School of Computing Sciences Pak-Austria Fachhochschule:
Institute of Applied Sciences and Technology, Haripur, Pakistan

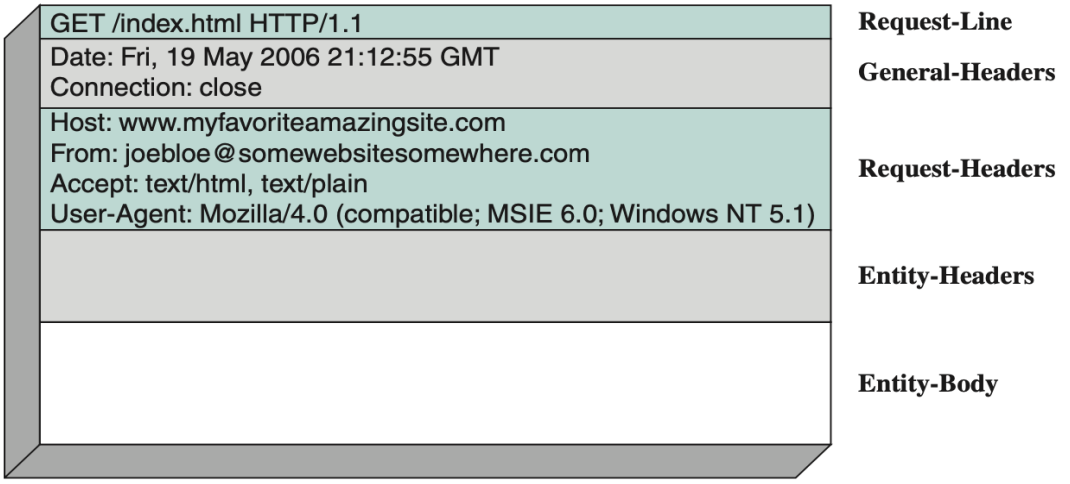
Lab 03
Analyze Hypertext Transfer Protocol (HTTP) Traffic using Wireshark

Course: COMP-352L (Computer Networks Lab)

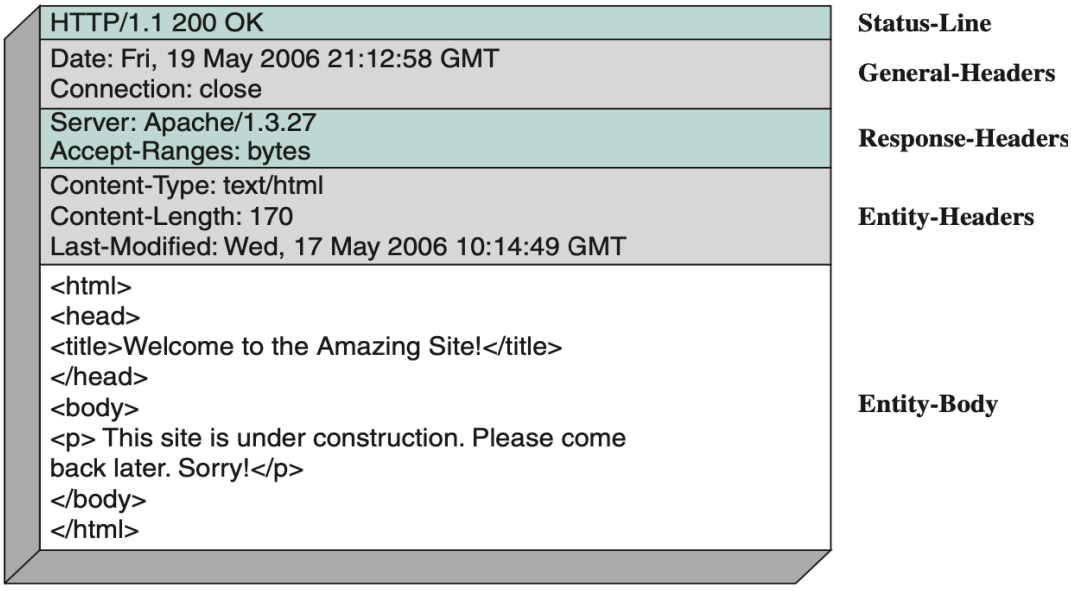
Lab Demonstrator: Jarullah Khan

HTTP(Hypertext Transfer Protocol)

- HTTP is an application layer protocol used by the world wide web to transport resources between client and server computers on the internet.
- HTTP is implemented in two programs: a **client** program (web browsers) and a **server** program.
- The client program and server program, executing on different end systems, talk to each other by exchanging **HTTP messages**.
- HTTP defines the structure of these messages and how the client and server exchange the messages.



(a) HTTP request



(b) HTTP response

- **URLs** describe the specific location of a resource on a particular server. They tell you exactly how to fetch a resource from a precise, fixed location.

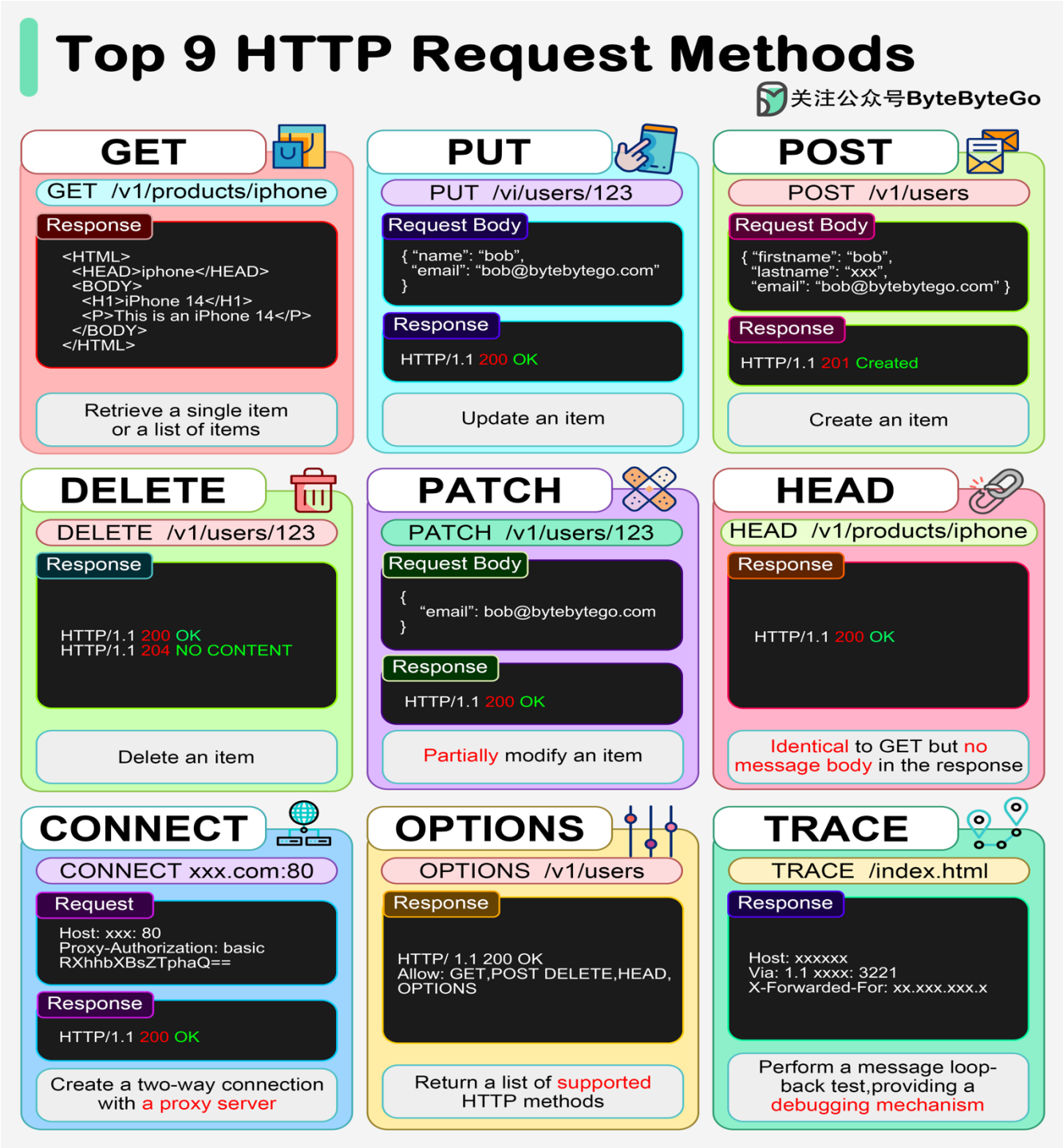
URL	Description
http://www.oreilly.com/index.html	The home URL for O'Reilly & Associates, Inc.
http://www.yahoo.com/images/logo.gif	The URL for the Yahoo! web site's logo
http://www.joes-hardware.com/inventory-check.cgi?item=12731	The URL for a program that checks if inventory item #12731 is in stock
ftp://joe:tools4u@ftp.joes-hardware.com/locking-pliers.gif	The URL for the *locking-pliers.gif* image file, using password-protected FTP as the access protocol

Resources

- A web resource is the source of web content.
- The simplest kind of web resource is a static file on the web server's filesystem. (These files can contain anything: text, images, audio, video etc.)
- Resources can also be software programs that generate content on demand.

Methods

- HTTP supports several different request commands, called ***HTTP methods***.
- Every HTTP request message has a method.
- The method tells the server what action to perform (fetch a web page, run a gateway program, delete a file etc.)



Status Codes

- Every HTTP response message comes back with a status code.
- The status code is a three-digit numeric code that tells the client if the request succeeded, or if other actions are required.
- The HTTP Status Code Registry is maintained at www.iana.org/assignments/http-status-codes.

HTTP Status Code

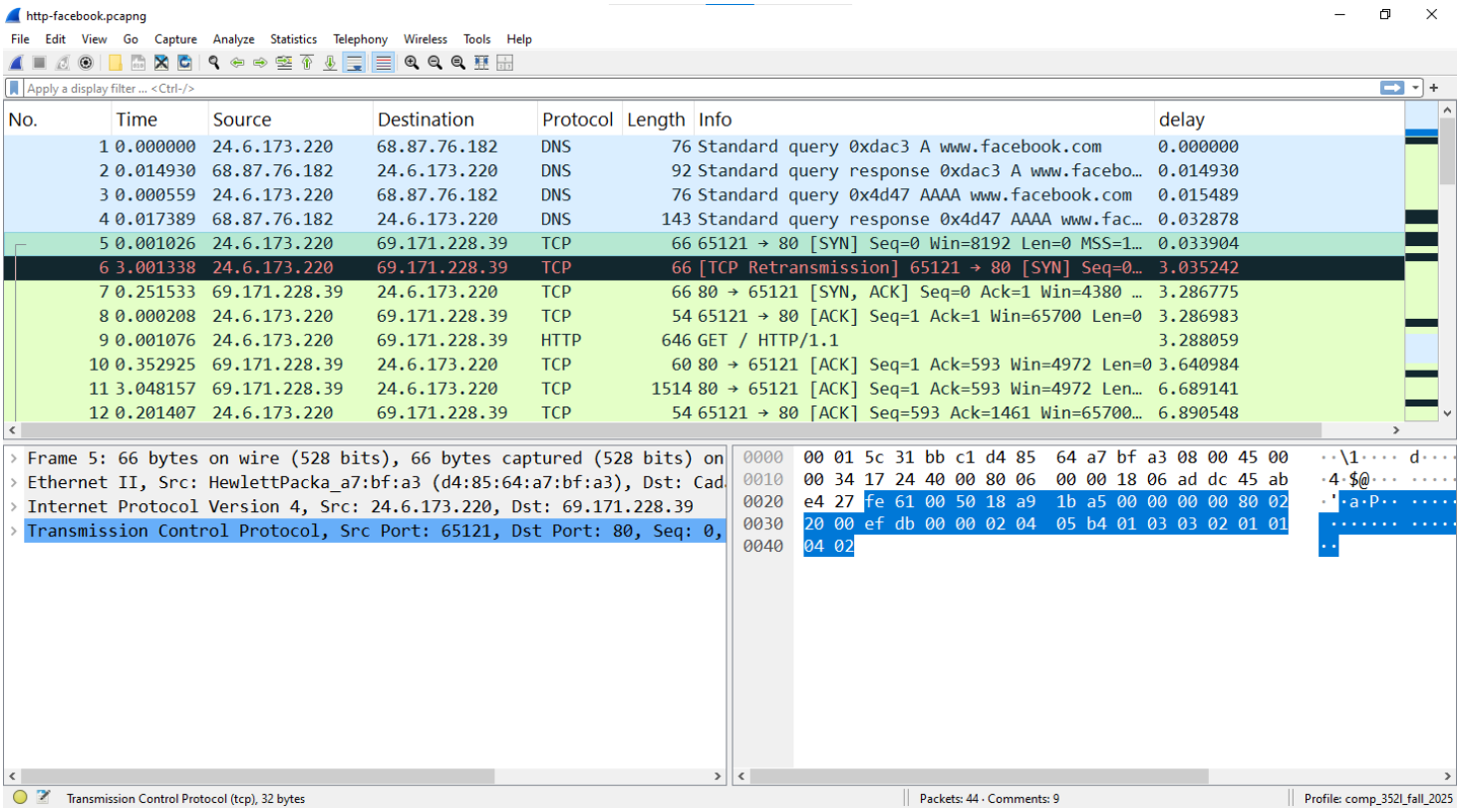
1XX Information	
100	Continue
101	Switching Protocols
102	Processing
103	Early Hints
2XX Success	
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
205	Reset Content
206	Partial Content
207	Multi-Status (WebDAV)
208	Already Reported (WebDAV)
226	IM Used (HTTP Delta Encoding)
3XX Redirection	
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
306	Unused
307	Temporary Redirect
308	Permanent Redirect
4XX Client Error	
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable

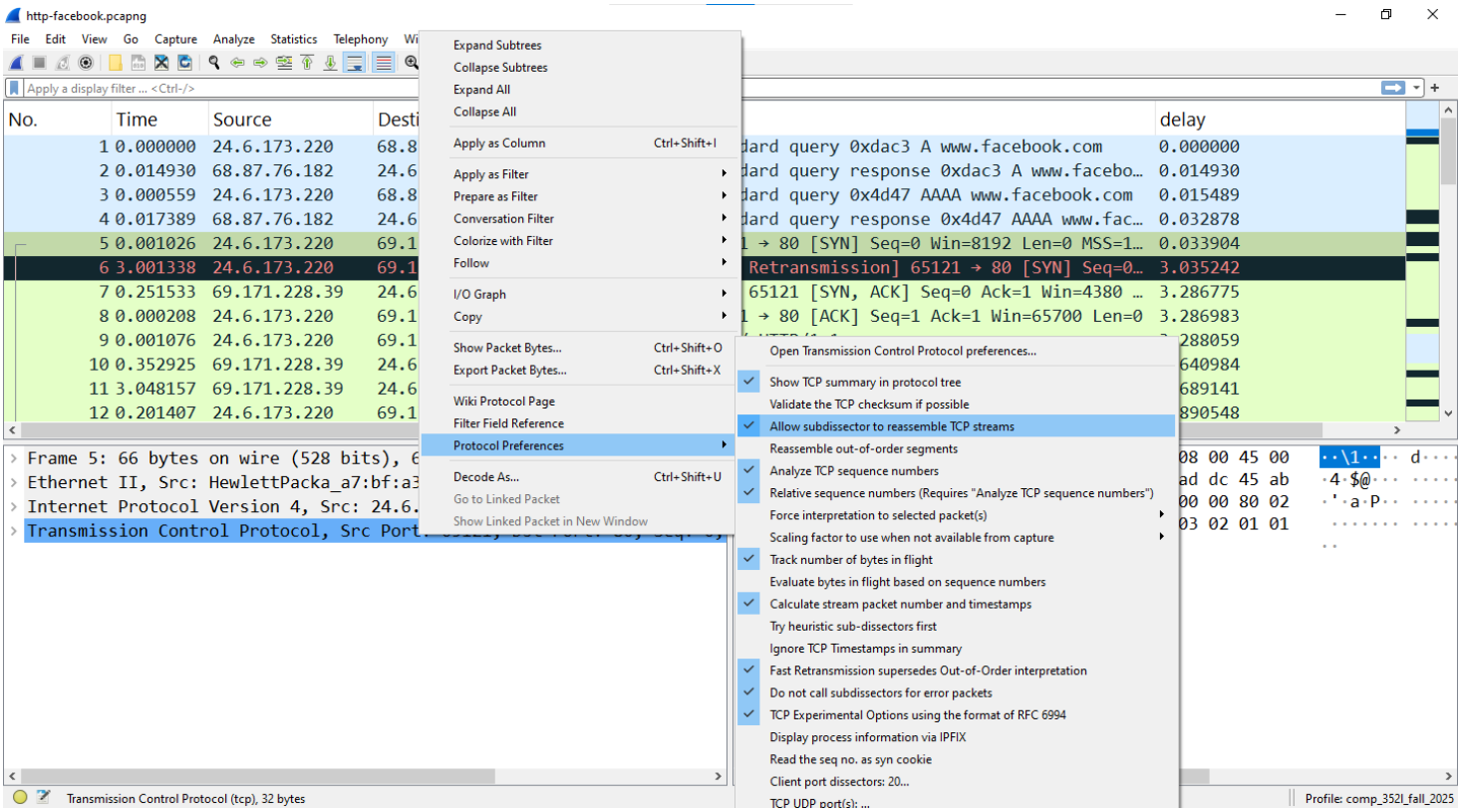
4XX Client (Continue)	
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Payload Too Large
414	URI Too Large
415	Unsupported Media Type
416	Range Not Satisfiable
417	Exception Failed
418	I'm a teapot
421	Misdirected Request
422	Unprocessable Entity (WebDAV)
423	Locked (WebDAV)
424	Failed Dependency (WebDAV)
425	Too Early
426	Upgrade Required
428	Precondition Required
429	Too Many Requests
431	Request Header Fields Too Large
451	Unavailable for Legal Reasons
499	Client Closed Request
5XX Server Error Responses	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
507	Insufficient Storage (WebDAV)
508	Loop Detected (WebDAV)
510	Not Extended
511	Network Authentication Required

✏️Disable Stream Reassembly to See HTTP More Clearly

For the clearest view of HTTP traffic in the Packet List pane, disable the TCP preference "Allow subdissector to reassemble TCP streams". This enables you to see all of the HTTP GET requests and the HTTP response codes in the Packet List pane.

One of the interesting response codes is the infamous 404 Not Found. Note that this is listed as a client error under the assumption that the client made a mistake in selecting the URL to visit. In truth, however, most 404 Not Found errors are sent in response to clients following broken links on websites.



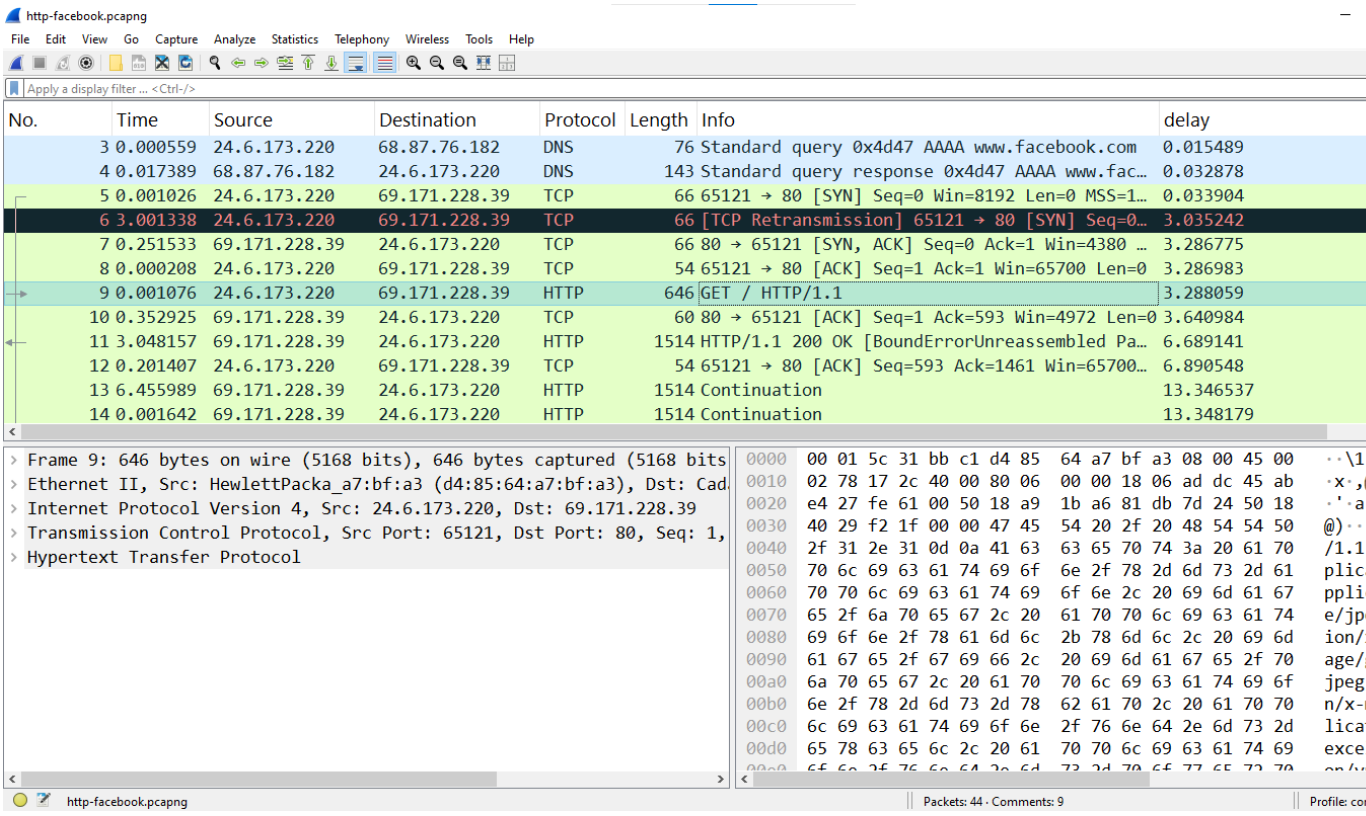


Analyze Normal HTTP Communications

Normal HTTP communications use a request/response communication style. Clients make requests of HTTP servers and servers respond with Status Codes and the requested Resource (if any).

The trace file we are working with is http-facebook.pcapng. You will notice some packet loss and very poor response times.

The client makes a three-way TCP handshake from port 65121 to port 80 (listed as http in the Info column because transport name resolution is enabled). By default, Wireshark is configured to dissect HTTP on 9 ports: 80, 3128, 3132, 5985, 8080, 8088, 11371, 1900 and 2869. HTTP communications can use other ports as well.

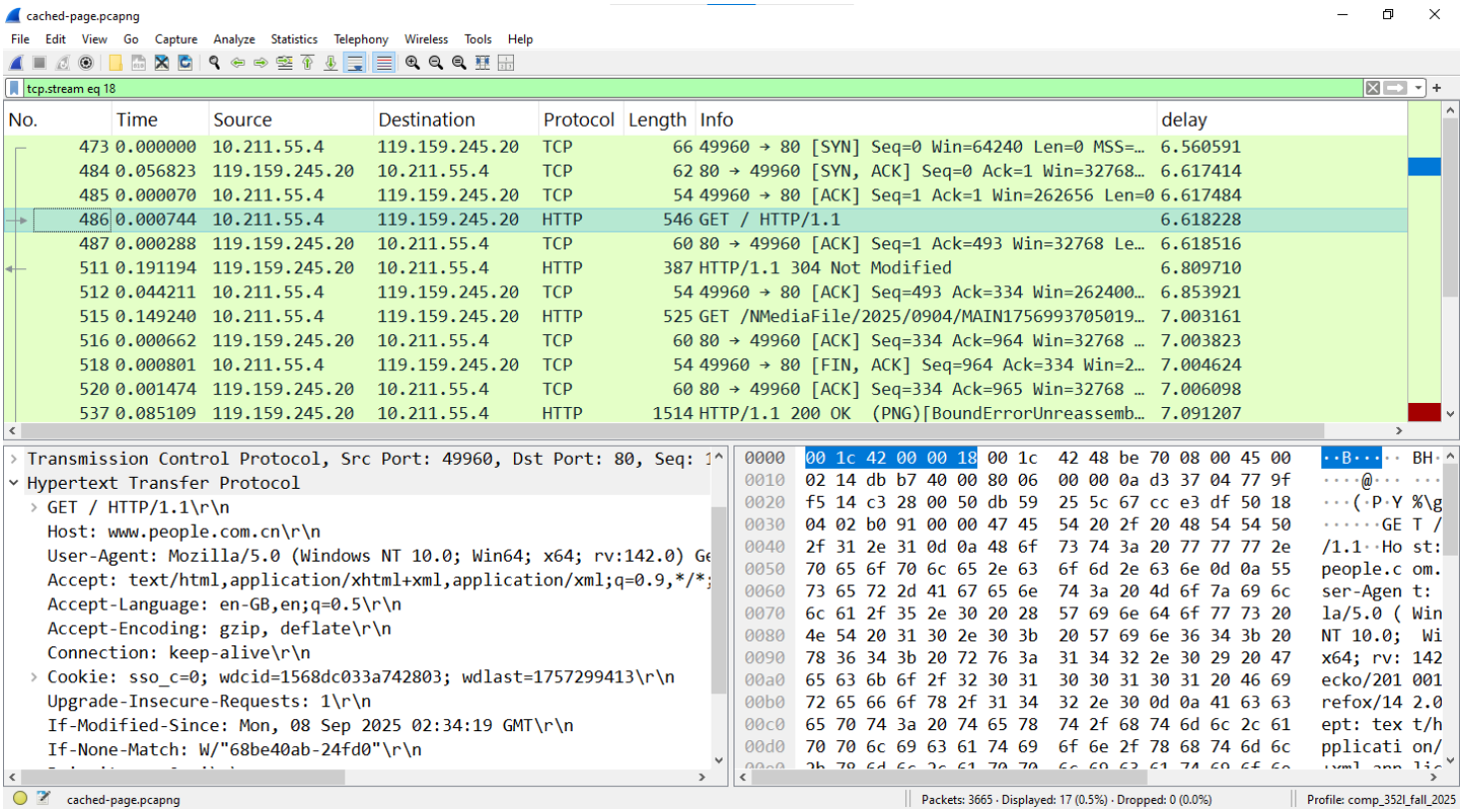


After the TCP connection is established successfully, the client makes an HTTP GET request for "/". The server responds with the Status Code 200 OK and begins sending the client the contents of the www.facebook.com main page. It takes five HTTP connections to view the main page at www.facebook.com.

All the HTTP Status Codes seen in http-facebook.pcapng are good—all 200 OK.

✏️Watch Out For Cache-Loaded Web Pages

If an HTTP client has visited a page recently and that page is cached locally, the client may send the ***IfModified-Since*** parameter and provide a date and time of the previous page download. If the server responds with a 304 Not Modified—the server will not resend the page that is already cached. This is an important part of HTTP to understand when analyzing HTTP performance. If a user complains of poor performance when accessing a website the first time only, they may be loading pages from cache—you may not be seeing a true full page download.



Analyze HTTP Problems

HTTP communication problems can occur because of problems in site name resolution, issues with the TCP connection process, HTTP requests for non-existent pages or items, packet loss as well as congestion at the HTTP server or client.

Everyone at some time has typed in the wrong website address. If the site name cannot be resolved, you cannot access the site. This would generate a DNS Name Error. It is important to pay attention to DNS traffic when analyzing web browsing problems.

In addition, HTTP connection problems may occur when the HTTP daemon is not running on the web server. When the HTTP daemon is not running on the server, the server responds with a TCP RST/ACK to the client's SYN. The connection cannot be established. This situation should be watched carefully as the SYN – RST/ACK pattern is seen during a port scan process as shown in Figure 267.

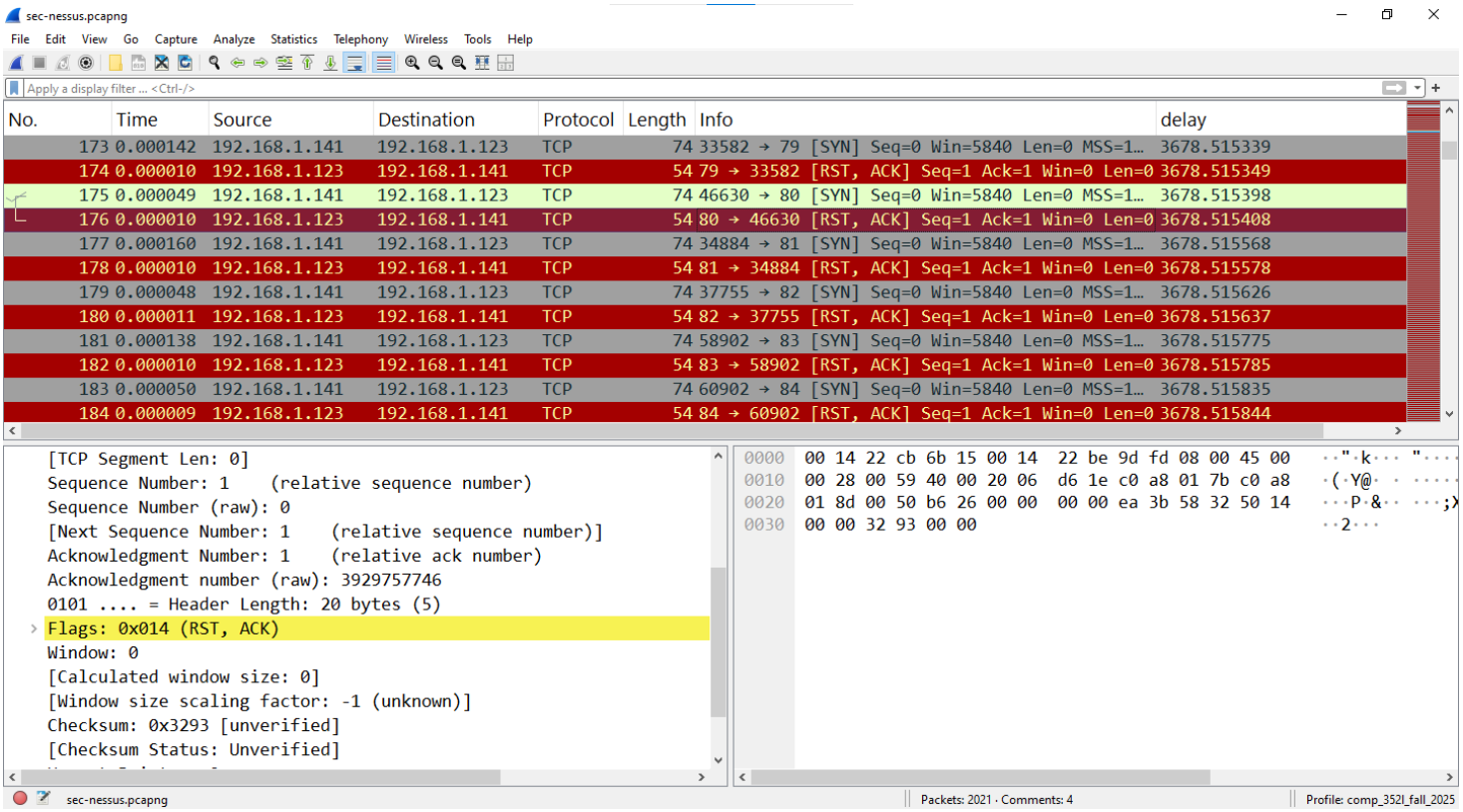


Figure 267. Multiple unsuccessful HTTP connection attempts create a stripe pattern in Wireshark [sec-nessus.pcapng]

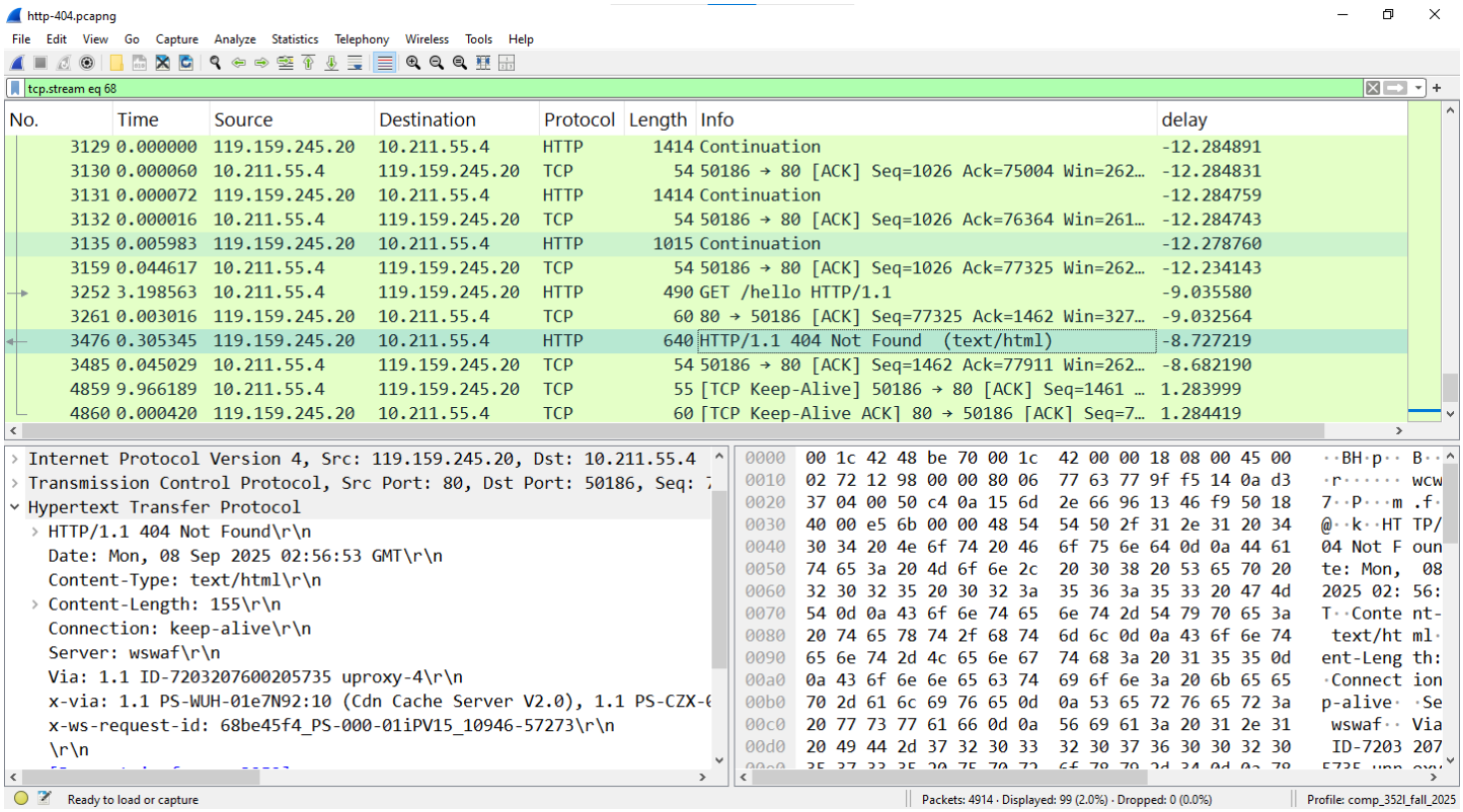
If the HTTP client connects successfully to the HTTP server, but then requests a page that is non-existent, HTTP 404 Not Found errors are generated by the web server.

Some redirection services will replace the standard 404 Not Found message with suggested links or redirect the HTTP client to another site completely. Build a coloring rule for HTTP client and server errors using `http.response.code >= 400`.

Figure 268 shows a problem when opening up a list of laptops for sale at the www.frys.com website. We were able to resolve the IP address for the site and the page exists. By following the TCP stream we can see the server respond with the page heading. The laptop items are not displayed on the page, however—the page is blank.

Looking at the trace file we can see the www.frys.com web server reported an internal server error. This is not a problem on the client's system or the network. This problem is likely caused by a database problem within Fry's web services infrastructure.

Figure 268. The frys.com server responds with



an internal server error [http-500error.pcapng]

Don’t Troubleshoot Large Delays before FIN or Reset Packets

Open http-fault-post.pcapng and set the Time column to Seconds Since Previous Displayed Packet. Notice a large delay before packet 29. Be careful here. Packet 29 has the FIN bit set. This indicates the client is finished sending information to the server. These packets (and packets with the Reset bit set) can be triggered long after the user has finished getting the required data. The user does not notice this delay so don’t spend your time troubleshooting delays before packets marked with the FIN (or Reset) bit.

In Figure 269 we are trying to fill out a form online (http-fault-post.pcapng). Upon clicking the Submit button, however, the client system appears to hang. In this case we can look at the HTTP traffic and observe a 403 Forbidden status code from the server. Following the TCP stream reveals clear text and HTML tags with more

information about the situation (we have removed the HTML tags):

> "The page cannot be displayed - you have attempted to execute a CGI, ISAPI, or other executable program from a directory that does not allow programs to be executed."

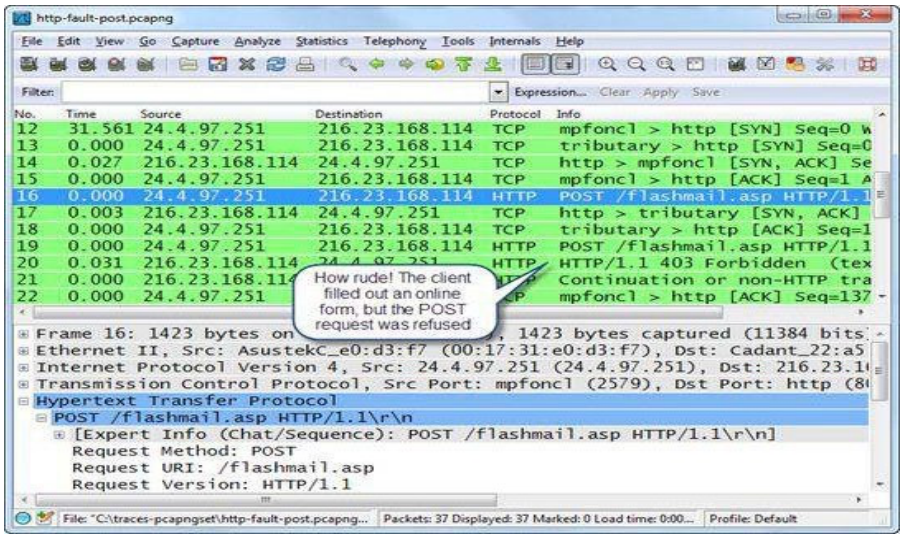
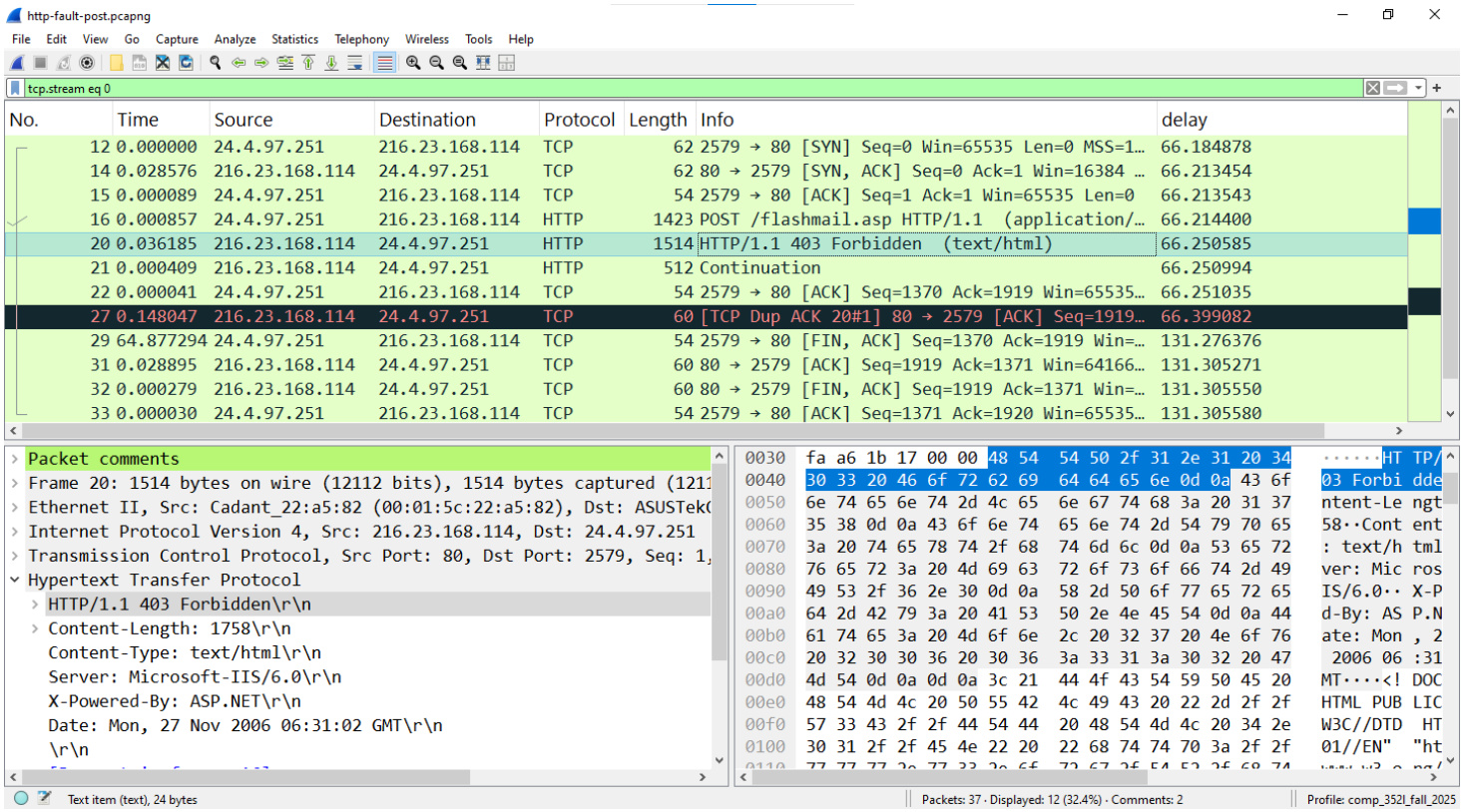


Figure 269. The client's POST is unsuccessful and indicates a web server problem [http-fault-post.pcapng]

Again, the problem does not appear to be a client issue and we do not see TCP transport errors as an issue. The problem is at the server.

When troubleshooting web browsing, look for TCP errors first before focusing on the HTTP traffic.



Dissect HTTP Packet Structures

HTTP packets are variable length. In this section we list some of the key areas in the HTTP packet structure. HTTP requests consist of a Method which defines the purpose of the HTTP request. HTTP responses contain a numerical response code referred to as a Status Code.

Figure 270 shows a GET request for the main Facebook page. The GET request contains the name of the target host, details about the browser issuing this GET request and information about what data types and format the browser will accept.

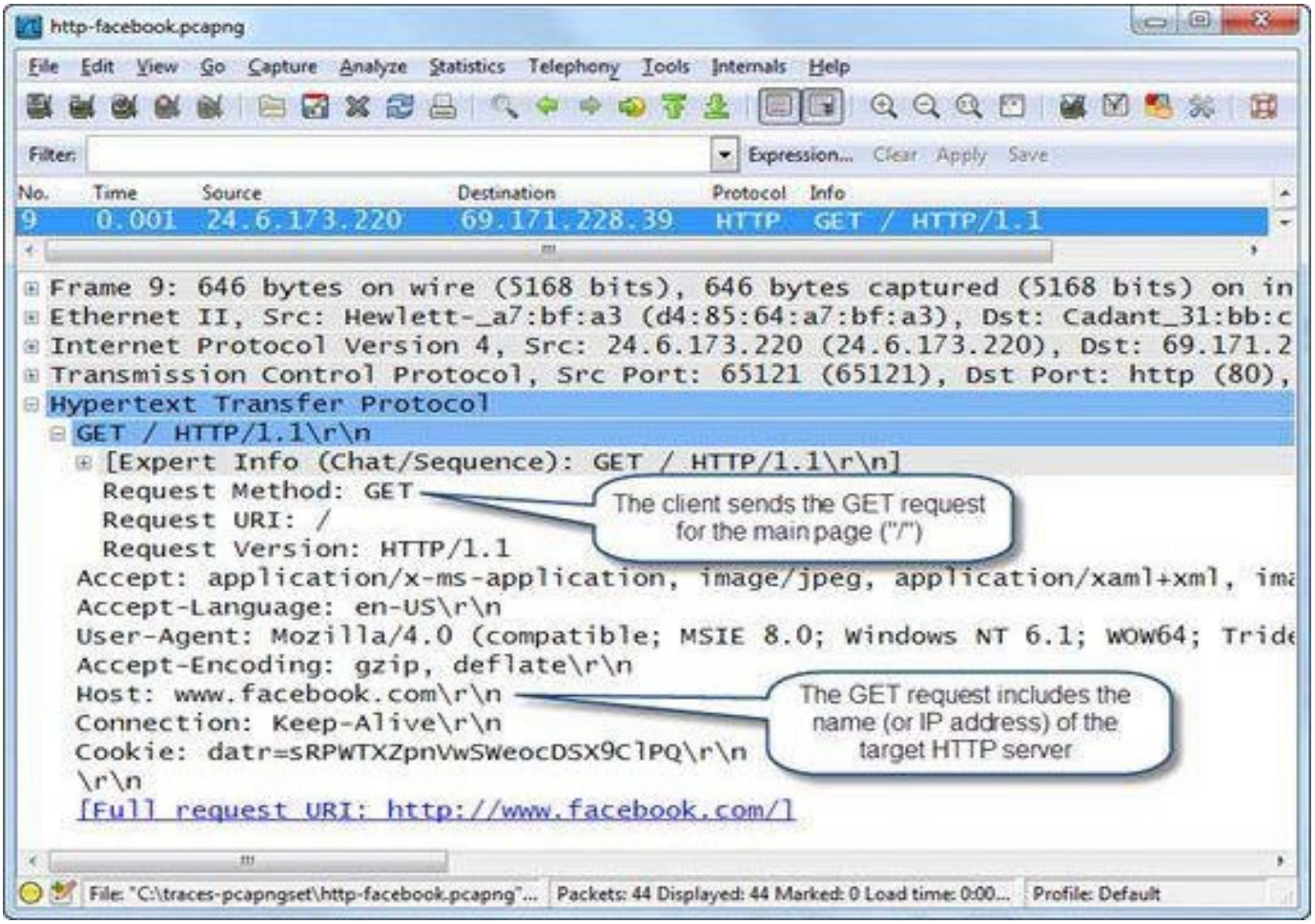


Figure 270. An HTTP GET request packet for the Facebook main page [http-facebook.pcapng]

HTTP Methods

Also referred to as the HTTP commands, the Methods define the purpose of the HTTP packet.

- **GET:** Retrieves information defined by the URI (Uniform Resource Indicator) field
- **HEAD:** Retrieves the meta data related to the desired URI
- **POST:** Sends data to the HTTP server
- **OPTIONS:** Determines the options associated with a resource
- **PUT:** Sends data to the HTTP server
- **DELETE:** Deletes the resource defined by the URI
- **TRACE:** Invokes a remote loopback so the client can see what the server received from the client; this is rarely seen as many companies disable this to protect against a Cross-Site Tracing vulnerability
- **CONNECT:** Connects to a proxy device

Host

The Host field is required in all HTTP/1.1 request messages. The Host field identifies the target internet host and port number of the resource being requested. In our previous example, the host is www.facebook.com. If no port number is specified, the default port for

the service (for example, port 80 for HTTP) is used.

Request Modifiers

HTTP requests and responses use request modifiers to provide details for the request. The following table lists the more commonly used request modifiers.

- **Accept:** Acceptable content types
- **Accept-Charset:** Acceptable character sets
- **Accept-Encoding:** Acceptable encodings
- **Accept-Language:** Acceptable languages
- **Accept-Ranges:** Server can accept range requests
- **Authorization:** Authentication credentials for HTTP authentication
- **Cache-Control:** Caching directives
- **Connection:** Type of connection preferred by user agent
- **Cookie:** HTTP cookie
- **Content-Length:** Length of the request body (bytes)
- **Content-Type:** Mime type of body (used with POST and PUT requests)
- **Date:** Date and time message sent
- **Expect :** Defines server behavior expected by client
- **If-Match:** Perform action if client-supplied information matches
- **IfModified-Since:** Provide date/time of cached data; 304 Not Modified if current
- **If-Range:** Request for range of missing information
- **If-Unmodified-Since:** Only send if unmodified since certain date/time
- **Max-Forwards :** Limit number of forwards through proxies or gateways
- **Proxy-Authorization:** Authorization credentials for proxy connection
- **Range:** Request only part of an entity
- **Referer**[110]**:** Address of previous website linking to current one
- **TE:** Transfer encodings accepted
- **UserAgent:** User agent—typically browser and operating system
- **Via:** Proxies traversed

Filter on HTTP or HTTPS Traffic

The capture filter syntax for HTTP or HTTPS traffic is `tcp port http` or `tcp port https`.

If HTTP or HTTPS are running on nonstandard ports, use the capture filter `tcp port x` where x denotes the port HTTP or HTTPS are using.

✏️Don't Use the `http` Filter to Analyze Web Browsing

This may seem quite counter-intuitive. If you examine the Protocol column in the Packet List pane, you will see which packets would meet this filter – just the packets that contain the value http in the Protocol column.

Examine the table on HTTP Filters/TCP Reassembly Settings to learn about the best filter to

use for HTTP analysis.

The display filter for HTTP is simply `http`. Note that this filter may not be the best display filter to use. The table below illustrates the difference between using `http` and `tcp.port==x` where x is the port used for the http session. The effectiveness of the http filter is dependent upon whether TCP reassembly is enabled or disabled in TCP preferences.

Filter: **`http`**
TCP Reassembly: **On**
Partial view of web browsing
session commands and response
codes visible
- no data packets
- no TCP handshake, FIN, RST, or ACK packets
TCP Reassembly: **Off**
Partial view of web browsing session

commands and response codes
visible all data packets visible
- no TCP handshake, FIN, RST, or ACK packets

`tcp.port==80`
TCP Reassembly: **On**
Total view of all packets in web browsing
session TCP Reassembly: **Off**
Total view of all packets in web browsing session

The filter for HTTP or HTTPS must be based on the port in use, such as port 443 for HTTPS (`tcp.port==443`). Alternately, you could use ssl as the display filter. HTTPS uses Transport Layer Security (TLS) which is based on SSL. Note that you will not see the TCP handshake process or ACK packets if you use ssl as the display filter. It is best to use a port number-based display filter to see all packets in an SSL conversation.

The following table lists additional HTTP/HTTPS display filters.

`http.request.method=="GET"` **or** **`http.request.method=="POST"`**
HTTP GET or POST requests

`http.response.code > 399`
HTTP 4xx or 5xx (client or server errors)

`http contains "IfModified-Since"`
Determine if a client has cached a page already

`http.host=="www.wireshark.org"`
Target host is www.wireshark.org

`http.user_agent contains "Firefox"`
HTTP client is using Firefox browser

`http.referer contains "wireshark.org"`
HTTP client has reached the current location from a link on wireshark.org

`tcp.port==443`
HTTPS

`ssl`
Secure Socket Layer (secure browsing session) - consider using tcp.port==443

`ssl.record.content_type==22`
TLSv1 handshake

`ssl.handshake.type==1`
TLSv1 Client Hello in handshake

`ssl.handshake.type==16`
TLSv1 Client Key exchange

`ssl.record.content_type==20`
TLSv1 Change Cipher Spec

`http.content_type contains "ocsp"`
Online Certificate Status Protocol (OCSP) is used

Export HTTP Objects

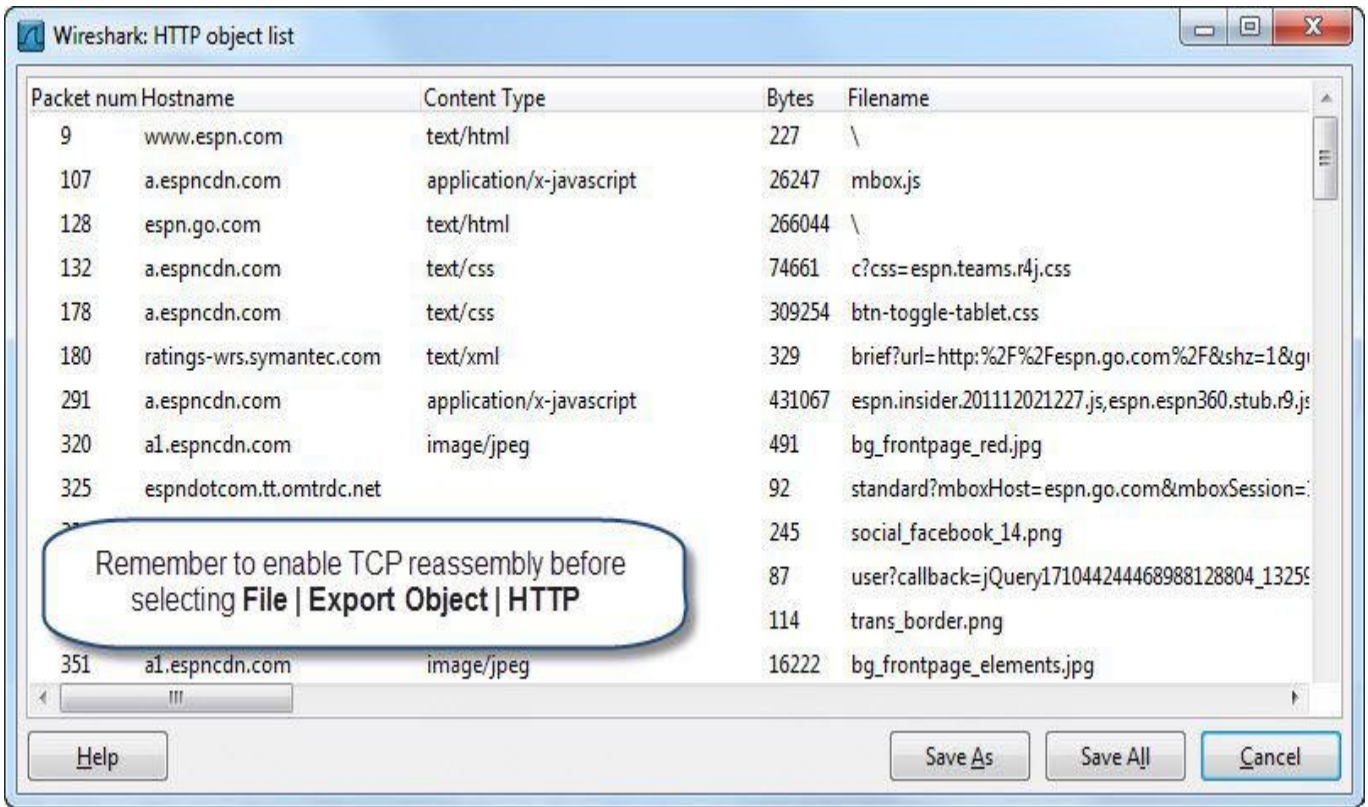


Figure 271. We can export the objects downloaded from a site we were referred to from ESPN's page [http-espn2012.pcapng]

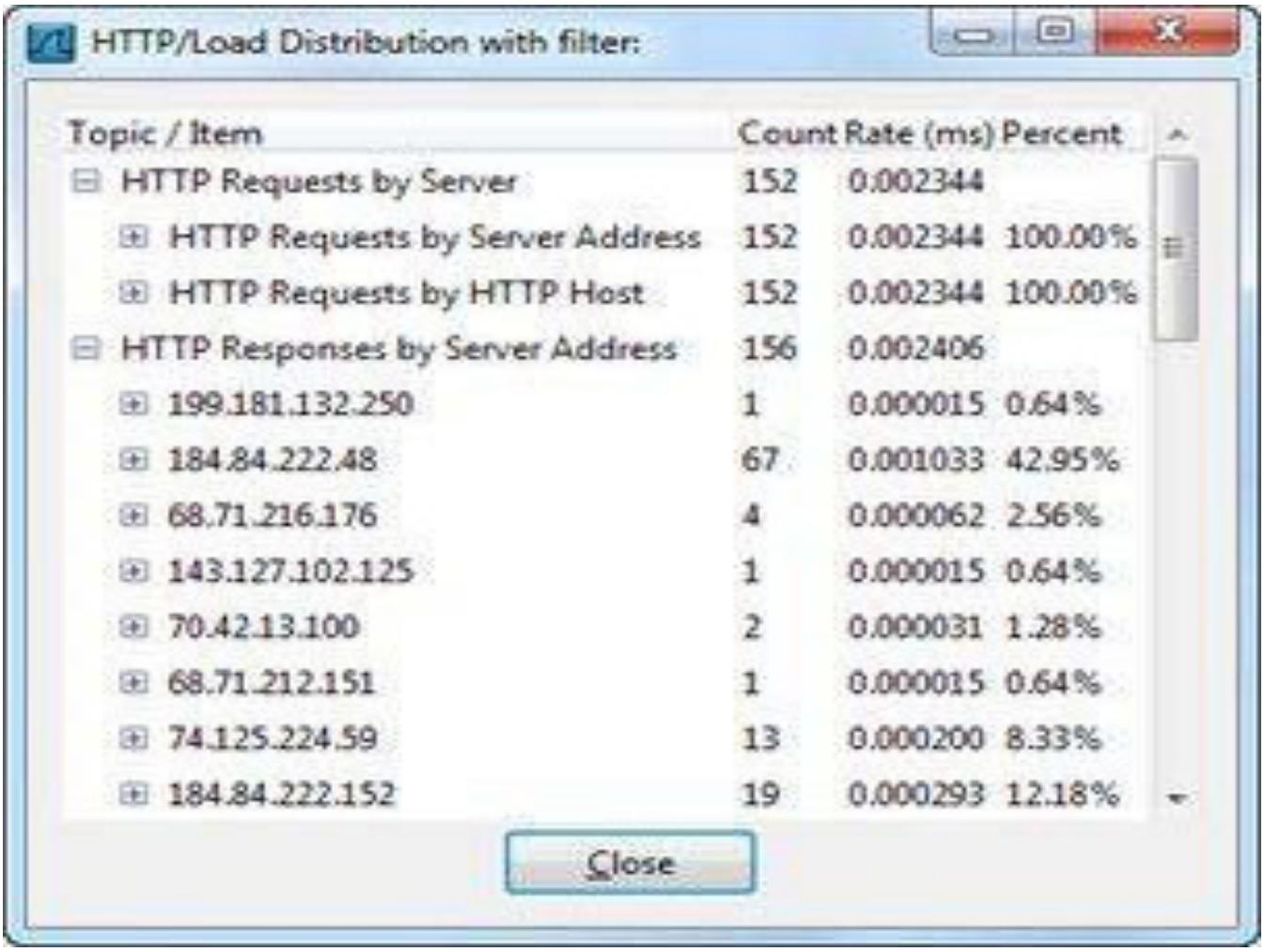
Display HTTP Statistics

Wireshark tracks HTTP statistics for load distribution, packet counters, and HTTP requests. Select **Statistics | HTTP** and select the type of statistic you are interested in.

You are provided an option to apply a display filter to the statistics. For example, if you have a trace file that contains web browsing session to numerous hosts, you might apply an `http.host==www.wireshark.org` display filter to examine statistics for web browsing sessions to www.wireshark.org only.

HTTP Load Distribution

HTTP Load Distribution lists the HTTP requests and responses by server. Expanding the HTTP Requests by HTTP Host section lists the hosts contacted and the number of request packets sent to each one.



The HTTP Load Distribution statistic is an excellent resource for determining web site redirections and dependencies. In Figure 272 we are viewing the HTTP referrals and dependencies when we browse to www.espn.com.

Examining this statistic, we learned that a simple browsing session to www.espn.com (http-espn2012.pcapng) creates HTTP sessions with 35 different servers that include content providing partners and advertisers. It is easy to understand why the www.espn.com site is slow to load.

Figure 272. The HTTP load distribution defines the number of servers we contacted when browsing to www.espn.com [http-espn2012.pcapng]

HTTP Packet Counter

When analyzing HTTP communications, the HTTP Packet Counter is invaluable because it lists the Status Code responses. Spotting 4xx Client Error or 5xx Server Error responses is simple.

Figure 273 shows the HTTP Packet Counter for another browsing session to www.espn.com (http-espn2007.pcapng). We can see some HTTP 301 and 302 redirections and a 404 Not Found response.

HTTP Requests

HTTP Requests lists each item requested of each HTTP server. In Figure 274, we are examining the HTTP Requests sent during our web browsing session to www.espn.com using the http-espn2011.pcapng file again. As you can see, we downloaded content from doubleclick.net during our browsing session.

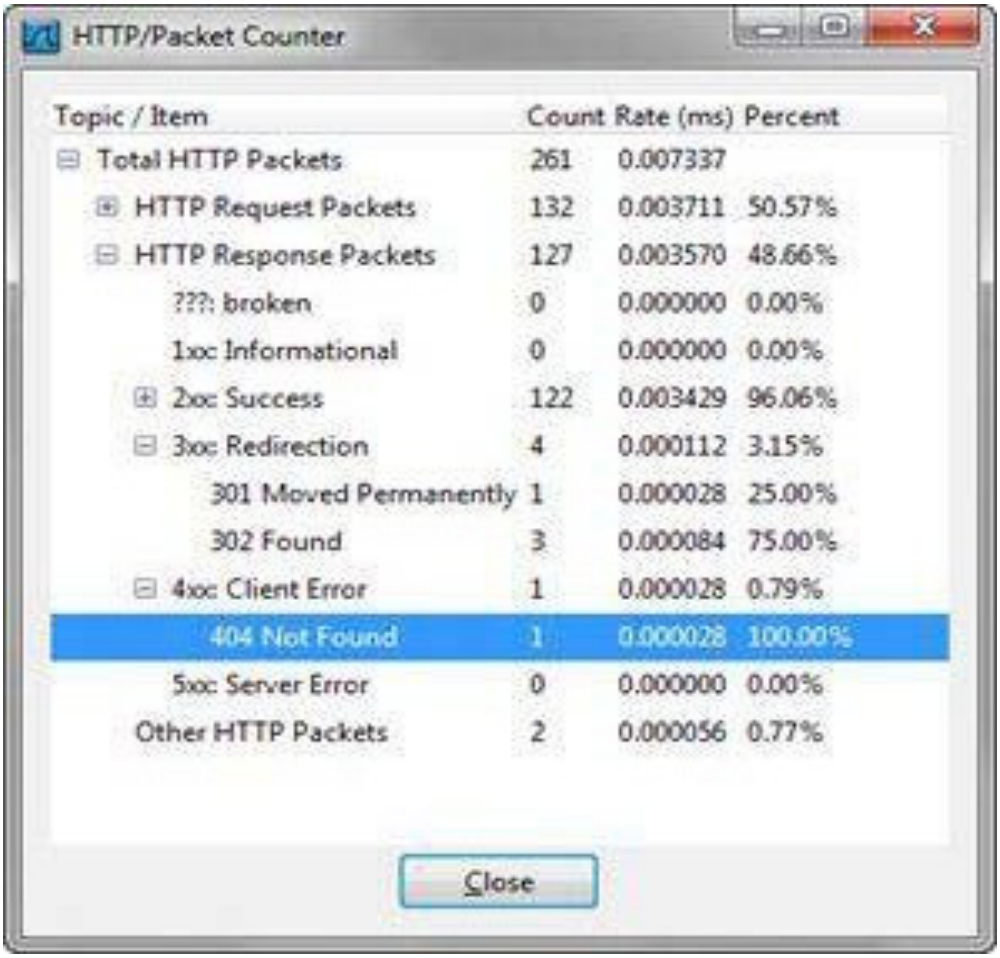


Figure 273. The HTTP Packet Counter displays the HTTP request method and the Status Code responses [http-espn2007.pcapng]

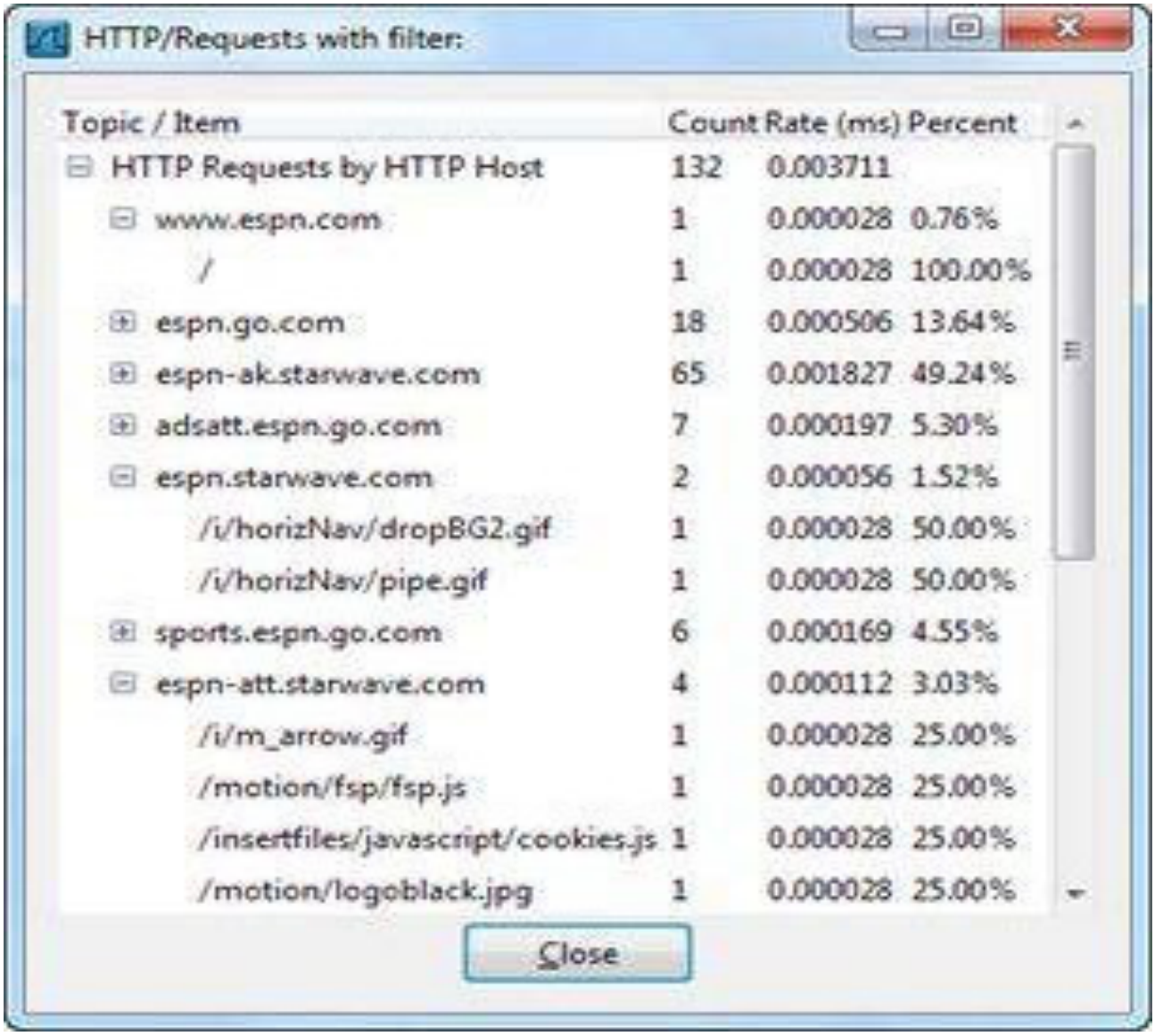


Figure 274. The HTTP Request statistic details each HTTP request made to each HTTP server [http-espn2007.pcapng]

Graph HTTP Traffic Flows

Flow Graphs provide a visual representation of the communications that occur during an HTTP session. This is an ideal statistic window to open when troubleshooting slow web browsing sessions. Each target host is listed in a column and every packet is listed in a row.

Create a Flow Graph to Spot Web Site Dependencies

Consider creating a Flow Graph based on your web browsing traffic. Capture your own browsing traffic to a popular website and notice the number of columns created because of other linked servers.

Select **Statistics | Flow Graph** to choose the three options for viewing the Flow Graph window, as shown in Figure 275.

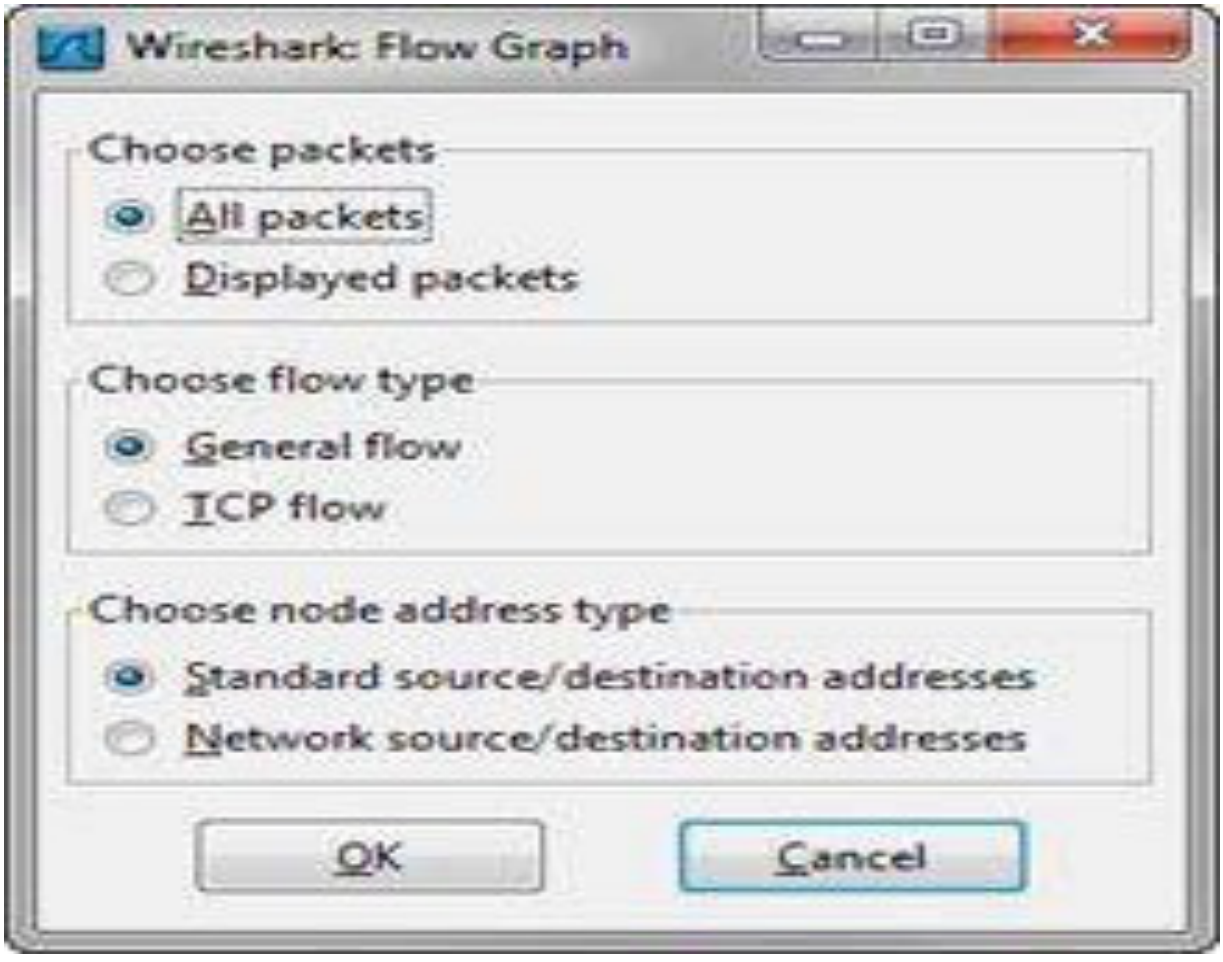


Figure 275. Create Flow Graphs based on the general flow or just the TCP header

Choose Packets

You can graph the flow of all packets in the trace file or just the displayed packets. If your trace file contains more than one conversation, you may want to filter on the conversation you want to graph and then open the Flow Graph window.

Choose Flow Type

The general flow view includes application-layer information such as requests and replies. The TCP Flow Graph shows just the TCP header values such as the sequence number and

acknowledgment number value and the TCP flag settings.

Choose Node Address Type

The Standard source/destination addresses option shows IP addresses of devices listed in the graph and is the recommended setting due to space constraints when many hosts are communicating with each other. Choose the network source/destination addresses if you are using network name resolution with Wireshark.

The Flow Graph contains IP address columns representing the HTTP client, the DNS server and the 25 HTTP servers that we contacted when browsing to www.espn.com.

Wireshark will highlight the corresponding packet when you click on a packet description in the IP address column.

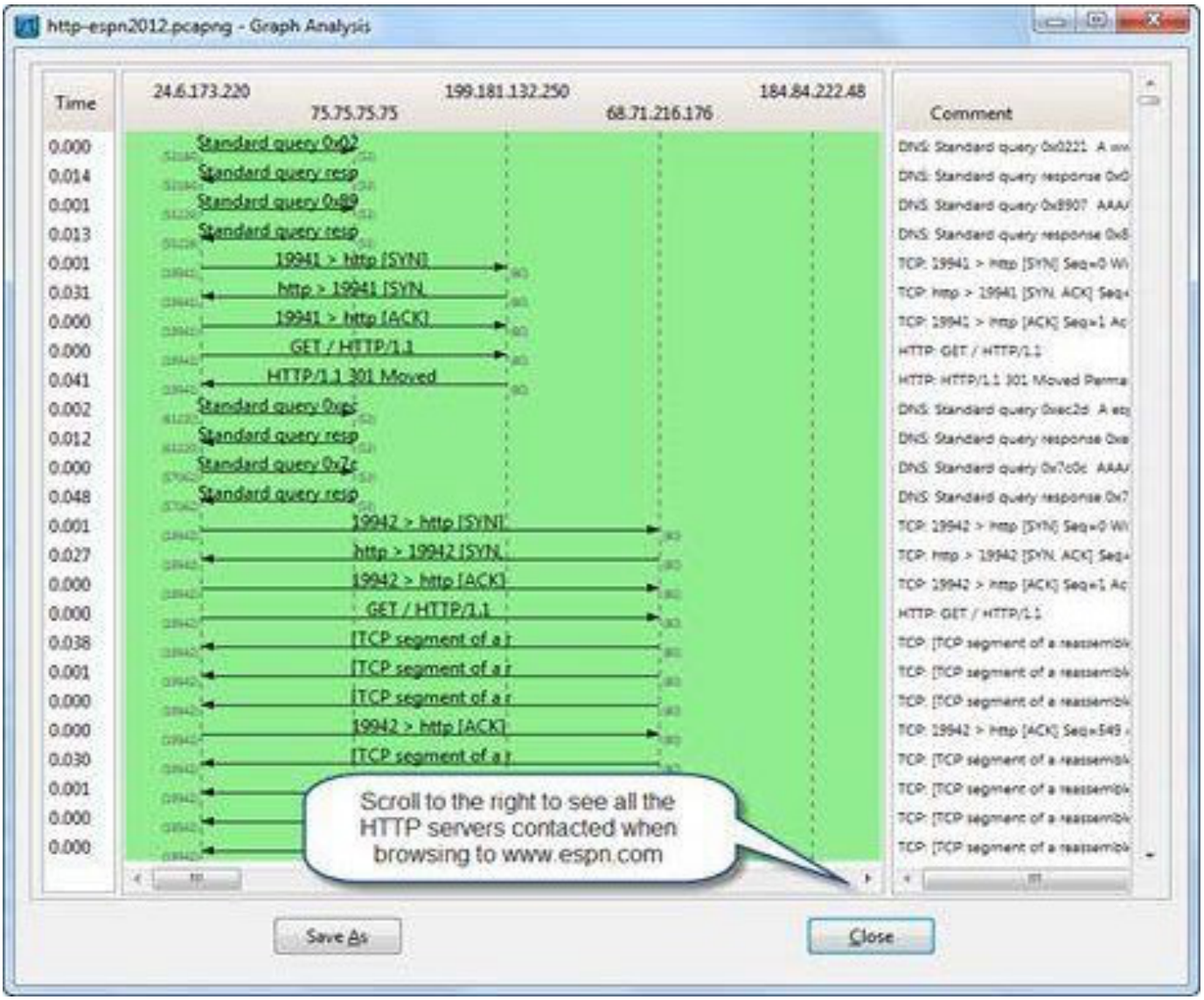


Figure 276. The Flow Graph adds a column for each host in the trace file [http-espn2012.pcapng]

After creating a Flow Graph, click **Save As** to save the contents of the Flow Graph in a text file. Depending on the number of IP address columns depicted in the Flow Graph, the text file width could be extremely wide and may print best in landscape mode.

Set HTTP Preferences

There are seven preference settings for HTTP communications as shown in the HTTP Preferences window in Figure 277.

One of the important settings that may need to be changed is the TCP Ports list. These are the ports associated with the HTTP dissector and the SSL/TLS port number to associate with the SSL/TLS dissector. Ensure the port used by your HTTP communications is listed in TCP Ports.

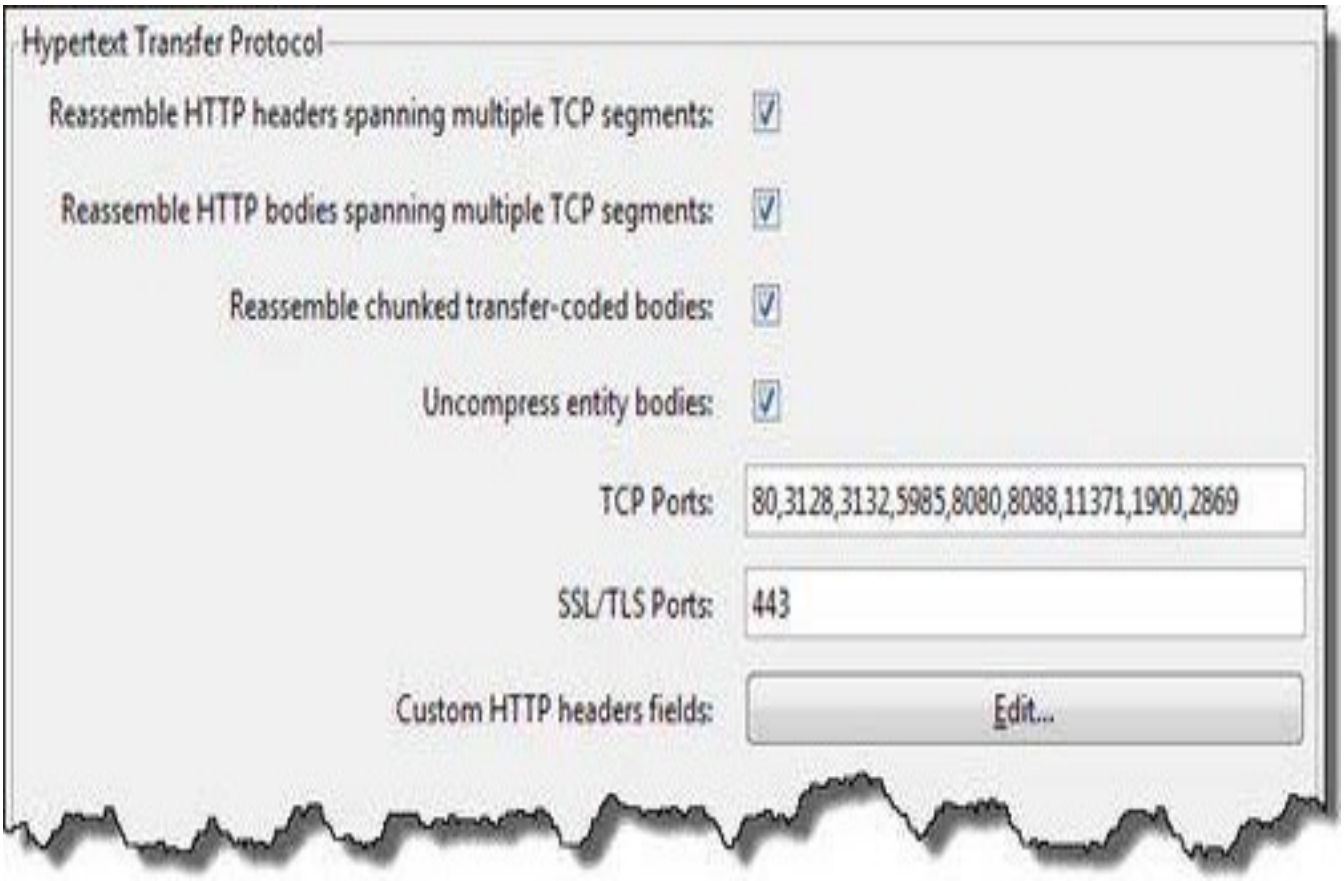


Figure 277. The HTTP preferences settings

Summary

HTTP uses a request/response model to transfer data between hosts. HTTP communications use TCP as the transport mechanism—the most commonly used HTTP port number is 80.

Clients send commands such as GET and POST to the HTTP server. HTTP servers respond with a numerical response code. Codes greater than 399 identify client and server errors. Many people are familiar with the dreaded 404 Not Found response seen when a page does

not exist.

When analyzing HTTP communications, watch for the IfModified-Since request modifier. This indicates that the client has a page in cache. If the server responds with code 304 Not Modified, the client will load the page from cache instead of across the network. This will affect your web loading time analysis.

Slow web browsing sessions can be caused by TCP problems as well as interdependencies on other web sites (such as advertisers), and non-optimized web sites. Wireshark enables you to rebuild web pages and export HTTP objects.

Practice What You've Learned

Download and use the trace files from the following link to practice analyzing HTTP communications.
https://drive.google.com/drive/folders/1SLCroQXf7Vb2gTfoFcwq6WEtn0NwsR4E?usp=sharing

Lab Tasks

1.	**Trace file: http-500error.pcapng** (This trace file depicts an HTTP error dealing with the server. 500 is defined as an Internal Error.) 1. How can you filter on all HTTP server errors?
2.	**Trace file: http-espn2007.pcapng** Select **Statistics | HTTP | HTTP Packet Counter**. 1. How many redirections are in this trace file? 2. How many client errors are in this trace file? 3. How many GET requests were required to launch the main page when browsing to www.espn.com?
3.	**Trace file: http-espn2011.pcapng** How many GET requests were required to launch the main page when browsing to www.espn.com? Is this more efficient now than in 2007?
4.	**Trace file: http-espn2012.pcapng** How many GET requests were required to launch the main page in 2012? How many sites did we have to connect to when loading the page? Were there any redirections?

5.	**Trace file: http-facebook.pcapng** 1. How many GET requests were required to launch the main page when browsing to www.facebook.com? 2. Why did the client have to send another DNS query so late in the process? 3. What was the last item to load? 4. Would this have slowed down the page loading process?
6.	**Trace file: http-fault-post.pcapng** 1. How many times did we try to send information up to the HTTP server? 2. Should you troubleshoot the 11 second delay before packet 29? 3. To what host are we trying to POST information?
7.	**Trace file: http-winpcap.pcapng** (This trace contains a web browsing session to www.winpcap.org.) 1. Does this client have any of the website elements in cache? 2. What is the largest delay in the trace file? 3. Should you troubleshoot this delay? 4. What operating system is running on the WinPcap server? 5. What is the size of the new.gif file? 6. What image does the new.gif file contain?
8.	**Trace file: http-winpcap.pcapng** This trace contains a web browsing session to www.winpcap.org. 1. Does this client have any of the website elements in cache? 2. What is the largest delay in the trace file? 3. Should you troubleshoot this delay? 4. What operating system is running on the WinPcap server?

	5. What is the size of the new.gif file? 6. What image does the new.gif file contain? 7. Will this connection support window scaling?

Review Questions

Q1

You are analyzing an HTTP session as a user browses a new website. What HTTP response code indicates the page was found locally?

Q2

How is an HTTP 404 Not Found categorized?

Q3

How can you determine that a client is loading web pages out of cache?

Q4

What display filter should you avoid if you want to view the TCP handshake and TCP ACKs during a web browsing session?

Q5

What is the HTTP request method used to send data up to an HTTP server?

Q6

What is the syntax for capture and display filters for HTTP traffic running over port 80?