TCP Performance Management for Dummies

Nalini Elkins
Inside Products, Inc.

Monday, August 8, 2011
Session Number 9285

Our SHARE Sessions – Orlando

- 9285: TCP/IP Performance Management for Dummies Monday, August 8, 2011: 11:00 AM-12:00 PM
- 9269: IPv6 Addressing Wednesday, August 10, 2011: 11:00 AM-12:00 PM
- 9289: Staying Ahead of Network Problems at DTCC Wednesday, August 10, 2011: 3:00 PM-4:00 PM

If TCP/IP Works Fine…

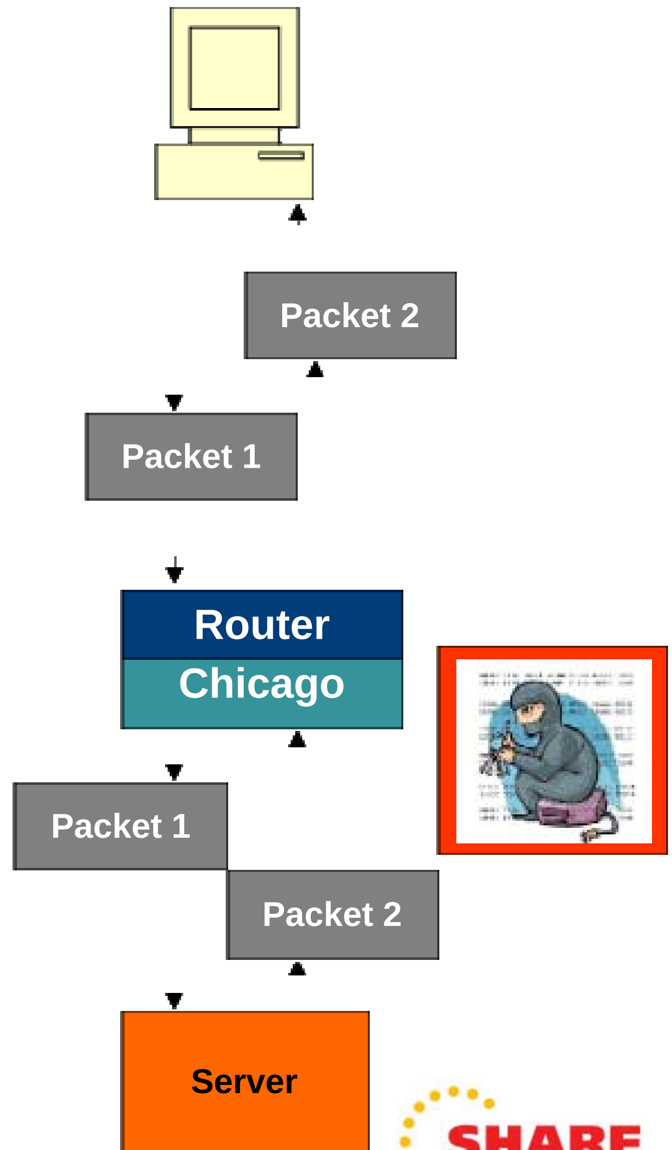
- Then, you don't need to look at anything. But, if it doesn't…
- This session is for the systems programmer who has to maintain TCP/IP, but doesn't really know where to start.

This session will tell you:

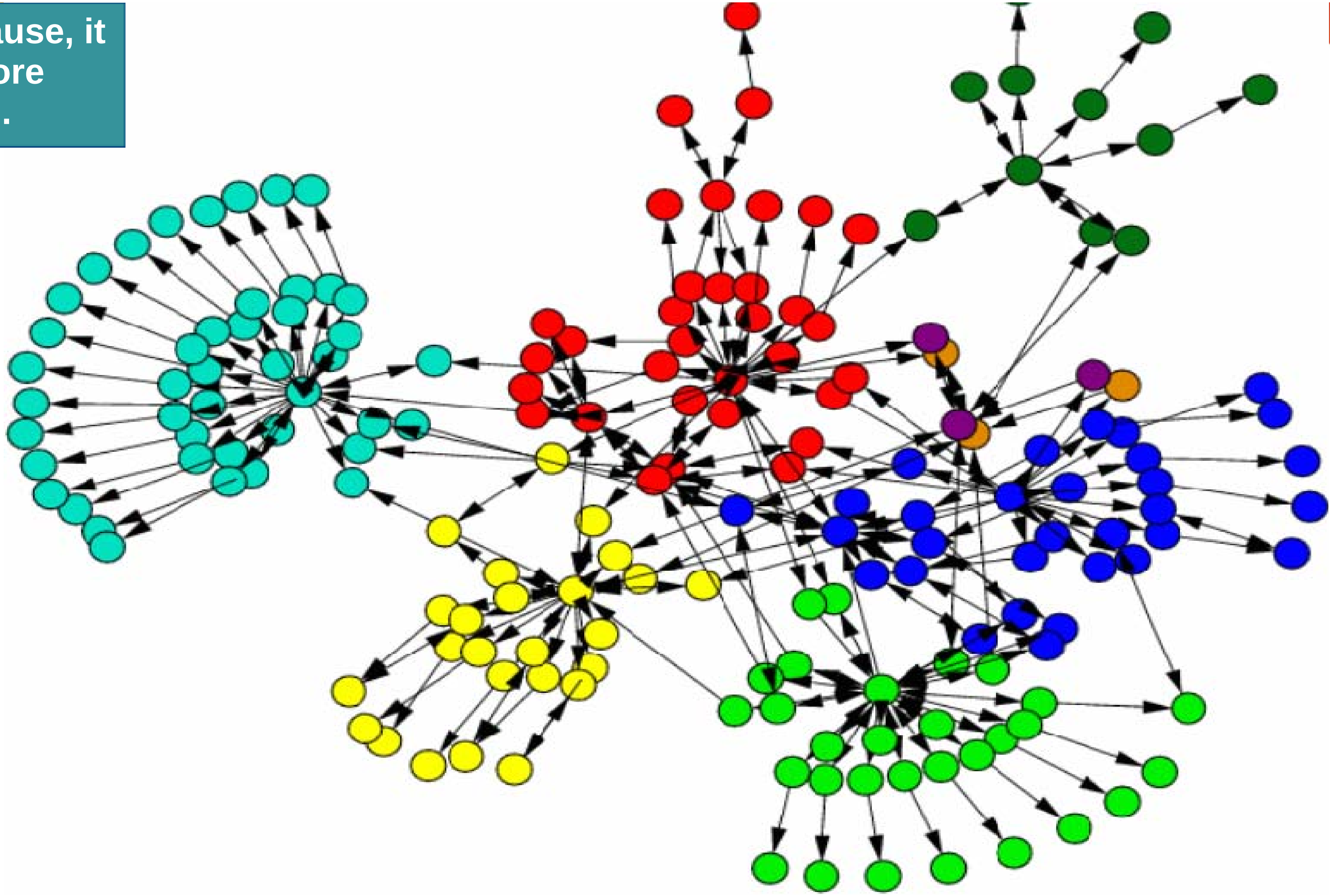
- What can cause problems on your TCP/IP network or stack.
- How to find problems.
- Where you may be able to change things.
- How to actually fix some things.

How Does TCP/IP Work

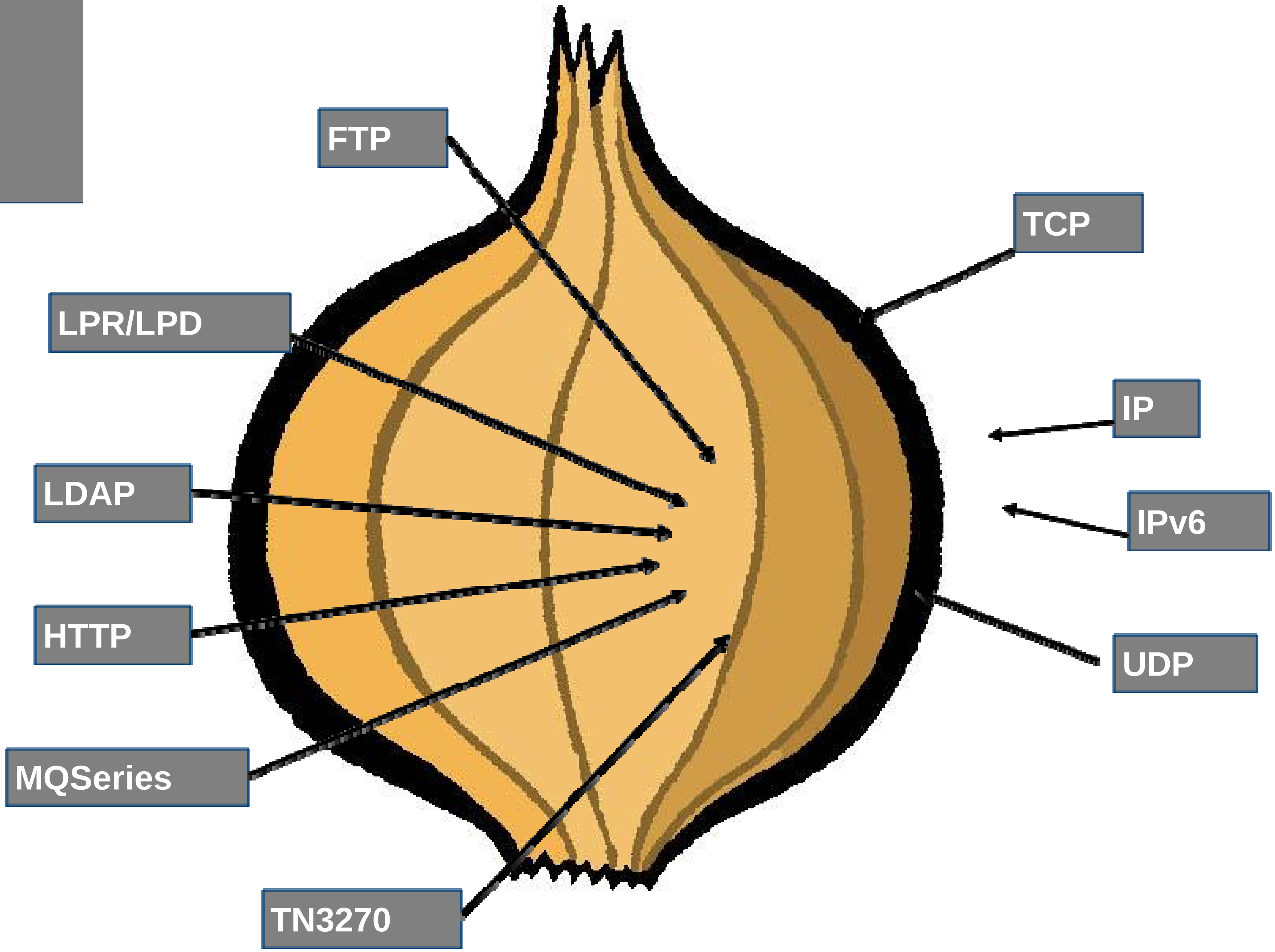
- Packets are sent from one host to another
- They go through some equipment in the middle – a router, switch, etc.
- Sometimes they get there, sometimes they don't.
- Sometimes some malicious person tries to snoop on them.
- What so hard about this?



Because, it is more like…



Network diagnostics involves decoding multiple layers of protocols.

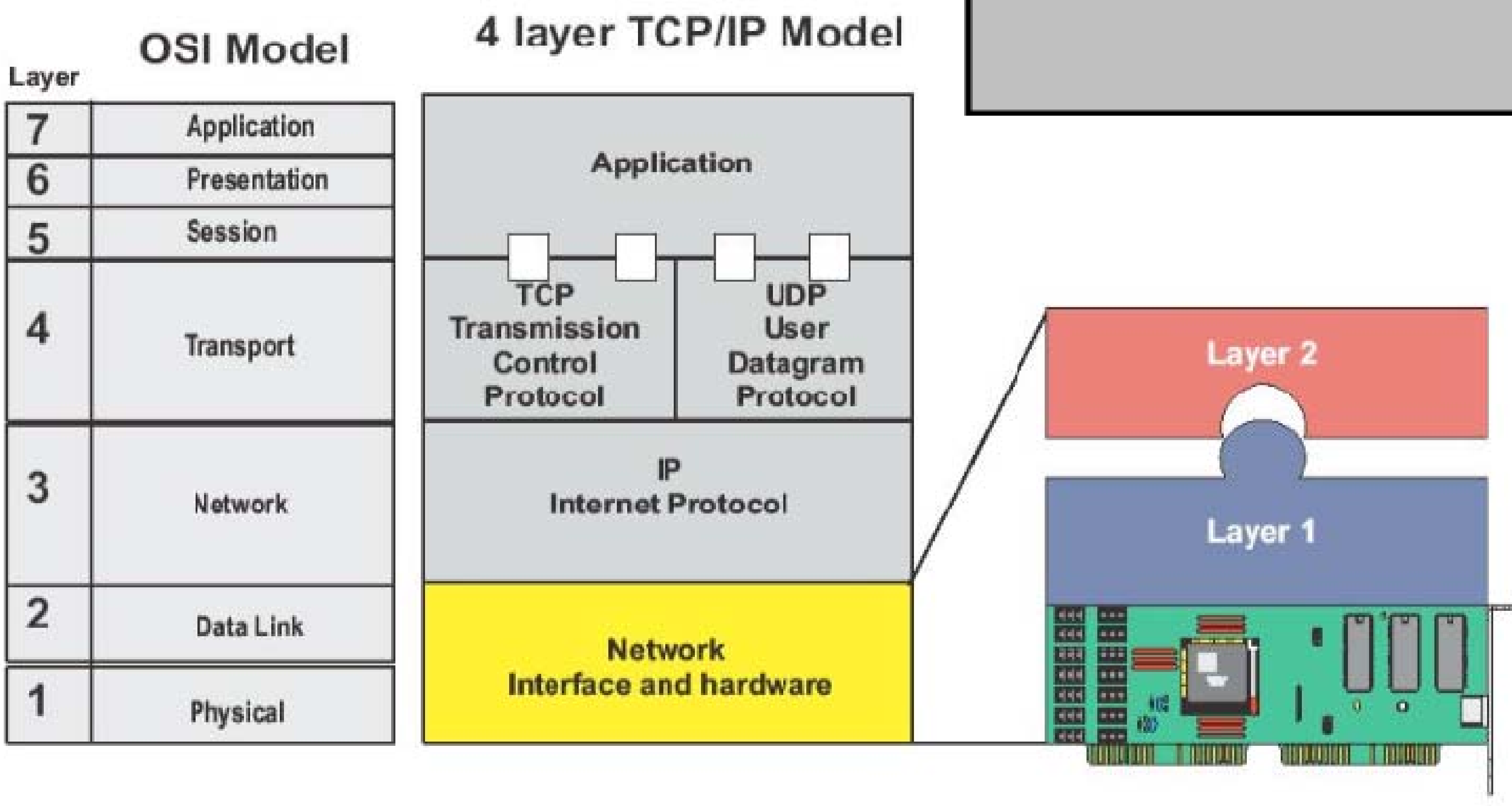


Core Internet Protocols

The core internet protocols are: TCP, UDP, IP, and ICMP (IPv4 and IPv6)

The TCP Profile:

- TCPConfig
- SOMAXCONN
- UDPConfig
- IPConfig
- IPConfig6



Basic TCP Functions

- Virtual circuit creation and management
- Network I/O management
- Reliability
- Application management

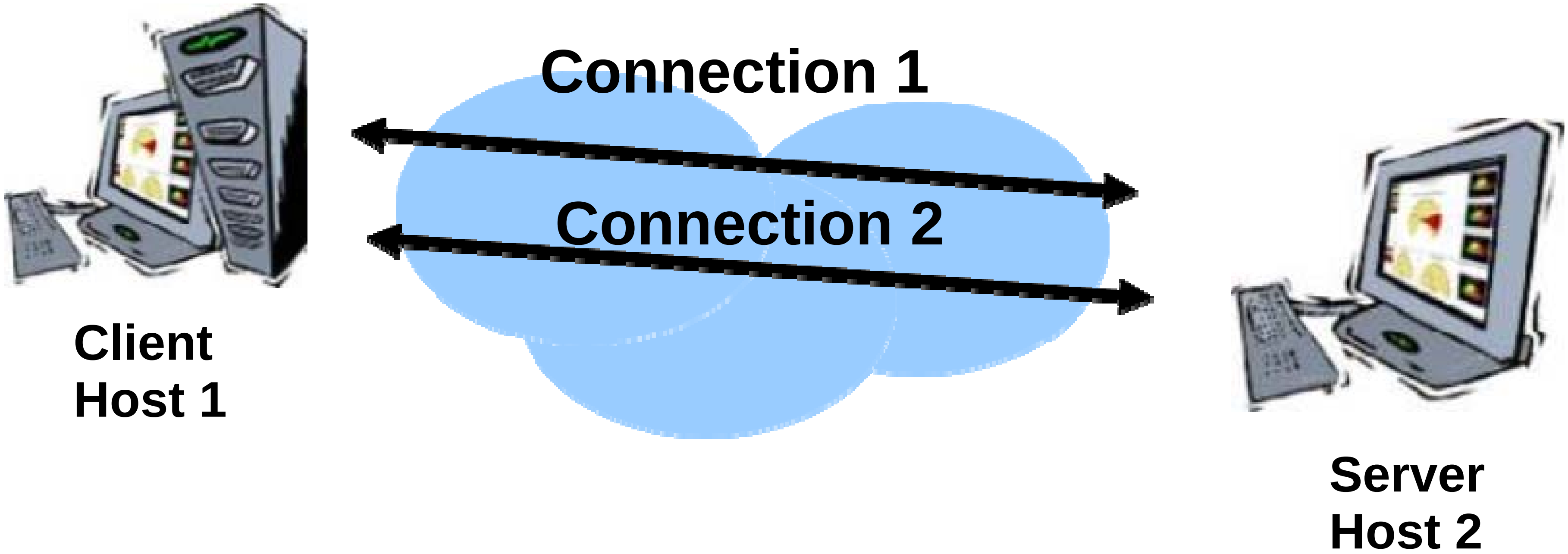
Virtual Circuit Functions

- TCP connection start up
- TCP shut down sequence
- Data transmission

TCP is a conversation

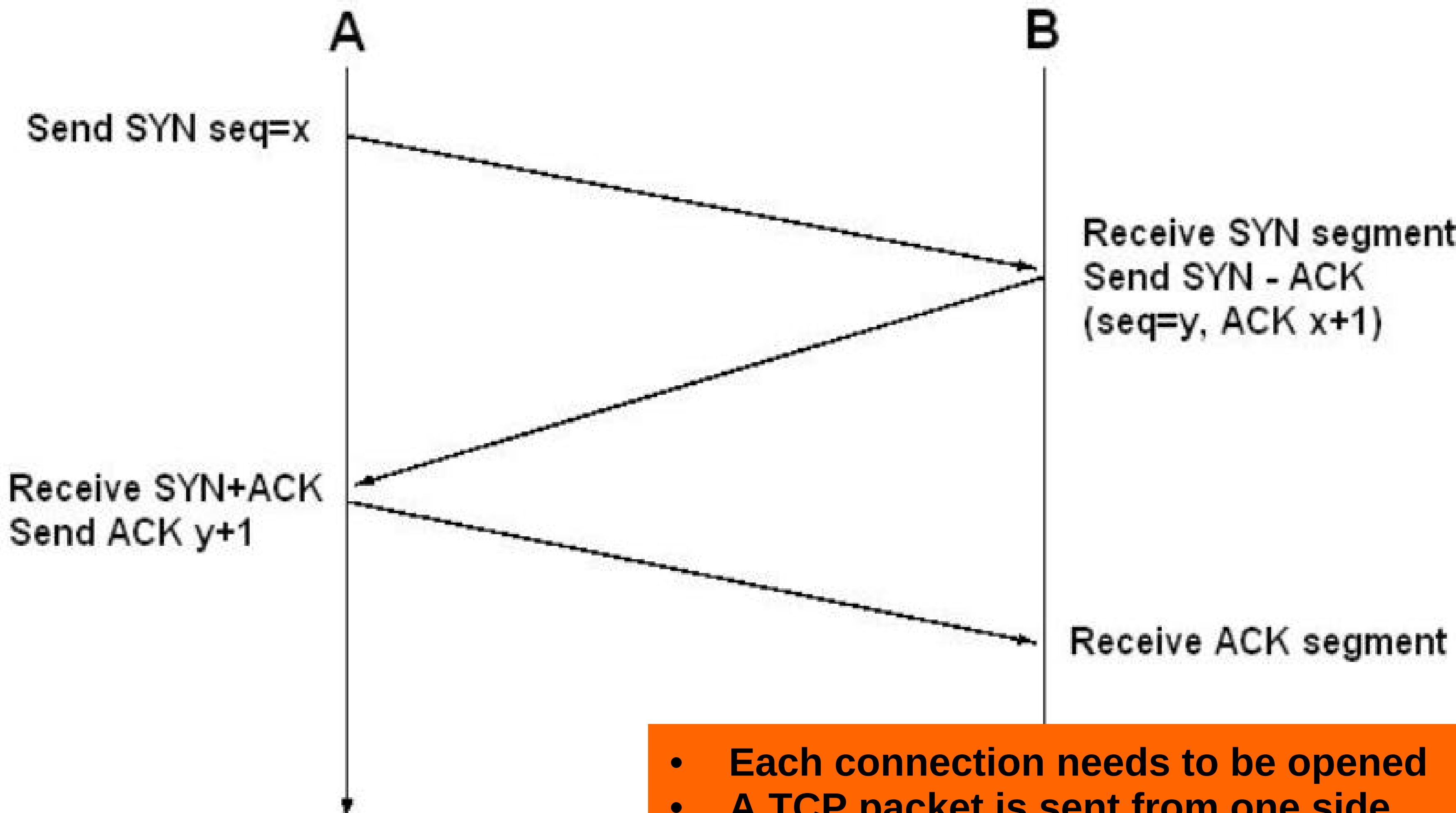
- *Hello, Dude!* (startup)
- *How's the surf?* (data transmission)
- *Later, man! Let's blow this popstand.* (close)

Virtual Circuit



- When two endpoints need to communicate with each other using TCP, a virtual circuit is established.
- The endpoints are the TCP listener running on the server and the remote client or foreign IP address.
- The virtual circuit provides the reliability, flow control and I/O management that make it different from UDP.
- There may be multiple connections between the client and server.

TCP Connection Establishment (3-Way Handshake)



X and Y are chosen randomly

- **Each connection needs to be opened**
- **A TCP packet is sent from one side**
- **Other side allocates buffers, other resources**
- **SYN -- SYN/ACK sequence**
- **Ready for data transmission**

TCP Header

Octet	Bits	Len	Name	Notes
0-1	-	2	Source port	-
2-3	-	2	Destination Port	-
4-7	-	4	Sequence number	position of last octet we sent.
8-11	-	4	Acknowledge Number	Next octet number we expect from the peer.
12	0-3	-	HLEN	4 bits. The number of 32 bit multiples (4 octets) in the TCP header including any 'options' fields.
12	4-7	-	Reserved	should be zero
13	-	1	Code bits	8 bits (6 used) valid if 1 bit 0 (URG) Urgent bit 1 (ACK) Acknowledgement bit 2 (PSH) Requests PUSH bit 3 (RST) Reset connection bit 4 (SYN) Sync sequence numbers bit 5 (FIN) sender finished
14-15	-	2	Window	Specifies the amount of data we can accept.
16-17	-	2	Checksum	Standard IP checksum
18-19	-	2	Urgent pointer	Points to end of urgent data.
TCP Options				
TCP data				

757 HOST1 PACKET 00000001 07:50:10.150650 Packet Trace
From Interface : GBE2 Device: QDIO Ethernet Full=60
Tod Clock : 2009/12/03 07:50:10.150649
Sequence # : 0 Flags: Pkt Ver2
Source Port : 3886 Dest Port: 5023 Asid: 0066 TCB: 00000000
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 60 ID Number: 0F74
Fragment : Offset: 0
TTL : 64 Protocol: TCP CheckSum: CACC FFFF
Source : xxx.194.129.5 ←
Destination : xxx.194.129.241
TCP
Source Port : 3886 () Destination Port: 5023 ()
Sequence Number : 3392023214 Ack Number: 0
Header Length : 40 Flags: Syn ←
Window Size : 65535 CheckSum: 6EDE FFFF Urgent Data: 0000
Option : Max Seg Size Len: 4 MSS: 8952
Option : NOP
Option : Window Scale OPT Len: 3 Shift: 1
Option : NOP
Option : NOP
Option : Timestamp Len: 10 Value: DA182CA6 Echo: 00000000

Open Packet

758 HOST1 PACKET 00000001 07:50:10.150761 Packet Trace
To Interface : GBE1 Device: QDIO Ethernet Full=60
Tod Clock : 2009/12/03 07:50:10.150761
Sequence # : 0 Flags: Pkt Ver2 Out
Source Port : 5023 Dest Port: 3886 Asid: 0066 TCB: 00000000
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 60 ID Number: F585
Fragment : Offset: 0
TTL : 64 Protocol: TCP CheckSum: E4BA FFFF
Source : xxx.194.129.241
Destination : xxx.194.129.5
TCP
Source Port : 5023 () Destination Port: 3886 ()
Sequence Number : 1441719441 Ack Number: 3392023215
Header Length : 40 Flags: Ack Syn
Window Size : 65535 CheckSum: 4AD2 FFFF Urgent Data Pointer: 0000
Option : Max Seg Size Len: 4 MSS: 1460
Option : NOP
Option : Window Scale OPT Len: 3 Shift: 0
Option : NOP
Option : NOP
Option : Timestamp Len: 10 Value: DA182CA7 Echo: DA182CA6

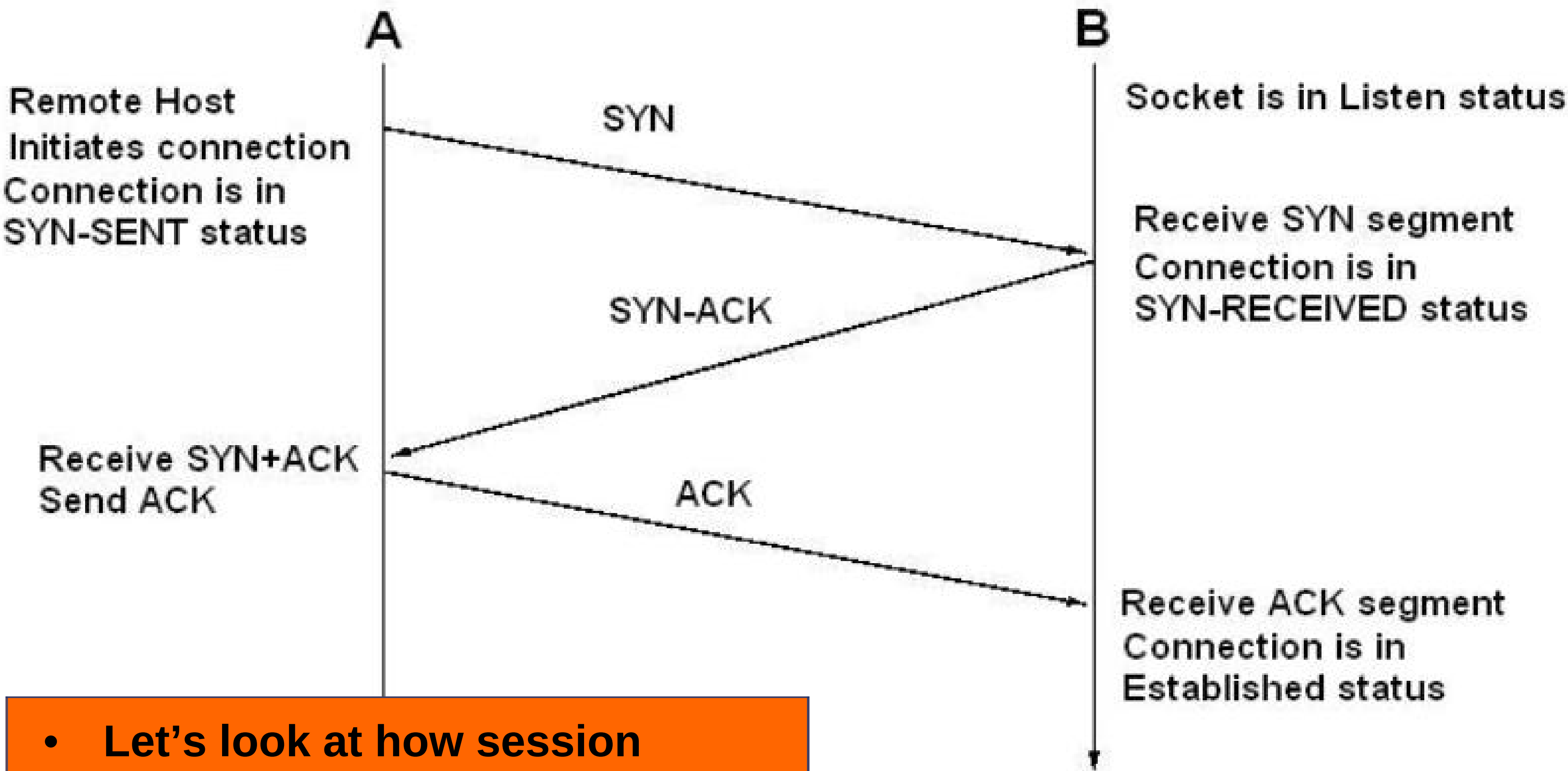
-	-	Packet Number	Time	F T	Interface	Device	Source Address	Source Port	Destination Address	Dest Port	Window	SYN	ACK	Delta in Milliseconds
1		1769	07:55:40.816412	F	LOSABP2	QDIO	[redacted]	1166	[redacted]	23	16384	Y	-	-
2		1770	07:55:40.816503	T	LOSABP1	QDIO	[redacted]	23	[redacted]	1166	32768	Y	Y	91
3		2112	07:55:41.851348	F	LOSABP1	QDIO	[redacted]	1141	[redacted]	23	64512	Y	-	-
4		2113	07:55:41.851424	T	LOSABP1	QDIO	[redacted]	23	[redacted]	1141	32768	Y	Y	76
5		2405	07:55:43.160732	F	LOSABP1	QDIO	[redacted]	1653	[redacted]	23	64512	Y	-	-
6		2406	07:55:43.160813	T	LOSABP2	QDIO	[redacted]	23	[redacted]	1653	32768	Y	Y	81
7		3240	07:55:44.871174	F	LOSABP2	QDIO	[redacted]	1167	[redacted]	23	16384	Y	-	-
8		3241	07:55:44.871257	T	LOSABP1	QDIO	[redacted]	23	[redacted]	1167	32768	Y	Y	83
9		5170	07:55:51.727295	F	LOSABP1	QDIO	[redacted]	2223	[redacted]	23	64512	Y	-	-
10		5171	07:55:51.727396	T	LOSABP1	QDIO	[redacted]	23	[redacted]	2223	32768	Y	Y	101
11		9203	07:56:09.381221	F	LOSABP2	QDIO	[redacted]	1159	[redacted]	23	65535	Y	-	-
12		9204	07:56:09.381313	T	LOSABP2	QDIO	[redacted]	23	[redacted]	1159	32768	Y	Y	92

- The session start times seem quite good.
- The absolute time is small and it is consistent.
- Next we will see a bad situation.

-	-	Packet Number	Time	F T	Interface	Device	Source Address	Source Port	Destination Address	Dest Port	Window	SYN	ACK	Delta in Milliseconds
1		3673	15:10:11.794716	T	GEMH0	QDIO	10.48.16.1	20	192.168.186.31	52266	32768	Y	-	-
2		3674	15:10:11.795610	F	GEMH0	QDIO	192.168.186.31	52266	10.48.16.1	20	24616	Y	Y	894
3		3734	15:10:12.319251	T	GEMH0	QDIO	10.48.16.1	20	192.168.186.31	52268	32768	Y	-	-
4		3735	15:10:12.320078	F	GEMH0	QDIO	192.168.186.31	52268	10.48.16.1	20	24616	Y	Y	827
5		4295	15:10:17.957071	T	GEMH0	QDIO	10.48.16.1	20	192.168.186.31	52270	32768	Y	-	-
6		4296	15:10:17.958014	F	GEMH0	QDIO	192.168.186.31	52270	10.48.16.1	20	24616	Y	Y	943
7		4355	15:10:18.552663	T	GEMH0	QDIO	10.48.16.1	20	192.168.186.31	52272	32768	Y	-	-
8		4356	15:10:18.553409	F	GEMH0	QDIO	192.168.186.31	52272	10.48.16.1	20	24616	Y	Y	746
9		4462	15:10:17.957071	T	GEMH0	QDIO	10.48.16.1	20	192.168.186.31	52270	32768	Y	-	-
10		4463	15:10:17.958014	F	GEMH0	QDIO	192.168.186.31	52270	10.48.16.1	20	24616	Y	Y	943
11		4522	15:10:18.552663	T	GEMH0	QDIO	10.48.16.1	20	192.168.186.31	52272	32768	Y	-	-
12		4523	15:10:18.553409	F	GEMH0	QDIO	192.168.186.31	52272	10.48.16.1	20	24616	Y	Y	746
45		12374	15:11:52.680784	T	GEMH0	QDIO	10.48.16.1	20	192.168.151.5	1105	32768	Y	-	-
46		12415	15:11:52.974294	F	GEMH0	QDIO	192.168.151.5	1105	10.48.16.1	20	24820	Y	Y	293,510

- Look at the last time! Why might this be?

TCP Connection Establishment Session Status



- **Let's look at how session establishment works.**
- **Why do you think that the SYN-ACK was sent after so much time?**

Viewing Session States

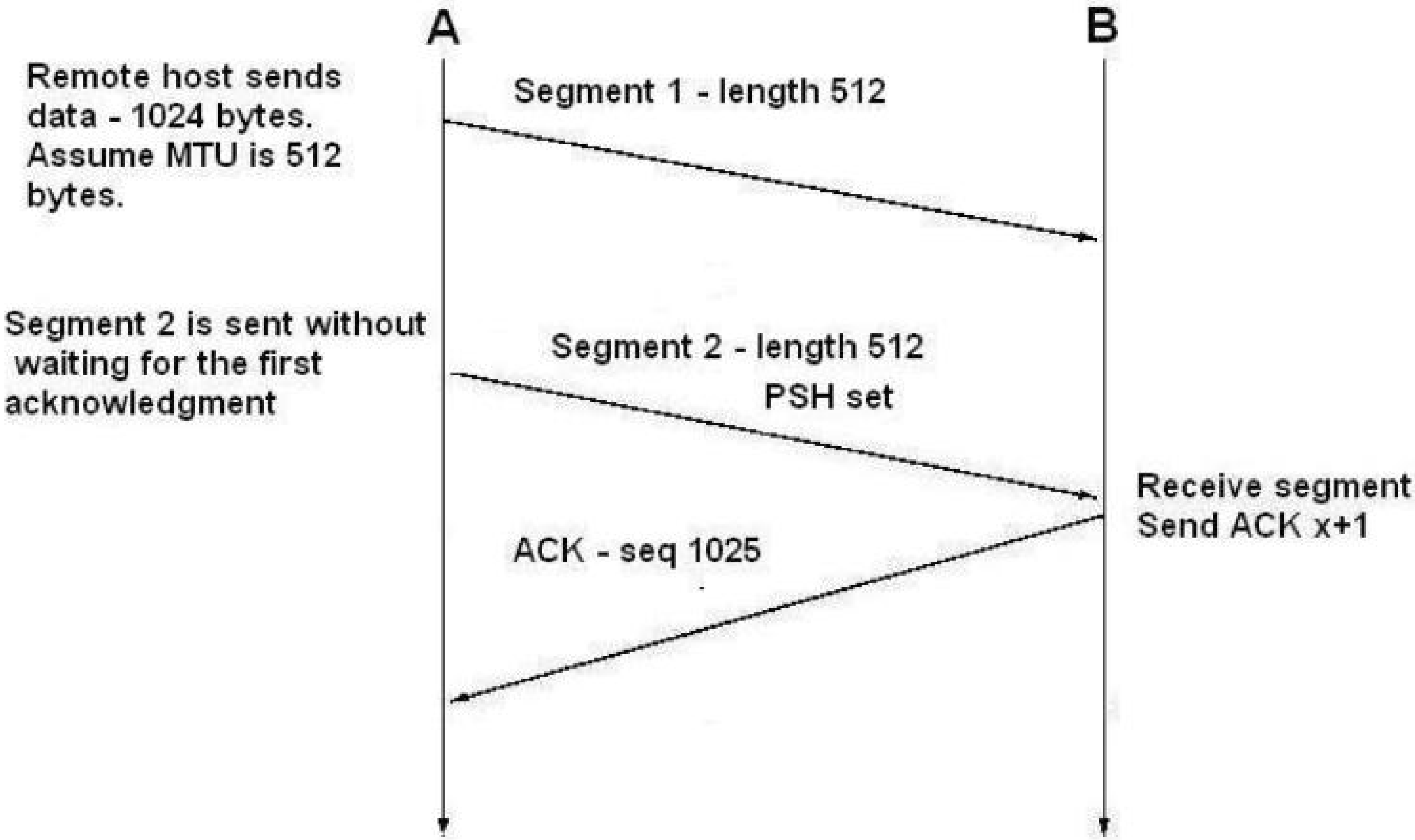
```
MVS TCP/IP NETSTAT CS V1R2          TCPIP NAME: TCPIP                  14:54:16
                    MVS TCP/IP REAL TIME NETWORK MONITOR
USER ID  B OUT      B IN       L PORT FOREIGN SOCKET                STATE
-------  -----      ----       ------ ---------------               -----
BPXOINIT 0000000000 0000000000 10007  0.0.0.0..0                    LISTEN
FTPD1    0000000000 0000000000 00021  0.0.0.0..0                    LISTEN
INETD4   0000000000 0000000000 00513  0.0.0.0..0                    LISTEN
INETD4   0000000000 0000000000 01023  0.0.0.0..0                    LISTEN
ITSCM    0000000000 0000000000 44444  0.0.0.0..0                    LISTEN
ITSCM    0000000000 0000000028 44444  65.113.138.108..3119          ESTABLSH
ITSHA    0000713502 0000026768 44441  65.113.138.108..4920          ESTABLSH
```

- How do you view session states? This is a Netstat Byteinfo command on z/OS.
- Notice Listen and Established connect states
- Established state will have foreign address

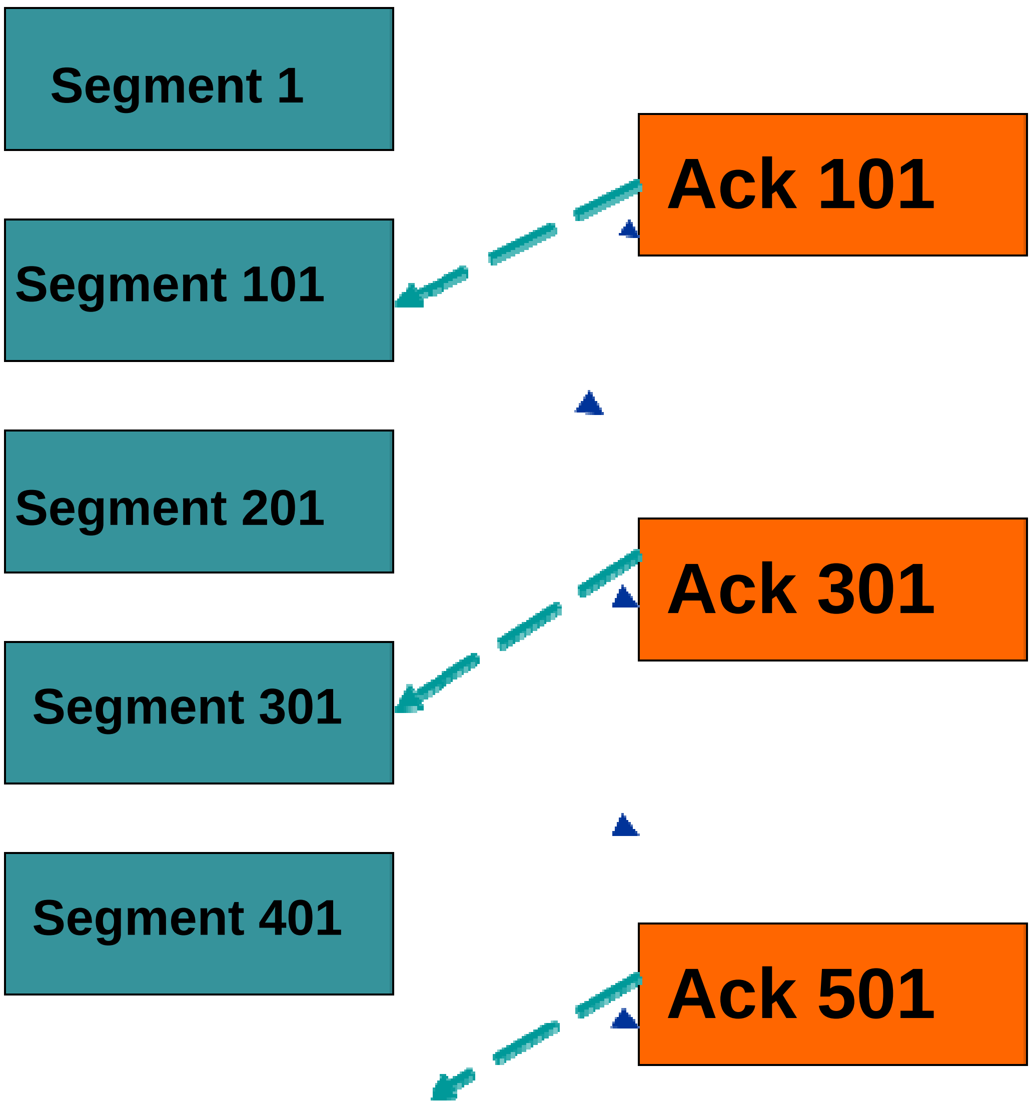
TCP Connect States (RFC793)

- A connection progresses through a series of states during its lifetime. The states are: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED. CLOSED is fictional because it represents the state when there is no TCB, and therefore, no connection.
- A TCP connection progresses from one state to another in response to events. The events are the user calls, OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS; the incoming segments, particularly those containing the SYN, ACK, RST and FIN flags; and timeouts.
- LISTEN - represents waiting for a connection request from any remote TCP and port. This is for a server port.
- SYN-SENT - represents waiting for a matching connection request after having sent a connection request.
- SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
- ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

TCP Data Transfer



How Does ACK Work?



- Assume each segment has 100 bytes.
- Ack is for all the bytes received and indicates the next byte of data that is expected
- Ack is not sent for each packet

15 HOST2 PACKET 00000001 13:31:53.385054 Packet Trace
From Interface : CISCO1 Device: CLAW Full=43
Tod Clock : 2009/10/14 13:31:53.385052
Sequence # : 0 Flags: Pkt Ver2 Adj
Source Port : 1080 Dest Port: 23 Asid: 0038 TCB: 00000000
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 43 ID Number: 0280
Fragment : Offset: 0
TTL : 125 Protocol: TCP CheckSum: DED2 FFFF
Source : 192.168.145.7
Destination : 10.201.0.2

TCP
Source Port : 1080 () Destination Port: 23 (telnet)
Sequence Number : 3010274514 Ack Number: 2247247751
Header Length : 20 Flags: Ack Psh
Window Size : 16592 CheckSum: 6878 FFFF Urgent Dat

Telnet: 3
0000 IAC,DO,BIN TRANS;

IP Header : 20
Data : 3 Data Length: 3

At: 2009/10/14 13:31:53.385052
From: 192.168.145.7:1080
To: 10.201.0.2:23 - 3 bytes
Sequence number: 3010274514
Expected ACK : 3010274517

16 HOST2 PACKET 00000001 13:31:53.385455 Packet Trace
To Interface : CISCO1 Device: CLAW Full=119
Tod Clock : 2009/10/14 13:31:53.385454
Sequence # : 0 Flags: Pkt Ver2 Adj Out
Source Port : 23 Dest Port: 1080 Asid: 0038 TCB: 00000000
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 119 ID Number: 01E9
Fragment : Offset: 0
TTL : 64 Protocol: TCP CheckSum: 1C1E FFFF
Source : 10.201.0.2
Destination : 192.168.145.7
TCP
Source Port : 23 (telnet) Destination Port: 1080 ()
Sequence Number : 2247247751 Ack Number: 3010274517
Header Length : 20 Flags: Ack Psh
Window Size : 65535 CheckSum: 5974 FFFF Urgent Data Ptr: 0000

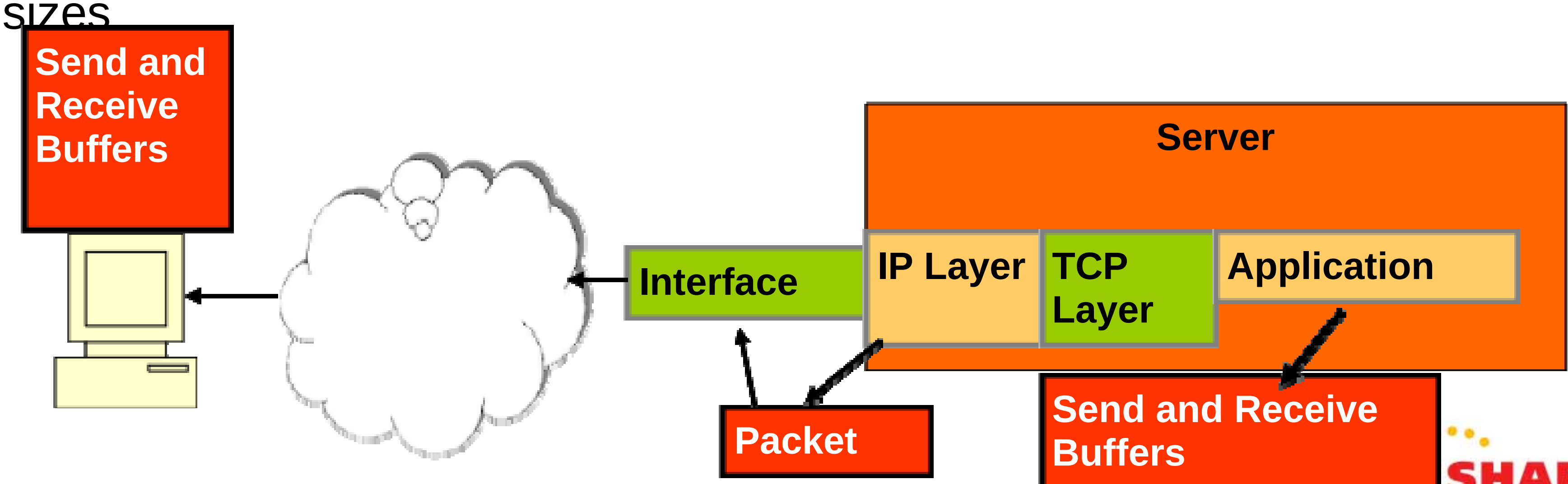
Telnet: 79
0000 (data=77),IAC,EOR;

IP Header : 20
000000 45000077 01E90000 40061C1E 0AC90002 C0A89107

Protocol Header : 20
000000 00170438 85F24787 B36D24D5 5018FFFF 59740000

TCP Buffers and MSS

- In the TCP Open, the TCP Send / Receive buffers and MSS are set.
- The Send / Receive buffers are pools used to hold data prior to sending it out across the physical adapter.
- TCP Send and Receive Buffers
  - o Can affect speed of transmission
  - o Match sender and receiver window sizes
- Max Segment Size
  - o Can affect speed of transmission
  - o Can be different in each direction.
- On most platforms each application can explicity set the buffer size via a socket option call.

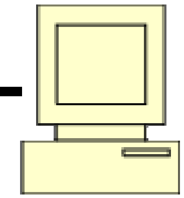


Congestion Window

z/OS Host

Interface GBE1 Port 2089

Pkt 368: 14:15:49.845144 – Set Window Size to 17520



Pkt 369: 14:15:49.845266 – Send Data -> 1448
Pkt 370: 14:15:49.845304 – Send Data -> 1448
Pkt 371: 14:15:49.845308 – Send Data -> 1448
Pkt 372: 14:15:49.845312 – Send Data -> 1448
Pkt 373: 14:15:49.845328 – Send Data -> 1448
Pkt 374: 14:15:49.845332 – Send Data -> 1448
Pkt 375: 14:15:49.845336 – Send Data -> 1448
Pkt 376: 14:15:49.845353 – Send Data -> 1448
Pkt 377: 14:15:49.845357 – Send Data -> 1448
Pkt 378: 14:15:49.845374 – Send Data -> 1448
Pkt 379: 14:15:49.845387 – Send Data -> 1448
Pkt 380: 14:15:49.845391 – Send Data -> 1448
Pkt 381: 14:15:49.845512 – Send Data -> 144

Remote: 10.2.161.4 Port 4987

(12 * 1448) + 144 = 17,520

Pkt 384 14:15:49.846338 – ACK

1 HOST2 PACKET 00000001 13:31:51.410583 Packet Trace
From Interface : CISCO1 Device: CLAW Full=48
Tod Clock : 2009/10/14 13:31:51.410582
Sequence # : 0 Flags: Pkt Ver2
Source Port : 1080 Dest Port: 23 Asid: 0038 TCB:
IpHeader: Version : 4 Header Length: 20
Tos : 00 QOS: Routine Normal Service
Packet Length : 48 ID Number: 026E
Fragment : Offset: 0
TTL : 125 Protocol: TCP CheckSum: DEDF FFFF
Source : 192.168.145.7
Destination : 10.201.0.2

TCP
Source Port : 1080 () Destination Port: 23 (telnet)
Sequence Number : 3010274480 Ack Number: 0
Header Length : 28 Flags: Syn
Window Size : 17520 CheckSum: 0A9C FFFF Urgent Data:0000
Option : Max Seg Size Len: 4 MSS: 1448 Option : NOP
Option : NOP
Option : SACK Permitted

IP Header : 20
000000 45000030 026E0000 7D06DEDF C0A89107 0AC90002

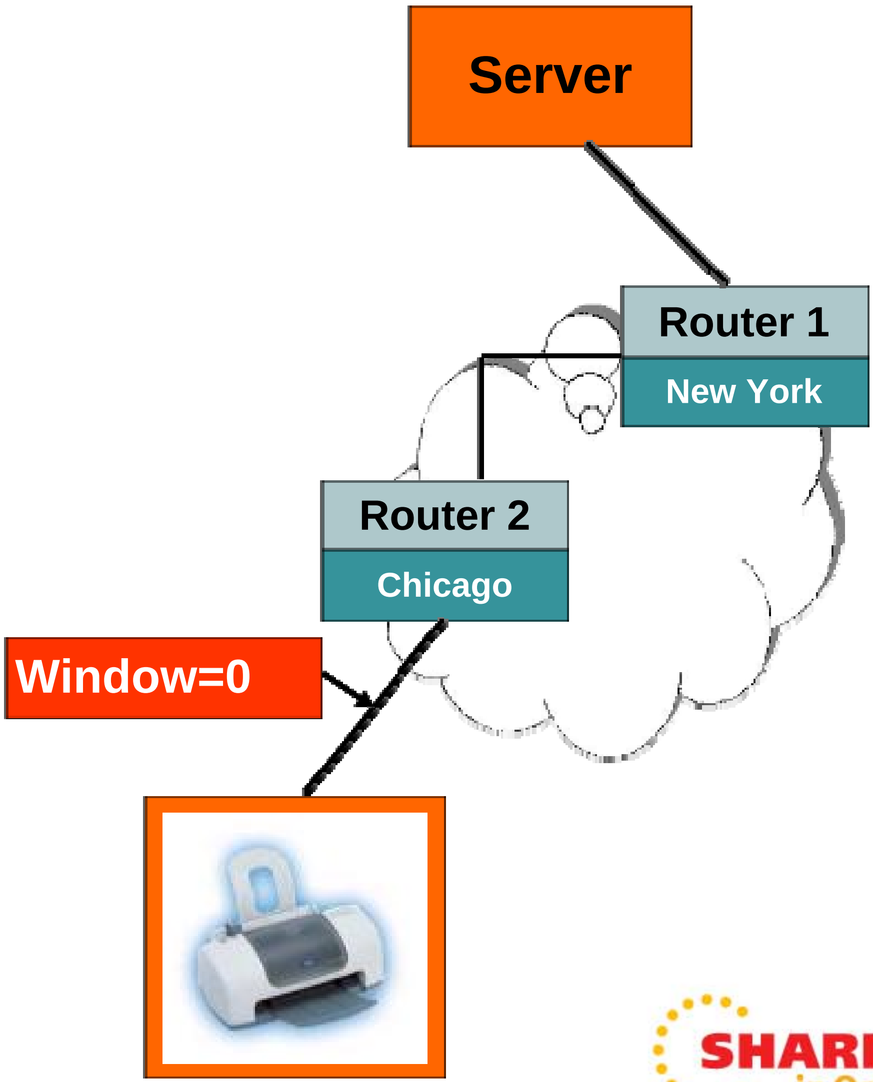
Protocol Header : 28
000000 04380017 B36D24B0 00000000 70024000 0A9C0000 02040550 01010402

-- SYN Packet: TCP Open Connection.
-- Will send out 13 packets (12 at 1,448 and 1 with remainder) before waiting for an ACK.

Congestion Window = 0

- If the window size is set to 0, then the sender should stop sending. The buffers may be full at the receiver side. Network printers are the most likely to advertise a 0 window.

- This may be because:
  - Paper jam in the printer
  - Printer is out of paper



Monitoring Congestion Window

```
CLIENT NAME: ITSCM                    CLIENT ID: 00000087
  LOCAL SOCKET: 192.168.1.231..44444
  FOREIGN SOCKET: 192.168.1.100..1684
  CONGESTIONWINDOW:    0000005840        SLOWSTARTTHRESHOLD: 0000065535
```

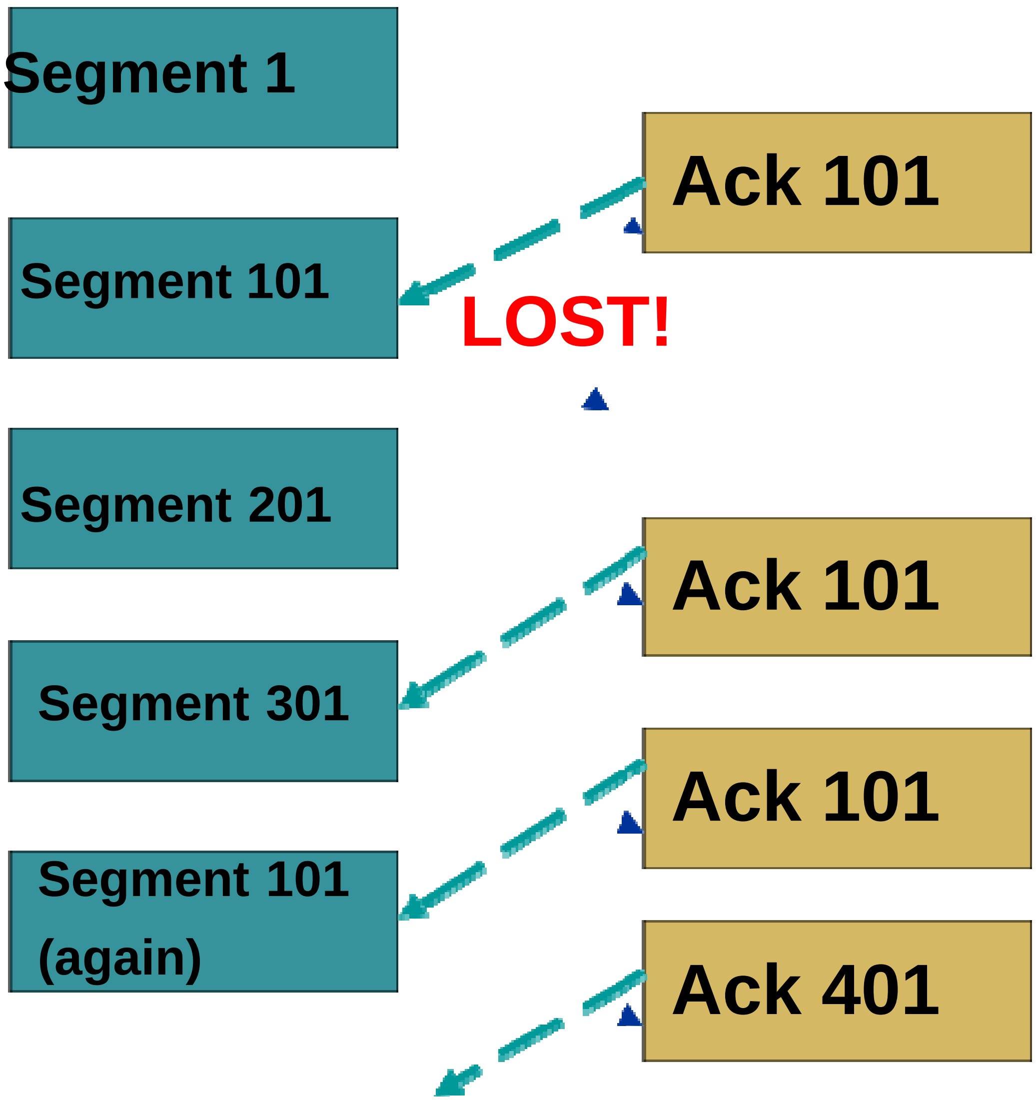
- **Netstat All or SNMP (z/OS MIB TCP connection entry)**
  - **Provide congestion window on a per connection basis**
  - **Must sample regularly to have meaning**

- **SMF Record 119 (TCP termination record) shows Congestion Window Size at time of connection close**

- **Network Management API – shows if congestion window ever went to 0 during the life of the connection.**

After z/OS 1.4, there is a new network management API which provides some measurements.
You need to write an application to retrieve the data periodically.

What is a Duplicate ACK?



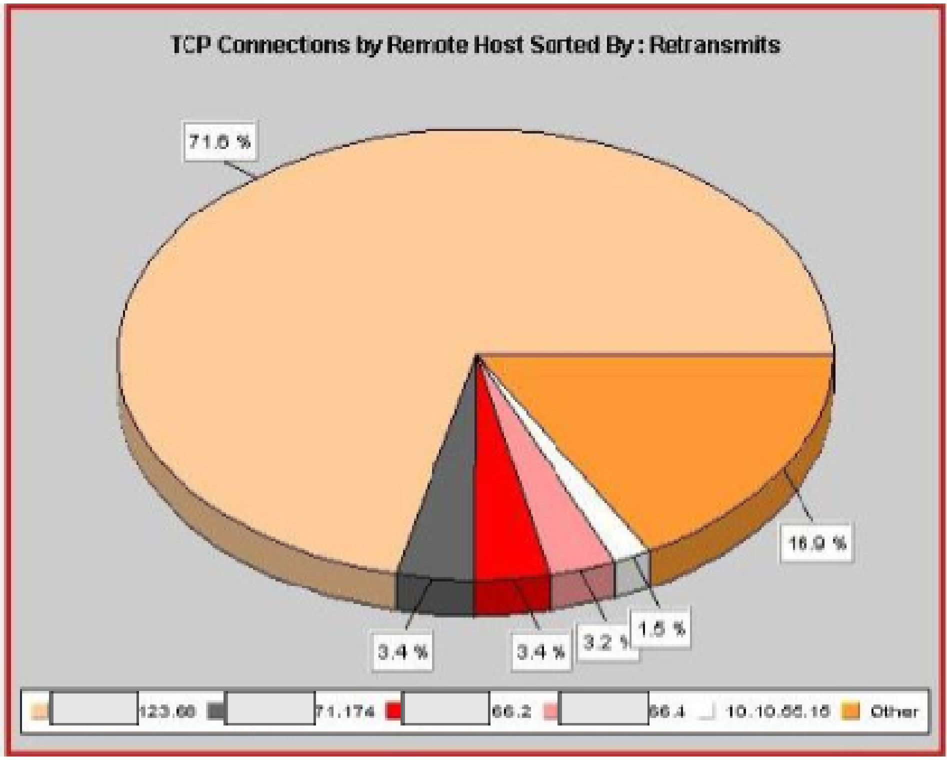
- Assume each segment has 100 bytes.
- Ack is for the next byte of data it is waiting for.
- A duplicate ack is sent when a packet is received and the sequence number indicates that it does not contain the byte you are waiting for.

TCP Retransmits By Remote Address

TCP Connections by Remote Host

Sorted By : Retransmits
Showing Entries 1 - 5
Yearly Report
[redacted]

-	Remote Host	Number of Connections		Avg Round Trip Time	Avg RTT Variance	Total Retransmits	
1	.123.68	67K	(0.64%)	281	138	1M	(71.61%)
2	.71.174	32K	(0.3%)	120	411	86K	(3.42%)
3	.66.2	23K	(0.22%)	106	82	85K	(3.38%)
4	.66.4	23K	(0.22%)	107	82	80K	(3.15%)
5	10.10.55.15	9K	(0.09%)	7	123	39K	(1.54%)
-	All Addresses	10M		-	-	2,536,933	



- Five remote addresses are responsible for over 80% of the retransmits.
- Duplicate acknowledgments show a similar pattern to the retransmits.

Monitoring Retransmits and Dup Acks

Listener Performance Profile

Error Log

Showing First Ten Thousand Entries

[redacted]

-	Local Address	Local Port	Remote Address	Avg Dup. ACKs
1	[redacted]	2389	[redacted].123.66	11544
2	[redacted]	23	[redacted].43.76	579
3	[redacted]	23	[redacted].220.107	438
4	[redacted]	23	[redacted].38.72	363
5	[redacted]	23	[redacted].54.82	93

- **Netstat All or SNMP (z/OS MIB TCP connection entry)**
  - **Provide retransmits / duplicate acknowledgments on a per connection basis**
  - **Must sample regularly to have meaning**
- **SMF Record 119 (TCP termination record) shows retransmits at time of connection close (after z/OS 1.8 will also show dup acks)**
- **Network Management API (SYSTCPCN) shows retransmits at time of connection close (after z/OS 1.8 will also show dup acks)**
- **Network Management API (EZBNMIFR) shows retransmits / duplicate acknowledgments on a per connection basis. Must sample regularly.**

Tuning TCP Saves Money

- Eliminate errors and unneeded traffic and benefit from:
  - Lower CPU usage
  - Less frequent hardware upgrades
  - Lower costs for MIPS-based software charges
  - Increased bandwidth availability
  - Increased technical staff productivity

- *Focus on problem solving and tuning.*

Before Tuning

Name	TCP Listeners	UDP Listeners	TCP Connections	TCP Segments In	TCP Segments Out	UDP Throughput In	UDP Throughput Out	Listener Errors	Needed Interfaces Down	Interface Bytes	IP Errors	TCP Errors	UDP Errors	ICMP Errors
	37	15	1,872	129,829	116,609	2,003	1,214	0	0	0	0	1,895	0	18
	86	17	1,469	159,649	166,982	6,662	6,743	20	0	0	0	2,501	72	134

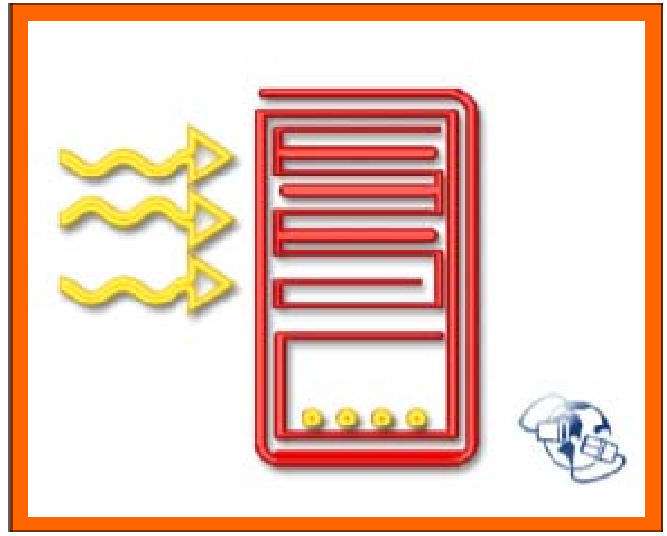
- Data from a recent Network Health Check reveal TCP, UDP, ICMP, and listener errors for both systems.
- Over 2,000 errors per 3-minute interval.
- With tuning these numbers fall significantly.
- Errors contribute to TCP/IP SRB usage.

After Tuning

Name	TCP Listeners	UDP Listeners	TCP Connections	TCP Segments In	TCP Segments Out	UDP Throughput In	UDP Throughput Out	Listener Errors	Needed Interfaces Down	Interface Bytes	IP Errors	TCP Errors	UDP Errors	ICMP Errors
	36	14	1,876	119,305	109,697	1,811	1,802	0	0	0	0	787	0	1
	84	15	392	107,540	115,269	4,622	4,621	0	0	0	0	359	0	0

- After a Health Check and tuning efforts lasting 2 -3 weeks, the listener and UDP errors for both systems have been completely eliminated.
- The ICMP errors for both systems are nearly eliminated.
- The TCP errors have been cut to 1/4 to 1/3 of what they used to be.
- TCP CPU usage dropped

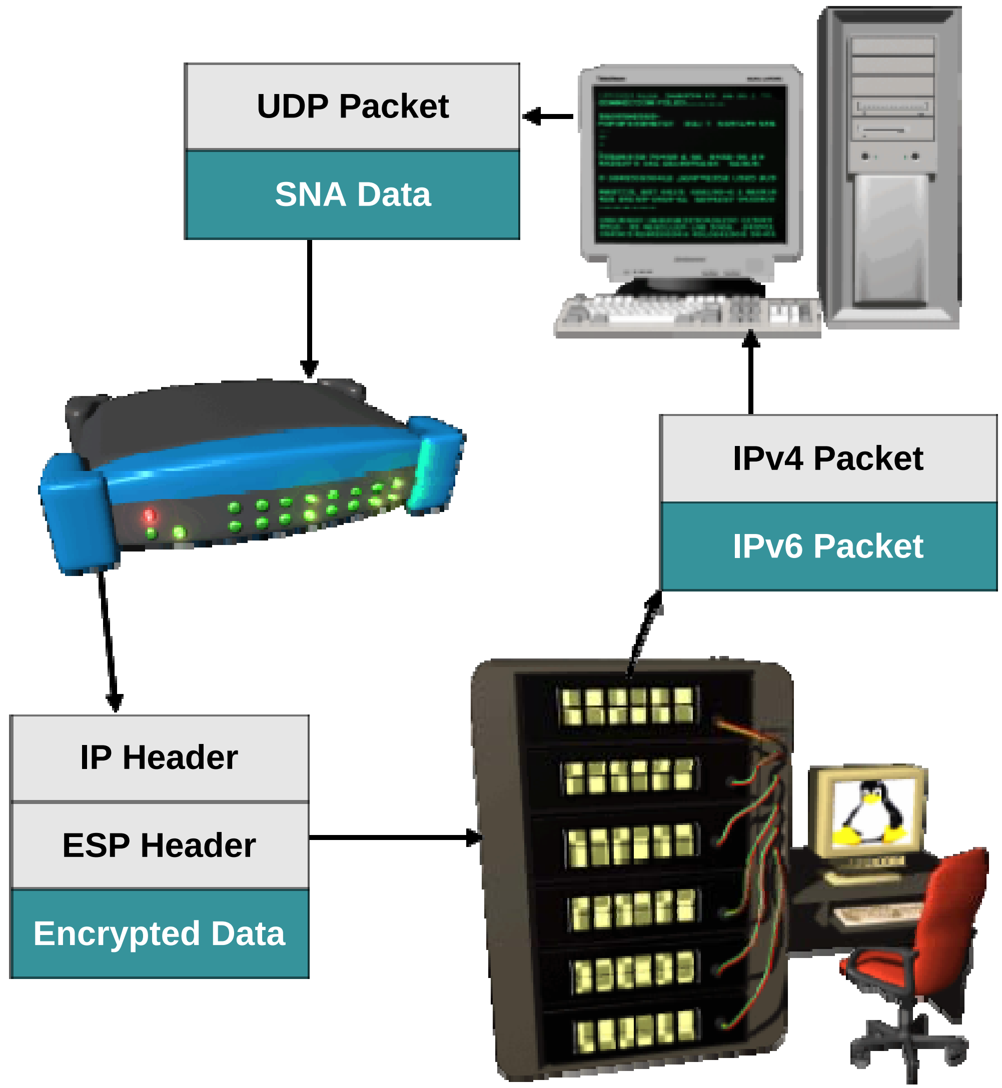
The Silent Killer

- You may not even realize you have problems with TCP/IP.
- Just as cholesterol in the heart can be a silent killer, retransmissions, excessive connections, and unneeded traffic can clog up the network.
- And… these problems are preventable!

Today's Complex Network Protocols

- Today, the packets we see are quite complex. They may have multiple protocols or multiple headers.
  - HPR over UDP
  - IPSec
  - Tunneling (6to4) for IPv6
- Why?
  - Because of the need to integrate legacy SNA networks with TCP/IP
  - Security
  - IPv4 and IPv6 are incompatible protocols



EE Trace Analysis

- Enterprise Extender uses HPR/RTP within IP/UDP.
- Let's decode the packets ourselves. We have to decode multiple headers.
- Inside some of the headers are indicators of congestion or problems
- RTP will retransmit data, if needed, so control information for retransmission is in some of the headers.
- RTP will try to adapt to changing network conditions, so some headers contain information needed for flow control.

EE Packet Headers

- IP Header (20 bytes)
- UDP Header (8 bytes)
- LLC Header (3 bytes)
- NLC (variable)
- RTP Header (20 bytes)
- Optional RTP Segments
- FID5 TH
- RH (3 bytes)
- RU – SNA Data

Inside Products Offerings

Classes: www.insidethestack.com/classes.html

- TCP Diagnostics
- TCP/IP Trace Analysis, EE Trace Analysis
- IP Security Protocols (SSL, IPSec)
- IPv6

Consulting: www.insidethestack.com/consulting.htm

- Network Health Check / EE Health Check
- TCP Problem Resolution
- Planning for the Future

Products: www.insidethestack.com/products.htm

- Inside the Stack, Early Warning System, Connection Log, Availability Checker
- TCP Problem Finder, EE Problem Finder, SSL Problem Finder
- TCP Response Time Monitor, TN3270 Response Time Reports
- 2cSNA (NetView replacement)

Webcasts: www.insidethestack.com/webcasts.htm