

WSU03: Wireshark Troubleshooting Network Performance

Appendix B: Trace File Catalog

Wireshark University™

The following pages describe the trace files included on the WSU03: Wireshark Troubleshooting Network Performance DVD.

No.	Name	Description
00003	2-specters-fighting.pcap	It's a cat fight! Watch the change of direction in the scan process when one aggressive honeypot gets scanned by another aggressive honeypot. Consider making an IO Graph with two filters (ip.src==24.6.138.50 && tcp.flags==0x02 in black and ip.src==24.6.137.85 && tcp.flags==0x02 in red). Set the tick interval to 0.01 seconds and the Y Axis scale to 10. Turn on the green graph line without any filter applied.
00048	anotherlousyhotelnetwork.pcap	This trace begins with a really slow DNS response. In fact, the client sends out two DNS queries. When the first DNS response arrives, the client shuts down the listening port and responds to the second DNS response with an ICMP Destination Unreachable/Port Unreachable. How much delay was caused by packet loss?
00161	arp-bootup.pcap	This is a classic client bootup sequence beginning with the DHCP Request/ACK sequence (indicating the client is still within its lease time) and moving on to the gratuitous ARP process before sending out an ARP to the default gateway defined in the DHCP ACK [P2]. This isn't a fast process however. Users typically accept sluggish boot processes, but not slow login sequences.
00047	arp-ping.pcap	This trace shows the startup sequence for using a newly-assigned IP address. What might be the cause for the delay between the Gratuitous ARP and the ARP for 10.1.0.1? Do you see recognizable ARP padding in the ICMP Echo request?

The following pages describe the trace files included on the WSU03: Wireshark Troubleshooting Network Performance DVD.

No.	Name	Description
00003	2-specters-fighting.pcap	It's a cat fight! Watch the change of direction in the scan process when one aggressive honeypot gets scanned by another aggressive honeypot. Consider making an IO Graph with two filters (ip.src==24.6.138.50 && tcp.flags==0x02 in black and ip.src==24.6.137.85 && tcp.flags==0x02 in red). Set the tick interval to 0.01 seconds and the Y Axis scale to 10. Turn on the green graph line without any filter applied.
00048	anotherlousyhotelnetwork.pcap	This trace begins with a really slow DNS response. In fact, the client sends out two DNS queries. When the first DNS response arrives, the client shuts down the listening port and responds to the second DNS response with an ICMP Destination Unreachable/Port Unreachable. How much delay was caused by packet loss?
00161	arp-bootup.pcap	This is a classic client bootup sequence beginning with the DHCP Request/ACK sequence (indicating the client is still within its lease time) and moving on to the gratuitous ARP process before sending out an ARP to the default gateway defined in the DHCP ACK [P2]. This isn't a fast process however. Users typically accept sluggish boot processes, but not slow login sequences.
00047	arp-ping.pcap	This trace shows the startup sequence for using a newly-assigned IP address. What might be the cause for the delay between the Gratuitous ARP and the ARP for 10.1.0.1? Do you see recognizable ARP padding in the ICMP Echo request?

00135	bittorrent-idle-crap.pcap	<p>We let a BitTorrent client sit idle for an extended period of time (24 hours) and then decided to check in on it to see if it was talking. Look at all the outbound handshakes! And after sending SYN packets to a number of these folks the BitTorrent client performs some DNS PTR queries to resolve their names. We think we were lucky – this client didn't make a connection to tracker.bittorrent.com. A BitTorrent tracker is a server that 'assists in the communication between peers.'</p> <p>Thanks, but no thanks.</p>
00134	bittorrent-launch-search-maddona.pcap	<p>Simply launching BitTorrent causes a connection to the BitTorrent website as well as surveymonkey.com, questionmarket.com and zedo.com (billing itself as Third Generation Technology of Ad Serving). Is this really the traffic you want on the network? The query for 'madonna' took 3.5 seconds – not the fastest bolt of lightning. Maybe they should talk to Google.</p>
00104	bit-torrent-startup-background.pcap	<p>Don't you just love that BitTorrent stuff? Well, your network doesn't! This is the background traffic from starting up BitTorrent on a clean system. Build an IO graph to see the packets/second and bytes/second rate. You couldn't pay me to put that junk on my network! What a bunch of crap! <IMHO></p>
00103	chargen.pcap	<p>I can't imagine why someone would want Character Generator (CHARGEN) services on their network. Maybe they don't know how to do connectivity tests using UDP or TCP traceroute? This trace depicts an ugly pair of devices exchanging CHARGEN traffic between them. What if one of them had selected a source port of 19? Life would get really painful really fast.</p>

00024	clientdying.pcap	A client system (172.16.1.10) is in trouble. After it boots up the CPU utilization climbs to 100% and the system locks up within 3 minutes. You can see many problems in the trace - incoming DCERPC communications and the client establishing TFTP and IRC communications to remote systems. Follow TCP Stream on the IRC communication. This client is infected with a variant of the sdbot worm.
00178	dhcpjerktakesaddressnopermission.pcap	The DHCP server is down, but the client remembers its last address and decided just to take it back. Of course it does a gratuitous ARP [P3]. The client uses router solicitation (ugh) to try to find a default gateway as well. Finally, 12 seconds in to the trace the DHCP server resurfaces [P8].
00044	dhcp-relay-serverside.pcap	Compare the source MAC address in the Ethernet header with the Client MAC Address inside the DHCP packet to note that this communication is coming from the DHCP Relay Agent to the DHCP Server. The client sends a Requested IP Address and begins with a DHCP Request because it is still within it's lease time.
00043	dhcp-renewtorebind.pcap	The DHCP client is unsuccessful in renewing its IP address from 10.1.0.1 so the client broadcasts the DHCP Request in hopes of finding a new DHCP server. When the DHCP client doesn't get an answer, it gives up its IP address and begins the process of discovery from scratch [P7].
00122	dhcp-server-slow.pcap	You know it's going to be a bad day when the first one you talk to ignores you. In this case the client sends a DHCP Discover out and waits six seconds without a reply (you could hear a pin drop). When the server does finally answer [P3] the client already has another Discover queued up and ready to send – out it goes [P4]. Let's hope the rest of the day goes better.

00019	dns-error-domain.pcap	When a client tries unsuccessfully to get the IP address of <code>www.us.gov</code> , it appends its domain information to the query. This, of course, generates a response of "No Such Name." The client is configured to "Append parent suffixes of the primary domain suffix."
00160	dns-misc.pcap	Compare the DNS lookups required to access <code>www.winpcap.org</code> , <code>www.msnbc.com</code> and <code>www.espn.com</code> . Check the DNS traffic generated when you connect to your corporate servers. Consider baselining that traffic.
00159	dns-misc2.pcap	Browsing to <code>www.cnn.com</code> , <code>www.microsoft.com</code> and <code>www.espn.com</code> generates lots of DNS queries and responses (filtered out in this trace) as they direct you to their ad and data streaming affiliates. Many of the server names imply they are advertising sites.
00120	dns-mxlookup.pcap	Our client is only looking for the Mail Exchange (MX) server for <code>www.packet-level.net</code> . The response [P2] includes the name servers and their IP addresses.
00157	dns-serverfailure.pcap	DNS server failures [P4] [P17] [P23-P25] don't indicate that the name was not found – they indicate that the server couldn't get a positive or negative response. Perhaps the upstream DNS server didn't respond in a timely manner (or at all). Consider building a display filter for <code>dns.flags.rcode==2</code> to view these DNS server failure responses.
00042	dns-slow.pcap	Compare how quickly the first and second DNS response packets come to the delay before the first DNS response is sent. Is the DNS response time better later in the trace [P98]?

00084	dns-ttl-issue.pcap	This client can't get to a website because DNS isn't resolving properly. It's not the normal Name Error or Server Error response, however. Something strange is happening. Look at the source of the ICMP packets, the type of ICMP packets and the original DNS query packet contents spit back inside those ICMP packets. Which device would you examine first?
00117	dnswalk.pcap	This DNS 'walking' operation begins by looking up the SOA (Start of Zone Authority) for a domain and then the NS (name servers) for the same domain. After doing some research on the name servers, the client begins a TCP-based Zone Transfer (AXFR) [P7]. Note the number and format of replies [P14].
00083	download-bad.pcap	The client complains that there is a problem with the Internet connection - they are trying to download the OpenOffice binary, but it is just taking too long. Use the Expert Info and Expert Info Composite to identify the problems in this trace file. What are the three primary causes for the poor performance?
00139	download-bad-to-good.pcap	This user tries to download the OpenOffice binary from one site, but the performance is pitiful. They cut off the download process and connect to a mirror site. The download process is much faster from the second site. Is the problem at the server? The client? The link? The file? The IO Graph tells the story – high spiky mountains are preferable to long rolling hills. Separate out the two TCP data streams and pull the Expert Info for each.
00082	download-good.pcap	The users are relatively happy with the download time required to obtain the OpenOffice binary depicted in this trace file. How long did the file transfer take? What is the average bytes/second rate? .

00102	espn-moved.pcap	You've got to kiss a lot of frogs to find a prince! And you've got to connect to a lot of ad servers and media streaming servers to get your sports fix! When viewing this trace, look at the Statistics > HTTP > Packet Counter to see how many redirections and 404 Not Found errors the client experiences when connecting to www.espn.com.
00144	evilprogram.pcap	A truly classic trace file of a system infected with the Stopguard browser hijack spyware/malware/scumware program. It's imperative that the client not reboot, but luckily we got the trace. Create a DNS filter to see the client look up Virtumonde's website. That's when the troubles begin. Check out www.spywarewarrior.com - a great reference for spyware/adware/malware/scumware, etc.
00133	fin-unhappy-client.pcap	This client has a personal problem. The HTTP server sends a FIN ACK and the client ACKs back. After that the client uses the TCP backoff algorithm at it sends a series of FIN ACKs back to the server in desperate hopes the server will send a final ACK to terminate the communications. Sorry – the server has moved on to other things.
00025	frag-needed.pcap	Something is definitely wrong here. Look inside the Destination Unreachable/Fragmentation Needed packets to locate the IP header Total Length field. Was the triggering packet too long? What is the MTU of the next hop advertised?
00142	ftp-download-good2.pcap	An FTP user wants to download a file, but not until it knows the size of the file [P12]. There are a few lost packets along the way, but nothing too significant. Consider setting Time Display Format > Seconds Since Beginning of Capture and then setting a Time Reference on the first data transfer packet [P16]. Scroll to the end of the trace to find the download time.

00101	ftp-failedupload.pcap	This is an interesting trace of an FTP file upload process that seemed to take forever and then generated an error. What happened here? Can you figure out which direction packet loss must have occurred on?
00039	ftp-filesizeproblem.pcap	In this case, an FTP download is unsuccessful because of a limit imposed by the FTP server. The response message is quite clear [P38].
00080	ftp-pasv_scrubbed.pcap	Someone went a little overboard when sanitizing a trace file. Can you learn anything worthwhile about this communication just by looking at the limited information available?
00190	ftp-pasv-fail.pcap	Although this FTP server seems to accept the PASV command [P30], when the user attempts to connect to the offered port, the server doesn't answer. Even though the FTP server shuts down the FTP data connection, it makes a snide remark to the client. [P39]
00018	ftp-putfile.pcap	The client uses the STOR command during an active FTP connection. Note the Wireshark decode of the PORT command packets [P16] [P37] [P55] [P71]. What data is being transferred across the secondary connections established by the server?
00028	ftp-transfer.pcap	This FTP transfer process consists of five TCP connections. Reviewing TCP Conversations information is the best way to see which connection supports the majority of the traffic. There are some problems as indicated in the Expert Info window as well.
00204	ftp-up-disconnect.pcap	Trying to upload a file (you can see the PPT file name in the trace), but the connection is lost. Note the retransmissions leading to a whole slew of FIN ACK packets.

00187	ftp-wrongpwds.pcap	This simple trace file depicts a user who provides the wrong FTP username and password several times. The username IEUser@ has been submitted because the user is performing FTP from inside Internet Explorer.
00116	gettime.pcap	A client connects to pool.ntp.org on port 123 to time sync its system. You can see the static structure used in both the NTP client mode and the NTP server mode. Notice that there are 12 IP addresses provided in the DNS response [P2].
00132	http-client-refuses.pcap	This client has more than one connection to a streaming video server, but nothing happens when they begin the stream viewing process. A quick review of the trace file indicates that the client rudely sends TCP RST packets [P28] [P29] to shut down the connections. The fault is at the client. Most likely a popup blocker process is getting in the way.
00147	http-espn.pcap	My favorite 'ugly' website (other than www.ebay.com) is www.espn.com. Consider selecting Statistics > HTTP > HTTP Packet Counter to view the number of redirections (Code 301 and 302) and client errors (Code 404). Now why is the client blamed for these 404 errors? It's ESPN's fault that we looked somewhere we weren't supposed to! Harrumph!
00175	http-fault-post.pcap	Although this company has a nice feedback form online, when you fill out the form and click submit they rudely send an HTTP error code 403 [P10] [P13]. Someone needs to let the webmaster know the form is broken!
00174	http-general.pcap	The HTTP client receives a 301 Moved Permanently redirection from the HTTP server. Follow TCP Streams makes it easier to follow what is happening in this trace file.

00050	http-msn.pcap	Browsing to www.msn.com is more complicated than it seems. When reviewing this trace, consider running Statistics > HTTP > Requests (do not apply a filter) to see how many systems the client actually connects to.
00115	http-partial-content.pcap	HTTP allows us to retrieve partial content of a file loaded on a web server. For example, perhaps we just want to get the artist information for an MP3 file, but we don't want to download the entire file yet. This trace depicts a user sending a partial content query as noted by the 'Range: bytes=x-y' header field. Cool!
00131	http-pushy-flash-movie.pcap	An interesting trace file of a video streaming application. If you watch the stream traffic [P136-P1033], you notice the MSS is at 1380 bytes, not the expected 1460 bytes. A closer look at the TCP handshake process reveals the server limiting the MSS. Why?
00078	http-slow-filexfer.pcap	Another problem file download caused by packet loss. Use the Expert Info Composite window to determine how many packets were lost in this file transfer.
00155	icmp-destination-unreachable.pcap	The client is trying to ping 10.4.88.88, but it appears that the local router can't locate the device on the next network. The local router sends an ICMP Destination Unreachable/Host Unreachable message indicating that it tried to ARP for the target, but didn't receive an answer. You MUST learn ICMP in depth to secure, optimize and troubleshoot your network effectively!
00077	icmp-lotsostuff.pcap	This trace contains some interesting ICMP traffic. If you look closely you can spot two systems that are behaving strangely on the network. What is triggering the ICMP Destination Unreachable/Protocol Unreachable responses?

00017	icmp-ping-2signatures.pcap	When you see ICMP echo request packets on the wire, check the payload to see if you can identify the application sending the data. Try using the following display filter: <i>data contains "from"</i> .
00016	icmp-ping-2signaturesagain.pcap	These two pings come from a Windows system (notice the partial alphabet in the padding) and a Network General Sniffer system (try using a <i>data contains "cinco"</i> filter).
00152	icmp-routersolicitation.pcap	Wow – my DHCP server [10.1.0.1] wasn't up when I needed to renew my IP address. When it did restart it offered me a different address and a bogus address for my default gateway. That triggered my system to perform ICMP Router Solicitation. This is typically not a good ICMP packet to see on the network. [Yes, you can see my name in the trace file. Why hide it – I'm the one that killed the DHCP server in the lab and I paid the price.]
00006	icmpstrange.pcap	The ICMP traffic in this trace is suspect. Consider filtering on the ICMP traffic only and examining the payload. Also consider looking at the IP Conversations information to see if traffic is flowing bi-directionally between any devices.
00151	icmp-traceroute-normal.pcap	This is a classic ICMP-based traceroute operation shows the dependency on the ICMP Time to Live Exceeded/Time to Live Exceeded in Transit response that is used to locate routers along a path. One of the routers doesn't generate these responses though.
00037	icmp-tracert-slow.pcap	This traceroute client is excruciatingly slow between some of the TTL increment sets. What triggers the ICMP Destination Unreachable/Port Unreachable message sent to the client [P64]? How many hops away is the target?

00138	io-ftpupload.pcap	No one will live long enough to upload files to this FTP server. Is the server at fault? The client? The network? Examine the Warnings and Notes in the Expert Info Composite to get the whole picture.
00036	ip-fragments.pcap	The client is sending fragment ICMP Echo packets to the target. Try setting up Wireshark with and without IP fragment reassembly to note the difference. Edit > Preferences > Protocols > IP.
00098	keepalive.pcap	Don't get sidetracked by the TCP Checksum Incorrect messages - this NIC does TCP Checksum Offloading so Wireshark doesn't see the correct checksum value before the packets are sent out. What is interesting in this trace is the keepalive process that is triggered by an application. It appears to read from a file at offset 3584 for a maximum of 512 bytes (check out that data in the responses) [P1] [P5] [P9] [P13] [P17]. Not a pretty site. What was that programmer thinking?
00126	live-chat.pcap	This live chat to a support line creates a nice secure connection. Oh, wait... make that 122 nice secure connections. Whazzup with that? Isn't that overkill? Look at Statistics > Packet Length to see how much of the traffic uses little itty bitty stinkin' packets!
00076	lost-route.pcap	Although the client can get to the Google toolbar page, it can't get to the Verio home page. It doesn't appear that there is a path to the target. Why aren't there any TCP RST packets or any ICMP Destination Unreachable packets.
00075	macof.pcap	Dug Song created Macof to flood network switch MAC address tables and cause them to go into 'hub mode.' This tool still wreaks havoc on the network. Can you identify the signature in this flood traffic?

00022	madclient.pcap	This client complained that the network wasn't working. A short packet capture reveals the truth. This application, although not recognized by Wireshark, identifies itself in the payload. In addition, it contains an error message that explains why things didn't work as the user hoped.
00072	nickname-packet-level.pcap	This query uses RWHOIS (Referral WHOIS) on port 4321. RWHOIS extends the capabilities of WHOIS in a hierarchical structure. This trace shows a successful RWHOIS query handled by root.whois.net.
00034	nosmtp.pcap	A user (10.1.0.1) complains that they cannot send email to the SMTP server (10.2.23.11). Examine the trace and determine if the fault lies with the client, the server or the network. .
00015	nst-axfr-refused.pcap	The AXFR command sticks out like a sore thumb. Someone is trying to do a DNS zone transfer and their request is being refused. Notice that this trace shows the two types of DNS queries - UPD-based and TCP-based.
00113	one-way-drops.pcap	When analyzing a one-half of a redundant link between switches, we learned that the switches were not load-balancing. They had created two one-way paths. Data was being lost download - we could tell by the retransmissions (we had to assume the server received duplicate ACKs that triggered the retransmissions).
00184	pop3problem.pcap	The POP email application gives no indication as to why it is taking so long to pick up mail. In this trace we can clearly see the problem lies with the email server that is sending back an '-ERR-' response [P5].

00182	pop-spamclog.pcap	Users can't get their email. It appears that their email programs just hang when they try to send/receive. In truth, we can see that there are spam messages with large attachments (.pif) that take a long time to download. Users need to be patient and the company needs to consider filtering this spam before it gets to the client systems. Follow TCP Stream on any POP packet to view the spam messages.
00068	proxy-problem.pcap	The client can't get off the network because of errors getting through the proxy server. You can read the proxy response in clear text - Follow TCP Stream. Also note the slow handshake response time. Not a good day for this user.
00136	sick-client.pcap	This client hits an IRC channel as user llll [P14] and later begins to do a scan on the network for anyone with port 139 open. Feels like a bot looking for other systems to infect. Look at the rapid rate of the scan – that's why the response are bunched up at the end of the trace. [Note: Turn off the colorization on this trace or your head may explode. We ran this trace through an IP address cleaner program but it didn't recalculate the checksums.] Symantec: Wargbot; MS: Graweg; Trend: Worm_IRCbot; McAfee: Mocbot; F-Secure: IRCBot.
00128	skype-call-connect-disconnect.pcap	While watching a Skype call being established and terminated, we noted the numerous UDP connections that were required and what appeared to be a command channel using TCP during the beginning and end of the call. Strange that it didn't terminate the TCP connections.
00011	smtp-fault.pcap	This trace shows what happens when the DNS lookup for an SMTP server works fine, but the actual connection attempt does not.

00181	smtp-normal.pcap	In a normal SMTP connection the user doesn't send a user name or password. Follow TCP Stream to clearly view the entire message. During the initial communication process, the server indicates that it supports pipelining and imposes no limitation on email size [P8].
00096	snmp.pcap	This trace includes a simple SNMP query-response pair of communications. Are they all looking for the same information? Perform some Internet research to locate .1.3.6.1.2.1.25.3.2.1.5.1 (or search for SNMP MIB-2.25.3.2.1.5.1). You'll notice the response code of INTEGER 5 indicates that the stated of the device is 'down.'
00067	ssl3session.pcap	During the establishment of this SSL connection (HTTPS) there appears to be communication problems causing retransmissions.
00198	sym-404.pcap	Look at all the 404 File Not Found responses during a Symantec LiveUpdate process. This type of update problem should be of concern to you.
00001	sym-failed.pcap	Filter on http to see what's happening more clearly. This will remove the handshake packets and the ACK responses. Why did this Symantec LiveUpdate process fail?
00095	sym-update2.pcap	This Symantec update process doesn't seem to work very well. Consider building a filter for <i>http.response.code == 404</i> and note the number of "File Not Found" responses. Now compare these two display filters and explain the difference in your results: (1) <i>!http.response.code == 404</i> and (2) <i>http.response.code !=404</i> . Good lesson!
00010	tcp-echo.pcap	You probably don't want to see this on the network - traffic to TCP Echo port (7). Consider the implications if both the source and destination ports were 7. Ugh.

00009	tcp-fin.pcap	This trace shows the normal TCP FIN process. There are actually two common variations of this process - FIN, FIN ACK, ACK or the four-packet process shown in this trace. Note that the TCP connection is still active and waiting for a timeout now.
00109	tcp-handshake-problem.pcap	An amazingly simple communication that went all wrong because of the TCP handshake. Each packet of the handshake looks good [P3-P5]. When the client begins sending data to the RWHOIS server, however, it receives SYN ACK packets in response. All this trouble just because the third packet of the handshake never arrived. The duplicate ACKs are asking for Sequence number 1 again - unfortunately two of the client's packets have this same sequence number. This will never get resolved.
00064	tcp-keepalive.pcap	An application that wants to keep the TCP connection open during a long idle time can trigger the TCP Keepalive function. This trace shows just such a process for traffic maintaining a connection between ports 1863 and 2042. Is there any data contained in these TCP Keepalive packets? How do you think Wireshark determines that these are TCP keepalives?
00094	tcp-low-mss.pcap	This HTTP file transfer is never going to achieve the maximum throughput potential because of a Maximum Segment Size (MSS) issue. Does the problem reside with the client or the HTTP server? The Flow Graph indicates that the client is communicating with more than one HTTP server. Is this problem evidence on the second server as well?
00055	tcpproblem.pcap	When you set the Time Display Format to Seconds Since Previous Packet, you can easily see the TCP retry process with five retransmissions of the packet that did not receive an ACK [P2].

00063	tcp-window-good.pcap	This trace depicts a large TCP window used for a file transfer. There are moments when the receiver advertises a Zero Window, however. How much delay do these Zero Window conditions inject into the file transfer. Analyze > Expert Info Composite > Notes > expand the Zero Window entry. As you click each Zero Window packet in the Expert Info Composite window, the trace file highlights the corresponding packet. What file is being transferred?
00093	tcp-wont-shutup.pcap	Which side of the communication is trying to terminate the TCP connection? What is the response from the other side of the connection? What size is the file being transferred? Do you think the entire graphic file was received by the client?
00092	telnet-questionable.pcap	Examine this trace to determine if the telnet client and telnet server agree on the communication settings. DO, DON'T are demands being made from the source to the target. WILL, WON'T are statements from the source indicate what it is willing or not willing to do. How long did it take to get to the Login prompt?
00168	telnet-refuse via rst.pcap	This client's telnet connection request is refused by the target in the traditional TCP RST/ACK method. An excessive number of RST/ACKs on the cable may be indication of a TCP port scan. In this case it is just one wayward telnet client.
00008	timesync.pcap	This trace shows a system performing a DNS query for tock.usno.navy.mil and then running a Network Time Protocol request on port 123.
00166	udp-general.pcap	DHCP, DNS, NetBIOS Name Service and Microsoft Messenger make up the UDP-based communications in this trace. You wouldn't wish this NetBIOS and Messenger traffic on your worst enemy – what a filthy network.

00089	using-time.pcap	Scroll through the entire trace file before making any assumptions on the cause of poor performance for this client. This trace illustrates the importance of paying attention to the time values in trace files. The ugliest communications are not necessarily the cause of poor performance. Identify the most time-consuming fault in this trace.
00088	video-streaming.pcap	Examine the "If-Modified-Since" lines in the HTTP GET requests from this client. How does that parameter affect the file download process? Are there any 404 response codes in this trace? Is the video stream bursty or steady?
00164	web-browse-ok.pcap	This user must have recently browsed to www.packet-level.com based on the high number of HTTP 304 Not Modified responses (39 in all) sent from the server. If the user complained of slow performance in the previous connection and we are troubleshooting the problem, we need to ensure they do not receive any HTTP 304 Not Modified responses – they should download all files and not pull them from cache.
00127	webcast-keepalive.pcap	This videocast ended 130 seconds into the trace file. Build an IO graph to watch the keepalive process as the video concluded and the client kept the connection alive [P2562] [P2572] by sending a GET request for <code>caption.aspx?filetype=1</code> every 6 seconds. This occurs all during the video download as well.
00107	whois-podbooks.pcap	This WHOIS query begins with a DNS lookup for ANY DNS entries related to <code>podbooks.com</code> and progresses to learn the contact email address. Next the researcher contacts the ARIN WHOIS database using port 43 (NICNAME), but the server says there isn't a match.

00106	window-frozen.pcap	A window frozen condition can kill file transfer speed. Set the Time Display Format to Seconds Since Beginning of Capture and right click on the first ZeroWindow packet [P30] to Set Time Reference. How much time did this condition waste?
00196	window-scaling-bad.pcap	It would have been nice to set up TCP window scaling for this HTTP connection. The client advertises a window scale of 2 (multiply the 65,535 window by 4) [P1], but the server doesn't support TCP window scaling. Sigh.
00195	window-scaling-good.pcap	Now this is the life! The client advertises a TCP window scale of 2 (multiply the window value by 4) and the server supports window scaling as well (although with a window scale of 0 which does it no good on the receive side of things). Check out Wireshark's ability to calculate the correct window size [P3] for the client. This is a feature you can turn on/off in the Preferences > TCP area.
00194	window-scaling-wishful.pcap	The client and the server can do window scaling, but when we look at the scaled value for the client [P3], it's only set at 5840. Shouldn't it be higher (5,840 times 4)? Highlight the TCP window field and you'll find out why we are still at 5,840. Bummer. It's a weird HTTP communication anyway.