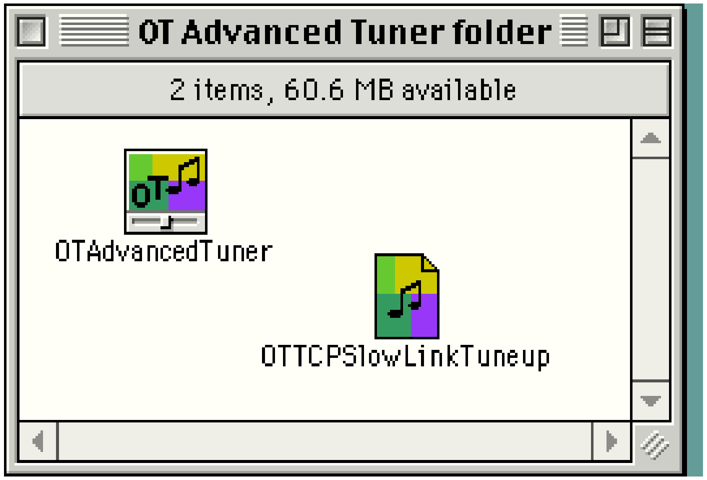
Tuning Open Transport

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.

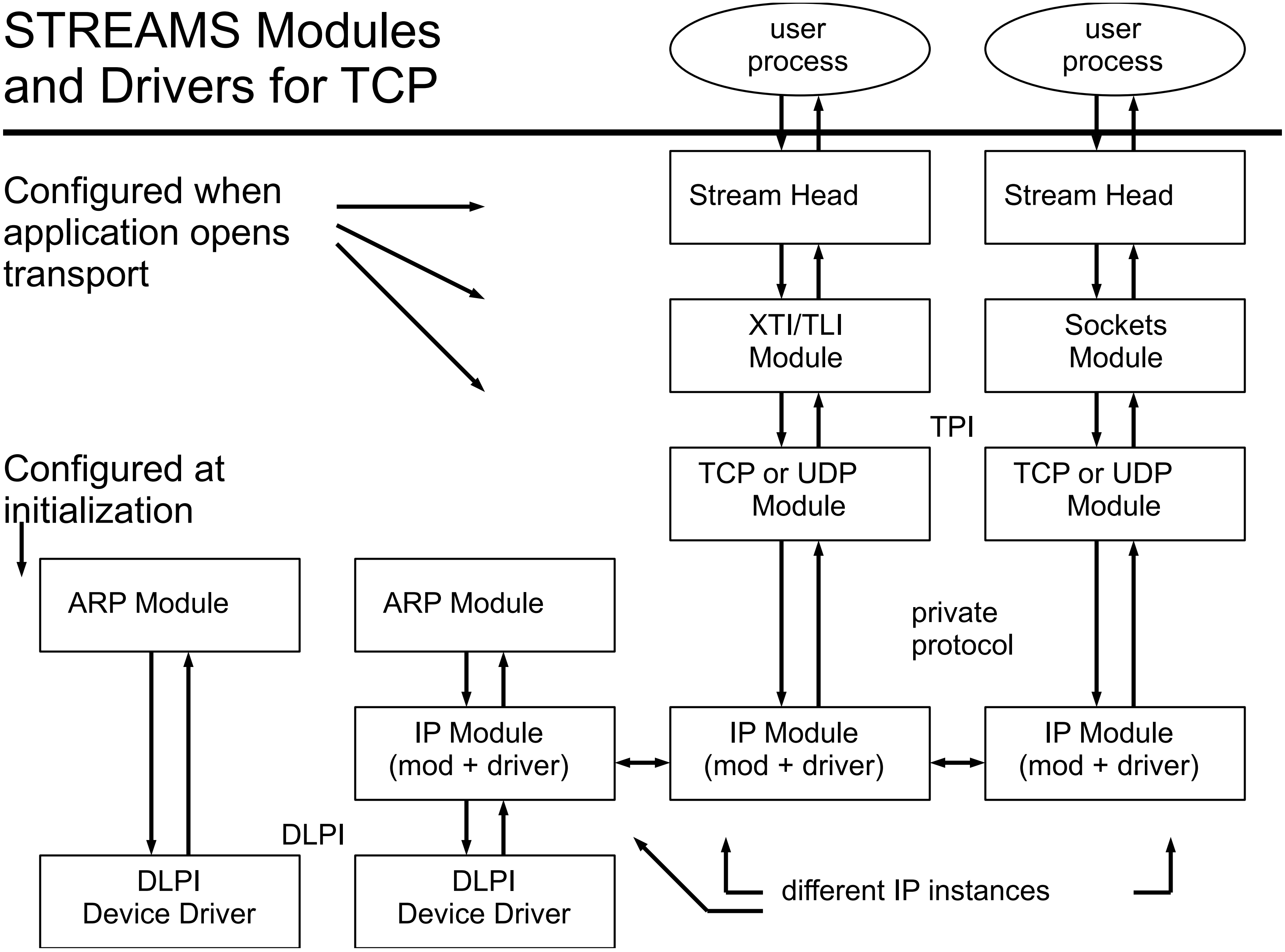
Best-in-class implementation used in Solaris, Digital UNIX, HP-UX 11 and other UNIX workstations.

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



Configured when application opens transport

Configured at initialization

user process

user process

Stream Head

Stream Head

XTI/TLI Module

Sockets Module

TPI

TCP or UDP Module

TCP or UDP Module

private protocol

ARP Module

ARP Module

IP Module (mod + driver)

IP Module (mod + driver)

IP Module (mod + driver)

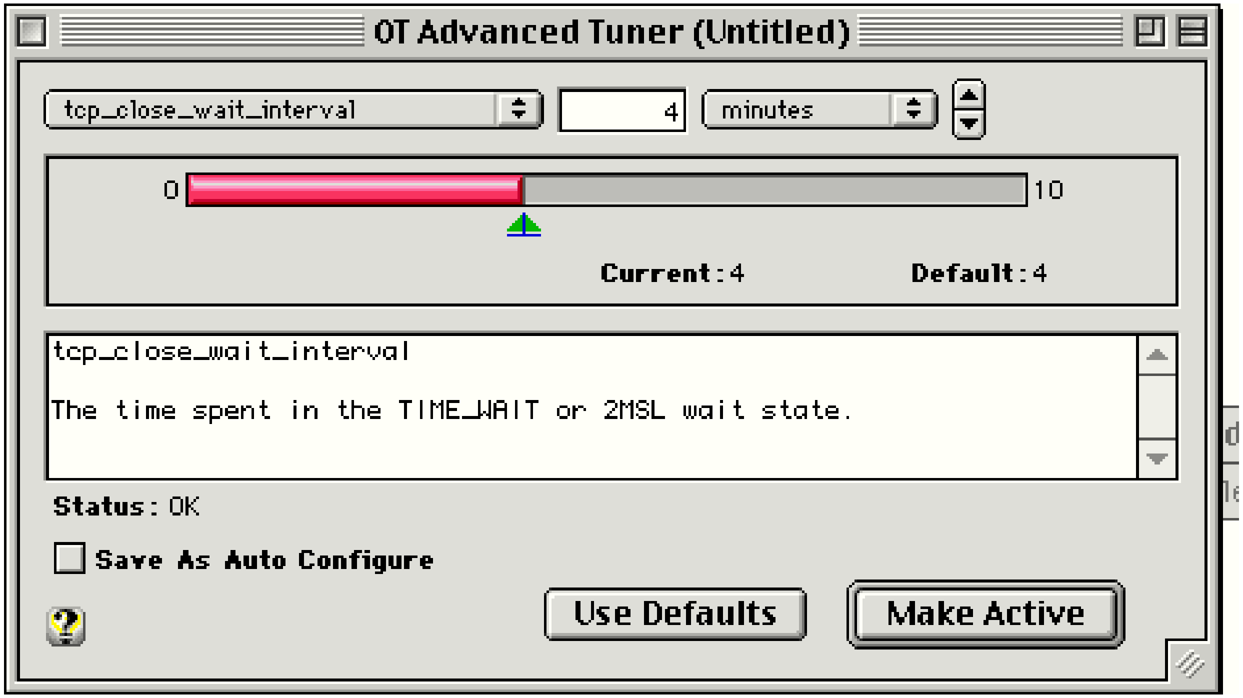
DLPI

DLPI Device Driver

DLPI Device Driver

different IP instances

The OT Advanced Tuner allows you to access the details



tcp_close_wait_interval 4 minutes

Current: 4 Default: 4

tcp_close_wait_interval

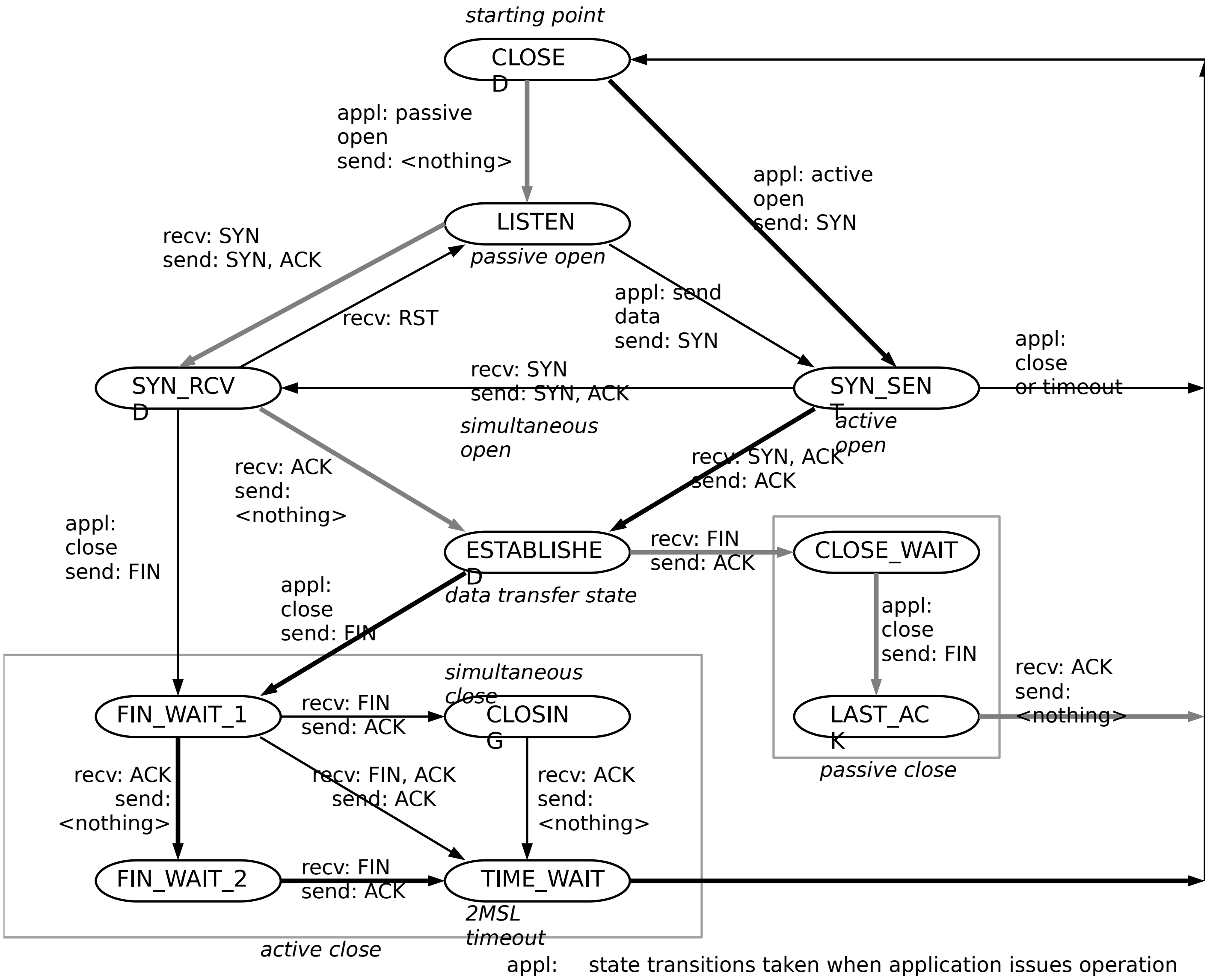
The time spent in the TIME_WAIT or 2MSL wait state.

Status: OK

□ Save As Auto Configure

<http://www.sustworks.com>

Tunable Parameters



appl: state transitions taken when application issues operation
recv: state transitions taken when segment received
send: what is sent for this transition

normal transitions for client
normal transitions for server

Some practical tuning examples

Slow Link Tune Up

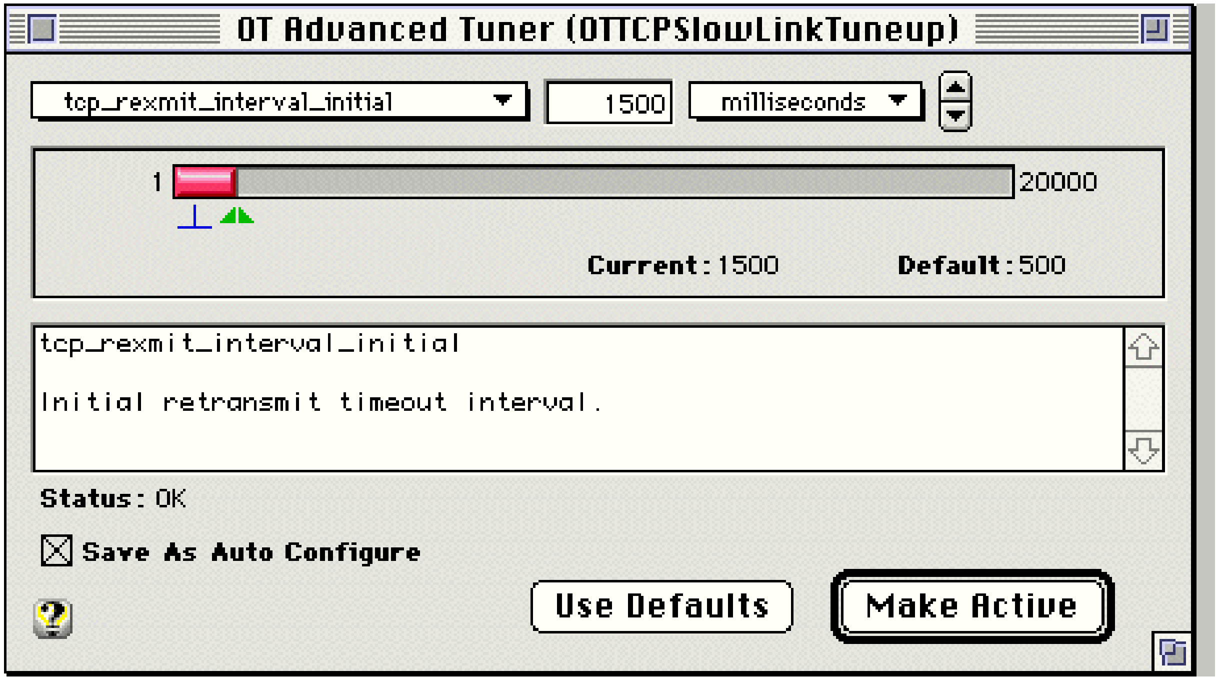
TCP Window Size

How to Download and Surf at the same time

Match the network MTU

Use a monitoring tool

Slow Link Tune Up (1)



tcp_rexmit_interval_initial 1500 milliseconds

Current: 1500 Default: 500

tcp_rexmit_interval_initial

Initial retransmit timeout interval.

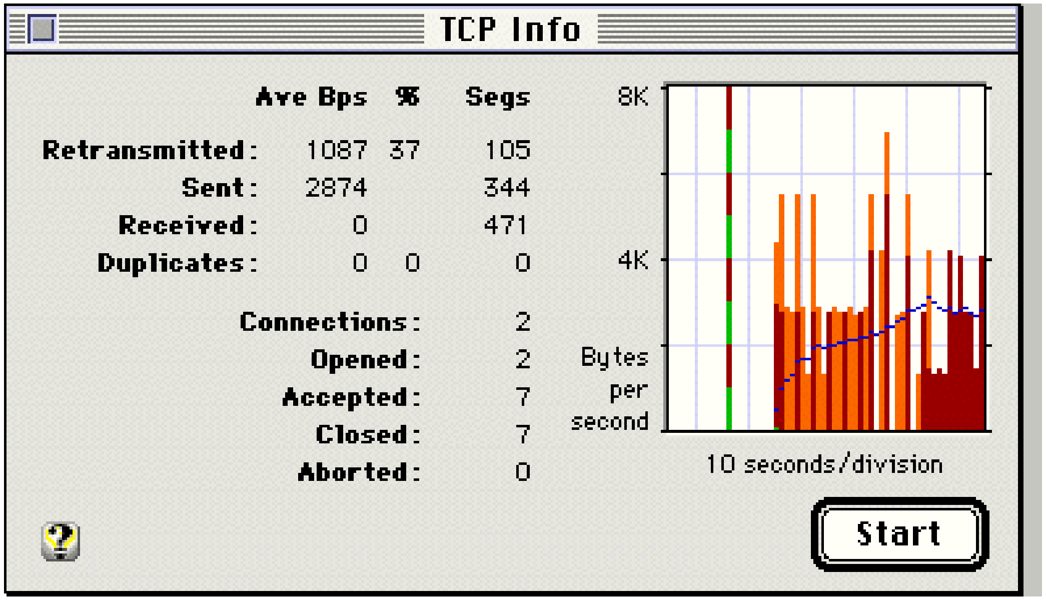
Status: OK

☒ Save As Auto Configure

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)

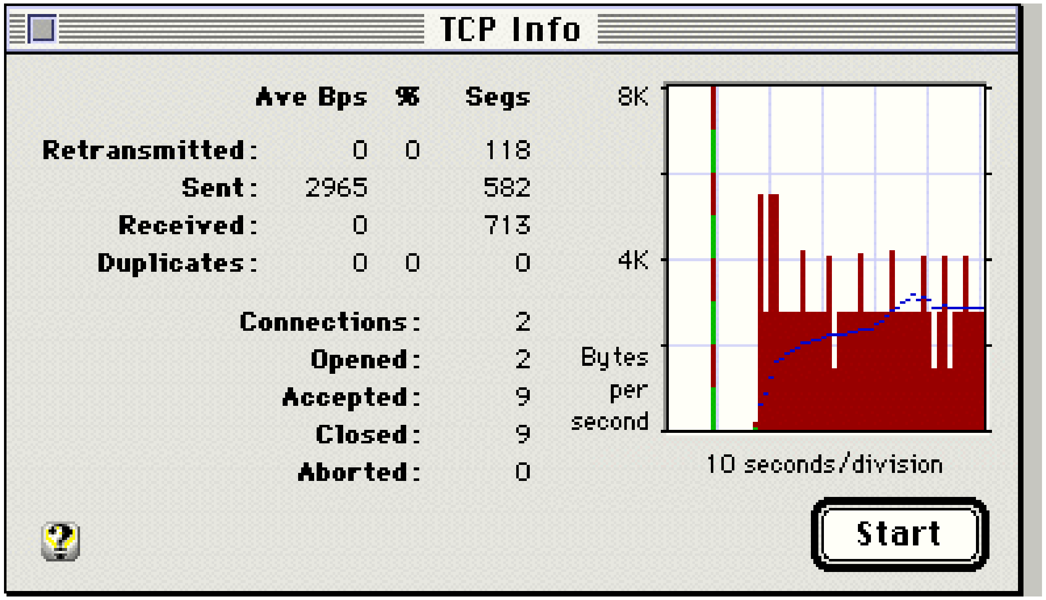


	Ave Bps	%	Segs
Retransmitted:	1087	37	105
Sent:	2874		344
Received:	0		471
Duplicates:	0	0	0

Connections:	2
Opened:	2
Accepted:	7
Closed:	7
Aborted:	0

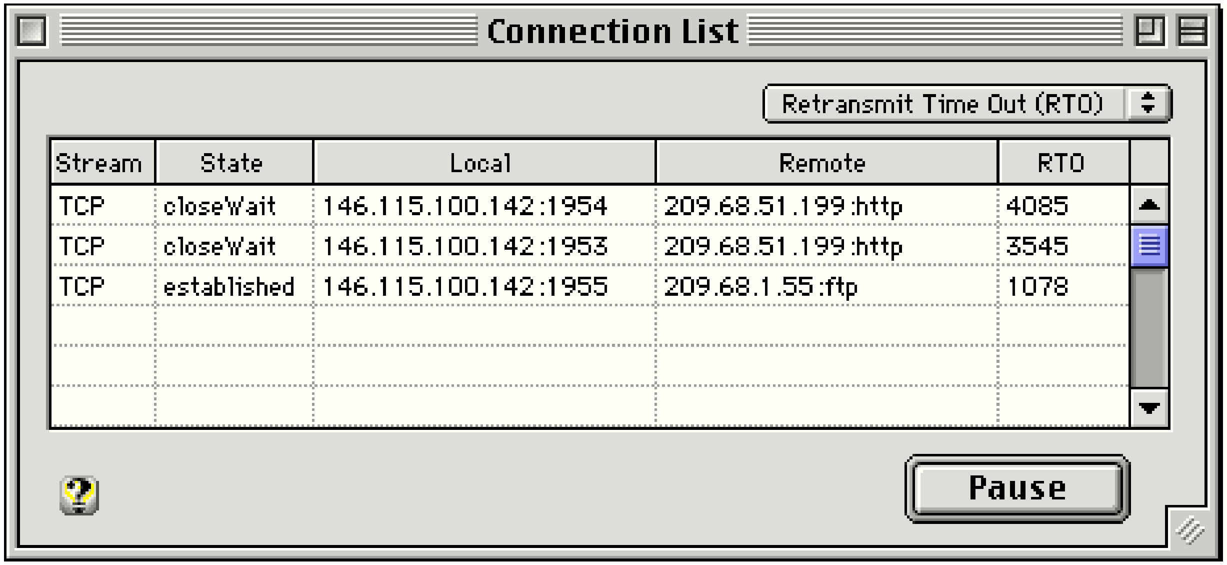
IPNetMonitor TCP Info window showing the start of an FTP upload over a 28.8k dial-up connection. The orange bars indicate retransmitted data is wasting much of the available bandwidth because the initial retransmit timeout interval is too small.

Slow Link Tune Up (3)



	Ave Bps	%	Segs
Retransmitted:	0	0	118
Sent:	2965		582
Received:	0		713
Duplicates:	0	0	0
Connections:			2
Opened:			2
Accepted:			9
Closed:			9
Aborted:			0

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.



Stream	State	Local	Remote	RTO
TCP	closeWait	146.115.100.142:1954	209.68.51.199:http	4085
TCP	closeWait	146.115.100.142:1953	209.68.51.199:http	3545
TCP	established	146.115.100.142:1955	209.68.1.55:ftp	1078

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.

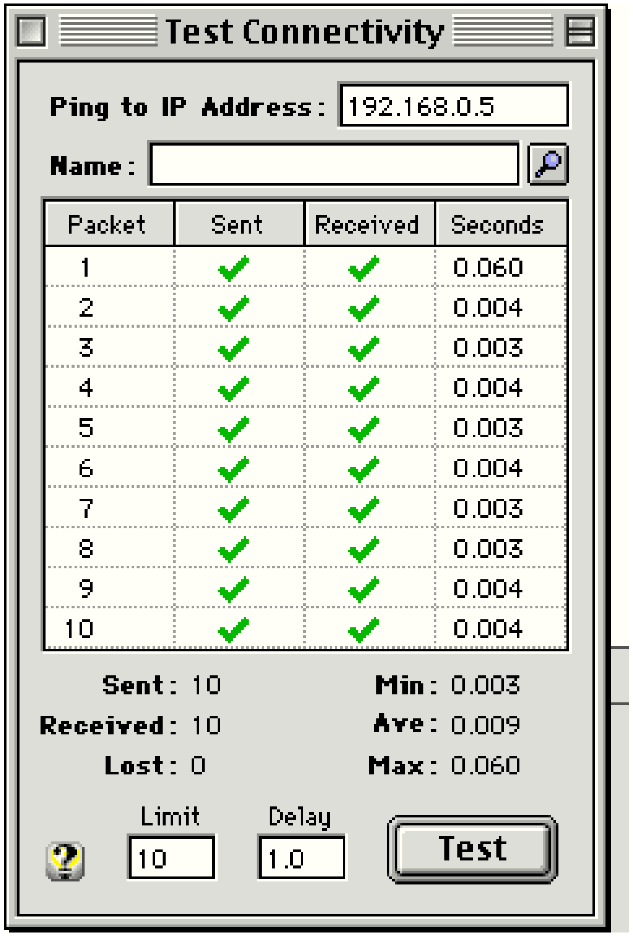
The capacity of the pipe is equal to the *bandwidth-delay product*.

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

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



Test Connectivity

Ping to IP Address: 192.168.0.5

Name:

Packet	Sent	Received	Seconds
1	✓	✓	0.060
2	✓	✓	0.004
3	✓	✓	0.003
4	✓	✓	0.004
5	✓	✓	0.003
6	✓	✓	0.004
7	✓	✓	0.003
8	✓	✓	0.003
9	✓	✓	0.004
10	✓	✓	0.004

Sent: 10 Min: 0.003
Received: 10 Ave: 0.009
Lost: 0 Max: 0.060

Limit: 10
Delay: 1.0

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)

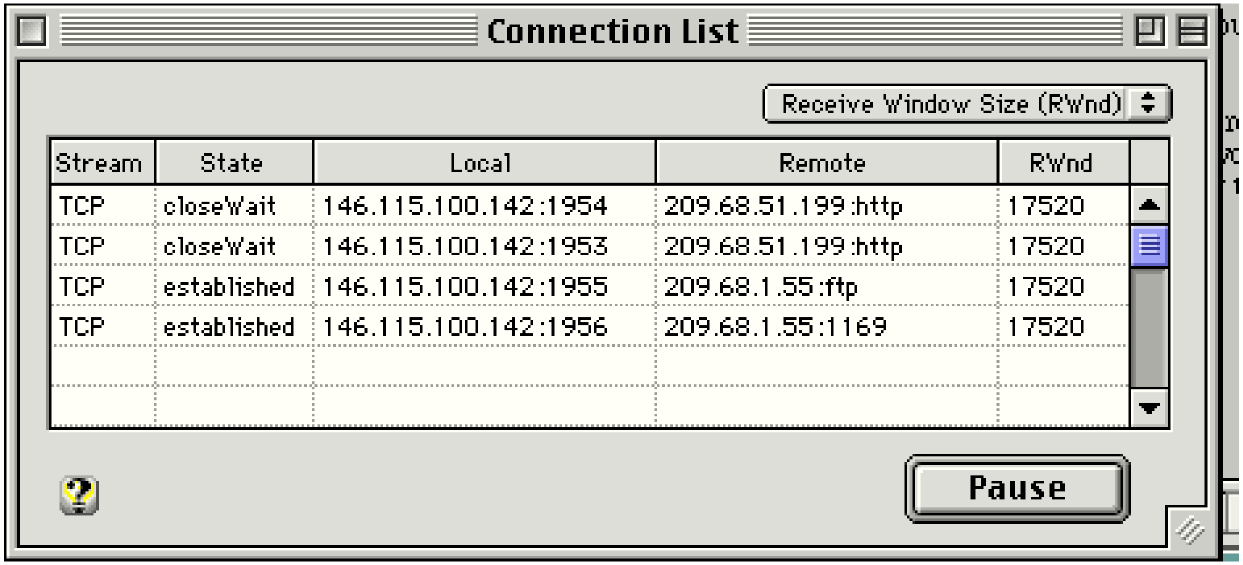
Cable Modems

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

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

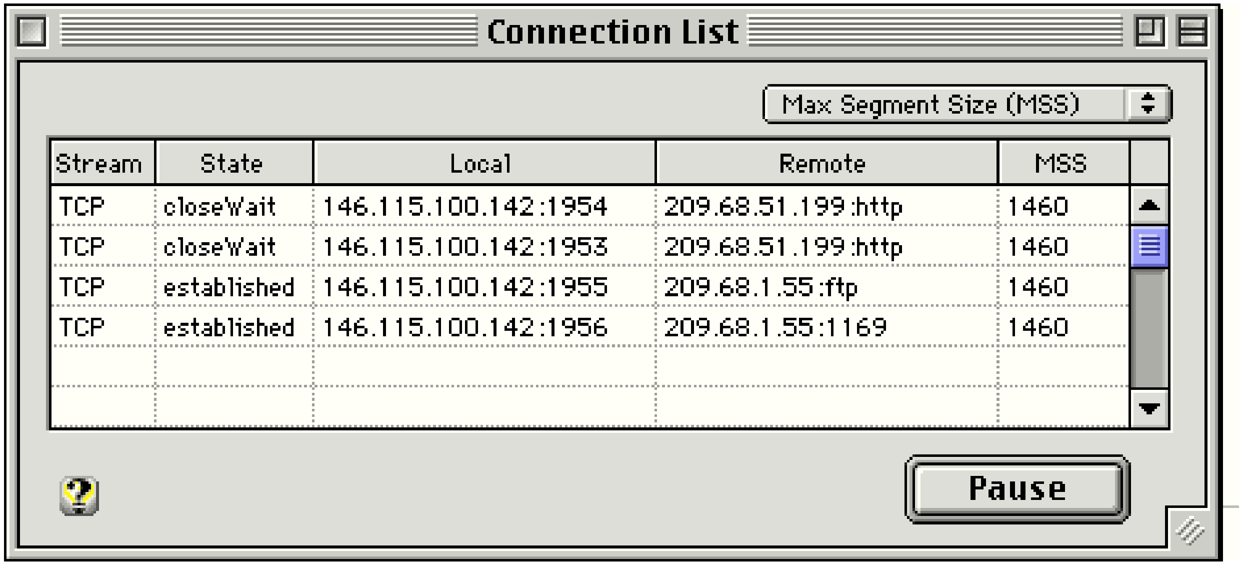
Can use IPNetMonitor to test the round trip time and monitor actual performance as the TCP Receive Window is adjusted.

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;



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



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

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			Measured	Measured	Calculated
Segment	MSS	Receive	Download	Ping time	Amount of
Size (MSS)	Mult	window(RWnd)	B/W	to another site	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.

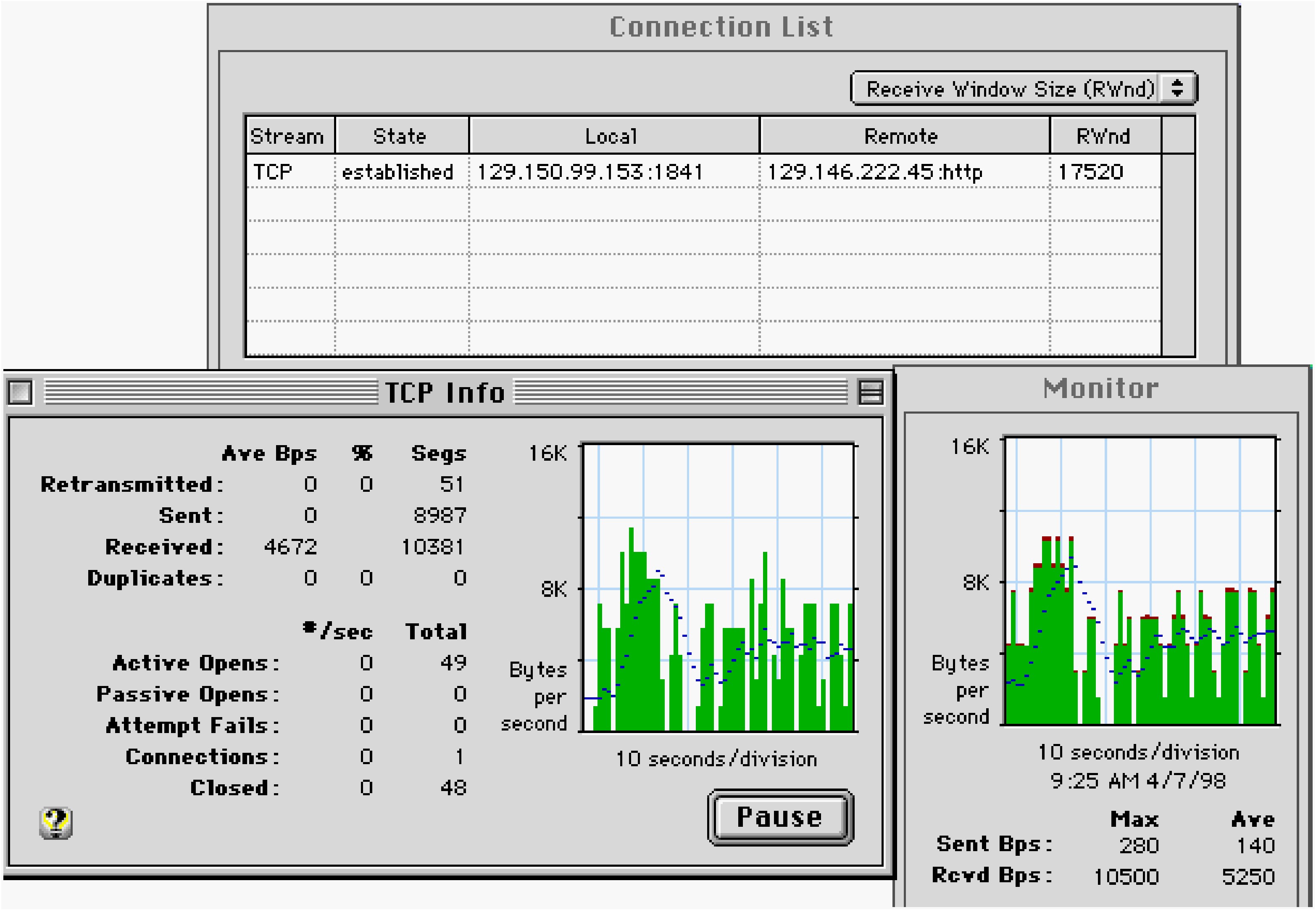
Why?

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



Stream	State	Local	Remote	RWnd
TCP	established	129.150.99.153:1841	129.146.222.45:http	17520

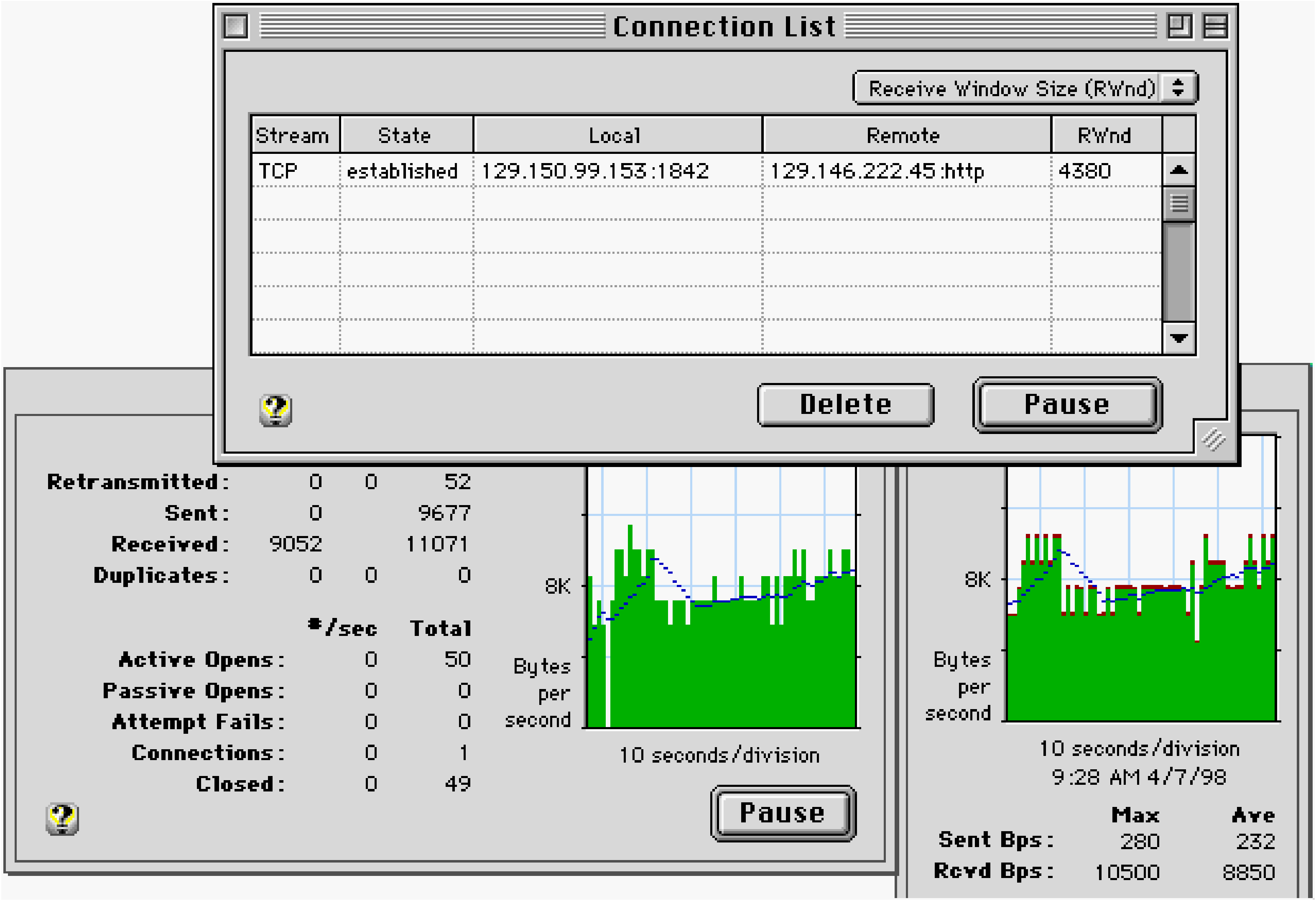
	Ave Bps	%	Segs
Retransmitted:	0	0	51
Sent:	0		8987
Received:	4672		10381
Duplicates:	0	0	0

	#/sec	Total
Active Opens:	0	49
Passive Opens:	0	0
Attempt Fails:	0	0
Connections:	0	1
Closed:	0	48

9:25 AM 4/7/98

	Max	Ave
Sent Bps:	280	140
Rcvd Bps:	10500	5250

Good 1460 x 3 PS



Connection List

Receive Window Size (RWnd)

Stream	State	Local	Remote	RWnd
TCP	established	129.150.99.153:1842	129.146.222.45:http	4380

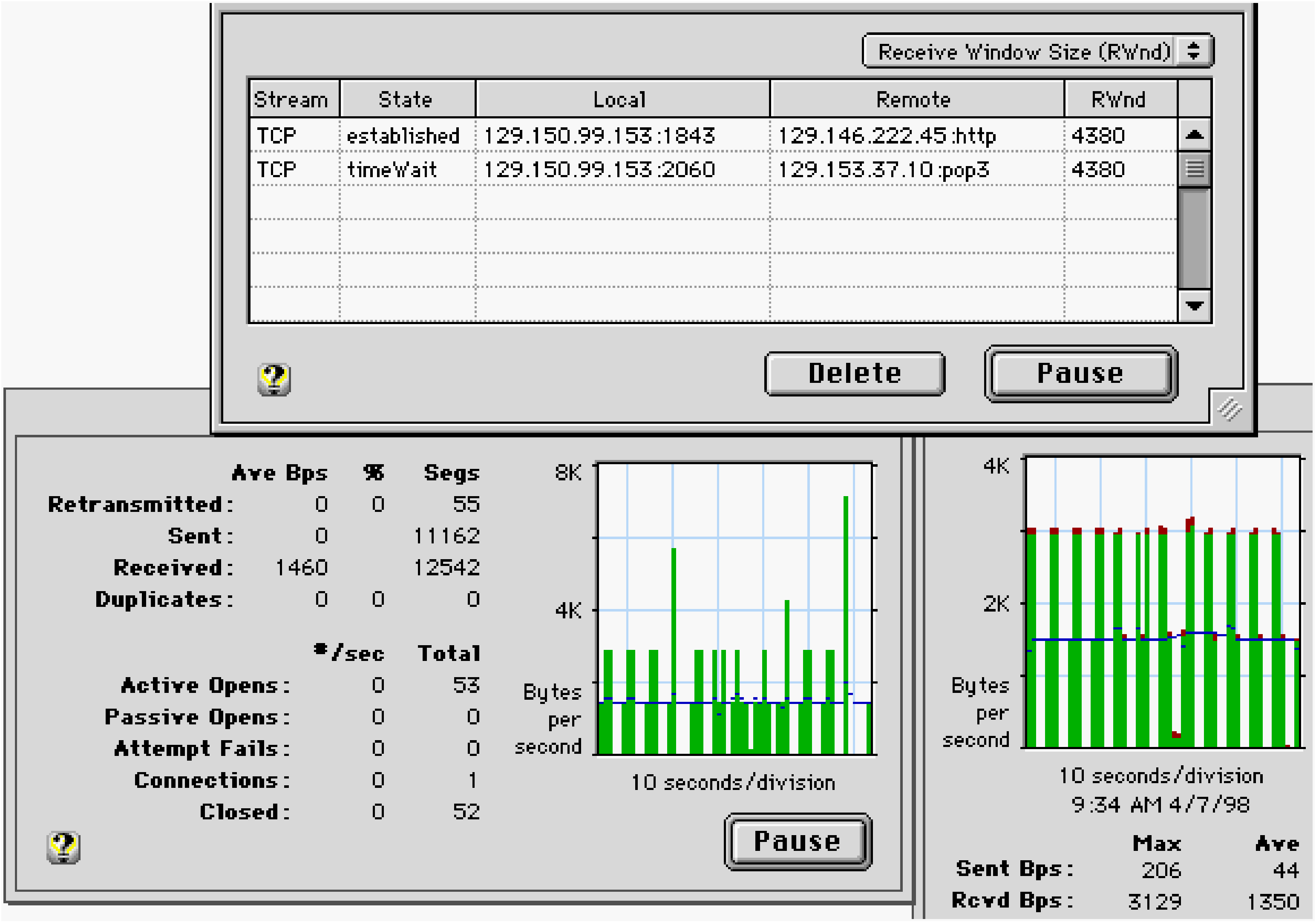
```
  Retransmitted:     0     0      52
           Sent:     0          9677
       Received:  9052         11071
     Duplicates:     0     0       0

                        #/sec  Total
     Active Opens:         0      50
    Passive Opens:         0       0
    Attempt Fails:         0       0
      Connections:         0       1
           Closed:         0      49
```

10 seconds/division
9:28 AM 4/7/98

```
             Max     Ave
 Sent Bps:   280     232
 Rcvd Bps: 10500    8850
```

Bad 1460 x 3 Z



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

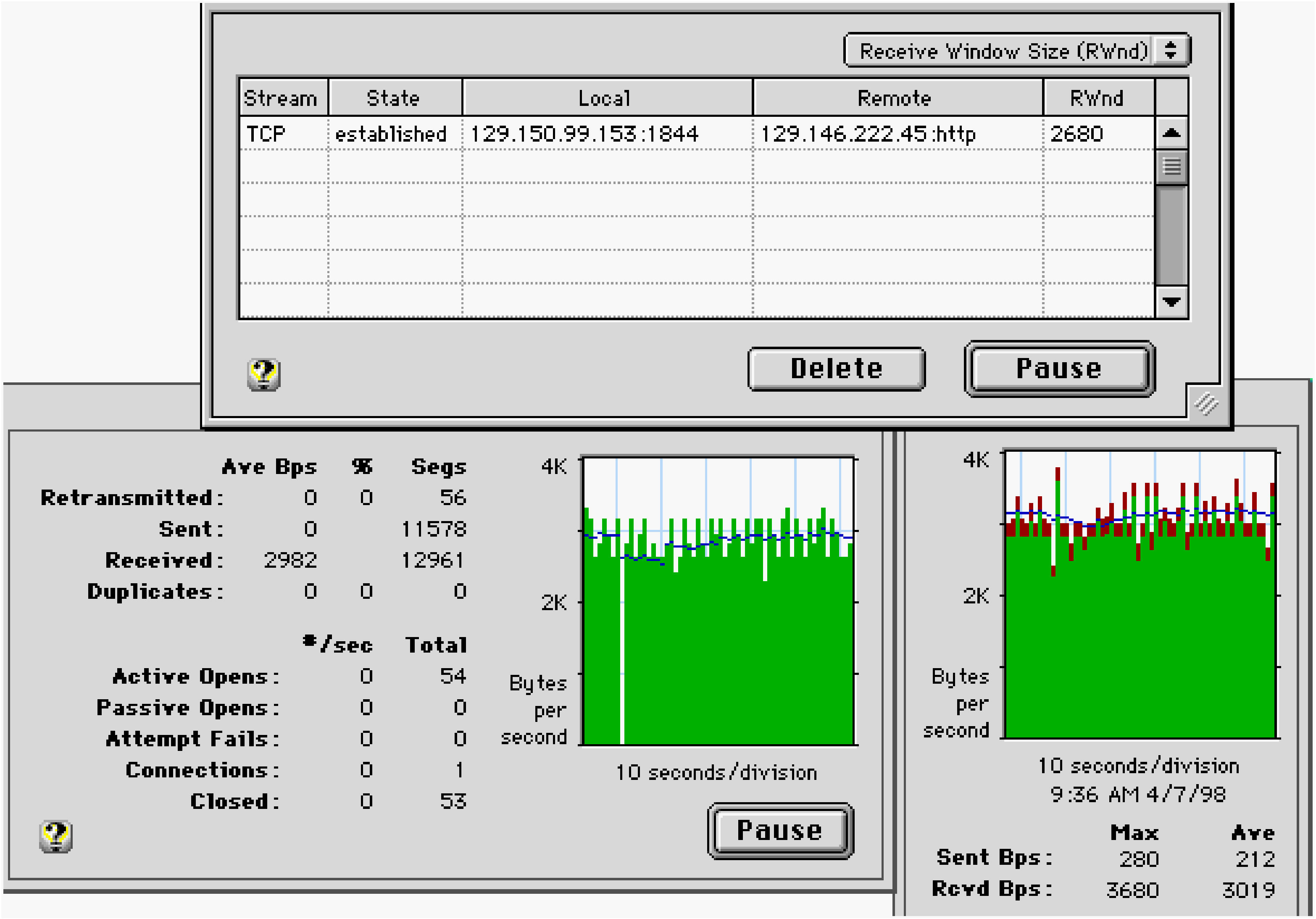
	Ave Bps	%	Segs
Retransmitted:	0	0	55
Sent:	0		11162
Received:	1460		12542
Duplicates:	0	0	0

	#/sec	Total
Active Opens:	0	53
Passive Opens:	0	0
Attempt Fails:	0	0
Connections:	0	1
Closed:	0	52

9:34 AM 4/7/98

	Max	Ave
Sent Bps:	206	44
Rcvd Bps:	3129	1350

Good 536 x 5 Z



Stream	State	Local	Remote	RWnd
TCP	established	129.150.99.153:1844	129.146.222.45:http	2680

	Ave Bps	%	Segs
Retransmitted:	0	0	56
Sent:	0		11578
Received:	2982		12961
Duplicates:	0	0	0

	#/sec	Total
Active Opens:	0	54
Passive Opens:	0	0
Attempt Fails:	0	0
Connections:	0	1
Closed:	0	53

	Max	Ave
Sent Bps:	280	212
Rcvd Bps:	3680	3019

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

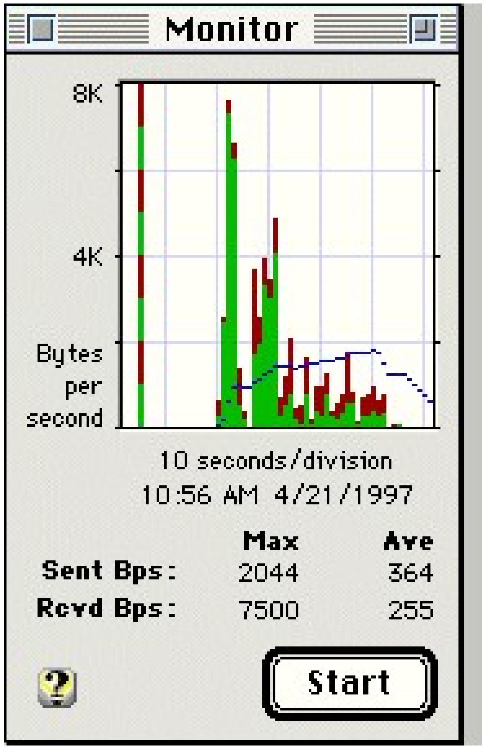
The MTU or Maximum Transmission Unit defines the largest segment or packet that can transmitted across a given network (without being split into smaller segments).

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



	Max	Ave
Sent Bps:	2044	364
Rcvd Bps:	7500	255

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