Part 1 – Start Wireshark and create some traffic to examine

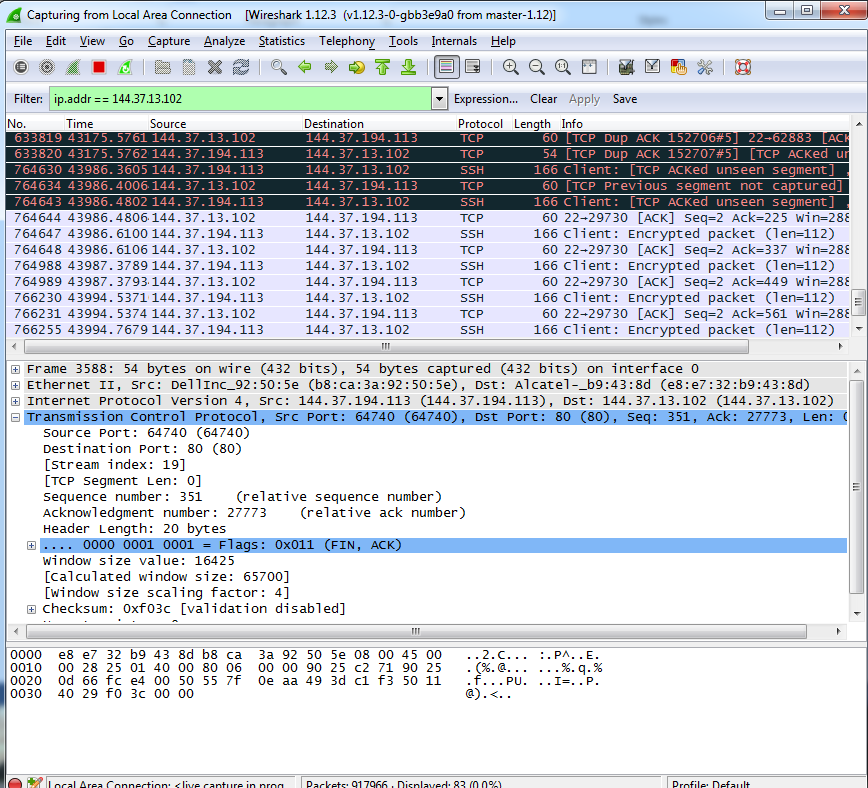
1. Start Wireshark.
2. Select “Ethernet” and click “start”.
3. For this lab we will be using Wireshark to examine the way TCP opens and closes connections.
4. Launch **PuTTY** from the start menu and connect to host empress.csusm.edu. Log in with your campus account. If PuTTY is not available on your system, download and install the file putty.exe from [www.putty.org](http://www.putty.org). Do NOT open PuTTY through CougarApps. If you do, the captured traffic will NOT be the same as instructions given here since your system would indirectly connect to empress through CougarApps but in this Lab we need a direct connection to empress server.
5. How to make sure that you run PuTTY on your PC, and NOT Citrix (Cougar Apps)? Click on search button and enter PuTTY. You may see a list of PuTTYs on your system. Select PuTTY. Do NOT select PuTTYgen or PuTTY Desktop app. Right click on it and choose “Open file location”, and make sure the path is on your PC, not on AppData. If you work with a Mac or Linux system, ssh to empress.
6. When you see a prompt [username@empress ~] you can type **exit**. Make sure you entered **exit**, and you returned from PuTTY, otherwise last steps of part 4 in this Lab will not work. Note that here you may run some commands like ls or cd. Then you could see those messages in Wireshark. However, since empress works with a secure ftp (not plain ftp), all messages are encoded.
7. At this point we can stop our packet capture by **clicking on the big red square** at the top of the Wireshark window.

Part 2 – Examine TCP Traffic

1. We are going to analyze the PuTTY session traffic first. In Wireshark’s filter box enter “**tcp.port == 22”.**

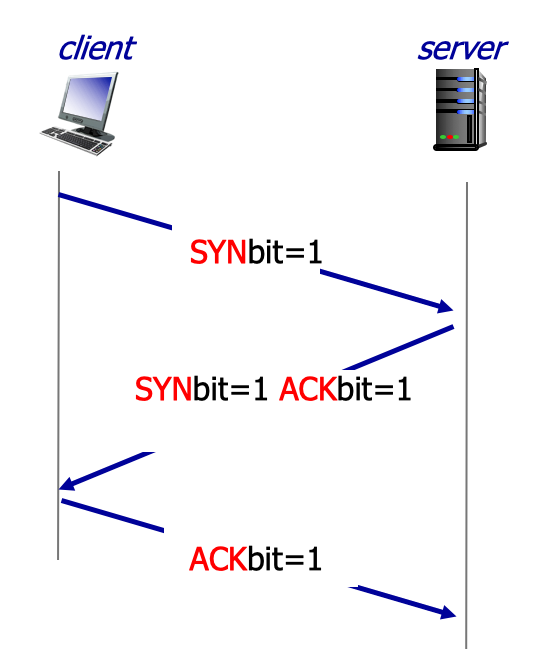
Note: Port 22 belongs to FTP protocol. PuTTY is a file transfer application and generates FTP traffic. empress.csusm.edu is a file server, and always listens to port 22.

1. We will be focusing on the IP and TCP header information and not the actual traffic or packet content. You have connected to the empress server using a TCP 3-way handshake and closed the connection (exit) using a 4-way close session. Wireshark has captured these exchanged messages. Now we will go over details of captured traffic in Part 3 (making a connection) and Part 4 (closing the connection).
2. The Wireshark window has three results sections: the “packet list” pane (top), the “packet details” pane (middle), and the “packet bytes” pane (bottom). Wireshark’s “Packet Details” pane is divided into multiple sections. These sections include IP: “Internet Protocol” and TCP: “Transmission Control Protocol”. You can click the plus sign to expand each section. **Expand the IP and TCP sections**.



Part 3 – Three-Way Handshake: TCP connection establishment

1. In this part of the lab we expect to see the three-way handshake between your host and empress.csusm.edu. **Select the first packet** of the stream from the packet list (top pane).



1. Use the “IP” section of the packet details pane to **fill out the darker areas of the IP header information**. Note that “IHL” is “Header Length”. Make sure that the destination IP is empress IP.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL = 20 | DSCP | ECN | Total Length = 52 | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live = 128 | | Protocol = TCP | | Header Checksum | |
| Source IP Address = 144.37.208.53 | | | | | |
| Destination IP Address = 144.37.1.24 | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

Note that in the IP header you have the source and destination IP addresses, but nothing about the actual packet to be sent.

1. Use the “TCP” section to **fill out the darker areas of the TCP headers**. Mark any of the flags that are set. Make sure that the SYN flag is set, and the ACK flag is not set (refer to TCP 3-way handshake).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port = 61870 | | | | | | | | | | | Destination Port = 22 |
| Sequence Number = 0 | | | | | | | | | | | |
| ACK Number (if set) = 0 | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG = Not set | ACK = Not set | PSH = Not set | RST = Not set | SYN = Set | FIN = Not set | Window Size |
| Check Sum | | | | | | | | | | | Urgent Pointer |
| Options | | | | | | | | | | | |

1. Note that the TCP header doesn’t contain source and destination IP addresses. Those are where? *\_\_\_The IP addresses are defined in the link layer: the IP header.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*
2. **Find the [Stream Index]** in the TCP protocol section(This section was 22). The [Stream index], and any value enclosed in [ ] is a Wireshark application value and not part of the actual packet that is being sent across the network. [Stream Index] is used by Wireshark to organize packets into streams. When we analyze traffic we want to make sure we are looking at the right piece of traffic and not traffic belonging to another session.
3. Now that we know the stream index we will filter on the stream traffic. In the Filter section enter **tcp.stream eq ###** where ### is the value for [Stream index]. Eg. if [Stream index] = 12 your filter should be tcp.stream eq 12
4. **Select the second packet** in the stream.
5. **Fill out the following IP header** with the information from Wireshark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL = 20 | DSCP | ECN | Total Length = 52 | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live = 61 | | Protocol = TCP | | Header Checksum | |
| Source IP Address = 144.37.1.24 | | | | | |
| Destination IP Address = 144.37.208.53 | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port = 22 | | | | | | | | | | | Destination Port = 61870 | |
| Sequence Number = 0 | | | | | | | | | | | | |
| ACK Number (if set) = 1 | | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG = Not set | ACK = Set | PSH = Not set | RST = Not set | SYN = Set | FIN = Not set | | Window Size |
| Check Sum | | | | | | | | | | | Urgent Pointer | |
| Options | | | | | | | | | | | | |

1. Compare this set of headers to the headers from the first packet. How do they differ?

They differ in the flags: the last has the acknowledgement flag set; the source and destination IP addresses switched their places; and the time-to-live of the last of the packets differ.

1. **Select the third packet** in the stream.
2. **Fill out the following IP header** will the information from Wireshark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL = 20 | DSCP | ECN | Total Length = 40 | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live = 128 | | Protocol = TCP | | Header Checksum | |
| Source IP Address = 144.37.208.53 | | | | | |
| Destination IP Address = 144.37.1.24 | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port = 61870 | | | | | | | | | | | Destination Port = 22 |
| Sequence Number = 1 | | | | | | | | | | | |
| ACK Number (if set) = 1 | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG = Not set | ACK = Set | PSH = Not set | RST = Not set | SYN = Not set | FIN = Not set | Window Size |
| Check Sum | | | | | | | | | | | Urgent Pointer |
| Options | | | | | | | | | | | |

1. You should have noticed that the first packet was the SYN, the second packet was the SYN, ACK and the third packet was the ACK. This is the TCP 3 way handshake, which is the start of the network connection.
2. **Select the fourth packet** in the stream. This one should have your IP address as the source and 144.37.1.24 as the destination.
3. **Fill out the following IP header** with the information from Wireshark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL = 20 | DSCP | ECN | Total Length = 552 | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live = 128 | | Protocol = TCP | | Header Checksum | |
| Source IP Address = 144.37.208.53 | | | | | |
| Destination IP Address = 144.37.1.24 | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port = 61870 | | | | | | | | | | | Destination Port = 22 |
| Sequence Number = 29 | | | | | | | | | | | |
| ACK Number (if set) = 22 | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG = Not set | ACK = Set | PSH = Set | RST = Not set | SYN = Not set | FIN = Not set | Window Size |
| Check Sum | | | | | | | | | | | Urgent Pointer |
| Options | | | | | | | | | | | |

1. Notice that in this packet’s TCP header the Psh bit is set. What does this do? Search “Push bit TCP” on Google to find out the reason why we set psh bit. Summarize the reason:

The push bit is set when your application sends segments of data, so a TCP receiving peer will push the data to the application buffer immediately if the push flag is set.

What sequence did we just record? \_\_\_\_\_29\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Note on Sequence numbers: If you look in the TCP header you will see there is a [Next sequence number] field. This is the Wireshark application calculating what the next sequence number would be from that source host. Wireshark calculates this value by adding the total length of data to the Sequence Number. Explore the items of TCP and find the field that represents the data size. What is the field name? \_TCP Segment Len\_\_\_\_\_\_\_

Part 4 – closing TCP connection

1. TCP closes a connection in 4 steps: 1. FIN, ACK, 2. ACK, 3. FIN, ACK, 4. ACK. Note that every single message acknowledges the last message.
2. **Scroll to the end** of the packet list.
3. **Select the FIN ACK packet** that is **forth** from the bottom. FIN, ACK will be in the info column of the packet list: FIN to finalize TCP connection from client side, and ACK to acknowledge the last packet sent to server before we start closing the connection.

Note1: In some cases, it happens that the source sends a FIN message and then receives a late ACK for a last data packet sent to the server. In such cases you will see an extra ACK in addition to the expected last four packets.

Note2: In some cases, you see only three packets from bottom. 1. FIN, ACK, 2. FIN, ACK, 3. ACK.

If that is the sequence of packets you see on Wireshark, the reason is the server has sent you an ACK message, but your system has not received that for some reason. In that case, you have missed the ACK message from server, so you see only three messages that close the connection. You can leave the steps 38, 39, 40 blank and mention the reason.

A close up of a logo

Description automatically generated

1. **Fill out the following IP header** with the information from Wireshark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL = 20 | DSCP | ECN | Total Length = 40 | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live = 128 | | Protocol = TCP | | Header Checksum | |
| Source IP Address = 144.37.208.53 | | | | | |
| Destination IP Address = 144.37.1.24 | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port = 61870 | | | | | | | | | | | Destination Port = 22 |
| Sequence Number = 2197 | | | | | | | | | | | |
| ACK Number (if set) = 3914 | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG = Not set | ACK = Set | PSH = Not set | RST = Not set | SYN = Not set | FIN = Set | Window Size |
| Check Sum | | | | | | | | | | | Urgent Pointer |
| Options | | | | | | | | | | | |

1. **Select the ACK packet that is the third** from the bottom (ACK from server).
2. **Fill out the following IP header** with the information from Wireshark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL | DSCP | ECN | Total Length | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live | | Protocol | | Header Checksum | |
| Source IP Address | | | | | |
| Destination IP Address | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

I haven’t received an ACK message: the server has sent me an ACK message, but my system has not received it for some reason.

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port | | | | | | | | | | | Destination Port |
| Sequence Number | | | | | | | | | | | |
| ACK Number (if set) | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN | Window Size |
| Check Sum | | | | | | | | | | | Urgent Pointer |
| Options | | | | | | | | | | | |

I haven’t received an ACK message: the server has sent me an ACK message, but my system has not received it for some reason.

1. **Select the FIN ACK packet that is 2nd** from the bottom (FIN to finalize the connection from server).
2. **Fill out the following IP header** with the information from Wireshark.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | IHL = 20 | DSCP | ECN | Total Length = 40 | |
| Identification | | | | Flags | Fragment Offset |
| Time to Live = 61 | | Protocol = TCP | | Header Checksum | |
| Source IP Address = 144.37.1.24 | | | | | |
| Destination IP Address = 144.37.208.53 | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | |

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Source Port = 22 | | | | | | | | | | Destination Port = 61870 | | |
| Sequence Number = 3914 | | | | | | | | | | | | |
| ACK Number (if set) = 2198 | | | | | | | | | | | | |
| Data Offset | Reserved | NS | CWR | ECE | URG = Not set | ACK = Set | PSH = Not set | RST = Not set | SYN = Not set | | FIN = Set | Window Size |
| Check Sum | | | | | | | | | | Urgent Pointer | | |
| Options | | | | | | | | | | | | |

1. **Select the last packet. Fill out the following IP header** will the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Version | | | | IHL = 20 | | | | DSCP | | | | | | ECN | | Total Length = 40 | | | | | | | | | | | | | | | |
| Identification | | | | | | | | | | | | | | | | Flags | | | Fragment Offset | | | | | | | | | | | | |
| Time to Live = 128 | | | | | | | | Protocol = TCP | | | | | | | | Header Checksum | | | | | | | | | | | | | | | |
| Source IP Address = 144.37.208.53 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Destination IP Address = 144.37.1.24 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Options - Only used if IHL is greater than 20 Bytes | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

1. **Fill out the following TCP header** with the information from Wireshark.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Source Port = 61870 | | | | | | | | | | | | | | | | Destination Port = 22 | | | | | | | | | | | | | | | |
| Sequence Number = 2198 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ACK Number (if set) = 3915 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Data Offset | | | | Reserved | | | NS | CWR | ECE | URG = Not set | ACK = Set | PSH = Not set | RST = Not set | SYN = Not set | FIN = Not set | Window Size | | | | | | | | | | | | | | | |
| Check Sum | | | | | | | | | | | | | | | | Urgent Pointer | | | | | | | | | | | | | | | |
| Options | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

1. Note: notice that the 4 packets were: (FIN, ACK), (ACK), (FIN, ACK), (ACK). This is what is known as the TCP 4-way handshake to teardown a connection. This tells us that the application has closed the ports correctly.

I haven’t received an ACK message: the server has sent me an ACK message, but my system has not received it for some reason.

Part 5 - TCP Traffic on a webpage

1. This time we want to check the TCP three-way handshake and closing connection on a webpage. Start capturing traffic at Wireshark. Click on this link: <https://www.whatsmyip.org/string-encoder-decoder/>. This causes a TCP three-way handshake between your system and this web server. In the opened webpage, enter a short string into the Base64 Encoder input box and encode it. You exchanged a few messages with the web server. Then click on one the links on the left-side of the webpage. This causes a TCP connection close. Stop capturing traffic.
2. To filter the traffic, use the IP address of web server. You can get the IP address simply by running a ping command in a command window: “ping www.whatsmyip.org”. This command returns some information including the IP address of the server and also whether it is up or not. When I ran it, it returned 204.11.35.98 as the server’s IP address. Now filter the traffic captured at Wireshark using “tcp && ip.addr==204.11.35.98”, which filters all the TCP traffic exchanged with the IP address of the webserver. Once you filtered the traffic, you may see two traffic streams with the server, one for regular messages and one for exceptions.
3. Select the first SYN request and find its stream number. Add that to the filter box, so that you filter the traffic using “tcp && ip.addr==204.11.35.98 && tcp.stream== ##” where ## is the tcp stream number. **Answer the following questions.**
4. Find the three-way handshake frames. For each one of SYN, SYNACK and ACK frames, answer the following questions:

What are the source and destination IP addresses of this frame?

* SYN: source IP address = 144.37.208.53; destination IP address = 204.11.35.98
* SYNACK: source IP address = 204.11.35.98; destination IP address = 144.37.208.53
* ACK: source IP address = 144.37.208.53; destination IP address = 204.11.35.98

What are the source and destination port numbers of this frame?

* SYN: source port = 62467; destination port = 443
* SYNACK: source port = 443; destination port = 62467
* ACK: source port = 62467; destination port = 443

What are the sequence number and Ack number (if set) for the TCP segment encapsulated in this frame?

* SYN: sequence number = 0; ack number = 0
* SYNACK: sequence number = 0; ack number = 1
* ACK: sequence number = 1; ack number = 1

What are the flags set for this frame? E.g. SYN, etc.

* SYN: [SYN]
* SYNACK: [ACK, SYN]
* ACK: [ACK]

1. Find the FIN frames, which close the TCP connection. For each one of FIN and FINACK frames, answer the following questions:

What are the source and destination IP addresses of this frame?

* FINACK(client): source IP address = 144.37.208.53; destination IP address = 204.11.35.98
* ACK(client): source IP address = 204.11.35.98; destination IP address = 144.37.208.53
* FINACK(server): source IP address = 204.11.35.98; destination IP address = 144.37.208.53
* ACK(server): source IP address = 144.37.208.53; destination IP address = 204.11.35.98

What are the source and destination port numbers of this frame?

* FINACK(client): source port = 62467; destination port = 443
* ACK(client): source port = 443; destination port = 62467
* FINACK(server): source port = 443; destination port = 62467
* ACK(server): source port = 62467; destination port = 443

What are the sequence number and Ack number (if set) for the TCP segment encapsulated in this frame?

* FINACK(client): sequence number = 1315; ack number = 839
* ACK(client): sequence number = 839; ack number = 1316
* FINACK(server): sequence number = 838; ack number = 1315
* ACK(server): sequence number = 1315; ack number = 839

What are the flags set for this frame? E.g. FIN, etc.

* FINACK(client): [FIN, ACK]
* ACK(client): [ACK]
* FINACK(server): [FIN, ACK]
* ACK(server): [ACK]