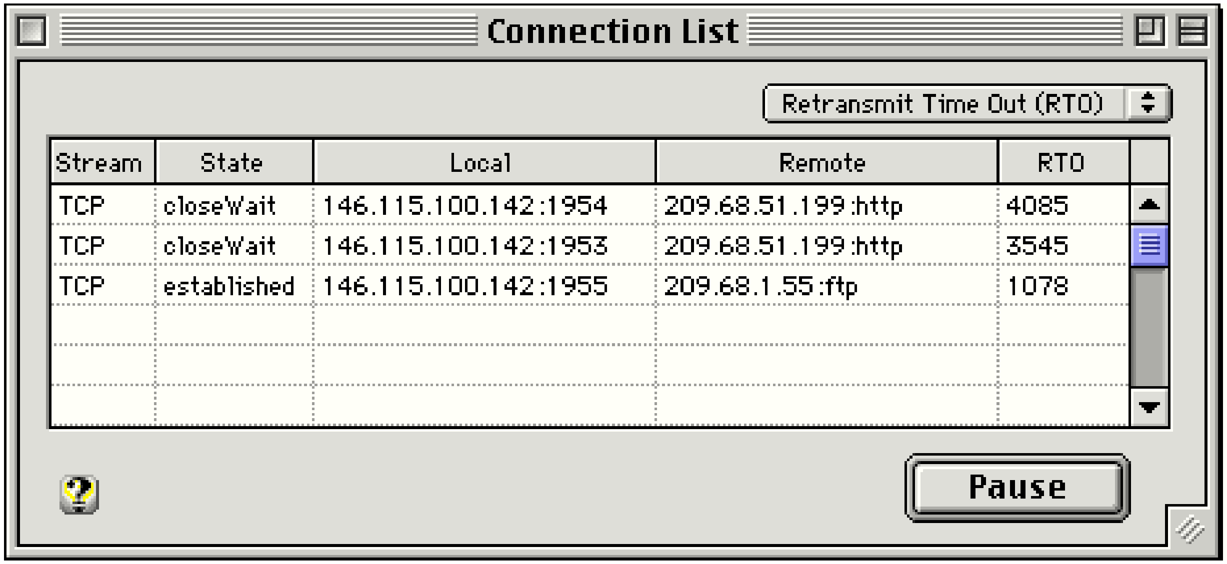


IPNetMonitor TCP Info window showing the start of an FTP upload over a 28.8k dial-up connection. The orange bars indicate retransmitted data is wasting much of the available bandwidth because the initial retransmit timeout interval is too small.

Slow Link Tune Up (3)



IPNetMonitor TCP Info window showing the same FTP upload after applying the TCP Slow Link Tuneup settings. The unnecessary retransmission has been eliminated by increasing the TCP retransmit timeout interval.



IPNetMonitor TCP Connection List window showing actual time out intervals (msec).

TCP Window Size (1)

The receiver tells the sender its receive buffer size.

The sender may transmit additional segments without waiting for an acknowledgement up to the receive buffer size.

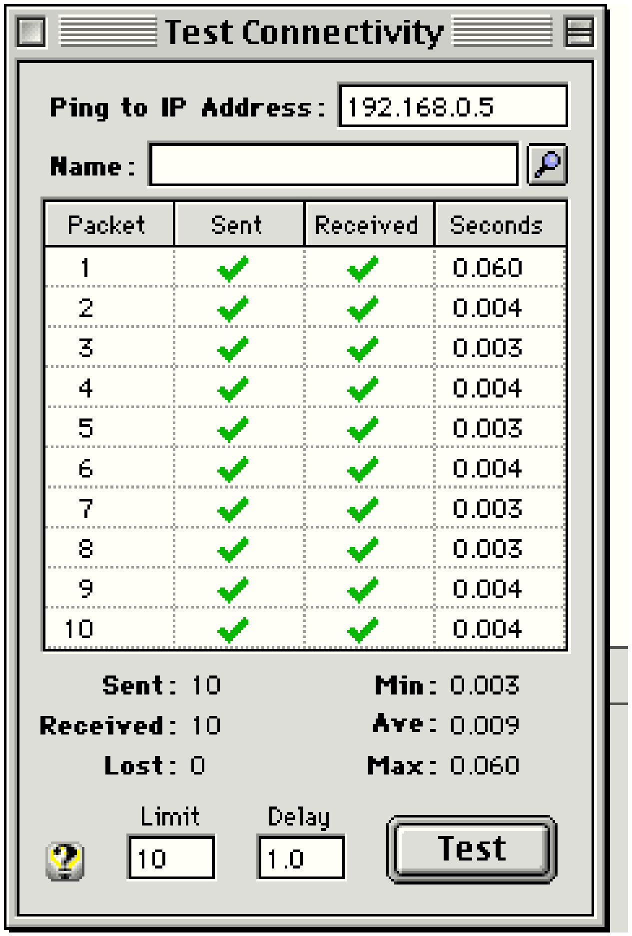
The capacity of the pipe is equal to the *bandwidth-delay product*.

$$\textit{capacity}\ \text{(bits)} = \textit{bandwidth}\ \text{(bits/sec)} \cdot \textit{round-trip time}\ \text{(sec)}$$

We want the window to be large enough that we can keep the pipe full.

The basic rule is you can't go any faster than the window size of the receiver divided by the round trip time [Stevens, "TCP/IP Illustrated", p356].

TCP Window Size



10Base-T Ethernet

Bandwidth 10 Mbps

Round trip time
approximately .004s

Capacity (bits) = 10,000,000
· .004
= 40,000 bits

Window Size = 40,000 / 8
= 5000 bytes

OT default MSS = 536
tcp_rwin_mss_multiplier = 10

Ethernet best case MSS = 1500
tcp_rwin_mss_multiplier = 4;

TCP Window Size (3)

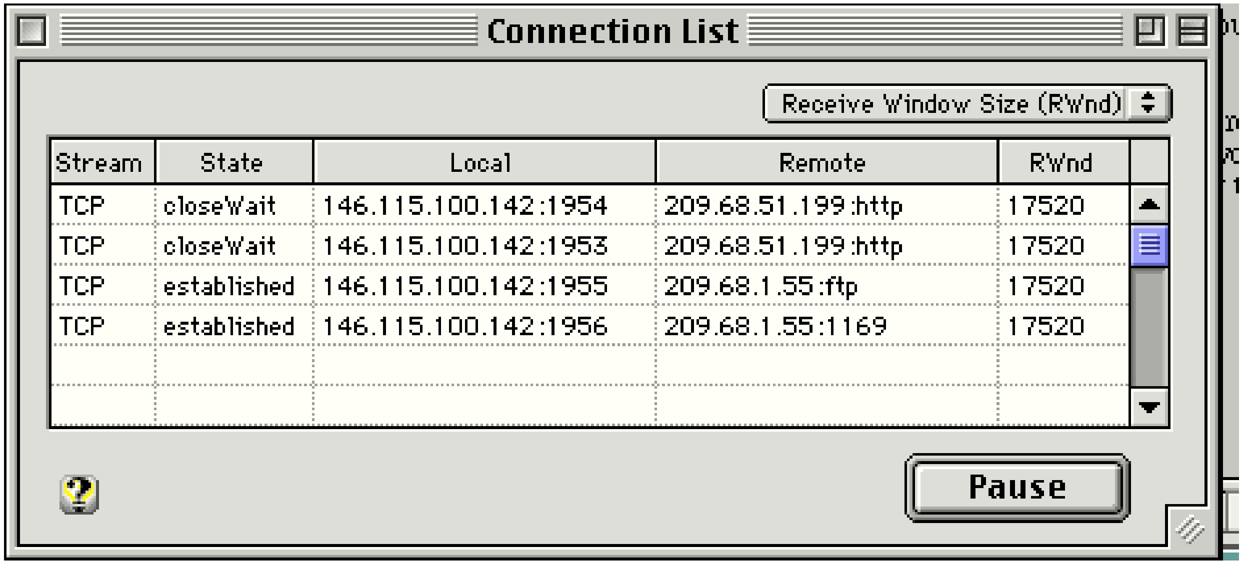
Cable Modems

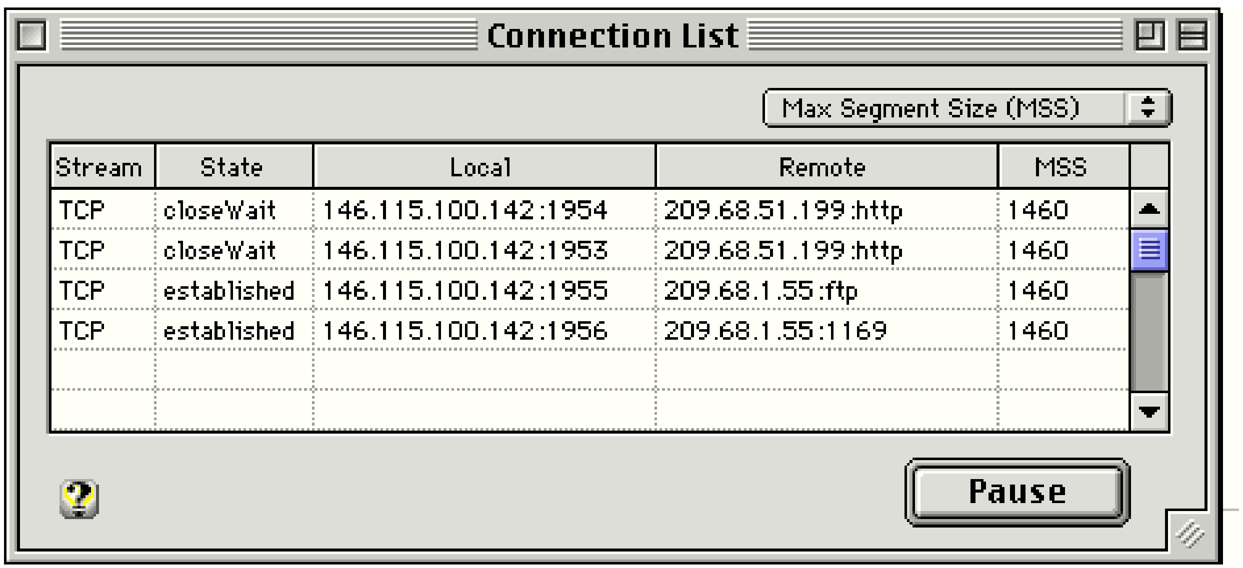
Cable modem connections can be highly asymmetric (much faster downstream than upstream).

Receive window may fill faster than acknowledgements can reach the sender.

Can use IPNetMonitor to test the round trip time and monitor actual performance as the TCP Receive Window is adjusted.

Bandwidth 10 Mbps
Round trip time approximately .080s
Capacity (bits) = 10,000,000 · .080 = 800,000 bits
Window Size = 800,000 / 8 = 100K bytes
tcp_rwin_mss_multiplier = 66;


Connection List
Receive Window Size (RWnd)
Stream State Local Remote RWnd
TCP closeWait 146.115.100.142:1954 209.68.51.199:http 17520
TCP closeWait 146.115.100.142:1953 209.68.51.199:http 17520
TCP established 146.115.100.142:1955 209.68.1.55:ftp 17520
TCP established 146.115.100.142:1956 209.68.1.55:1169 17520
Pause


Connection List
Max Segment Size (MSS)
Stream State Local Remote MSS
TCP closeWait 146.115.100.142:1954 209.68.51.199:http 1460
TCP closeWait 146.115.100.142:1953 209.68.51.199:http 1460
TCP established 146.115.100.142:1955 209.68.1.55:ftp 1460
TCP established 146.115.100.142:1956 209.68.1.55:1169 1460
Pause

How To Browse While Downloading

Ever try to browse while using FTP to download over a 28.8 modem? The download hogs the connection making browsing unresponsive.

We can see why this happens from the IPNetMonitor Connection List Window:
MSS = 1460 (note MSS = MTU - 40)
RWnd = 17520 (tcp_rwin_mss_multiplier = 12)

A 28.8K modem might yield a sustained download rate of 3Kbps for already compressed files. With a receive window of 17520, this means up to 6 seconds of FTP data may be in flight -- meaning that URL clicks are waiting behind 6 seconds worth of FTP data.

We can use the OT Advanced Tuner to adjust these settings for better responsiveness **AND** better download performance at the same time!

Show me the numbers!

Using the OT Advanced Tuner to set the *tcp_mss_max* and *tcp_rwin_mss_multiplier* parameters (these results were contributed by Alan Charlesworth).

Maximum Segment Size (MSS)	MSS Mult	Receive window(RWnd)	Measured Download B/W	Measured Ping time to another site	Calculated Amount of FTP data in flight
1,460 bytes	12	17,520 bytes	1.8 KBps	4.4 sec	9.5 sec worth
1,460	9	13,140	2.7	4.0	4.8
1,460	6	8,760	2.8	2.6	3.0
1,460	3	4,380	2.7	1.1	1.6
536	8	4,288	2.7	1.2	1.6
268	16	4,288	2.7	1.4	1.6
536	6	3,216	2.7	1.0	1.6
1,460	2	2,920	2.7	0.7	1.1
536	4	2,144	2.7	0.5	0.8
268	8	2,144	2.6	0.7	0.8
1,460	1	1,460	2.1	0.3	0.7
536	2	1,072	2.4	0.3	0.4
268	4	1,072	2.5	0.3	0.4
536	1	536	1.3	0.2	0.4
268	2	536	1.4	0.2	0.4
268	1	268	0.9	0.2	0.3

The first line is Open Transports default settings which give large packets and a large window. It delivers both poor bandwidth and poor interactivity over a modem connection.

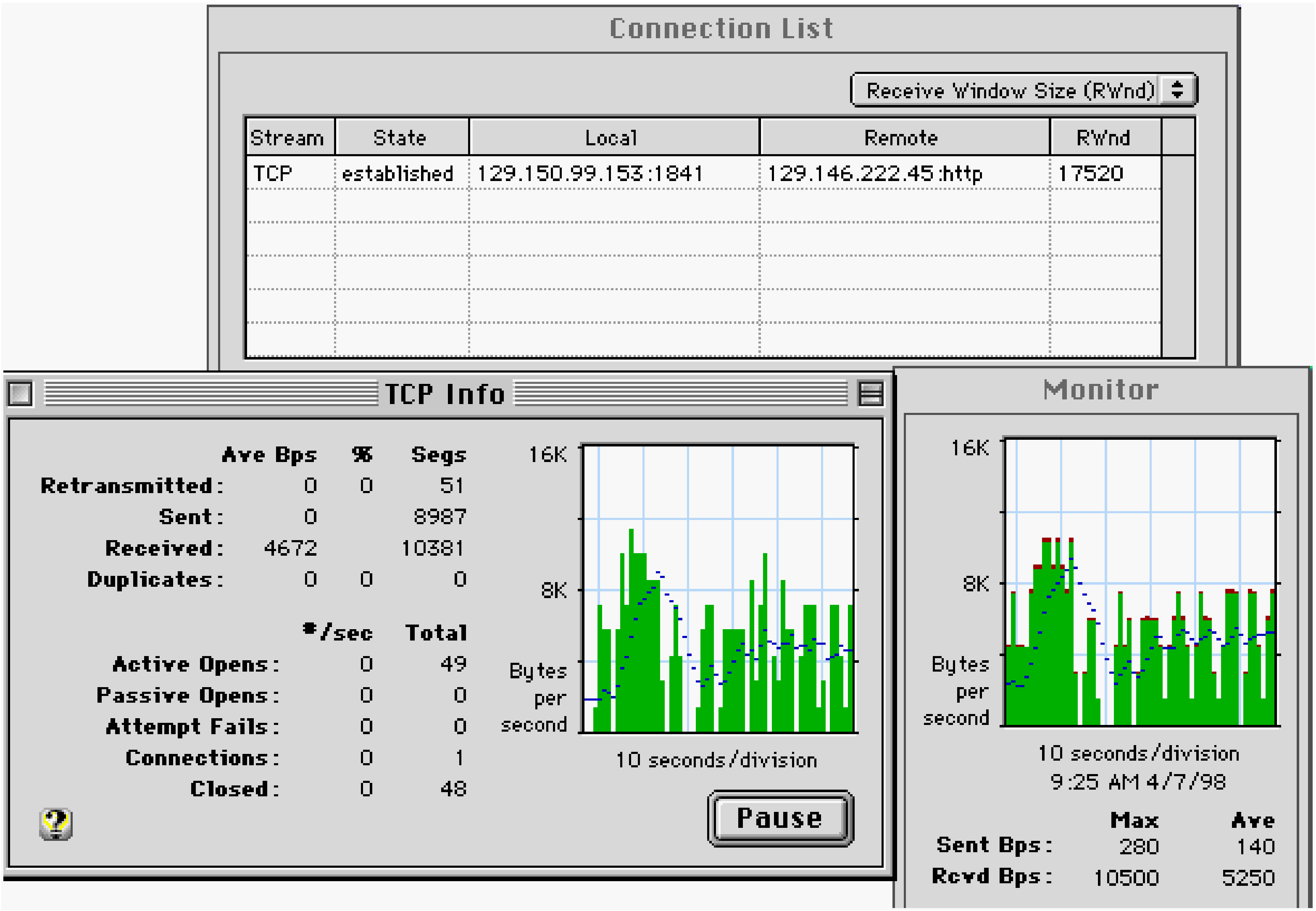
Why?

A fast Ethernet connection would benefit from long packets and a large receive window, but modems are so slow that smaller settings are a better match. A short 500 byte packet still takes a long 0.15 seconds to move through a 31K modem connection.

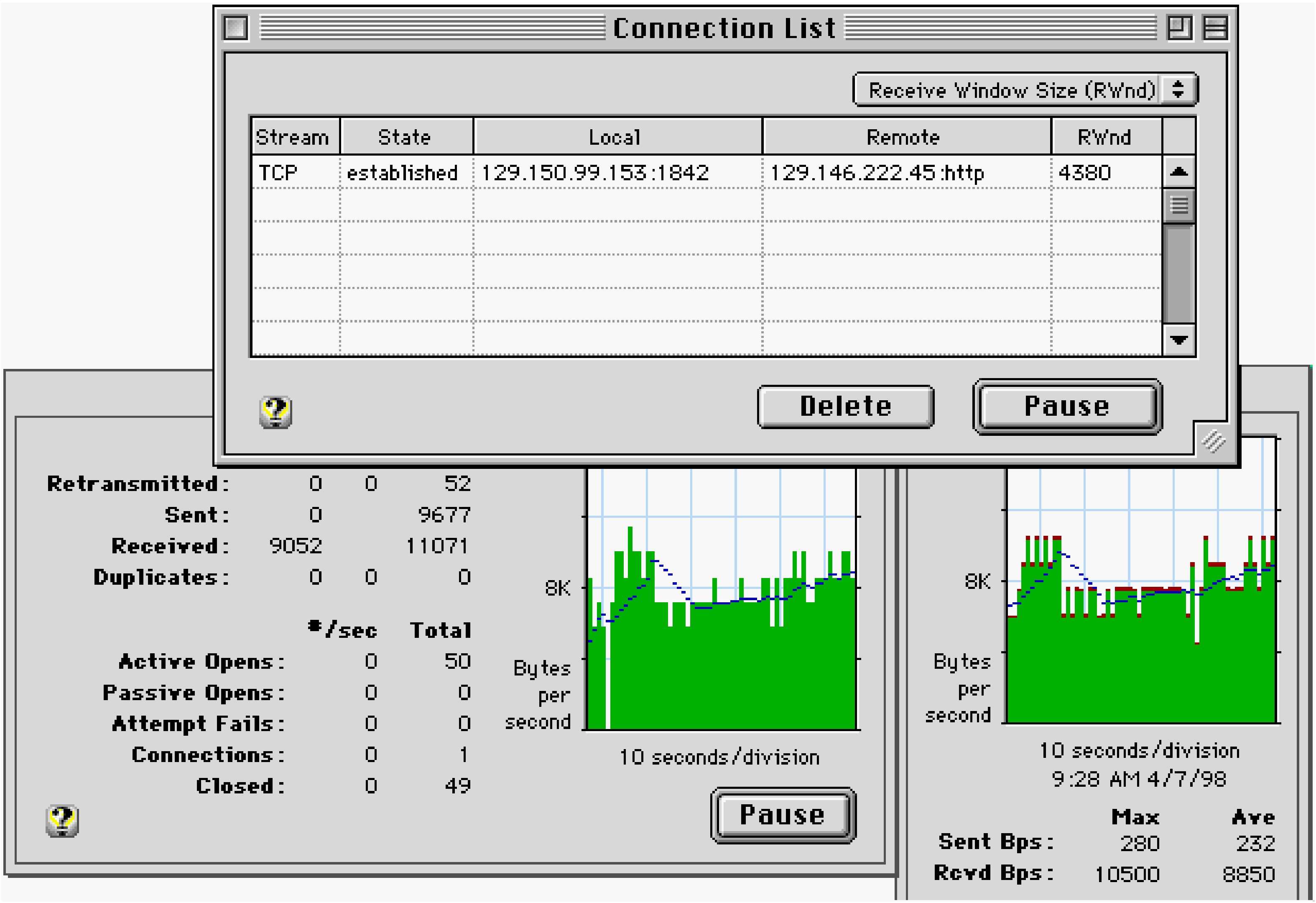
You only need a receive window that is big enough to let a sufficient amount of data be in flight to cover the latency to the target site. It appears that a second's worth of data is about enough for a modem connecton. This gives good combination of download speed and interactive responsiveness.

The low FTP bandwidth observed when using the OT defaults is probably caused by duplicate packets, presumably because the long time through the receive window pipeline has caused the sender to time out and resend.

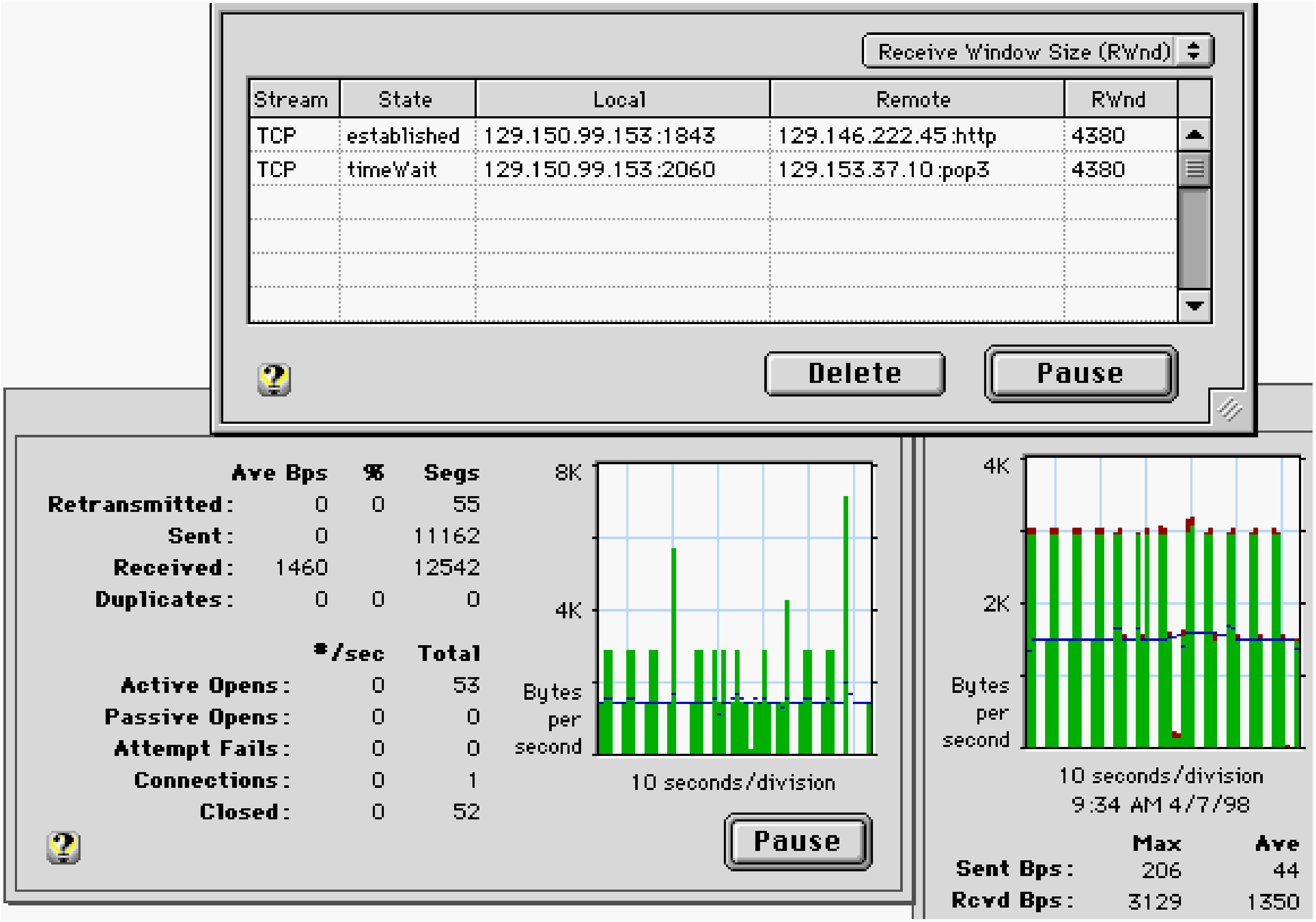
Bad 1460 x 12 PS



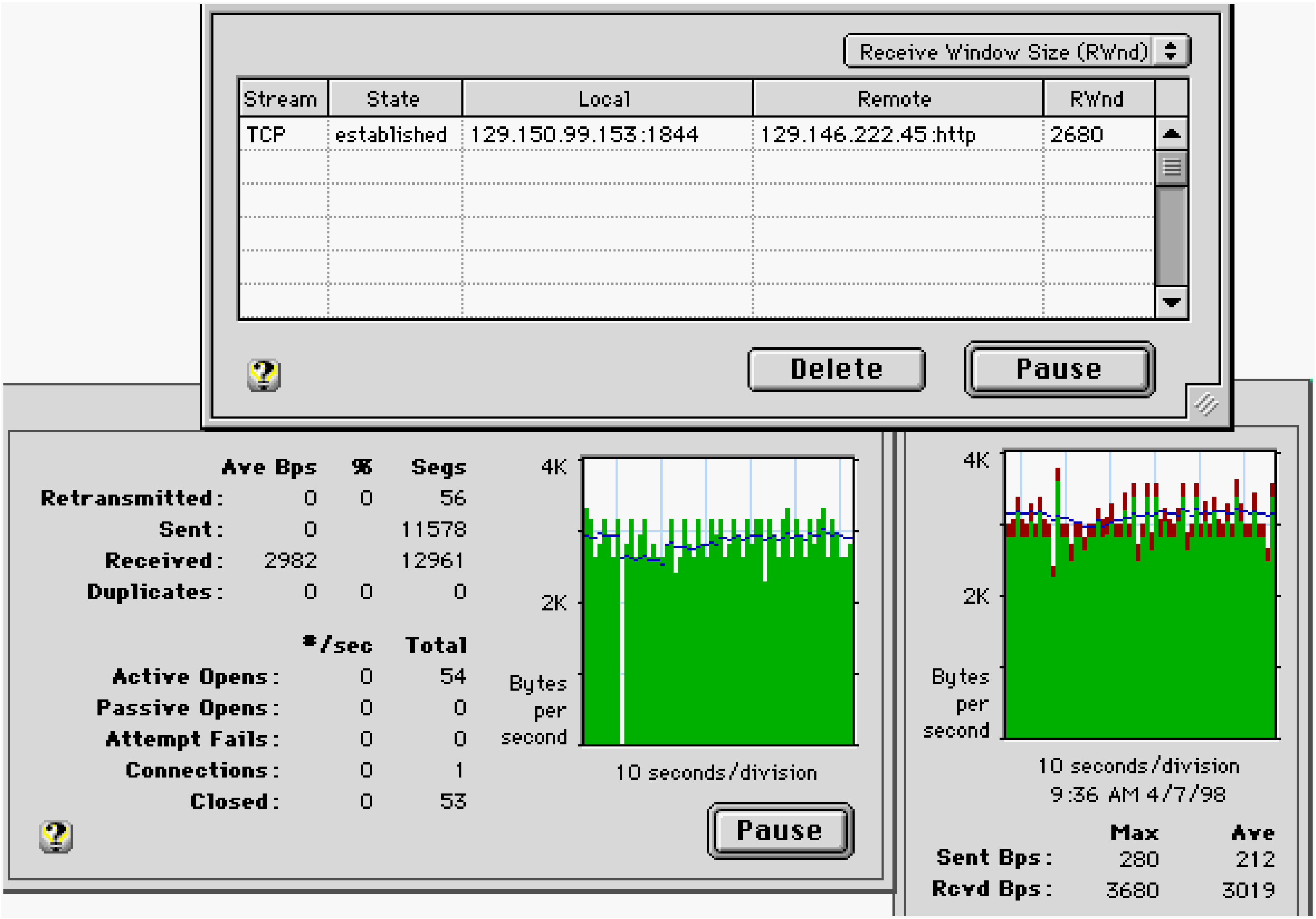
Good 1460 x 3 PS



Bad 1460 x 3 Z


Receive Window Size (RWnd)
Stream
State
Local
Remote
RWnd
TCP
established
129.150.99.153:1843
129.146.222.45:http
4380
TCP
timeWait
129.150.99.153:2060
129.153.37.10:pop3
4380
Delete
Pause
Ave Bps
%
Segs
Retransmitted:
0
0
55
Sent:
0
11162
Received:
1460
12542
Duplicates:
0
0
0
#/sec
Total
Active Opens:
0
53
Passive Opens:
0
0
Attempt Fails:
0
0
Connections:
0
1
Closed:
0
52
8K
4K
Bytes per second
10 seconds/division
Pause
4K
2K
Bytes per second
10 seconds/division
9:34 AM 4/7/98
Max
Ave
Sent Bps:
206
44
Rcvd Bps:
3129
1350

Good 536 x 5 Z



I Use These Settings

```
#auto
tcp_mss_max=536
tcp_rwin_mss_multiplier=6
```

```
tcp_rexmit_interval_initial=3000
tcp_rexmit_interval_min=3000
tcp_conn_grace_period=3000
#end
```

You can download this tuner file from
<http://www.sustworks.com>

Of course, these conclusions are for modem connections, and would change for connections like ISDN, ADSL, or cable modems with different latency and bandwidth characteristics.

Match the Network MTU

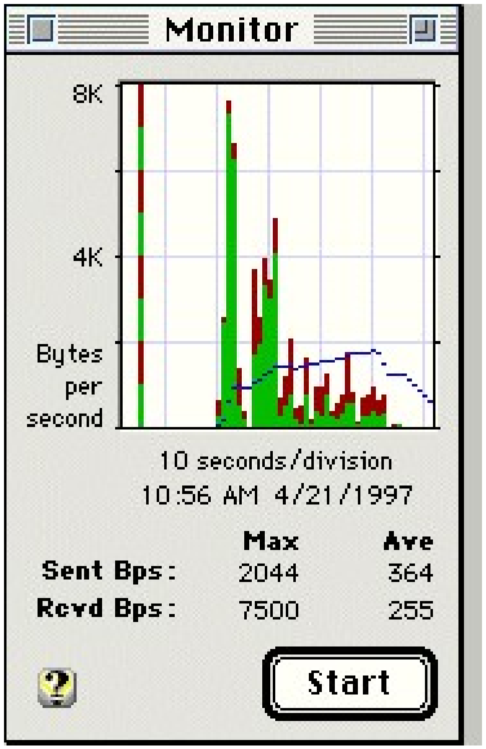
The MTU or Maximum Transmission Unit defines the largest segment or packet that can transmitted across a given network (without being split into smaller segments).

For best performance, we normally want to use the largest packet size possible to reduce per packet overhead.

Open Transport uses an automatic path MTU discovery technique, but if your network connection requires a specific MTU, it may be necessary to set this explicitly for best performance or compatibility.

> “I've been using OT Tuner for a few months now, and it's a life saver. I use a Cisco 766 router to connect to my ISP, and it insists on sending larger blocks than my ISP's router (a US Robotics Total Control) wants to receive. Thus it never sees the end of a long packet, and I'm limited to sending files or email messages of around 1500 bytes or smaller.
>
> With OT Tuner, I'm able to set the Maximum Segment Size, and everything works well….”

Monitoring Tools



IPNetMonitor
<http://www.sustworks.com>

AG Group
<http://www.aggroup.com>

Neon Software
<http://www.neon.com>

What The Tools Look At

Ethernet packets

IP traffic

TCP duplicate and retransmitted data

TCP throughput

Round Trip Time

SNMP MIB-II statistics (RWnd, RTO, MSS,...)

Suggested References

For more information on how TCP/IP works and TCP tuning parameters, may I suggest :

"TCP/IP Illustrated, Volume 1" by W. Richard Stevens;

Appendix E, "Configurable Options" describes the tunable parameters available under Solaris which are mostly the same as Open Transport. An updated version of this appendix is available from Stevens web page at:

<http://www.kohala.com/~rstevens/tcpipiv1.appe.update1.ps>

This can provide a good starting point for understanding many of the tunable parameters you can set using the OT Advanced Tuner. [Notice this is a Postscript file.]

Another excellent reference for understanding TCP/IP is:

“Internetworking With TCP/IP, Volume 1”, by Douglas E. Comer

Questions & Answers