SHARKFEST '13

Wireshark Developer and User Conference

Mike Canney

Principal Network Analyst

getpackets.com

My contact info

contact

Mike Canney,

Principal Network Analyst, getpackets.com

- canney@getpackets.com
- 319.389.1137

Capture Strategies

capture

Capture to Disk Appliance (CDA)

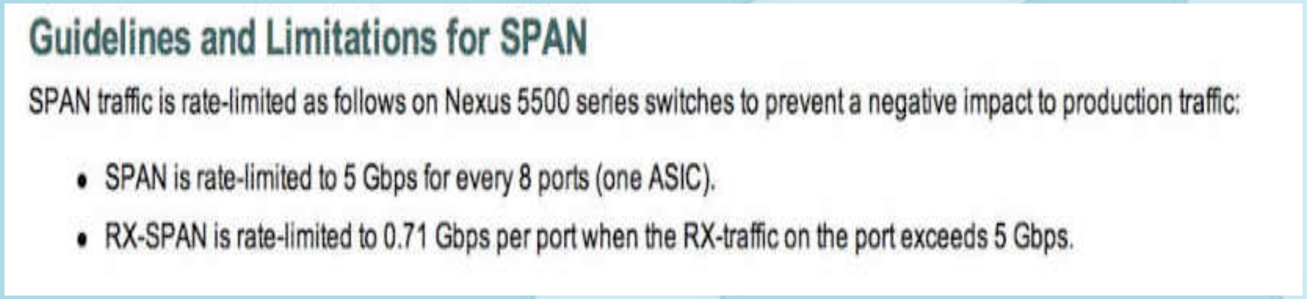
Commercial vs. Free Capture

capture

- Define your capture strategy
  - Data Rates
  - What are my goals? Troubleshooting vs. Statistical information.
  - Do I need to capture every packet?

SPAN vs. TAP

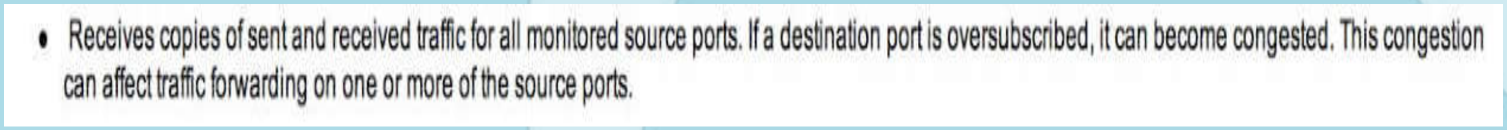
SPAN



Guidelines and Limitations for SPAN

SPAN traffic is rate-limited as follows on Nexus 5500 series switches to prevent a negative impact to production traffic:

- SPAN is rate-limited to 5 Gbps for every 8 ports (one ASIC).
- RX-SPAN is rate-limited to 0.71 Gbps per port when the RX-traffic on the port exceeds 5 Gbps.



- Receives copies of sent and received traffic for all monitored source ports. If a destination port is oversubscribed, it can become congested. This congestion can affect traffic forwarding on one or more of the source ports.

Capture to Disk Appliance (on a budget)

budget

- What is needed?
  - dumpcap is a command line utility included with the Wireshark download to enable ring buffer captures.
  - Use an inexpensive PC or laptop (best to have 2 NICs or more).
  - Basic batch file to initiate capture.
  - Cascade Pilot

Dumpcap Example

example

```
cd \program files (x86)\wireshark
dumpcap -i 1 -s 128 -b files:100 -b filesize:
2000000 –w c:\traces\internet
\headersonly1.pcap
```

This is a basic batch file that will capture off of interface 1, slice the packets to 128 bytes, write 100 trace files of ~2 Gigabytes, and write the trace file out to a pcap file.

So why did I write multiple 2 Gig trace files?

trace file

- Pilot!
- Pilot can easily read HUGE trace files.
- This allows us to utilize our CDA in ways no other analyzer can.
- I personally have sliced and diced 100 GB trace files in Pilot in a matter of seconds.

So how does this all work together?

practice

- Directory full of 2GB trace files, all time stamped based on when they were written to disk.
- User calls in and complains that “the network” is slow.
- Locate that trace file based on time and date and launch Pilot.

Instructor Demo

demo

Troubleshooting user
“Network Issue”

Think about what you just saw

hmmm

- From a 2 GB trace file we were able to:
  - Look at the total Network throughput.
  - See what applications were consuming the bandwidth.
  - Identify the user that was responsible for consuming the bandwidth.
  - Identify the URI's the user was hitting and what the response times were.
  - Drill down to the packets involved in the slow web response time in Wireshark.
- All in a matter of a few seconds.

Much better and Still on a Budget…

- vShark
  - Cascade Virtual Shark
  - Low cost, up to 2 TB of capture capability
  - Can analyze within a ESX server as well as build a stand alone CDA
  - Great for 1 GB SPANs and below
  - My new, CDA of choice

Application Analysis

Apps

Using Pilot and Wireshark to troubleshoot Application issues

So why focus on the Application?

- In many cases it is the Network Engineers that have the tool set to help pinpoint where the problem exists.
- "It's not the Network!" - The Network is guilty until proven innocent.
- Application performance issues can impact your business/customers ability to make money.
- User Response time is "Relative".
- Intermittent performance issues (moving target).

focus

The “moving target”

- Analyzer placement - Two options
  - Move the analyzers as needed
  - Capture anywhere and everywhere
- To defend the Network multiple capture points of the problem is often the best solution.

Scenario X

Your company has a remote site that is connected back to the Data Center via a T1. There are 25-30 users at this site and they are complaining that the Network is slow. Not only is is slow but it is down. When you check your bandwidth usage you see that it is only ~300Kbps. You decide to take a trace file at the Data Center and at the Remote site to see what is going on.

Why are there so many application issues?

- Applications are typically developed in a "golden" environment
  - Fastest PCs
  - High Bandwidth/low latency
- When applications move from test (LAN) to production (WAN) the phone starts ringing with complaints coming in.

help

The Application QA Lifecycle

qa cycle

- In most organizations, applications go through a QA process
- Typical QA/App developers test the following:
  - Functional tests
  - Regression tests
  - Stress tests (server)
  - Rinse and Repeat
- What is often missing is “Networkability” testing
- All QA Lifecycles should include Networkability testing

Application Networkablility Testing

testing

- Identify key business transactions, number of users and network conditions the application will be deployed in.
- Simulation vs. Emulation
  - Simulation is very quick, often gives you rough numbers of how an application will perform over different network conditions.
  - Emulation is the only way to determine when an application will “fail” under those conditions.
- A Combination of both is recommended.

Top Causes for Poor Application Performance

Top 6

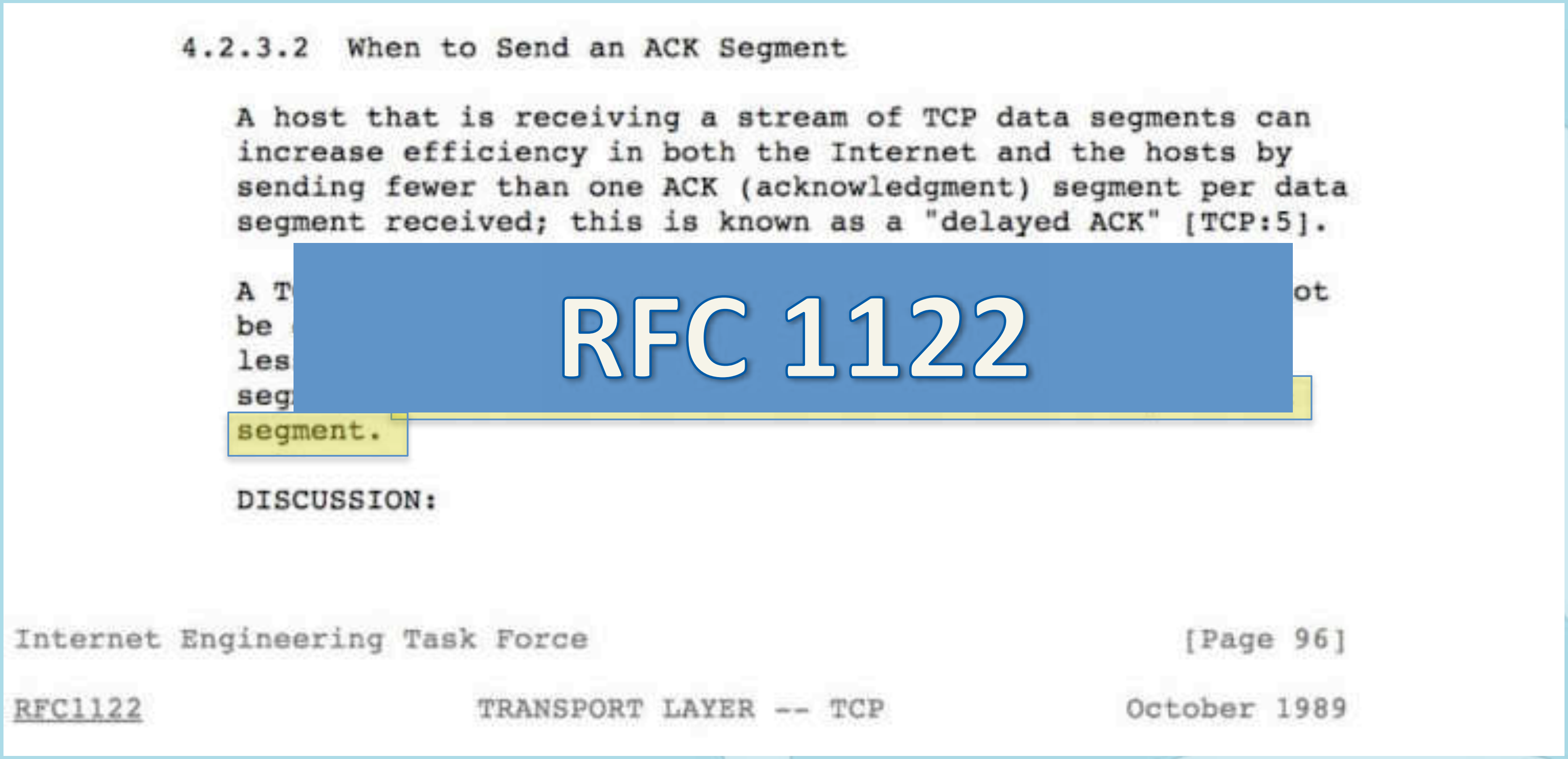
- TCP
- Application Turns
- Misconfigured Devices
- Layer 7 Bottlenecks
- Congestion (network)
- Processing Delay

Causes for Slow Application Performance

TCP

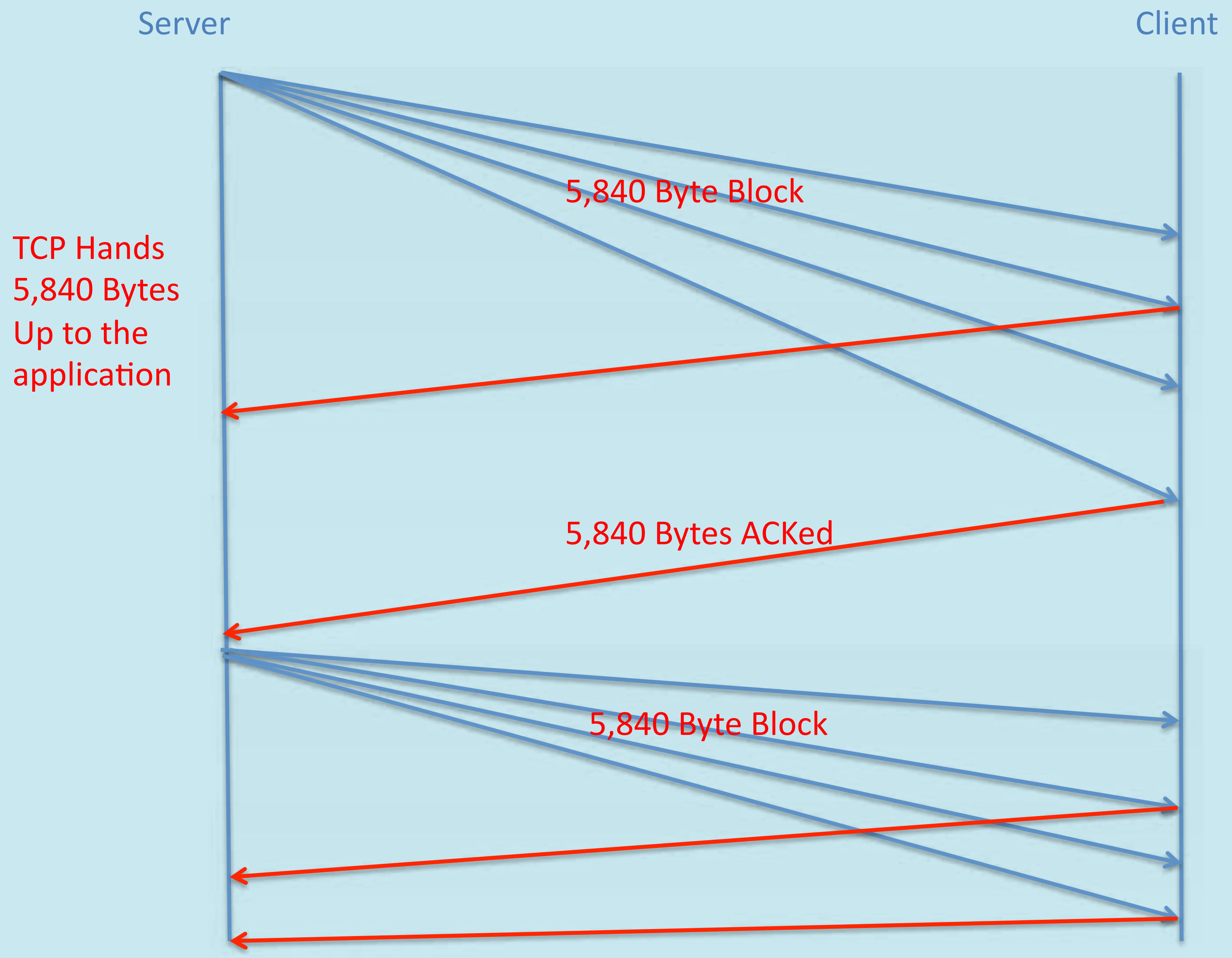
tcp

When are TCP ACKs sent?



```
4.2.3.2  When to Send an ACK Segment

   A host that is receiving a stream of TCP data segments can
   increase efficiency in both the Internet and the hosts by
   sending fewer than one ACK (acknowledgment) segment per data
   segment received; this is known as a "delayed ACK" [TCP:5].

   A T                                                      ot
   be
   les
   seg
   segment.

   DISCUSSION:


Internet Engineering Task Force                            [Page 96]

RFC1122              TRANSPORT LAYER -- TCP             October 1989
```

RFC 1122


Server
Client
5,840 Byte Block
TCP Hands
5,840 Bytes
Up to the
application
5,840 Bytes ACKed
5,840 Byte Block

TCP Window Size

- The TCP Window Size defines the host’s receive buffer.
- Large Window Sizes can sometimes help overcome the impact of latency.
- Depending on how the application was written, advertised TCP Window Size may not have an impact at all (more on this later).

size

TCP Inflight Data

- The amount of unacknowledged TCP data that is on the wire at any given time.
- TCP inflight data in limited by the following:
  - TCP Retransmissions
  - TCP Window Size
  - Application block size
- The amount of TCP inflight data will never exceed the receiving devices advertised TCP Window Size.

Scenario VII – The Slow File Transfer

- A customer is having an issue with their FTP to one of their customers.   They had recently installed a dedicated FTP server for this specific customer. They had a 10Mbps MPLS connection and as expected, the Network was being blamed for the poor file transfers. When we checked the Bandwidth usage on the MPLS connection all looked well. In fact, they were hardly using any of their bandwidth. The customer was planning to upgrade the MPLS connection from 10Mbps to 20Mbps in hopes to speed up the file transfers but decided to let us help them take a look at the issue before making the purchase of the additional bandwidth.

Causes for Slow Application Performance

Application Turns

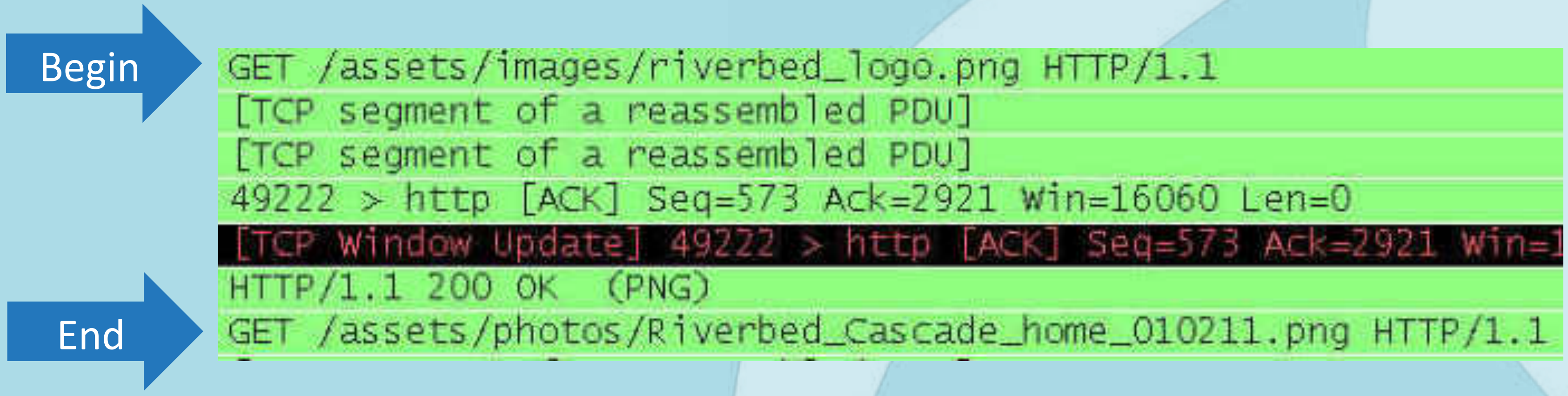
turns

Application Turns

turns

- An Application Turn is a request/response pair
- For each "turn" the application must wait the full round trip delay.
- The greater the number of turns, the worse the application will perform over a WAN (latency bound).

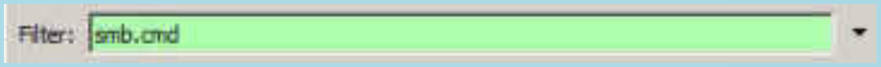
App Turn

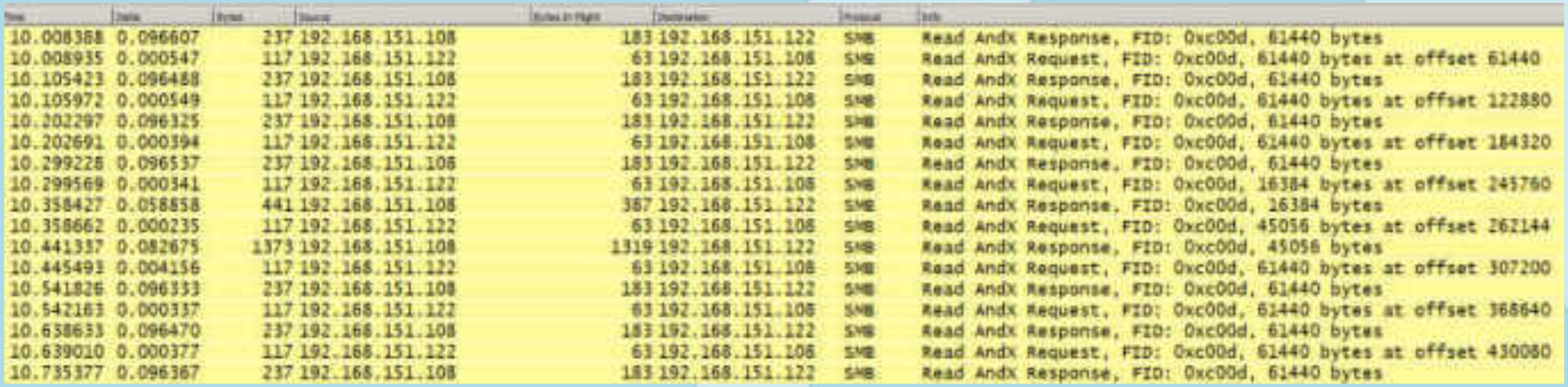


```
GET /assets/images/riverbed_logo.png HTTP/1.1
[TCP segment of a reassembled PDU]
[TCP segment of a reassembled PDU]
49222 > http [ACK] Seq=573 Ack=2921 Win=16060 Len=0
[TCP Window Update] 49222 > http [ACK] Seq=573 Ack=2921 Win=1
HTTP/1.1 200 OK  (PNG)
GET /assets/photos/Riverbed_Cascade_home_010211.png HTTP/1.1
```

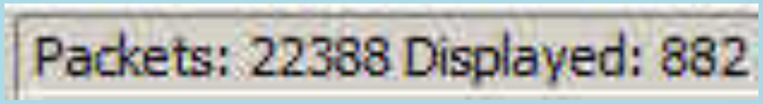
Example in Wireshark

Display Filter:


Filter: smb.cmd



Time	Delta	Bytes	Source	Bytes in Flight	Destination	Protocol	Info
10.008388	0.096607	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes
10.008935	0.000547	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 61440 bytes at offset 61440
10.105423	0.096488	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes
10.105972	0.000549	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 61440 bytes at offset 122880
10.202297	0.096325	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes
10.202691	0.000394	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 61440 bytes at offset 184320
10.299228	0.096537	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes
10.299569	0.000341	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 16384 bytes at offset 245760
10.358427	0.058858	441	192.168.151.108	387	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 16384 bytes
10.358662	0.000235	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 45056 bytes at offset 262144
10.441337	0.082675	1373	192.168.151.108	1319	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 45056 bytes
10.445493	0.004156	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 61440 bytes at offset 307200
10.541826	0.096333	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes
10.542163	0.000337	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 61440 bytes at offset 368640
10.638633	0.096470	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes
10.639010	0.000377	117	192.168.151.122	63	192.168.151.108	SMB	Read AndX Request, FID: 0xc00d, 61440 bytes at offset 430080
10.735377	0.096367	237	192.168.151.108	183	192.168.151.122	SMB	Read AndX Response, FID: 0xc00d, 61440 bytes


Packets: 22388 Displayed: 882

882 Application Turns in this trace

App Turns and Latency

latency

- It is fairly easy to determine App Turns impact on end user response time
  - Multiply the number of App Turns by the round trip delay:
    - 10,000 turns * .050 ms delay = 500 seconds due to latency
- Note, this has nothing to do with Bandwidth or the Size of the WAN Circuit

So what causes all these App Turns?

cause

- Size of a fetch in a Data Base call
- Number of files that are being accessed
- Loading single images in a Web Page instead of using an image map
- Number of bytes being retrieved and how they are being retrieved (block size)

Pop Quiz!

- We have an application that uses a 16,384 KB block size
- This application is going to be rolled out over a DS3 with 40 ms of round trip latency.
- Users are complaining that the network is slow and they need to upgrade the DS3 to a 1 Gbps dedicated link.
- What kind of throughput should I expect?

Hint…

- Forget about the bandwidth, it has nothing to do with the throughput

Answer

- Very simple calculation
- Throughput = Blocksize / Round Trip Delay

- (16,384 / .04) * 8 = 3,278,800 bits per second

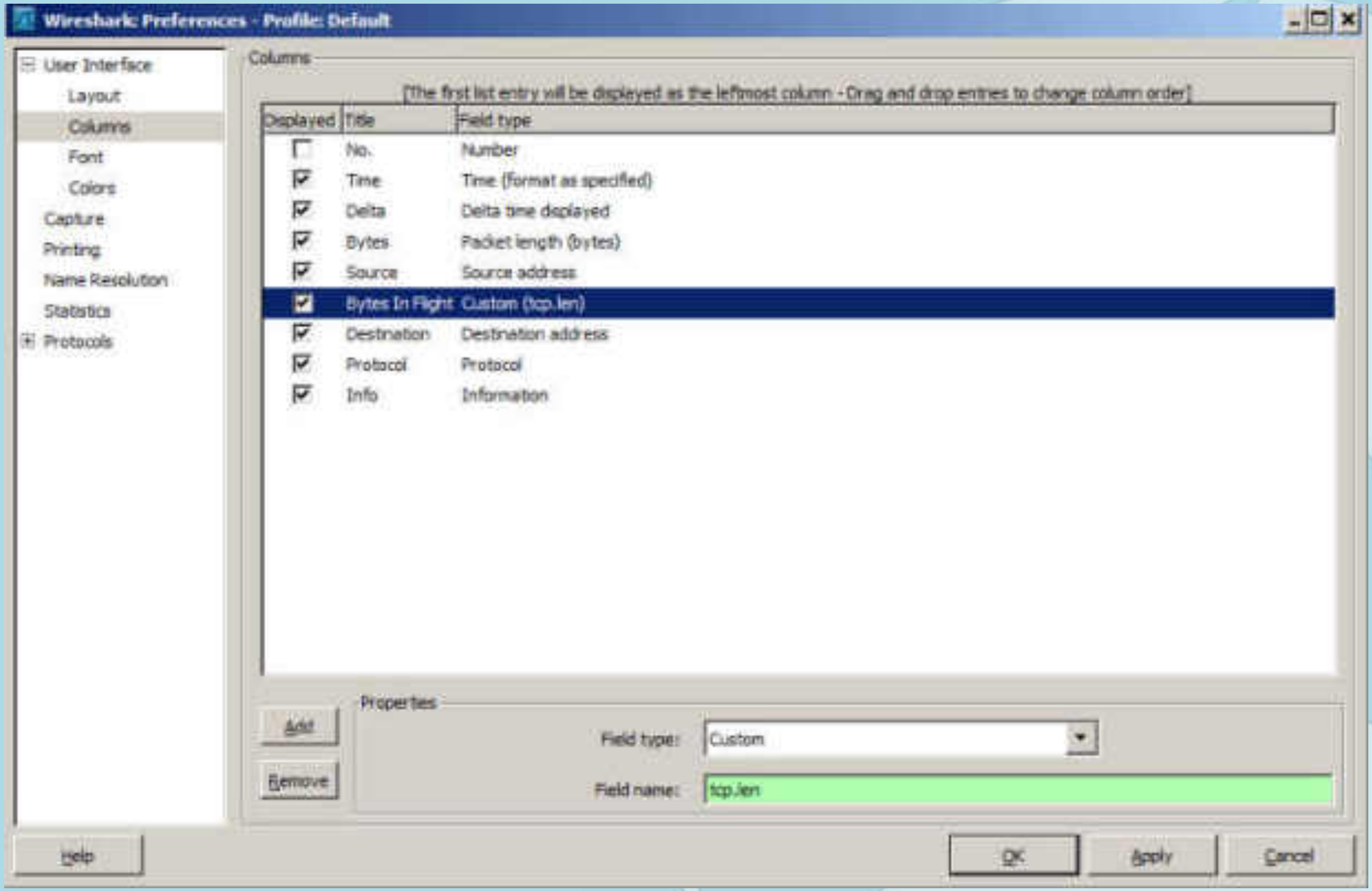
- Again, note, this has NOTHING to do with bandwidth it's all on the application.

Remember to Calculate BDP

- Bandwidth Delay Product
- BW * RT Latency /8 = offered load to fill the pipe

- (44,000,000 * .04) /8 = 220,000 bits per second to fill the DS3

TCP Inflight Data in Wireshark



Wireshark: Preferences - Profile: Default

Columns

[The first list entry will be displayed as the leftmost column - Drag and drop entries to change column order]

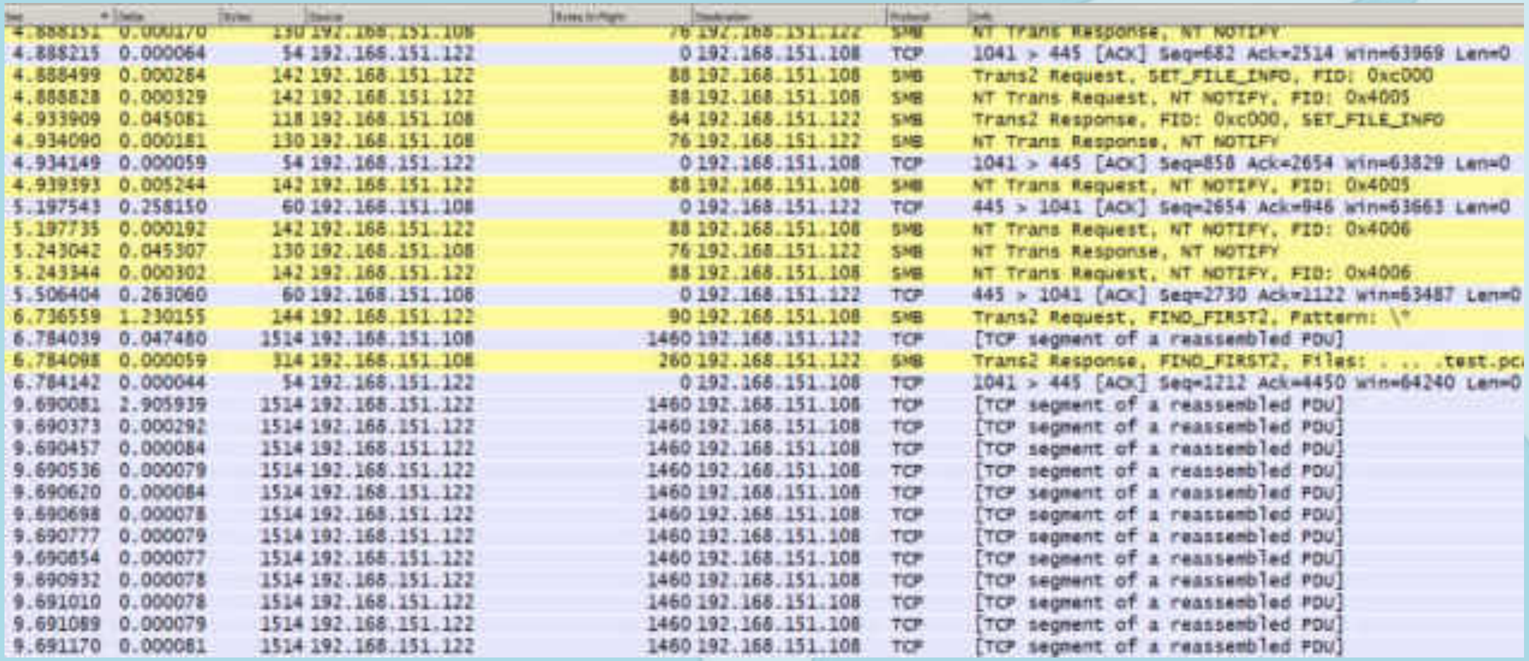
Displayed	Title	Field type
☐	No.	Number
✓	Time	Time (format as specified)
✓	Delta	Delta time displayed
✓	Bytes	Packet length (bytes)
✓	Source	Source address
✓	Bytes In Flight	Custom (tcp.len)
✓	Destination	Destination address
✓	Protocol	Protocol
✓	Info	Information

Properties

Field type: Custom

Field name: tcp.len

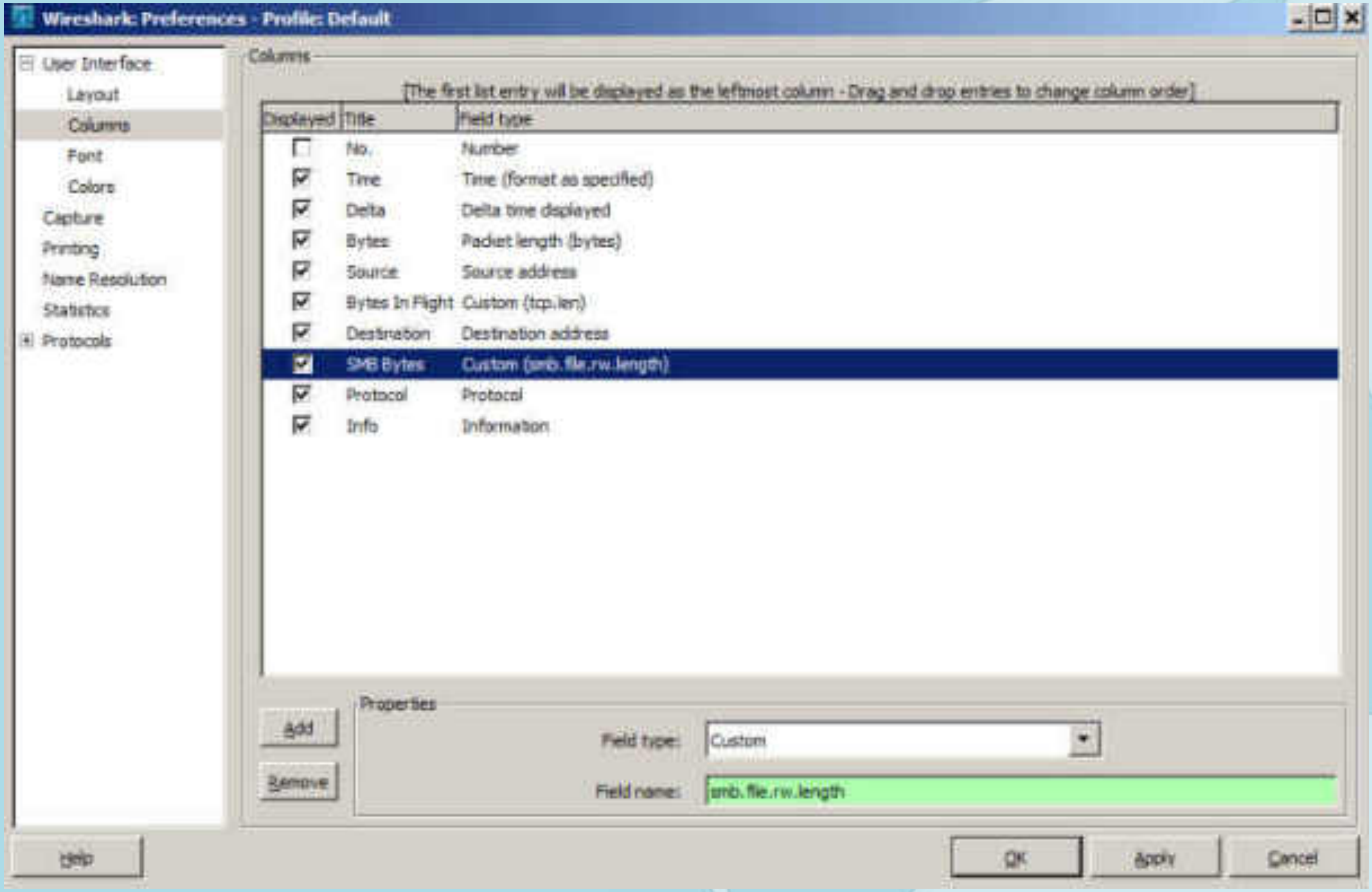
TCP Inflight Data in Wireshark



Time	Delta	Bytes	Source	Bytes in Flight	Destination	Protocol	Info
4.888151	0.000170	130	192.168.151.108	76	192.168.151.122	SMB	NT Trans Response, NT NOTIFY
4.888215	0.000064	54	192.168.151.122	0	192.168.151.108	TCP	1041 > 445 [ACK] Seq=682 Ack=2514 Win=63969 Len=0
4.888499	0.000284	142	192.168.151.122	88	192.168.151.108	SMB	Trans2 Request, SET_FILE_INFO, FID: 0xc000
4.888828	0.000329	142	192.168.151.122	88	192.168.151.108	SMB	NT Trans Request, NT NOTIFY, FID: 0x4005
4.933909	0.045081	118	192.168.151.108	64	192.168.151.122	SMB	Trans2 Response, FID: 0xc000, SET_FILE_INFO
4.934090	0.000181	130	192.168.151.108	76	192.168.151.122	SMB	NT Trans Response, NT NOTIFY
4.934149	0.000059	54	192.168.151.122	0	192.168.151.108	TCP	1041 > 445 [ACK] Seq=858 Ack=2654 Win=63829 Len=0
4.939393	0.005244	142	192.168.151.122	88	192.168.151.108	SMB	NT Trans Request, NT NOTIFY, FID: 0x4005
5.197543	0.258150	60	192.168.151.108	0	192.168.151.122	TCP	445 > 1041 [ACK] Seq=2654 Ack=946 Win=63663 Len=0
5.197735	0.000192	142	192.168.151.122	88	192.168.151.108	SMB	NT Trans Request, NT NOTIFY, FID: 0x4006
5.243042	0.045307	130	192.168.151.108	76	192.168.151.122	SMB	NT Trans Response, NT NOTIFY
5.243344	0.000302	142	192.168.151.122	88	192.168.151.108	SMB	NT Trans Request, NT NOTIFY, FID: 0x4006
5.506404	0.263060	60	192.168.151.108	0	192.168.151.122	TCP	445 > 1041 [ACK] Seq=2730 Ack=1122 Win=63487 Len=0
6.736559	1.230155	144	192.168.151.122	90	192.168.151.108	SMB	Trans2 Request, FIND_FIRST2, Pattern: *
6.784039	0.047480	1514	192.168.151.108	1460	192.168.151.122	TCP	[TCP segment of a reassembled PDU]
6.784098	0.000059	314	192.168.151.108	260	192.168.151.122	SMB	Trans2 Response, FIND_FIRST2, Files: . .. .test.pc
6.784142	0.000044	54	192.168.151.122	0	192.168.151.108	TCP	1041 > 445 [ACK] Seq=1212 Ack=4450 Win=64240 Len=0
9.690081	2.905939	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690373	0.000292	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690457	0.000084	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690536	0.000079	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690620	0.000084	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690698	0.000078	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690777	0.000079	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690854	0.000077	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.690932	0.000078	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.691010	0.000078	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.691089	0.000079	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]
9.691170	0.000081	1514	192.168.151.122	1460	192.168.151.108	TCP	[TCP segment of a reassembled PDU]

The Bytes in Flight Column shows us how much payload is in each packet.

Easier way for SMB/CIFS



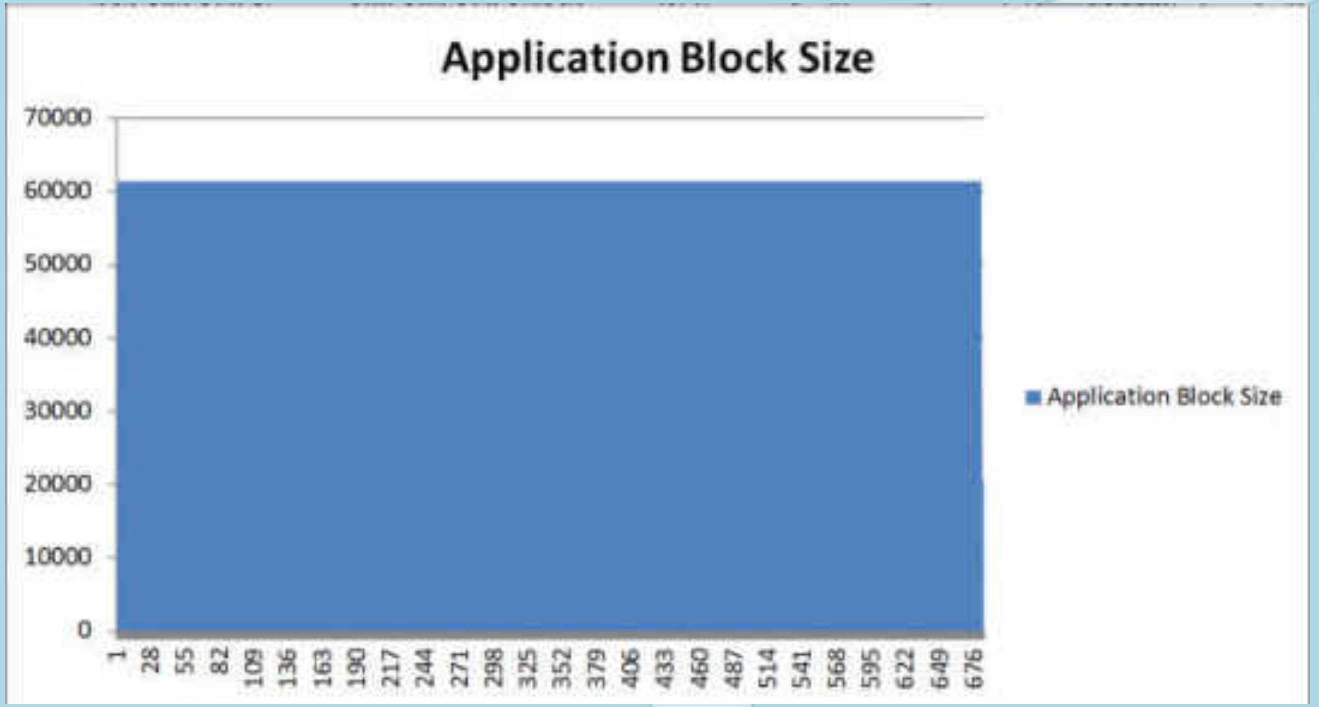
[The first list entry will be displayed as the leftmost column - Drag and drop entries to change column order]

Displayed	Title	Field type
☐	No.	Number
✓	Time	Time (format as specified)
✓	Delta	Delta time displayed
✓	Bytes	Packet length (bytes)
✓	Source	Source address
✓	Bytes In Flight	Custom (tcp.len)
✓	Destination	Destination address
✓	SMB Bytes	Custom (smb.file.rw.length)
✓	Protocol	Protocol
✓	Info	Information

Field type: Custom

Field name: smb.file.rw.length

TCP Inflight Data in Wireshark



Graphed in Excel

TCP Retransmissions

- Every time a TCP segment is sent, a retransmission timer is started.
- When the Acknowledgement for that segment is received the timer is stopped.
- If the retransmission timer expires before the Acknowledgement is received, the TCP segment is retransmitted.

tcp

TCP Retransmissions

tcp flow

- Excessive TCP Retransmissions can have a huge impact on application performance.
- Not only does the data have to get resent, but TCP flow control (Slow Start) kicks into action.

ULPs (upper layer protocols)

- TCP often gets blamed for the ULPs problem.
  - The application hands down to TCP amount of data to go retrieve (application block size)
  - TCP then is responsible for reliably getting that data back to the application layer
    - TCP has certain parameters in which to work with and can usually be tuned based on bandwidth and latency
    - Many times too much focus is put on “tuning” TCP as the fix for poor performance in the network

- If the TCP advertised receive window is set to 64K and the application is only handing down to TCP requests for 16K, where is the bottleneck?

ULPs (upper layer protocols)

Case in point: CIFS/SMB

ulp

Troubleshooting CIFS/SMB

cifs/smb

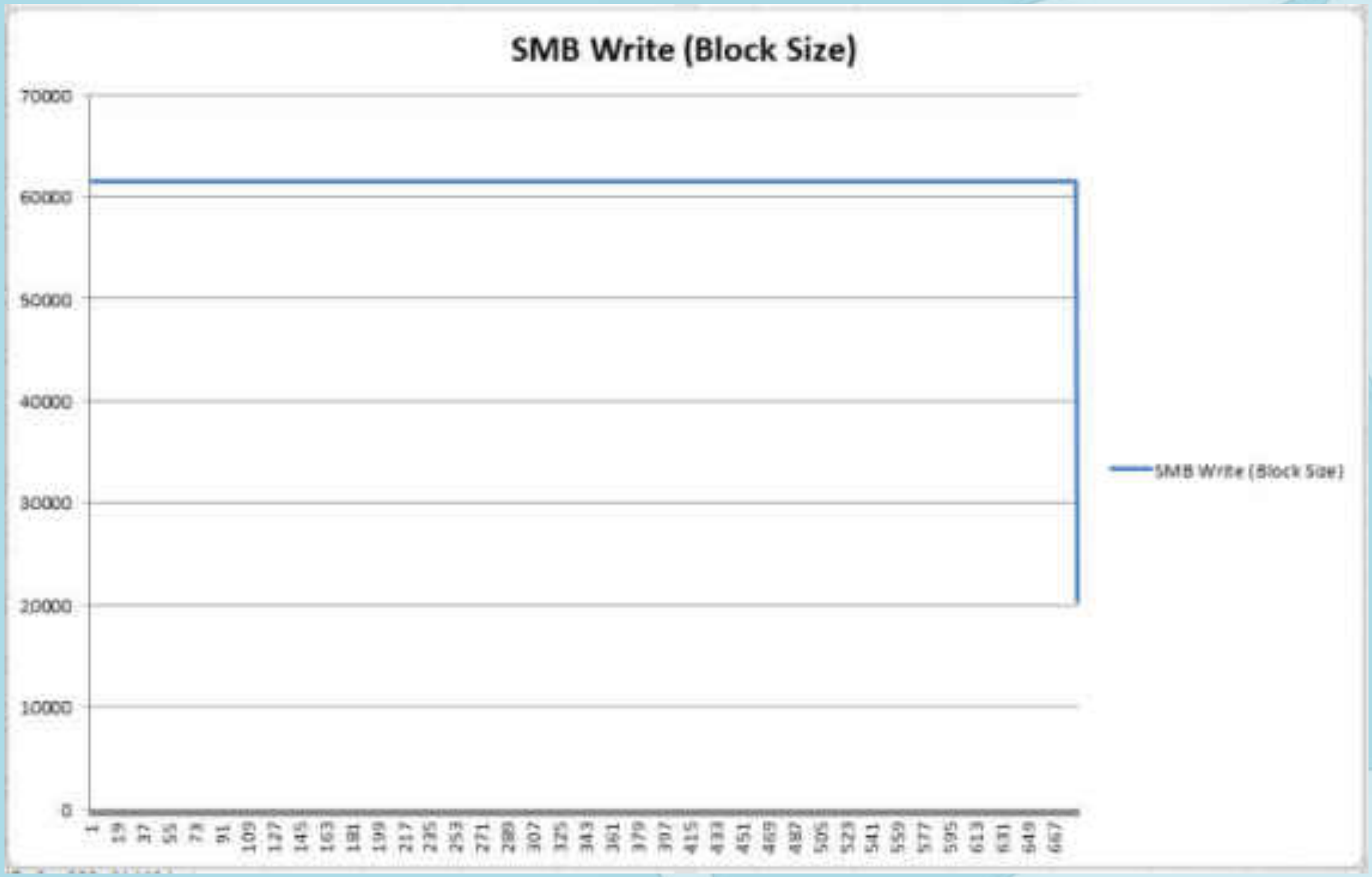
- Arguably the most common File Transfer method used in businesses today.
- SMB was NOT developed with the WAN in mind.
- One of the most "chatty" protocols/ applications I run into (with the exception of poorly written SQL).

CIFS/SMB Quiz

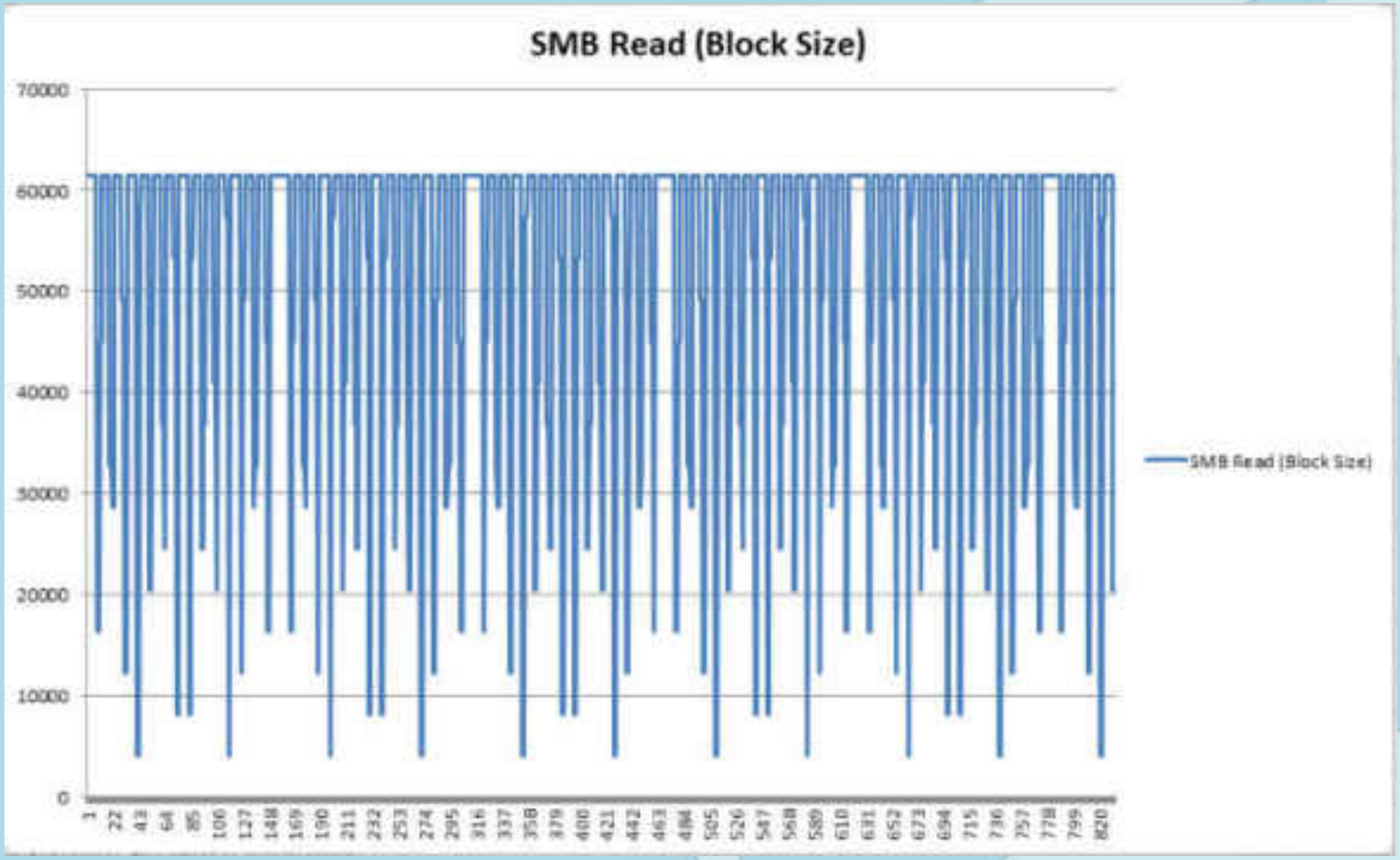
- What is faster using MS File Sharing?
  – Pushing a file to a file server?
  – Pulling a file from a file server?

quiz

ULPs (upper layer protocols)



ULPs (upper layer protocols)



CIFS/SMB

cifs / smb

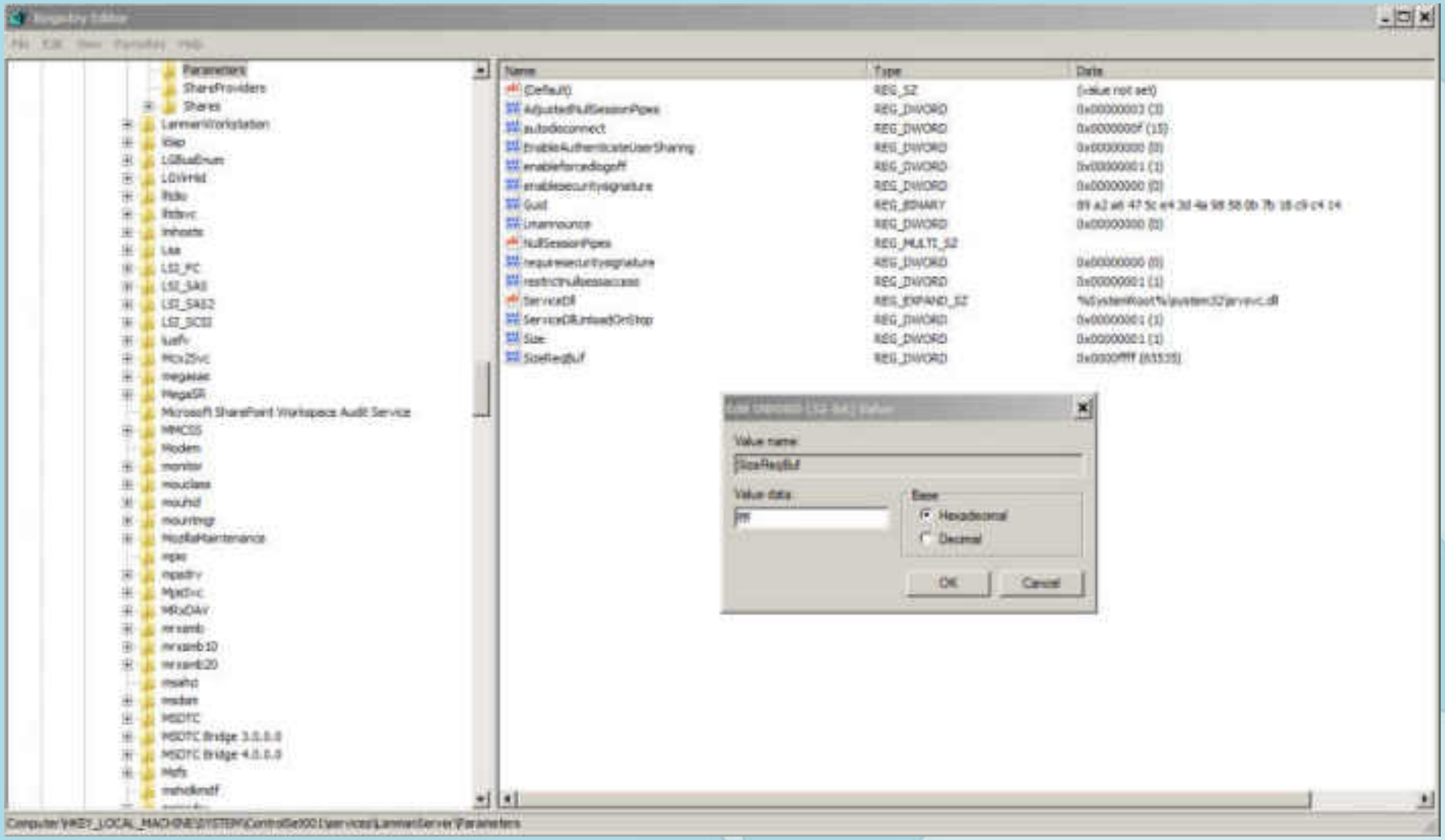
- What is faster using MS File Sharing?
  - Pushing a file to a file server?
  - Pulling a file from a file server?
    - SMB Write (Pushing the file) can almost be 2X as fast as pulling (SMB Read)
    - Depends on the Latency

CIFS/SMB Tuning

tuning

- SMB Maximum Transmit Buffer Size
  - Negotiated MaxBufferSize in the Negotiate Protocol response
  - Default for Windows servers is typically 16644 (dependent upon physical memory)
  - Client default typically 4356

CIF/SMB Tuning



Name	Type	Data
(Default)	REG_SZ	(value not set)
AdjustedNullSessionPipes	REG_DWORD	0x00000003 (3)
autodisconnect	REG_DWORD	0x0000000f (15)
EnableAuthenticateUserSharing	REG_DWORD	0x00000000 (0)
enableforcedlogoff	REG_DWORD	0x00000001 (1)
enablesecuritysignature	REG_DWORD	0x00000000 (0)
Guid	REG_BINARY	89 a2 a6 47 5c e4 3d 4a 98 58 0b 7b 18 c9 c4 14
Lmannounce	REG_DWORD	0x00000000 (0)
NullSessionPipes	REG_MULTI_SZ	
requiresecuritysignature	REG_DWORD	0x00000000 (0)
restrictnullsessaccess	REG_DWORD	0x00000001 (1)
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\system32\srvsvc.dll
ServiceDllUnloadOnStop	REG_DWORD	0x00000001 (1)
Size	REG_DWORD	0x00000001 (1)
SizeReqBuf	REG_DWORD	0x0000ffff (65535)

Edit DWORD (32-bit) Value

Value name: SizeReqBuf

Value data: ffff

Base: [✓ Hexadecimal] Decimal

OK Cancel

Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\LanmanServer\Parameters

CIFS/SMB Tuning

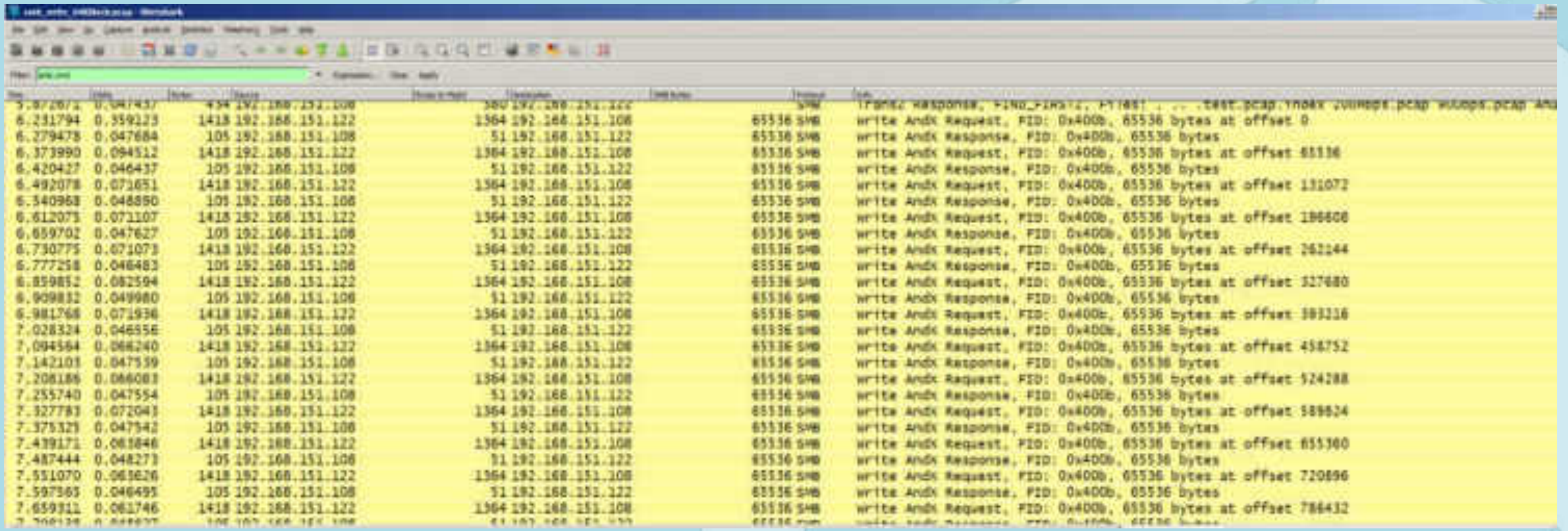
- Caveat:
  - SMB is extremely dependent upon the API
    - Even though you set the max buffer size to 64K, windows “share” data will always get truncated to 60K (61440) even though the server can support 64K

CIFS/SMB Tuning

tuning

- Custom SMB APIs
  - The Windows limitation can be exceeded by programs written to use SMB as they file transfer protocol

CIFS/SMB Tuning



```
5.872671 0.047437    454 192.168.151.108    360 192.168.151.122          SMB  Trans2 Response, FIND_FIRST2, Files: . .. .test.pcap.index 100Mbps.pcap 400bps.pcap
6.231794 0.359123   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 0
6.279478 0.047684    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
6.373990 0.094512   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 65536
6.420427 0.046437    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
6.492078 0.071651   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 131072
6.540968 0.048890    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
6.612075 0.071107   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 196608
6.659702 0.047627    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
6.730775 0.071073   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 262144
6.777258 0.046483    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
6.859852 0.082594   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 327680
6.909832 0.049980    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
6.981768 0.071936   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 393216
7.028324 0.046556    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
7.094564 0.066240   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 458752
7.142103 0.047539    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
7.208186 0.066083   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 524288
7.255740 0.047554    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
7.327783 0.072043   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 589824
7.375325 0.047542    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
7.439171 0.063846   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 655360
7.487444 0.048273    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
7.551070 0.063626   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 720896
7.597565 0.046495    105 192.168.151.108     51 192.168.151.122    65536 SMB  Write AndX Response, FID: 0x400b, 65536 bytes
7.659311 0.061746   1418 192.168.151.122   1364 192.168.151.108    65536 SMB  Write AndX Request, FID: 0x400b, 65536 bytes at offset 786432
```

Note the SMB writes of 65,536

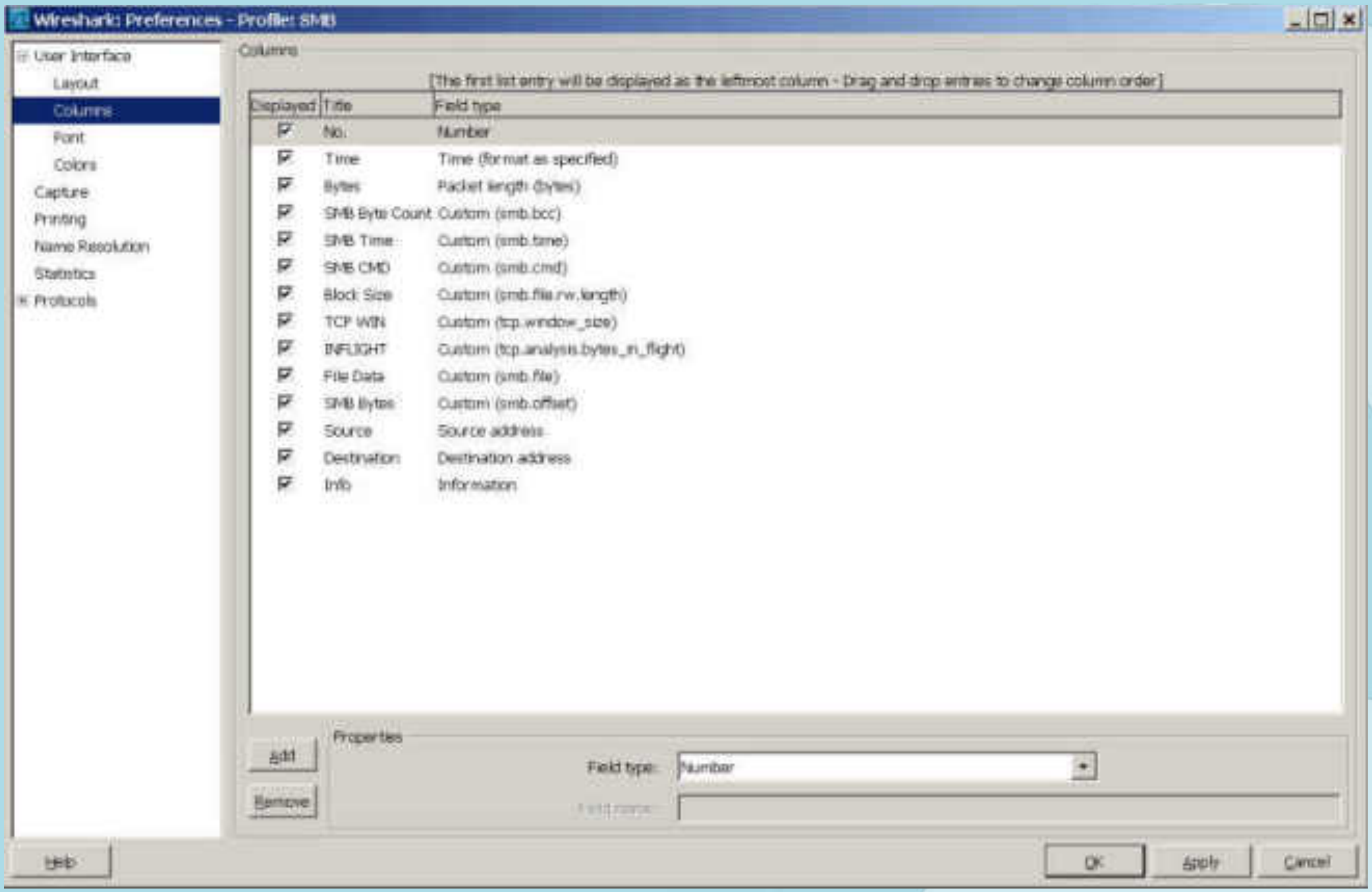
This is a file transfer using a custom API on a Windows XP machine

Instructor Demo of SMB Profiles

demo

Demo of SMB Tracefiles

My personal SMB Profile



[The first list entry will be displayed as the leftmost column - Drag and drop entries to change column order]

Displayed	Title	Field type
✓	No.	Number
✓	Time	Time (format as specified)
✓	Bytes	Packet length (bytes)
✓	SMB Byte Count	Custom (smb.bcc)
✓	SMB Time	Custom (smb.time)
✓	SMB CMD	Custom (smb.cmd)
✓	Block Size	Custom (smb.file.rw.length)
✓	TCP WIN	Custom (tcp.window_size)
✓	INFLIGHT	Custom (tcp.analysis.bytes_in_flight)
✓	File Data	Custom (smb.file)
✓	SMB Bytes	Custom (smb.offset)
✓	Source	Source address
✓	Destination	Destination address
✓	Info	Information

Field type: Number

Take Away Points

points

- Building your own CDA is easy to do and may fit in a majority of the areas you need to capture from
- Pilot, Pilot, Pilot, it's not just a fancy reporting engine for Wireshark!
- Test your applications "Networkability" before they hit production.
- Use the Wireshark Profiles, they will save you a ton of time.

SHARKFEST '13

Wireshark Developer and User Conference

Mike Canney

Principal Network Analyst

canney@getpackets.com