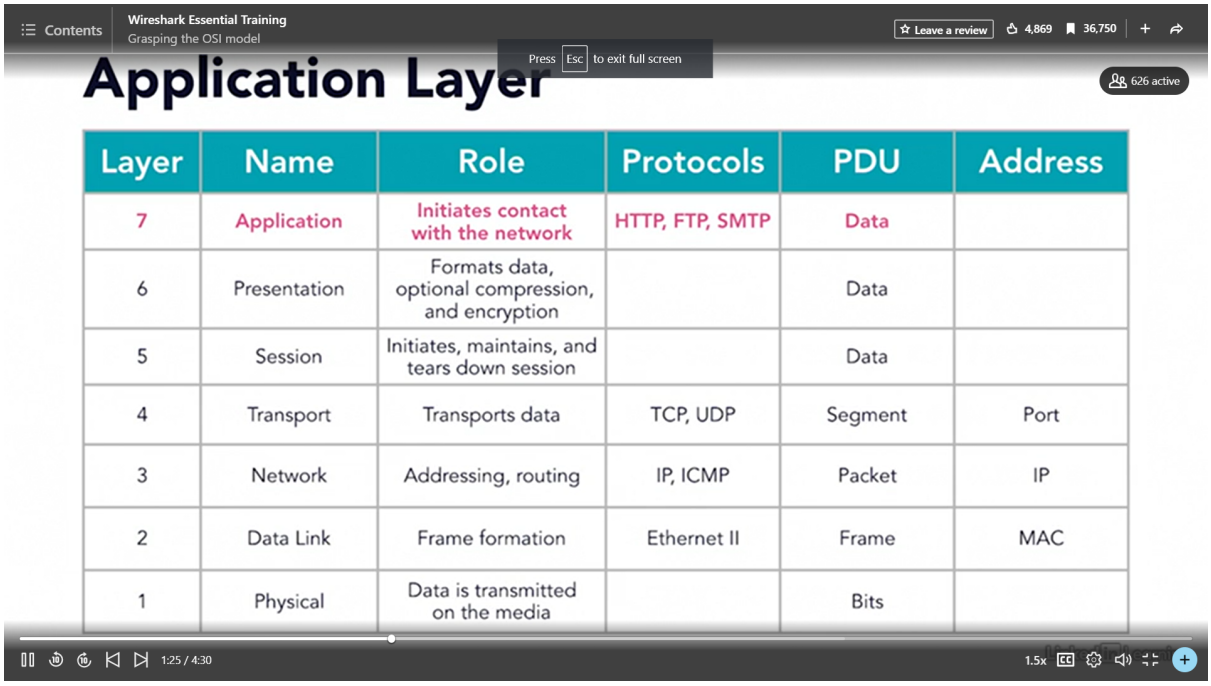
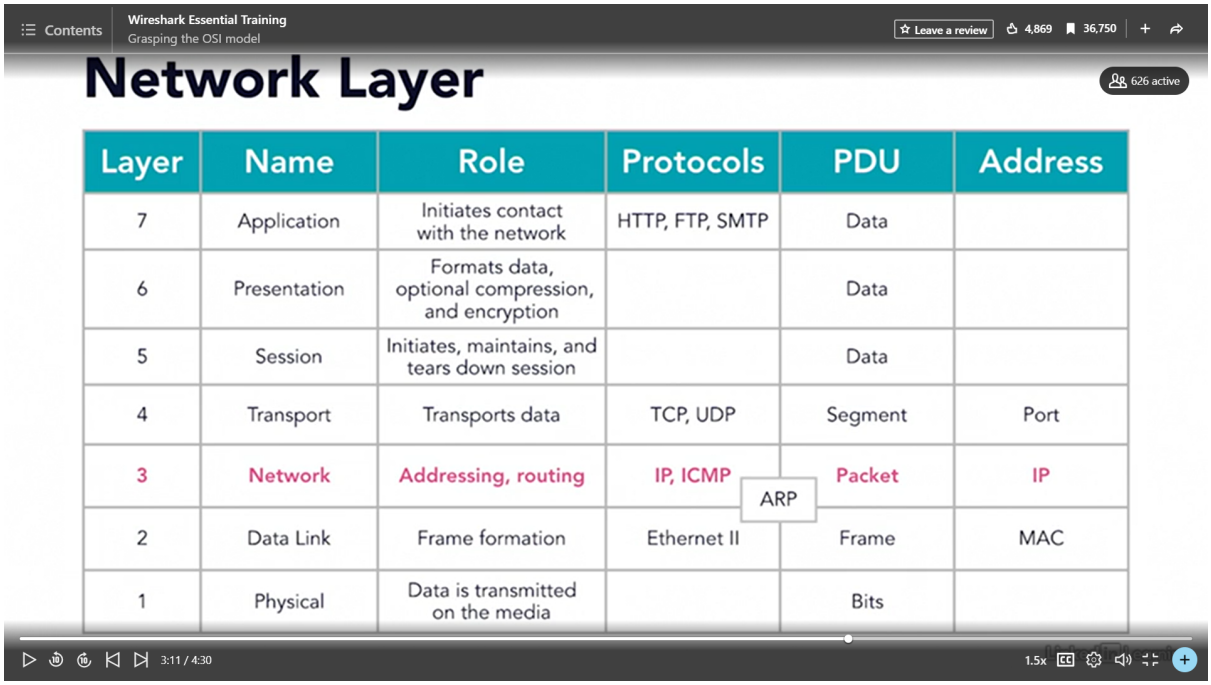
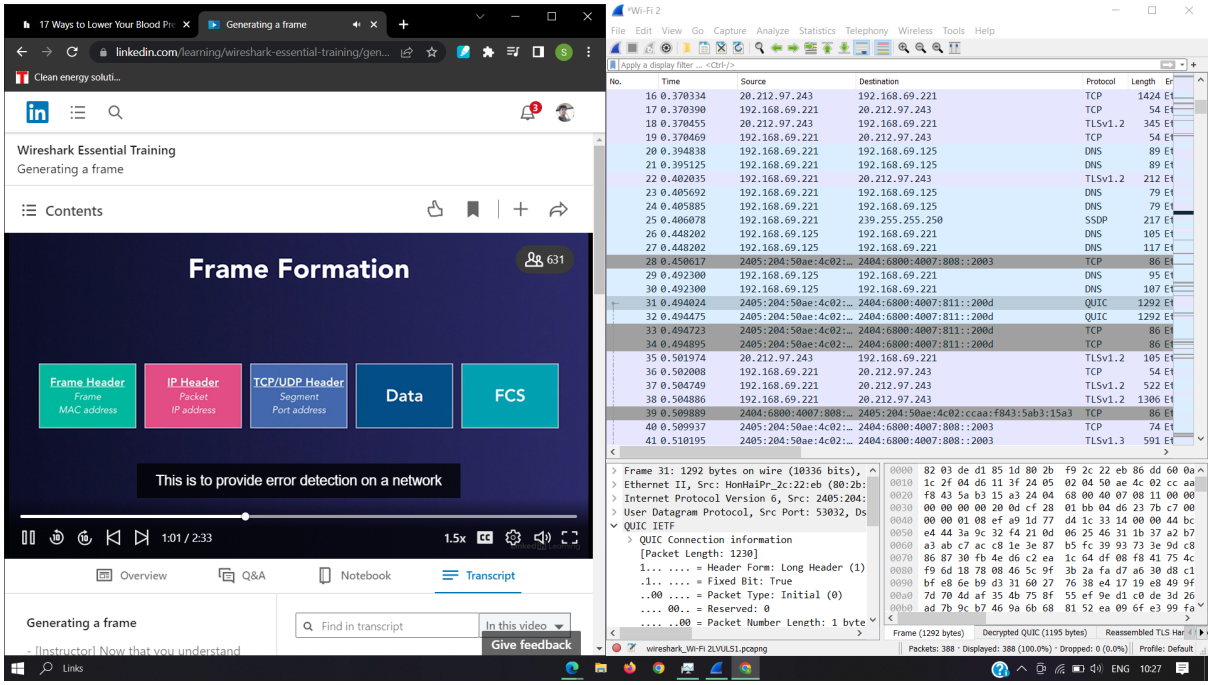
OSI MODEL and WireShark



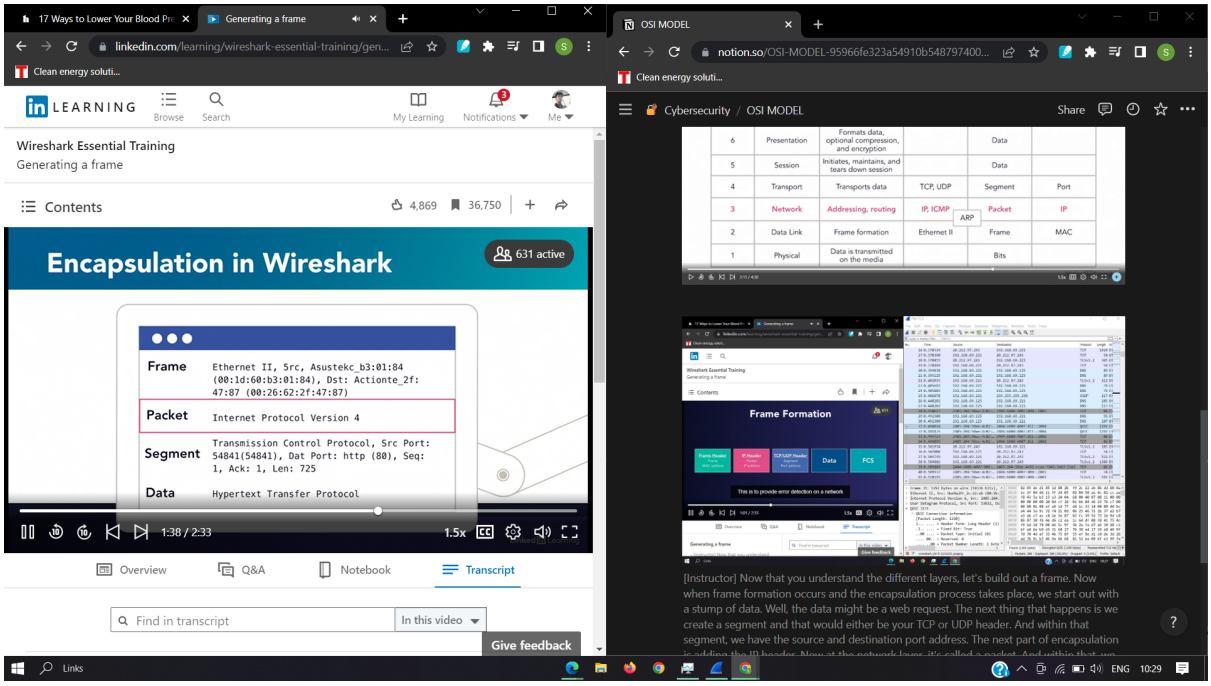
Layer	Name	Role	Protocols	PDU	Address
7	Application	Initiates contact with the network	HTTP, FTP, SMTP	Data	
6	Presentation	Formats data, optional compression, and encryption		Data	
5	Session	Initiates, maintains, and tears down session		Data	
4	Transport	Transports data	TCP, UDP	Segment	Port
3	Network	Addressing, routing	IP, ICMP	Packet	IP
2	Data Link	Frame formation	Ethernet II	Frame	MAC
1	Physical	Data is transmitted on the media		Bits	

Let's start with the Application Layer. Application Layer is Layer 7, **and this is all about initiating contact with the network.** Usually, a user who would initiate, perhaps getting a webpage. The protocols that are used are HTTP, FTP, or SMTP. The protocol data unit at this layer is simply data. Nothing magical has happened, it's just simply data. And there's no addresses that are needed. The Presentation Layer will take the next step. It will format the data. It provides optional compression and encryption. And the protocol data unit at this point is simply data. Layer 5 is the Session Layer. This is all about initiating, maintaining, and tearing down a session. Again, the protocol data unit at this point is simply data. Now, when we go to the Transport Layer, this is where we start the encapsulation process. The Transport Layer is responsible for transporting data. Depending on how you need it transported, you might choose a connection-oriented protocol such as TCP or connectionless protocol such as UDP. The protocol data unit at this point is a segment. And at this point, we need a port address. We need the source and destination port address, which will logically associate with an appropriate application. Layer 3 is the Network Layer. This is all about the network. We provide addressing and routing. And when we talk about the different protocols that are in this layer, we know, of course, there is IP; and, of course, we also need ICMP; but let's take a look at that protocol that I put right in between Layer 3 and Layer 2, which is Address Resolution Protocol. Now, Address Resolution Protocol is a unique protocol and that there is no IP header. And that's because it's where it needs to be. There's no routing involved, it's simply resolving an IP address to a MAC address. The protocol data unit at this point is a packet, and the address is an IP address. Layer 2 or the Data Link Layer is all about proper frame formation. The protocol is Ethernet II, which is the most widely used protocol on the local area network. And the protocol data unit at this point is a frame. And the address is a MAC address. Now, it's all ready and packaged up, it's encapsulated with proper frame formation, and it has all the necessary addresses. It, then, is broken down into a stream of bits. So it's able to be transmitted on whatever media you need it to be transmitted on. There are no addresses and no protocols involved. Again, it's just sending it off on a stream of bits. So if you aren't already familiar with the OSI model, get a better understanding of each of the layers of the OSI model, what it does, the protocol data units, and the addressing. All in all, these skills will help you to analyze traffic better.


Contents
Wireshark Essential Training
Grasping the OSI model
Leave a review
4,869
36,750
626 active
Network Layer
Layer
Name
Role
Protocols
PDU
Address
7
Application
Initiates contact with the network
HTTP, FTP, SMTP
Data
6
Presentation
Formats data, optional compression, and encryption
Data
5
Session
Initiates, maintains, and tears down session
Data
4
Transport
Transports data
TCP, UDP
Segment
Port
3
Network
Addressing, routing
IP, ICMP
ARP
Packet
IP
2
Data Link
Frame formation
Ethernet II
Frame
MAC
1
Physical
Data is transmitted on the media
Bits
3:11 / 4:30
1.5x



[Instructor] Now that you understand the different layers, let's build out a frame. Now when frame formation occurs and the encapsulation process takes place, we start out with a stump of data. Well, the data might be a web request. The next thing that happens is we create a segment and that would either be your TCP or UDP header. And within that segment, we have the source and destination port address. The next part of encapsulation is adding the IP header. Now at the network layer, it's called a packet. And within that, we have a source and destination IP address. The last part of this frame formation is the frame header. And within that, of course, we have the MAC address. And with a frame, not only is there a header, we also have a trailer. That is the frame check sequence. This is to provide error detection on a network by creating a cyclic redundancy check value that the network will check as it's traveling along on its way. Now, if we look at that frame formation and the encapsulation process, we can relate it to how it's presented to us in Wireshark. So take a look at this encapsulation and all the way at the top is, of course, your frame header. Frame is ethernet two and you see the source and destination MAC addresses within that frame header.



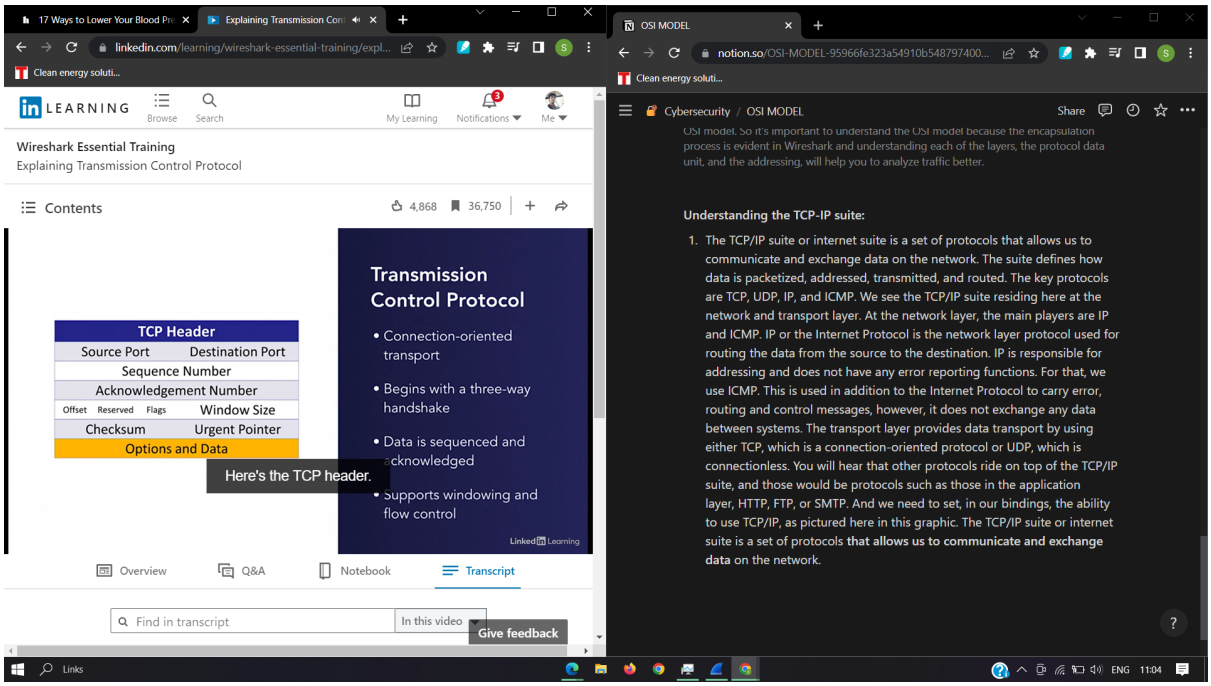
Now, if we look at that frame formation and the encapsulation process, we can relate it to how it's presented to us in Wireshark. So take a look at this encapsulation and all the way at the top is, of course, your frame header. Frame is ethernet two and you see the source and destination MAC addresses within that frame header. The next represents the network layer in the form of a packet. And there you see the IP header and of course, within that would be your source and destination IP address. Then we move to the transport layer where you see a segment and in this case, it's transmission control protocol. And there you have the source and destination ports. Tucked within that frame is the data. And here we see HTTP and it might be a web request. Keep in mind, not all frames have data **this is an example of a fully encapsulated frame,** just to show you how it relates in Wireshark to the OSI model. So it's important to understand the OSI model because the encapsulation process is evident in Wireshark and understanding each of the layers, the protocol data unit, and the addressing, will help you to analyze traffic better.

Understanding the TCP-IP suite:

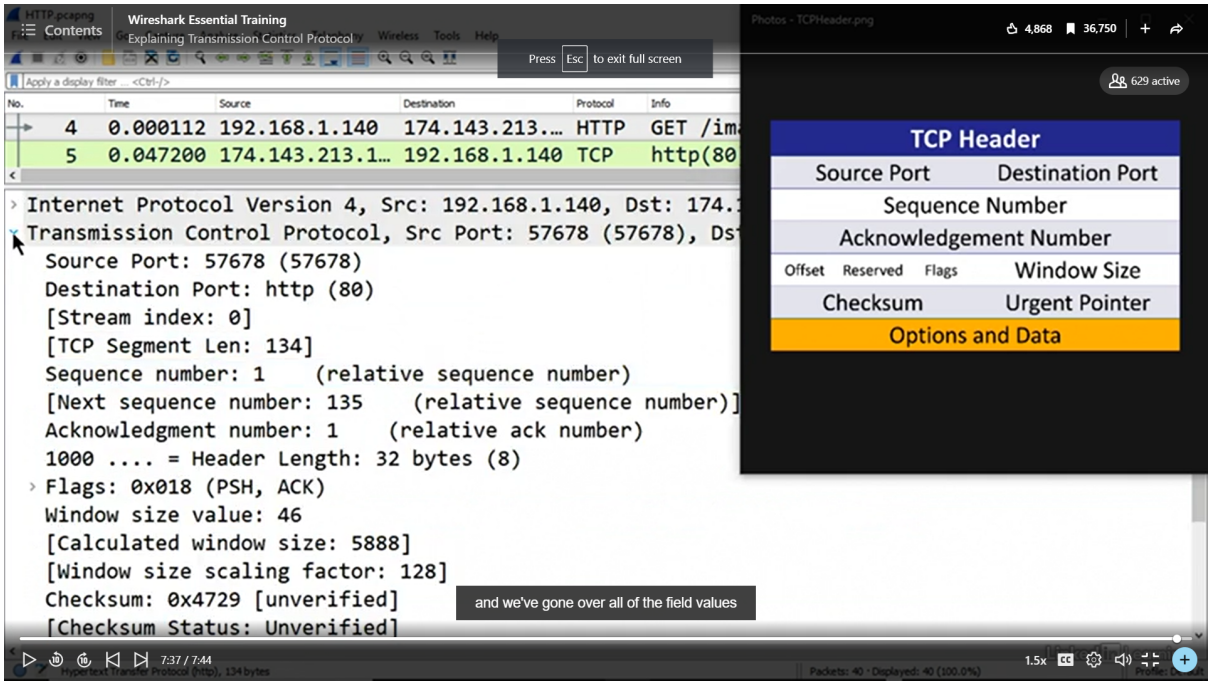
1. The TCP/IP suite or internet suite is a set of protocols that allows us to communicate and exchange data on the network. The suite defines how data is packetized, addressed, transmitted, and routed. The key protocols are TCP, UDP, IP, and ICMP. We see the TCP/IP suite residing here at the network and transport layer. At the network layer, the main players are IP and ICMP. IP or the Internet Protocol is the network layer protocol used for routing the data from the source to the destination. IP is responsible for addressing and does not have any error reporting functions. For that, we use ICMP. This is used in addition to the Internet Protocol to carry error, routing and control messages, however, it does not exchange any data between systems. The transport layer provides data

transport by using either TCP, which is a connection-oriented protocol or UDP, which is connectionless. You will hear that other protocols ride on top of the TCP/IP suite, and those would be protocols such as those in the application layer, HTTP, FTP, or SMTP. And we need to set, in our bindings, the ability to use TCP/IP, as pictured here in this graphic. The TCP/IP suite or internet suite is a set of protocols **that allows us to communicate and exchange data** on the network.

Explaining Transmission Control Protocol:



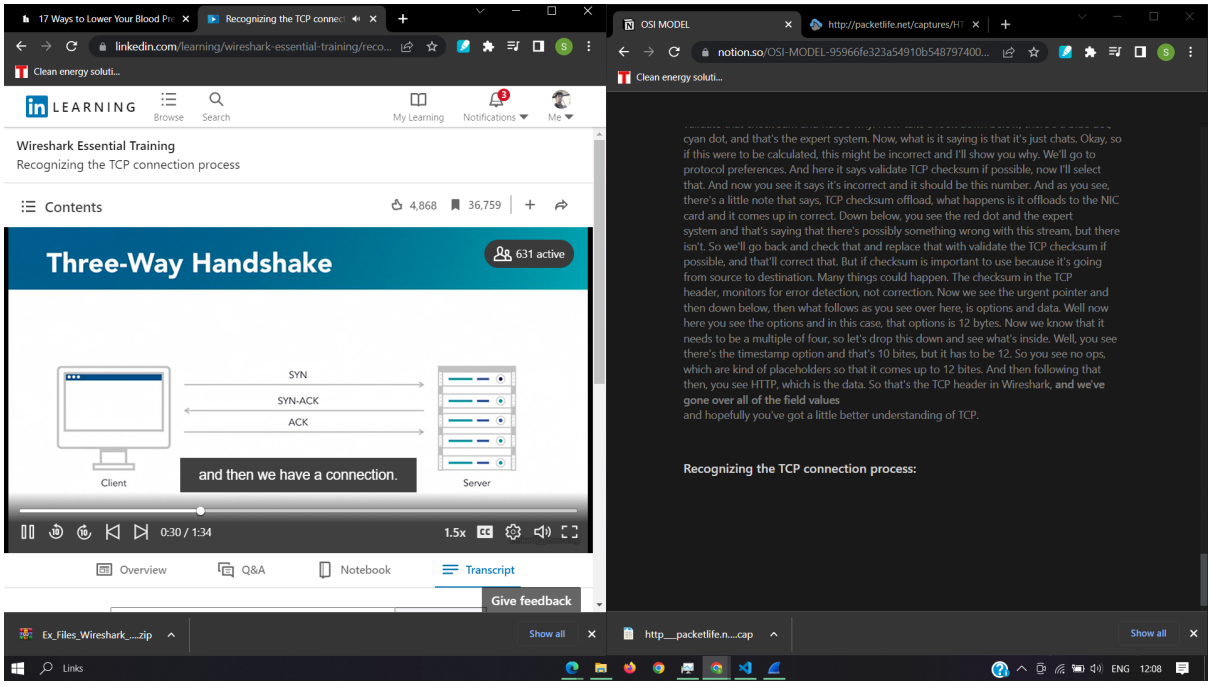
[Instructor] Transmission control protocol resides at layer four or the transport layer of the OSI model. Transmission control protocol provides connection oriented transport. TCP begins with a three-way handshake and data is sequenced and acknowledged. TCP supports windowing and flow control so as not to overwhelm the host. Here's the TCP header.



And now we'll go into Wireshark and take a look at those field values. If you'd like to follow along, we'll go to cloudshark. I downloaded HTTP.cap and you can go to export and download file and bring it up in Wireshark. **(Download link: https://www.cloudshark.org/captures/0012f52602a3)**Now, once in Wireshark, what I did was I'll show you here, it starts out with the three panels. I go to view and take off packet bites. I want a little more landscape. Now we'll bring this up and go to Frame 4. Let's just take a look at our encapsulation. Frame 4 is simply metadata about that frame. Here's the frame header, ethernet two, the network layer header IP, the transport layer header, TCP, and then the data HTTP. We're going to open up TCP. Now we can take a look, I'll float this on top, so you can refer to the TCP header. All right, first we see the source and destination port. The source port is 57678. Now sent from the client, that's simply saying when you come back, send the data to 57678. The destination port is 80, which is associated with HTTP. Here, we see stream index zero. There are no field value stream index, that's a value calculated by Wireshark. This helps you to keep track of the many different streams in Wireshark and if we were to look at conversations, we could go to stream index zero or stream 14 or whatever you'd like. Here, we see TCP segment length 134. Now, where did that value come from? Well, I'll scroll down here and I'll place my cursor on the HTTP header and here you see 134 bytes. Again, that's just a number calculated in Wireshark. Now here we see the sequence number one, and it's a relative sequence number. What that means it's in relationship to this stream, sequence number one, it's not the actual value. If it were, it's a large number, let's take a look. I'll right click. And I'll go to protocol preferences. Now here we can uncheck relative sequence numbers. And as you see, it's a large number. It's really hard to calculate values with that large number so we'll go back to relative sequence number by going to protocol preferences, and then we'll put relative sequence number back. For example, here it says next sequence number 135. Well, that's 134 plus one, and that's, what's calculated as what's coming next. Here, you see the acknowledgement number one and again, it's a relative acknowledgement number. And now here is the header length, 32 bytes. Normally a TCP header will be about 20 bytes. Now when we add options, that adds to the length. So let's take a look, I'll place my cursor up here at the TCP header and then down below, we see that the TCP header is 32 bytes. And we'll scroll up here and let's take a look at the flag section. Now, there's a lot going on with the flags. When you're ready to go a little more advanced, you can find out about all the flags and why

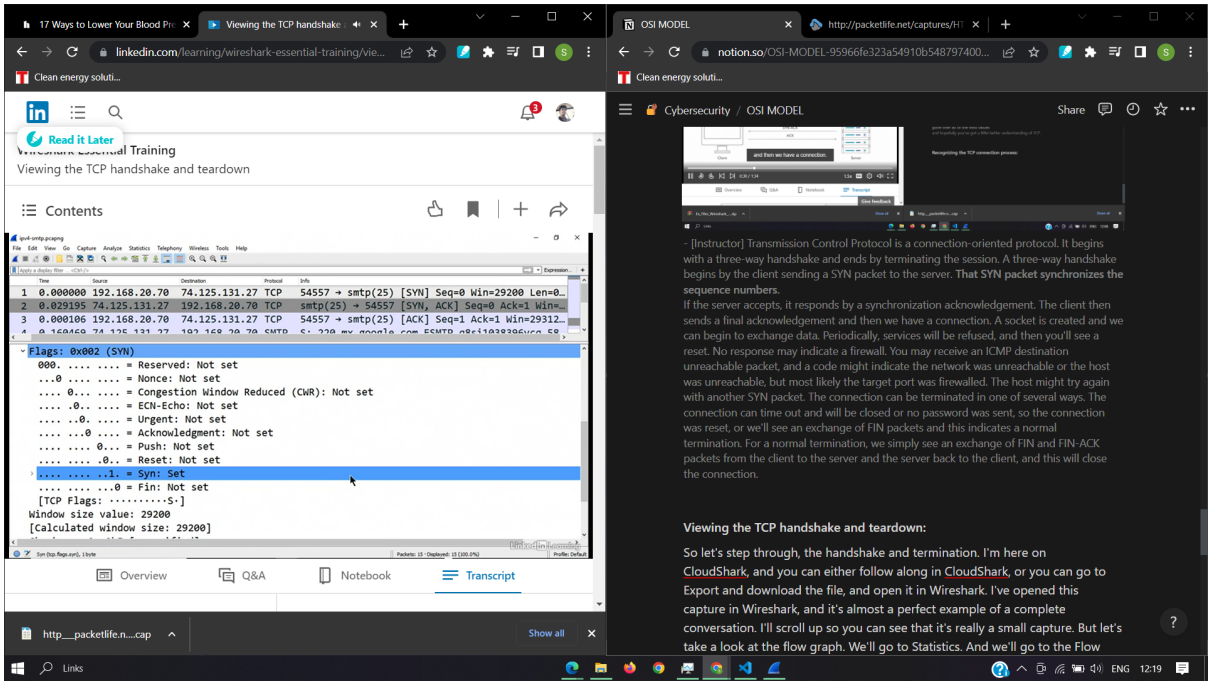
we keep track of those is certainly your devices are monitoring what's happening during the conversation. The flags tell the story about what's going on at that point in time. Now some common ones that you'll see of course, is the syn flag and that's going to synchronize the sequence numbers. We see acknowledgement and that's of course acknowledging the data push. And the fin flag is when the conversation is over we exchange fin packets. And reset is when it set, it means I don't want to talk to you and the conversation has reset or closed. So a lot goes on with flags and it would be good to get familiar with those as you advance, because it tells a lot about the story and it is used for a number of different reasons. Right now we'll go to window size. Now window size, it says 46, but then there's a value here in brackets, it says calculated window size, that's a larger number. Well that's because the actual value is pretty small. And in the eighties, the window size of 46 would maybe be normal. But as you see, there's a scaling factor of 128, which gives us the true value of 5888. That's so the server knows about how much data I can send to the host without overwhelming the host. It will update if the value goes up or down, for example, if the host buffer were to get too full, the number would shrink. And as it processes the data, the number would then expand. Now we'll look at the check sum. It says unverified. Well, in a lot of cases, you'll find that we're not going to try to validate that check sum and here's why. Now take a look down below, there's a blue dot, cyan dot, and that's the expert system. Now, what is it saying is that it's just chats. Okay, so if this were to be calculated, this might be incorrect and I'll show you why. We'll go to protocol preferences. And here it says validate TCP checksum if possible, now I'll select that. And now you see it says it's incorrect and it should be this number. And as you see, there's a little note that says, TCP checksum offload, what happens is it offloads to the NIC card and it comes up in correct. Down below, you see the red dot and the expert system and that's saying that there's possibly something wrong with this stream, but there isn't. So we'll go back and check that and replace that with validate the TCP checksum if possible, and that'll correct that. But if checksum is important to use because it's going from source to destination. Many things could happen. The checksum in the TCP header, monitors for error detection, not correction. Now we see the urgent pointer and then down below, then what follows as you see over here, is options and data. Well now here you see the options and in this case, that options is 12 bytes. Now we know that it needs to be a multiple of four, so let's drop this down and see what's inside. Well, you see there's the timestamp option and that's 10 bites, but it has to be 12. So you see no ops, which are kind of placeholders so that it comes up to 12 bites. And then following that then, you see HTTP, which is the data. So that's the TCP header in Wireshark, **and we've gone over all of the field values**
and hopefully you've got a little better understanding of TCP.

Recognizing the TCP connection process:



- [Instructor] Transmission Control Protocol is a connection-oriented protocol. It begins with a three-way handshake and ends by terminating the session. A three-way handshake begins by the client sending a SYN packet to the server. **That SYN packet synchronizes the sequence numbers.** If the server accepts, it responds by a synchronization acknowledgement. The client then sends a final acknowledgement and then we have a connection. A socket is created and we can begin to exchange data. Periodically, services will be refused, and then you'll see a reset. No response may indicate a firewall. You may receive an ICMP destination unreachable packet, and a code might indicate the network was unreachable or the host was unreachable, but most likely the target port was firewalled. The host might try again with another SYN packet. The connection can be terminated in one of several ways. The connection can time out and will be closed or no password was sent, so the connection was reset, or we'll see an exchange of FIN packets and this indicates a normal termination. For a normal termination, we simply see an exchange of FIN and FIN-ACK packets from the client to the server and the server back to the client, and this will close the connection.

Viewing the TCP handshake and teardown:



So let's step through, the handshake and termination. I'm here on CloudShark, and you can either follow along in CloudShark, or you can go to Export and download the file, and open it in Wireshark. I've opened this capture in Wireshark, and it's almost a perfect example of a complete conversation. I'll scroll up so you can see that it's really a small capture. But let's take a look at the flow graph. We'll go to Statistics. And we'll go to the Flow Graph. And there's no need to filter it because its entire conversation is simply with the client to the server. Now let's take a look at the top three packets. There's where you see your three-way handshake. And then down below we see an exchange of data, and then somewhere in the middle, you see quit and closing connection. So we now know it's time to terminate the session. And the last four packets are where we close the conversation. I'll close that. **And now let's take a look at the three-way handshake.**

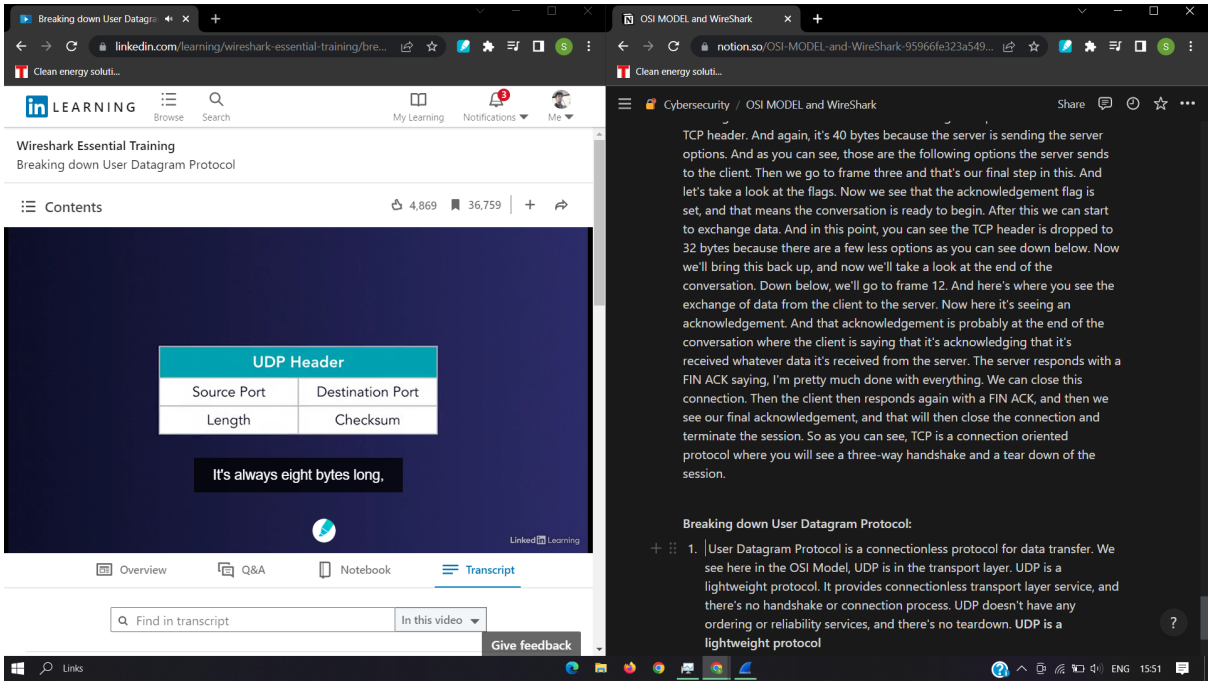
I'll scroll up here and give us a little more room where we can take a look at frame one. Let's just first start with frame one. Now when we look at frame one, we have a frame header, we have the IP header, and the TCP header. As you notice there's no data. There won't be any because it's just a handshake. No data is exchanged. You can also see then, up here in the upper right-hand corner, length equals zero. So let's take a look at the TCP header. This indicates what's going on if we take a look at the flags, and I'll drop this down, and here you see the SYN flag set. Again, that's from the client to the server, trying to initiate a conversation. Now with that first packet, going from the client to the server, comes with it some options. If I place my cursor on the TCP header and then look down below the TCP header is 40 bytes. The TCP header is 20 bytes at least, but with options it's more, now at this

point it's 40 bytes. But let's take a look at the options. Here's your options. And we'll drop this down and you can see, what it is, is the client communicating to the server what options would I like to agree upon. And so you understand, for example, the following. Client is seeing that the maximum segment size is 1,460 bytes. Also SACK is permitted and SACK is selective acknowledgment. Meaning keep sending the data, even if it's not in order, I'll put it in the correct order. Here we see timestamps and windows scaling. Now keep in mind. It has to be a multiple of four. Now with each of these, and if you look down below, you can see the size of each of those options, window scaling is three bytes. So we have to add a no op, which is essentially a placeholder, to bring that total up to 20 bytes. And then that with the TCP header gives us our 40 bytes. The server then responds with a SYN-ACK. And I'll scroll down here, and here you can see the acknowledgement flag and the SYN flag. Now keep in mind that SYN flag is set only on the first two packets of the three-way handshake. After that, you won't see it again in that conversation. Now we'll bring this up and take a look at the TCP header. And again, it's 40 bytes because the server is sending the server options. And as you can see, those are the following options the server sends to the client. Then we go to frame three and that's our final step in this. And let's take a look at the flags. Now we see that the acknowledgement flag is set, and that means the conversation is ready to begin. After this we can start to exchange data. And in this point, you can see the TCP header is dropped to 32 bytes because there are a few less options as you can see down below. Now we'll bring this back up, and now we'll take a look at the end of the conversation. Down below, we'll go to frame 12. And here's where you see the exchange of data from the client to the server. Now here it's seeing an acknowledgement. And that acknowledgement is probably at the end of the conversation where the client is saying that it's acknowledging that it's received whatever data it's received from the server. The server responds with a FIN ACK saying, I'm pretty much done with everything. We can close this connection. Then the client then responds again with a FIN ACK, and then we see our final acknowledgement, and that will then close the connection and terminate the session. So as you can see, TCP is a connection oriented protocol where you will see a three-way handshake and a tear down of the session.

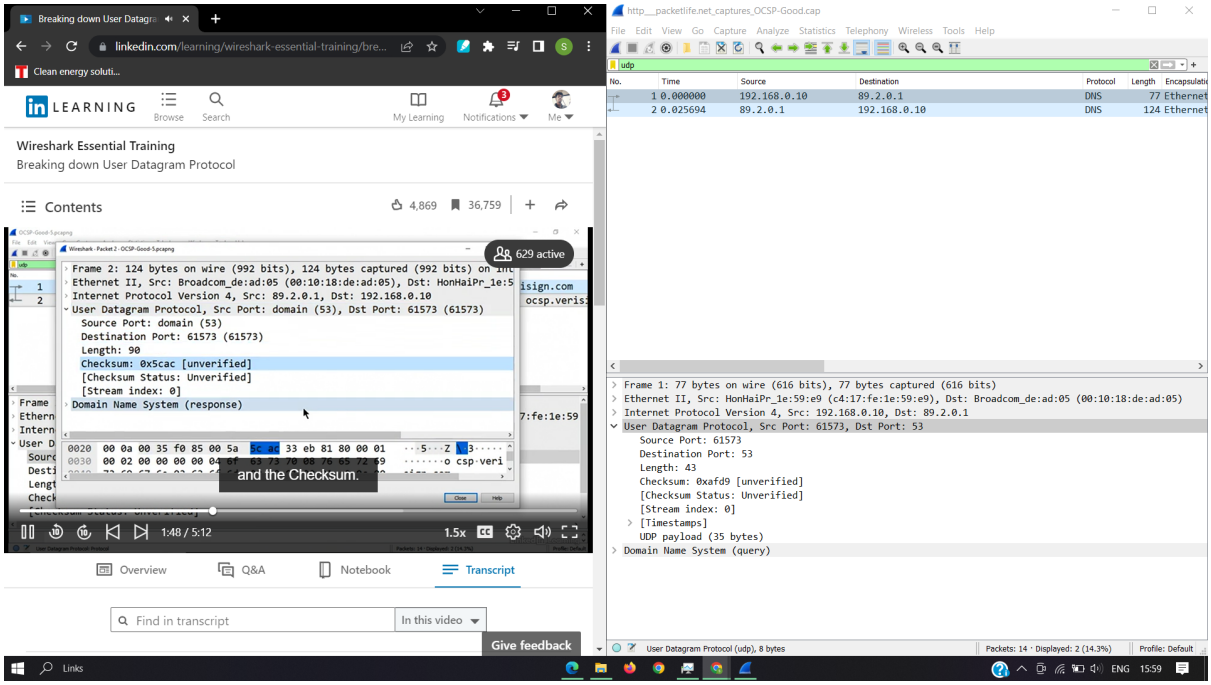
Breaking down User Datagram Protocol:

1. User Datagram Protocol is a connectionless protocol for data transfer. We see here in the OSI Model, UDP is in the transport layer. UDP is a lightweight protocol. It provides connectionless transport layer service, and there's no

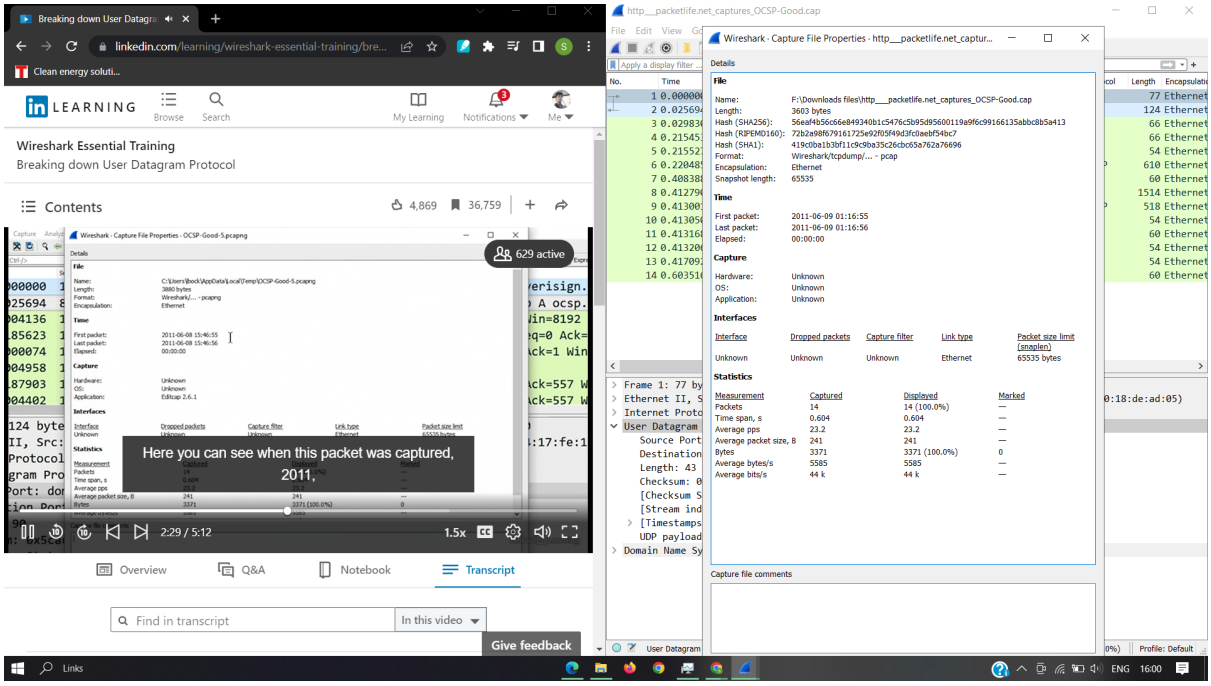
handshake or connection process. UDP doesn't have any ordering or reliability services, and there's no teardown. **UDP is a lightweight protocol** that's great for time-sensitive applications such as DNS, Voice over IP, routing information protocol, or trivial file transfer protocol



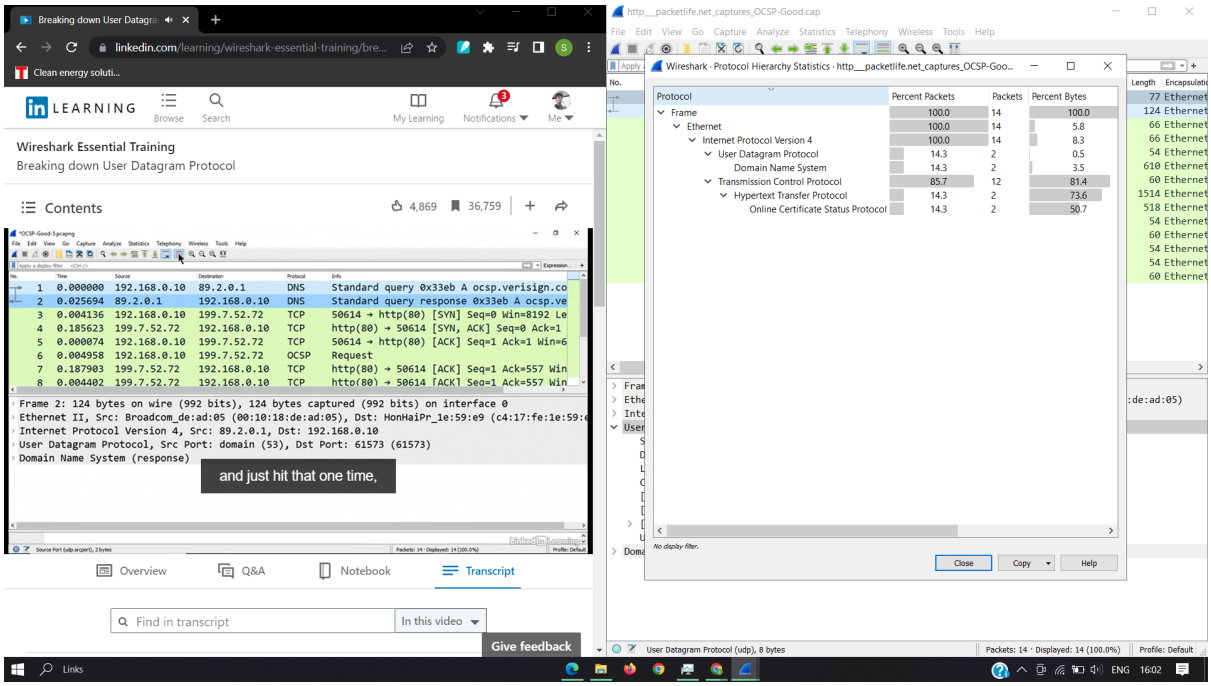
Here we see the UDP Header. It's always eight bytes long, and it includes the Source Port, Destination Port, Length, and Checksum. Now, that Checksum is used for error detection, not correction. The UDP Checksum is optional with IP version four. However, it is mandatory with IP version six primarily because IP version six doesn't have a checksum.



It's always eight bytes long, and it includes the Source Port, Destination Port, Length, and Checksum. Now, that Checksum is used for error detection, not correction. The UDP Checksum is optional with IP version four. However, it is mandatory with IP version six primarily because IP version six doesn't have a checksum. Now let's take a look at a UDP header. I'm at CloudShark, and if you'd like to follow along, you can download this file. I've opened it up in Wireshark, and what I'll do up here is go to the Display filter and I'll type udp. So there are my DNS packets, and what we'll do is just take a look at the User Datagram Protocol. I'll double-click and pop out this frame. And, again, this lightweight protocol only has four fields: the Source Port, the Destination Port, the Length, and the Checksum. **Now, that's pretty much it for UDP.**



Let's go to Statistics. If we go to Statistics, you can see there are a lot of choices, and for more advanced training, we will get into those. But for something really simple, let's just take a look at Capture File Properties. Now, when you open this, this tells us a summary about this file. Now, you can see the capture and what it knows about the hardware, the operating system. Again, what it knows, it will put in there. Here you can see when this packet was captured, 2011, **and some more information about the statistics**

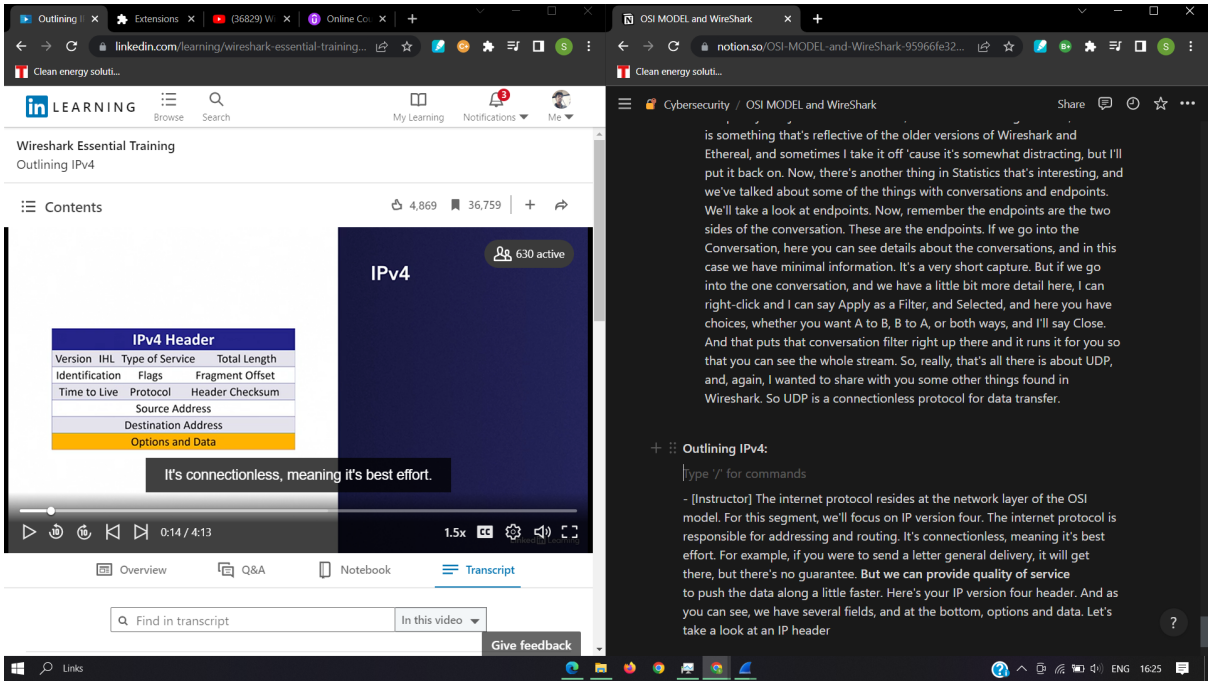


And another handy tool is a Protocol Hierarchy. Now, this is helpful if you are taking a look at protocols that are traveling on your network and you might want to see if there's anything that shouldn't be on your network. So this is a simple capture, nothing's really standing out, but, again, it's a nice handy tool. I'll close that.

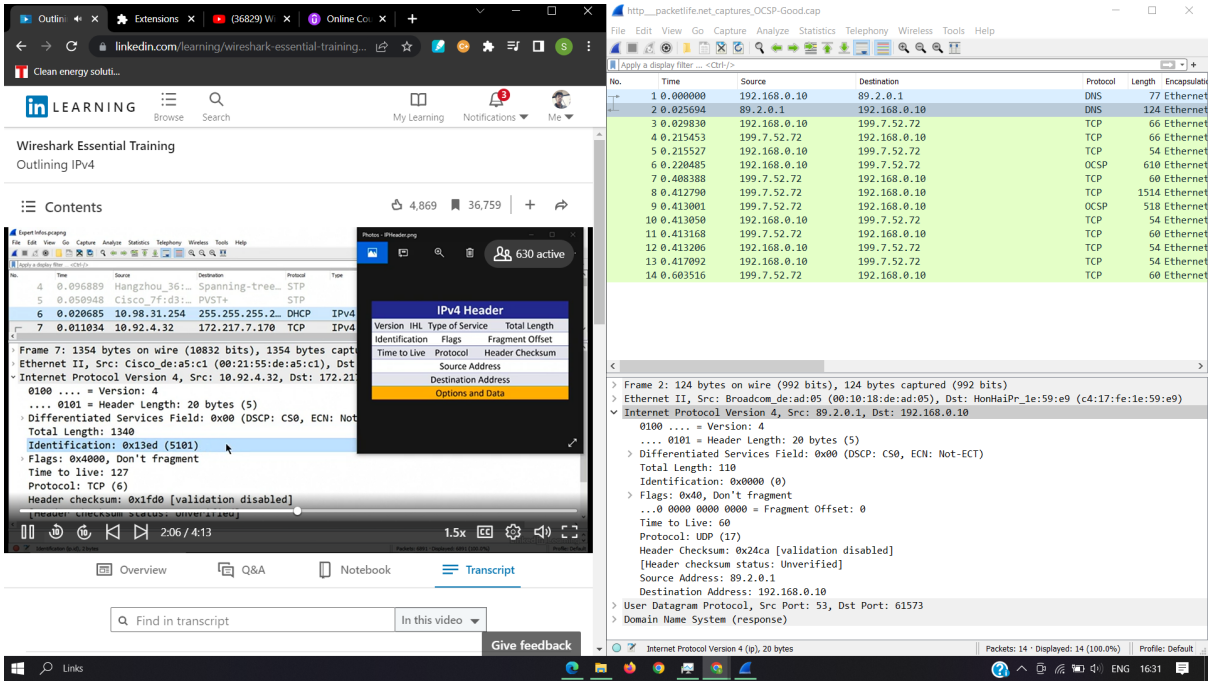
1. User Datagram Protocol is a connectionless protocol for data transfer. We see here in the OSI Model, UDP is in the transport layer. UDP is a lightweight protocol. It provides connectionless transport layer service, and there's no handshake or connection process. UDP doesn't have any ordering or reliability services, and there's no teardown. UDP is a lightweight protocol that's great for time-sensitive applications such as DNS, Voice over IP, routing information protocol, or trivial file transfer protocol. Here we see the UDP Header. It's always eight bytes long, and it includes the Source Port, Destination Port, Length, and Checksum. Now, that Checksum is used for error detection, not correction. The UDP Checksum is optional with IP version four. However, it is mandatory with IP version six primarily because IP version six doesn't have a checksum. Now let's take a look at a UDP header. I'm at CloudShark, and if you'd like to follow along, you can download this file. I've opened it up in Wireshark, and what I'll do up here is go to the Display filter and I'll type udp. So there are my DNS packets, and what we'll do is just take a look at the User Datagram Protocol. I'll double-click and pop out this frame. And, again, this lightweight protocol only has four fields: the Source Port, the Destination Port, the Length, and the Checksum. Now, that's pretty much it for UDP. So while I'm in here, let's take a look at some other things in Wireshark. Let's go to Statistics. If we go to Statistics, you can see there are a lot of choices, and for more advanced training, we will get into those. But for something really simple, let's just take a look at Capture File Properties. Now, when you open this, this tells us a summary about this file. Now, you can see the capture and what it knows about the hardware, the operating system. Again, what it knows, it will put in there. Here you can see when this packet was captured, 2011, and some more information about the statistics. Now, down below here, in the next generation, we are able to put comments in the file. Now, this would be handy if we were to share this information with coworkers. So, for example, I might want to say, "Take a look at frame 29." And I'll just say Save Comments, and then we'll close it. Now, if you take a look up here, there is an asterisk that is right by the title. That is because it knows that there's a comment, and you will want to save that so someone can reference it when they open it up. Well, we'll go back into Statistics again. I'll drop this down. And another handy tool is a Protocol Hierarchy. Now, this is helpful if you are taking a look at protocols that are traveling on your network and you might want to see if there's anything that shouldn't be on your network. So this is a simple capture, nothing's really standing out, but, again, it's a nice handy tool. I'll close that. Another thing is we talked about coloring rules. Now, they're there and in some cases, I do take them off because they're distracting. If you go up here under Telephony and just hit that one time, it takes the coloring rules off, and that is something that's

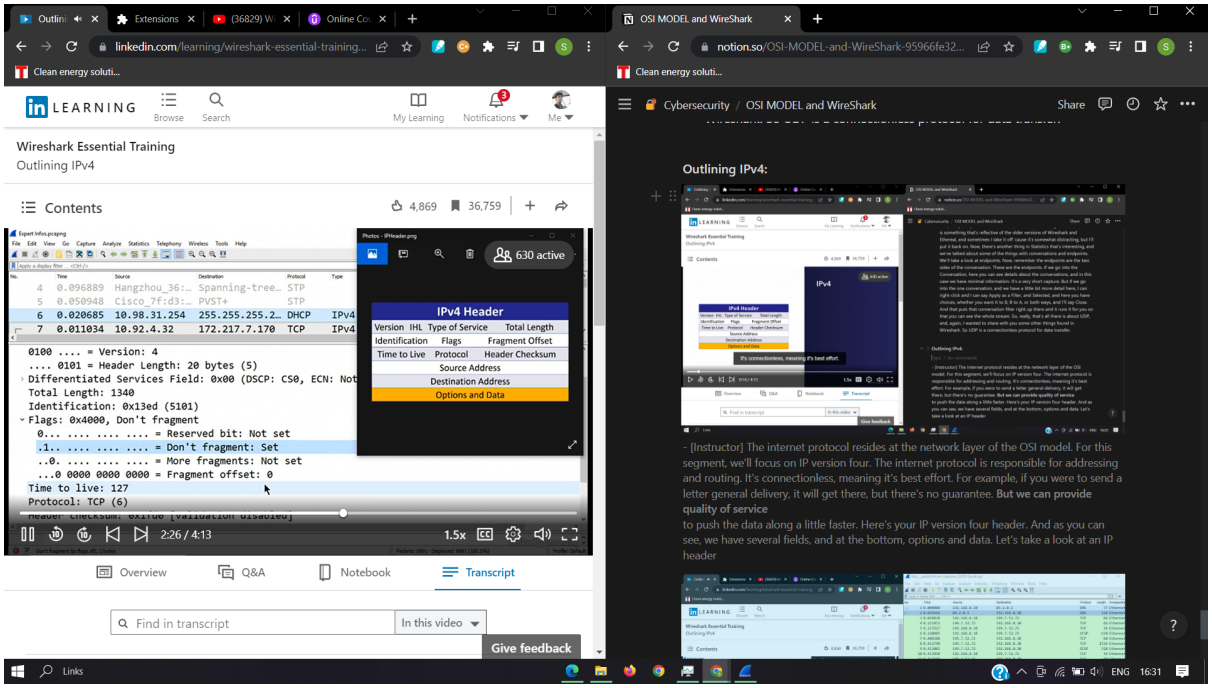
reflective of the older versions of Wireshark and Ethereal, and sometimes I take it off 'cause it's somewhat distracting, but I'll put it back on. Now, there's another thing in Statistics that's interesting, and we've talked about some of the things with conversations and endpoints. We'll take a look at endpoints. Now, remember the endpoints are the two sides of the conversation. These are the endpoints. If we go into the Conversation, here you can see details about the conversations, and in this case we have minimal information. It's a very short capture. But if we go into the one conversation, and we have a little bit more detail here, I can right-click and I can say Apply as a Filter, and Selected, and here you have choices, whether you want A to B, B to A, or both ways, and I'll say Close. And that puts that conversation filter right up there and it runs it for you so that you can see the whole stream. So, really, that's all there is about UDP, and, again, I wanted to share with you some other things found in Wireshark. So UDP is a connectionless protocol for data transfer.

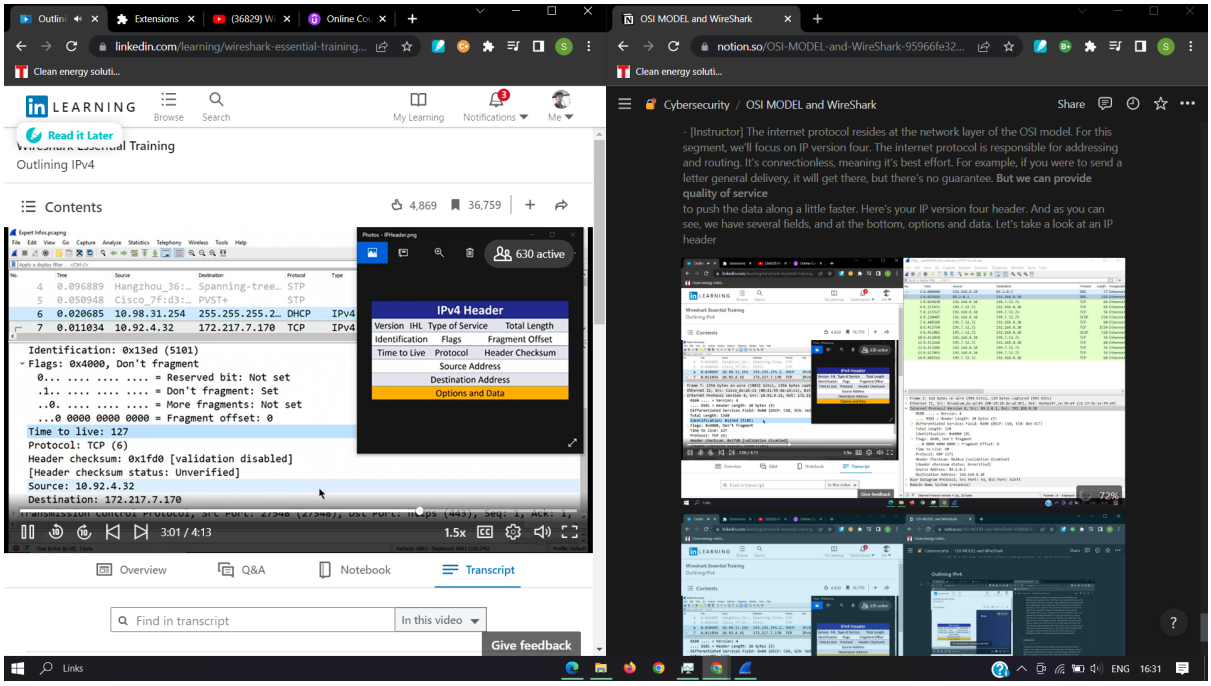
Outlining IPv4:



- [Instructor] The internet protocol resides at the network layer of the OSI model. For this segment, we'll focus on IP version four. The internet protocol is responsible for addressing and routing. It's connectionless, meaning it's best effort. For example, if you were to send a letter general delivery, it will get there, but there's no guarantee. **But we can provide quality of service** to push the data along a little faster. Here's your IP version four header. And as you can see, we have several fields, and at the bottom, options and data. Let's take a look at an IP header


linkedin.com/learning/wireshark-essential-training...
LEARNING
Wireshark Essential Training
Outlining IPv4
Contents
4,869
36,759
630 active
IPv4 Header
Version IHL Type of Service Total Length
Identification Flags Fragment Offset
Time to Live Protocol Header Checksum
Source Address
Destination Address
Options and Data
2:06 / 4:13
1.5x
Overview
Q&A
Notebook
Transcript
Find in transcript
In this video
Give feedback
http___packetlife.net_captures_OCSP-Good.cap
Frame 2: 124 bytes on wire (992 bits), 124 bytes captured (992 bits)
Ethernet II, Src: Broadcom_de:ad:05 (00:10:18:de:ad:05), Dst: HonHaiPr_1e:59:e9 (c4:17:fe:1e:59:e9)
Internet Protocol Version 4, Src: 89.2.0.1, Dst: 192.168.0.10
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 110
Identification: 0x0000 (0)
Flags: 0x40, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 60
Protocol: UDP (17)
Header Checksum: 0x24ca [validation disabled]
[Header checksum status: Unverified]
Source Address: 89.2.0.1
Destination Address: 192.168.0.10
User Datagram Protocol, Src Port: 53, Dst Port: 61573
Domain Name System (response)

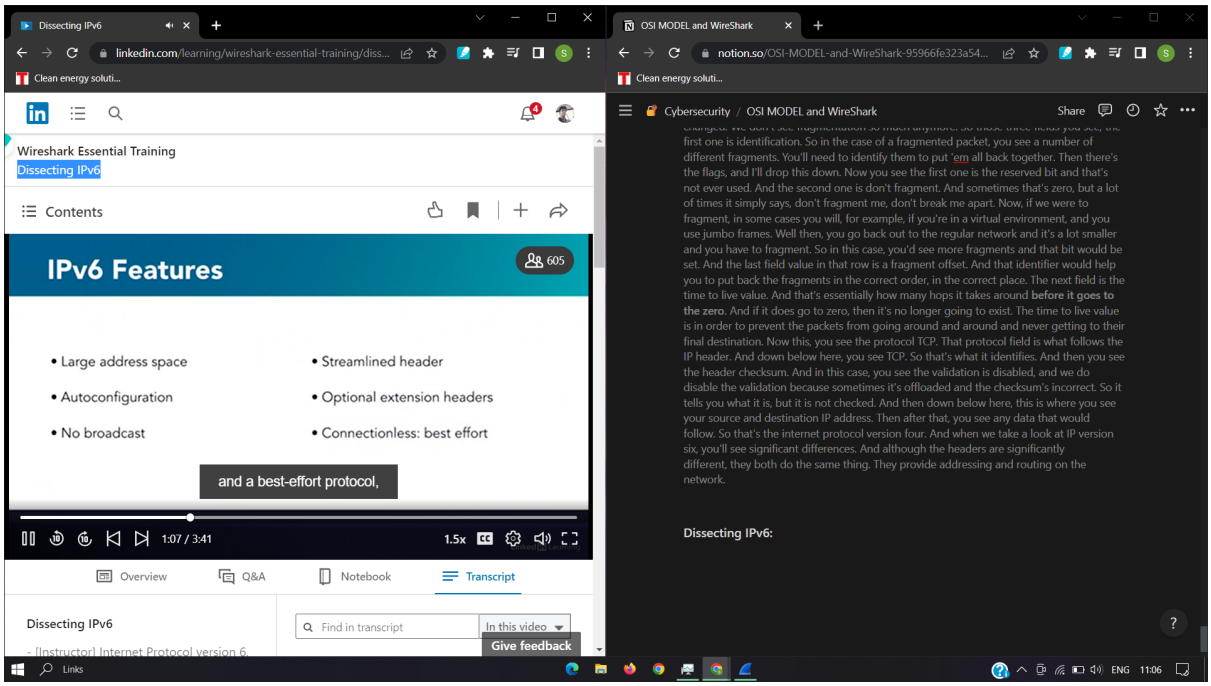

linkedin.com/learning/wireshark-essential-training...
notion.so/OSI-MODEL-and-WireShark-95966fe32...
Cybersecurity / OSI MODEL and WireShark
Share
Outlining IPv4:
- [Instructor] The internet protocol resides at the network layer of the OSI model. For this segment, we'll focus on IP version four. The internet protocol is responsible for addressing and routing. It's connectionless, meaning it's best effort. For example, if you were to send a letter general delivery, it will get there, but there's no guarantee. But we can provide quality of service
to push the data along a little faster. Here's your IP version four header. And as you can see, we have several fields, and at the bottom, options and data. Let's take a look at an IP header

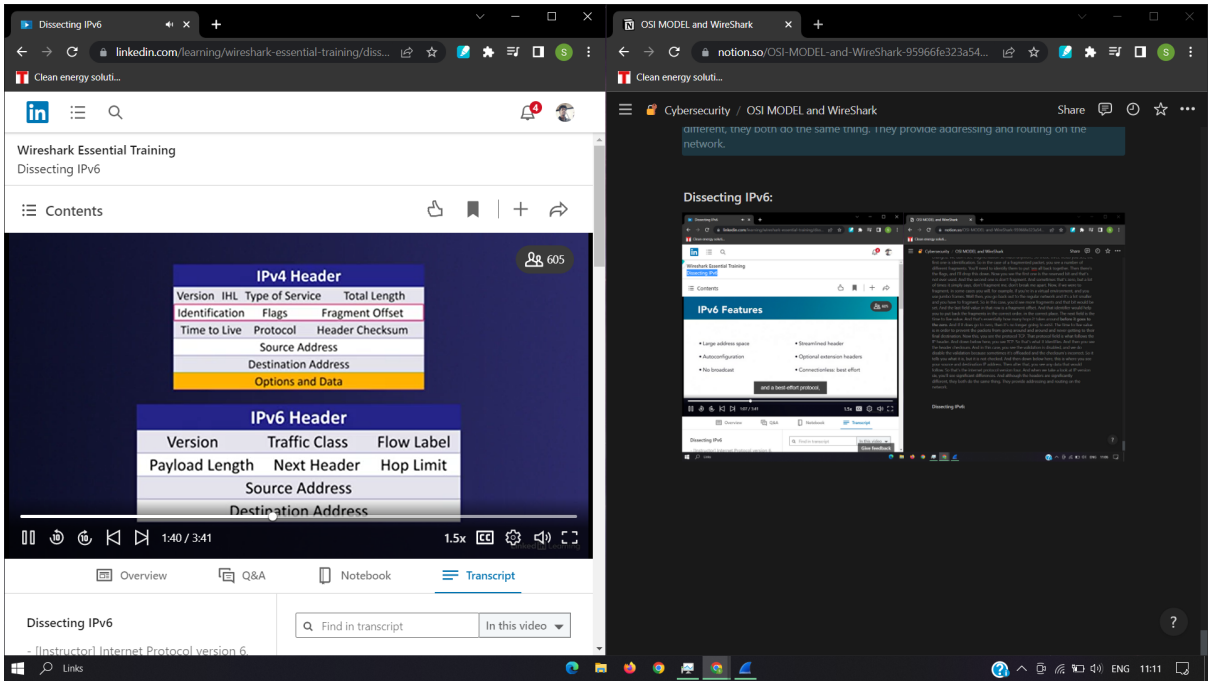


I'm in Wireshark and I've expanded one of the frames, and we'll take a look at that header. And on the right-hand side, you could follow along. All right, the first field is simply version. And in this case, it is version four. Not much there to show. This tells us the header length, and it is 20 bytes. And I can confirm if we go and place my cursor up on the top, and then look down below, that it is 20 bytes. This field value here is called differentiated services. That relates to what's called the type of service. Now keep in mind, zero is best effort. It's not prioritized, but in certain cases, we want the data to be prioritized and moved along a little faster. The total length is the header and any data that follows. And here we see 1340. Now the next three fields relate to fragmentation. Those are identification, flags, and fragment offset. The specification stated that IP was responsible for addressing and fragmentation. Now that was a long time ago, but things have changed. We don't see fragmentation so much anymore. So those three fields you see, the first one is identification. So in the case of a fragmented packet, you see a number of different fragments. You'll need to identify them to put 'em all back together. Then there's the flags, and I'll drop this down. Now you see the first one is the reserved bit and that's not ever used. And the second one is don't fragment. And sometimes that's zero, but a lot of times it simply says, don't fragment me, don't break me apart. Now, if we were to fragment, in some cases you will, for example, if you're in a virtual environment, and you use jumbo frames. Well then, you go back out to the regular network and it's a lot smaller and you have to fragment. So in this case, you'd see more fragments and that bit would be set. And the last field value in that row is a fragment offset. And that identifier would help you to put back the fragments in the correct order, in the correct place. The next field is the time to live value. And that's essentially how many hops it takes around **before it goes to the zero.** And if it does go to zero, then it's no longer going to exist. The time to live value is in order to prevent the packets from going around and around and never getting to their final destination. Now this, you see the protocol TCP. That protocol field is what follows the IP header. And down below here, you see TCP. So that's what it identifies. And then you see the header checksum. And in this case, you see the validation is disabled, and we do disable the validation because sometimes it's offloaded and the checksum's incorrect. So it tells you what it is, but it is not checked. And then down below here, this is where you see your source and destination IP address. Then after that, you see any data that would follow. So that's the internet protocol version four. And when we take a look at IP version

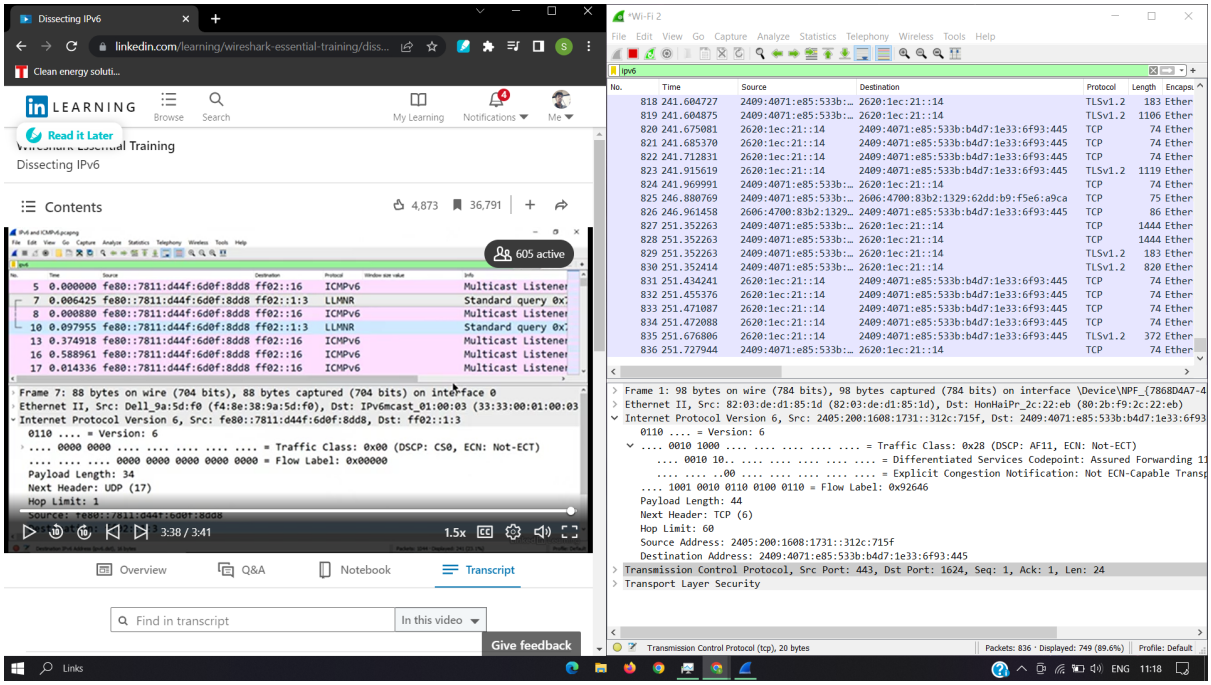
six, you'll see significant differences. And although the headers are significantly different, they both do the same thing. They provide addressing and routing on the network.

Dissecting IPv6:



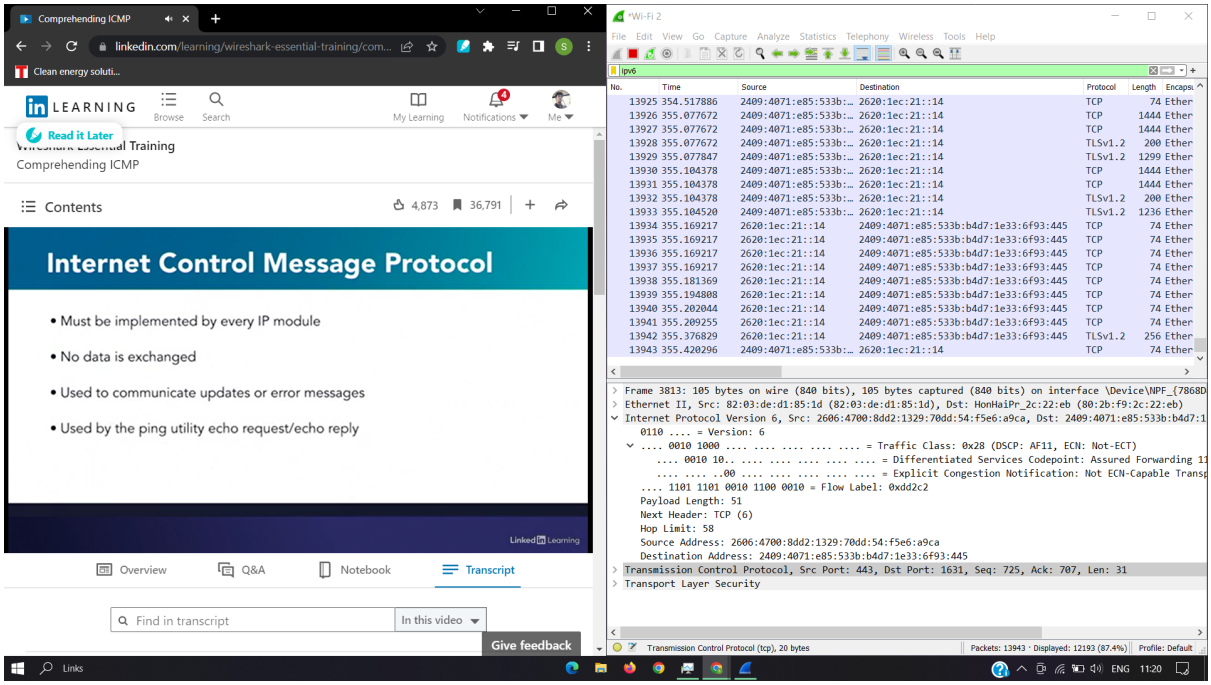


Let's compare the IP version 4 header with IP version 6 header. Now we'll take a look at the field values. First, the Internet Header Length doesn't exist in an IP version 6 header. Those three fields, the identification, flags, and fragment offset had to do with fragmentation. IP version 6 seeks not to fragment, so there are no fields that deal with fragmentation. You can fragment, but you'll need to use an extension header. The time to live value still exists in an IP version 6 header, however it's called, hop limit. And there is no header checksum in an IP version 6 header.

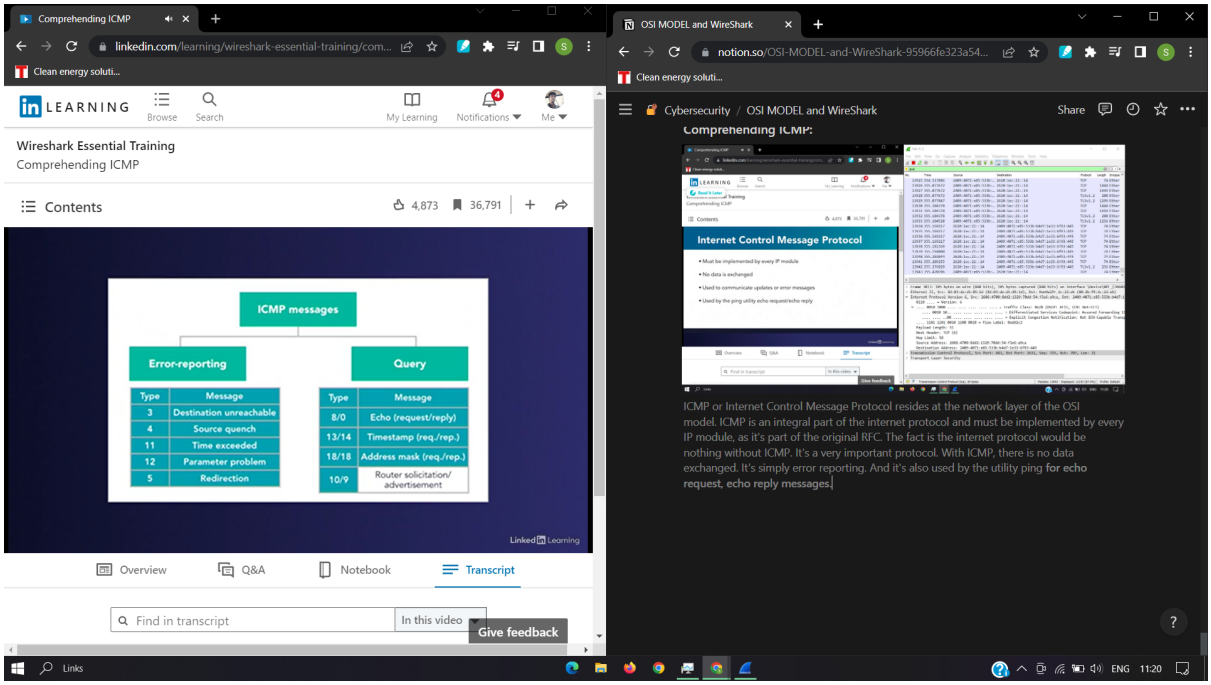


I've opened this packet capture and I filtered by IP version 6. Now let's take a look at the field values. Okay, I'll put this up at the top so you can take a look at it. And we'll go through each of those field values. There aren't a lot of them, but let's take a look at, the first is the version. It says version 6. Not much to see there, we know that that's what the version is. But the next field value is called traffic class. This has to do with quality of service. The default is zero, meaning best effort. And then we see the flow label. The value is zero and that's because the flow label is experimental and not really to be used. Now we look at the payload length. The payload length is any headers or data that follows the IP header. So let's take a look. The UDP header is eight bytes, and the Link-Local Multicast Name Resolution protocol, here we see 26 bytes. So 26 plus eight equals 34. And there we see UDP that follows the IP header, and that's what follows it. And then we see the hop limit. The last two field values are the source and the destination address. In general, you're going to see a lot more IP version 6 on our networks, **so you'll need to get familiar with IP version 6.**

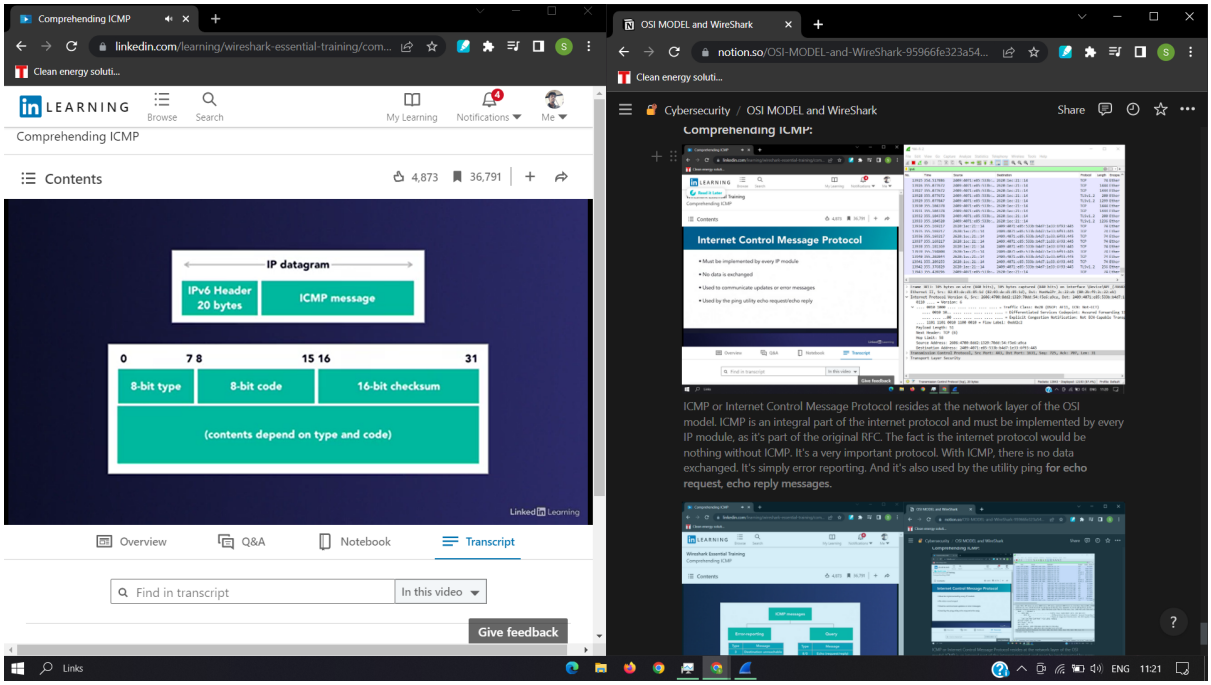
Comprehending ICMP:

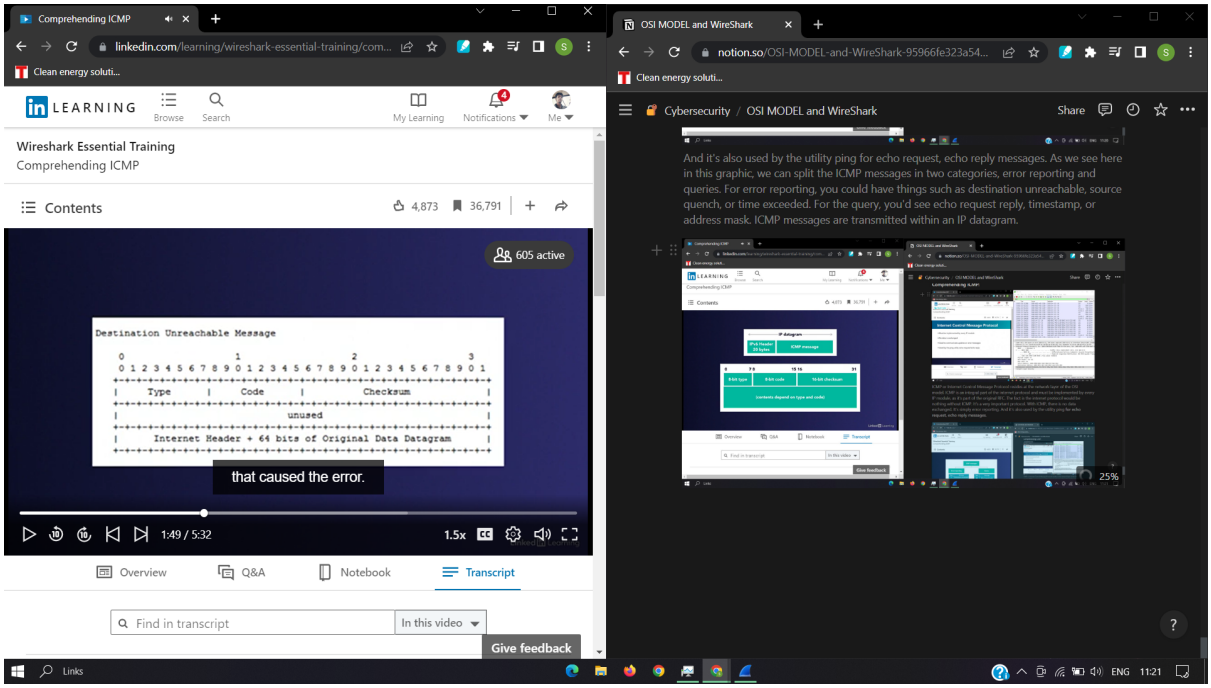


ICMP or Internet Control Message Protocol resides at the network layer of the OSI model. ICMP is an integral part of the internet protocol and must be implemented by every IP module, as it's part of the original RFC. The fact is the internet protocol would be nothing without ICMP. It's a very important protocol. With ICMP, there is no data exchanged. It's simply error reporting. And it's also used by the utility ping **for echo request, echo reply messages.**

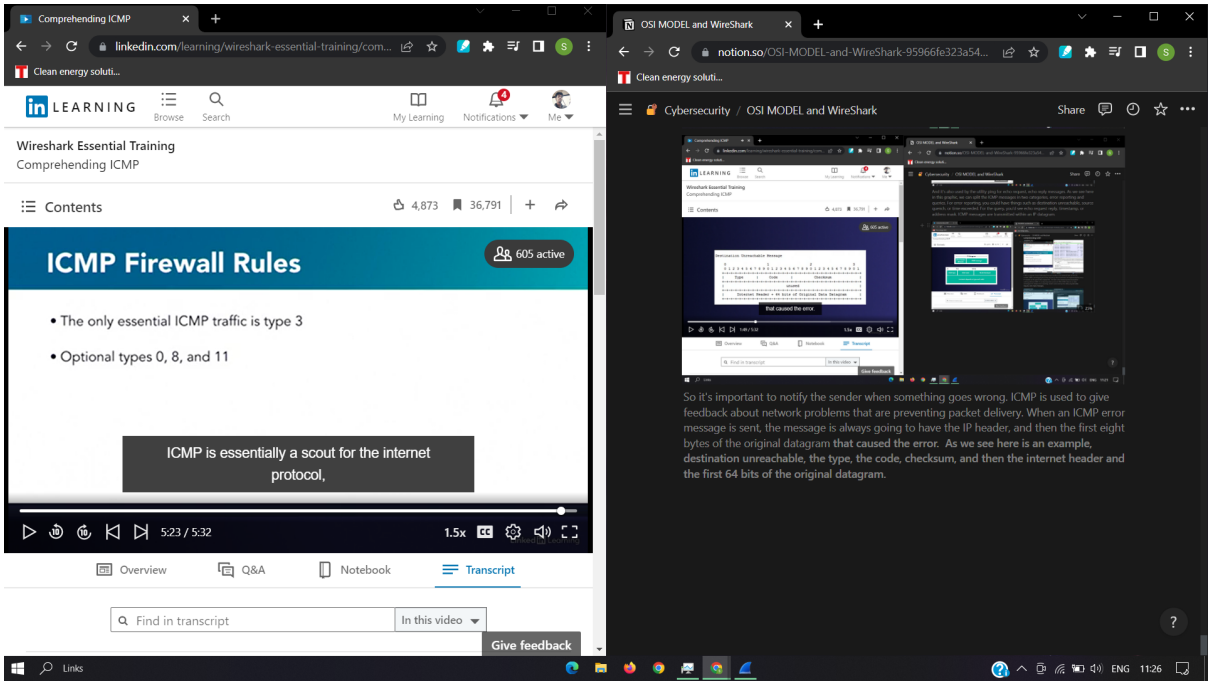


And it's also used by the utility ping for echo request, echo reply messages. As we see here in this graphic, we can split the ICMP messages in two categories, error reporting and queries. For error reporting, you could have things such as destination unreachable, source quench, or time exceeded. For the query, you'd see echo request reply, timestamp, or address mask. ICMP messages are transmitted within an IP datagram.





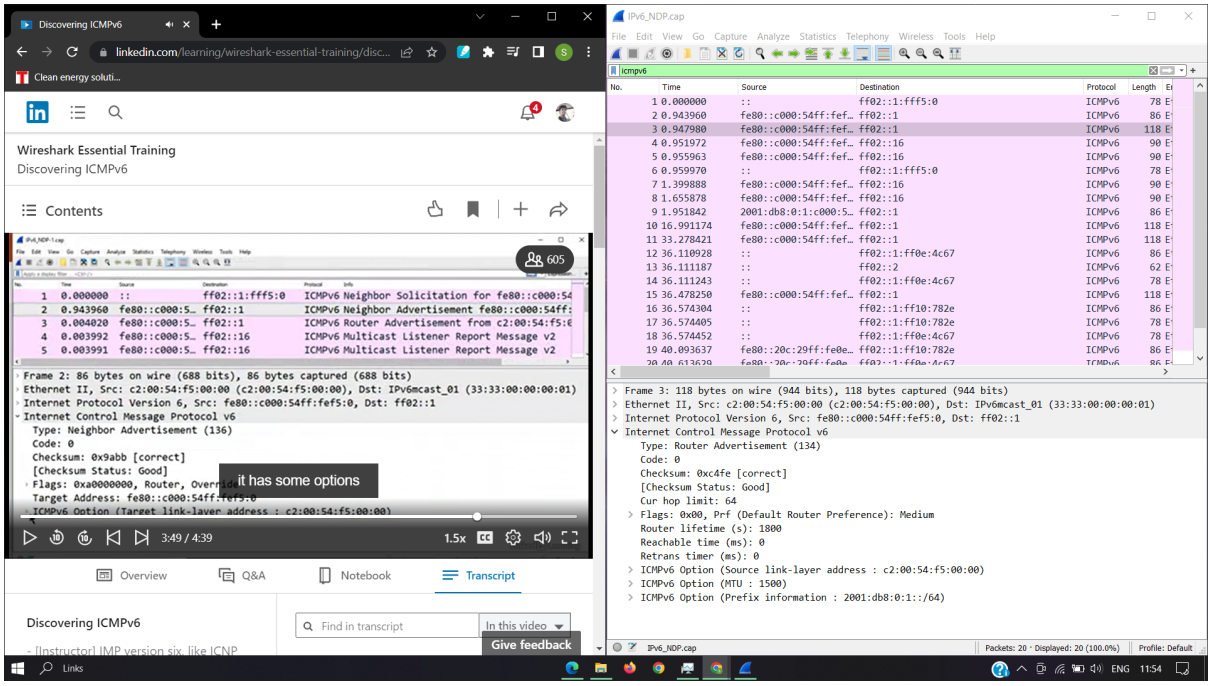
So it's important to notify the sender when something goes wrong. ICMP is used to give feedback about network problems that are preventing packet delivery. When an ICMP error message is sent, the message is always going to have the IP header, and then the first eight bytes of the original datagram **that caused the error. As we see here is an example, destination unreachable, the type, the code, checksum, and then the internet header and the first 64 bits of the original datagram.**



ICMP is a powerful protocol, but we have to understand that it's also used in malicious ways. For example, ICMP is used in reconnaissance by Kali Linux. It could be used to do reconnaissance, to do a ping sweep, or to evade firewall rules. So the question is, what type of ICMP packets are are you going to allow? The only really essential ICMP traffic is Type 3, that's your destination unreachable, and then the codes that fall within that. The others are optional, the types 0, 8, and 11. Those would be optional dependent on whether you would like to allow those on your network or not. So Internet Control Message Protocol or ICMP is used by routers, intermediary devices, or hosts **to communicate updates or error information.**

ICMP is essentially a scout for the internet protocol, making sure the route is clear and communicating any errors.

Discovering ICMPv6:



IMP version six, like ICNP version four, resides at the network layer of the OSI model. ICMP version six is a very important protocol. It must be implemented by every IP module. Now we've said this before, but now let's take a look, we'll go to the source. I'm at RFC 4443. And then right here, it states ICMP version six is an integral part of IP version six, and the base protocol must be fully implemented by every IP version six node. Now just like ICMP version four, node data is exchanged. It's used to communicate updates or error messages, and it's also used by the ping utility echo request/echo reply. Now here we see the IP datagram where we see the IP version six header followed by the ICMP version six message. Now that ICMP message is just like IP version four, where you see the type, the code, the checksum, and then the contents will depend on the type and the code. But unlike ICMP version four, ICMP version six has a lot bigger role. Keep in mind, IP version six does not have any AARP or IGMP. As a result ICMP version six assumes those two roles. The messages include error messages, informational messages, neighbor discovery messages, including the neighbor discovery protocol and group membership messages. That includes multicast listener discovery messages. We also of course, have error messages that include destination unreachable, packet too big, time exceeded and parameter problems. The informational messages include, echo request and echo reply. IMET packet captures on packet life, and you can follow along if you like. We'll take two simple examples. ICMP version six, echos and IP version six ndp.cap. Now I've opened up the echos and that's just a test for reachability. And as you can see, we have our ping request, ping reply, pretty simple. And down below you can take a look, you have your IP header and then your ICMP version six header, and then the information within it. Well, now sometimes you might want to just follow what is happening there with ICMP and if you right click and we say follow, well, we can't because there's no transport layer protocol. This is an NSSL stream or an HTTP stream, but we can do this. Let's go to statistics in flow graph. You can open the flow graph and there you can see the test for reachability with the request and the replies. And then you see the neighbor discovery protocol, and there's a lot of activity going on within this. We see the multicast listener report, neighbor solicitation, and neighbor advertisement. Again, I'll take a look in statistics in the flow graph. And here you can see the flow as the traffic is going back and forth, communicating to other devices on the network. And if we did take a look at the neighbor advertisement and we'll blow out here, ICMP version six, **it has some options**

and it's telling a little bit about what's happening. Here you see the link layer address. Here we see a router advertisement, and there are some options there as well. And we'll drop this down and scroll down. And with this, you see the link layer or MAC address and MTU that's 1500 and then prefix information. So ICMP version six, like ICMP version four, is a very important protocol. But unlike ICMP version four, it assumes a couple of different roles. And that is because IP version six doesn't use AARP or IGMP, and that is taken care of by ICMP version six.

Solution: Evaluating a pcap:

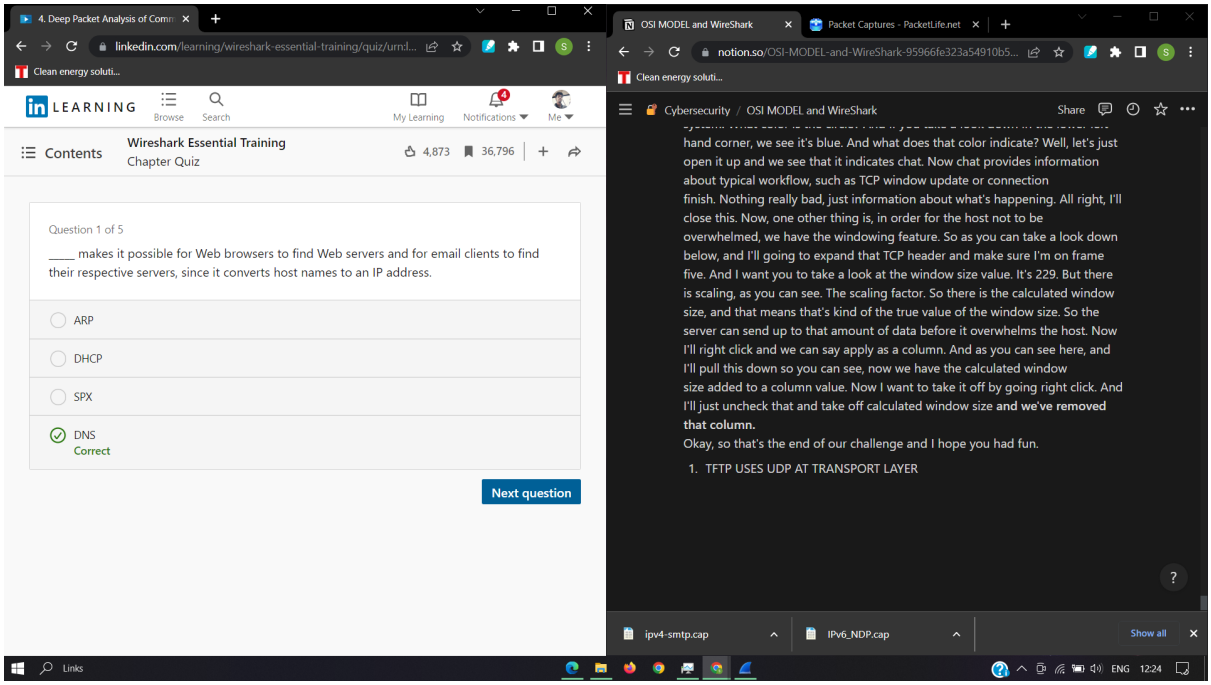
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bd687cfe-f398-4540-aa95-1ca861594dc9/03_10_Challenge.pdf

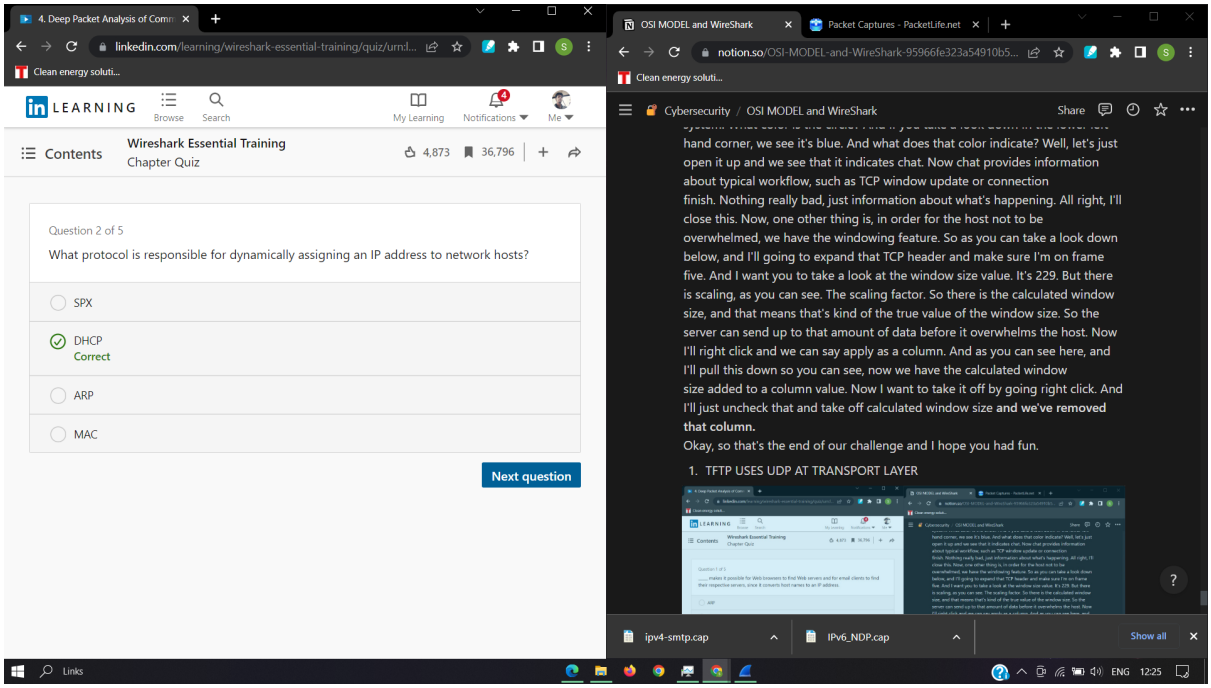
Go to the first link and select **ipv4-smtp.cap and download it and open it in wireshark**

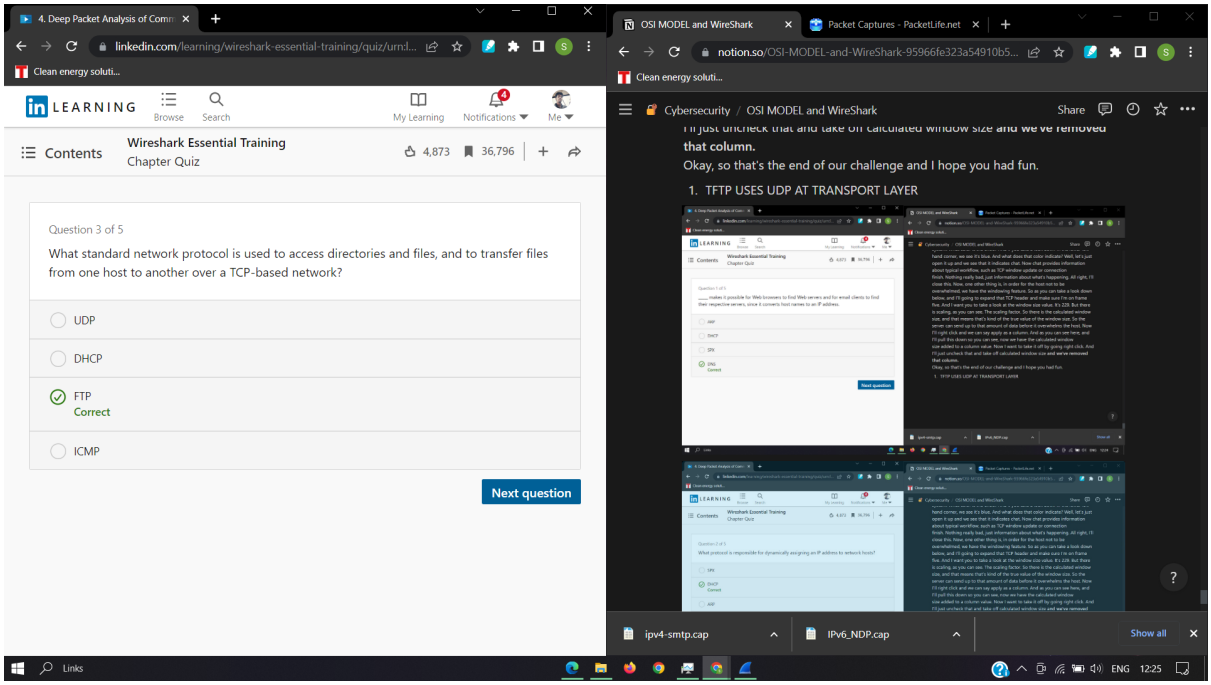
Okay, how'd you do? You'll want to have the instructions handy so that you can follow along. Now I have the packet capture open and the first question is click on statistics and protocol hierarchy. Now we go to statistics and there's a lot of things that we can choose from, but what we want is protocol hierarchy. And we'll select that. And this tells us about the protocols that are within this packet capture. And it wants to know how many simple mail transfer protocol packets there are. And here we can see there are five. Now we'll close the window. Now the next thing is we want to take a look at the conversations. Click on statistics and then conversations. But before I go there, I want to show you the next line is endpoints. Conversations is communication between two endpoints. So let's throw our focus on that. Now once you open it, and as you can see that the tabs are set up so we can look at ethernet, IP version four, IP version six, TCP, or UDP. So we wanted to take a look at the ethernet conversation and then down below it says, name resolution. Check and uncheck the name resolution box and what are the results? So we'll check it and then we'll uncheck it. Now, when I check it, we see some writing. Now that represents the first six bytes of the Mac address. In the frame header, the addressing we use is the Mac address. And the first six bytes represents the organizational unique identifier. And as you see, we see Hewlett Packard and VMware. Now using this name resolution won't hurt anything because it comes from a file. Now we'll close that. Now we want to find the first TCP handshake. And what I want to do is we take a look at this. And as we see in the right hand side, the intelligence scroll bar, but for this exercise, I want to take off the coloring roles and we'll just select right here. All right. So now we see that frame one, two, and three, or packets one, two, and three are the three-way handshake. Now the three-way handshake is what's going to establish that conversation. So the question is, number four, what is the IP address of the host that

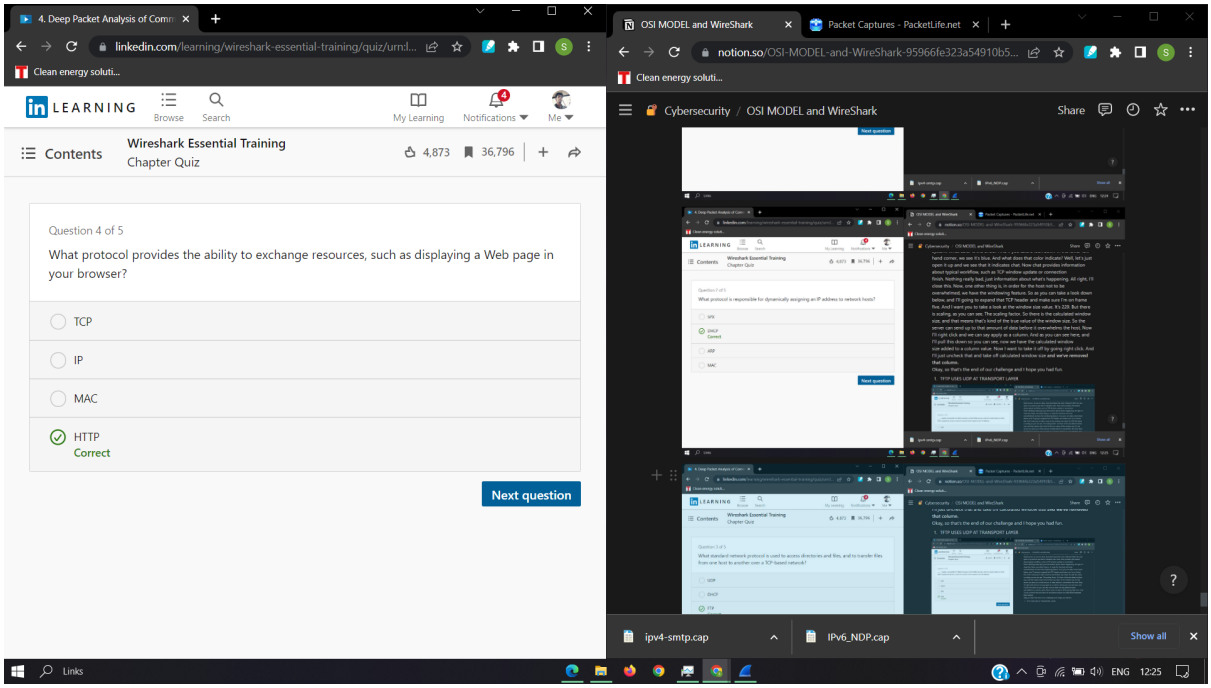
started the handshake? All right, so we'll go down below and you can see here, it's up above, under source, but I'll take a look at it down below. And here we see the source is 192.168.20.70. I'll pull this up. And we'll place my cursor on that TCP header. And I want you to take a look down below. That shows us the size of the TCP header on frame one. Now the destination port will indicate what application the host wants to use on the serving host. So let's take a look here and we'll drop down the TCP header. And here we see what is the client's port. And this is that temporary port that just throws it out there and says, when you return the data, return it to port 54557. So that's the client's port, 54557. Now the question is, what application does the host want to use on the server? While the destination port is 25, which is related to SMTP, which is an email server. All right, we'll pull this back up. And the next thing is, number seven, go to frame five and expand the IP header. We'll go to frame five and then we'll expand the IP header. And the question is, what is the time to live value? And I'll place my cursor there. Time to live for this frame is 64. I'll pull that up. Number eight says, in the lower left corner is the expert system. What color is the circle? And if you take a look down in the lower left-hand corner, we see it's blue. And what does that color indicate? Well, let's just open it up and we see that it indicates chat. Now chat provides information about typical workflow, such as TCP window update or connection finish. Nothing really bad, just information about what's happening. All right, I'll close this. Now, one other thing is, in order for the host not to be overwhelmed, we have the windowing feature. So as you can take a look down below, and I'll going to expand that TCP header and make sure I'm on frame five. And I want you to take a look at the window size value. It's 229. But there is scaling, as you can see. The scaling factor. So there is the calculated window size, and that means that's kind of the true value of the window size. So the server can send up to that amount of data before it overwhelms the host. Now I'll right click and we can say apply as a column. And as you can see here, and I'll pull this down so you can see, now we have the calculated window size added to a column value. Now I want to take it off by going right click. And I'll just uncheck that and take off calculated window size **and we've removed that column.**
Okay, so that's the end of our challenge and I hope you had fun.

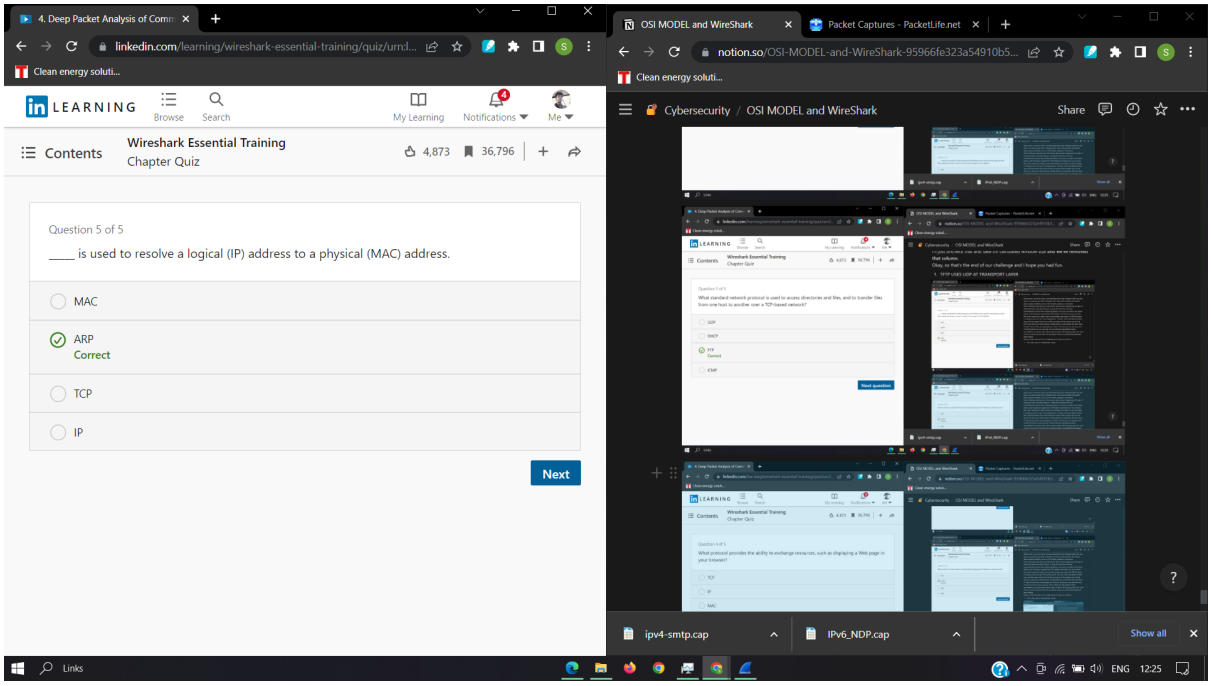
1. TFTP USES UDP AT TRANSPORT LAYER


4. Deep Packet Analysis of Comm
linkedin.com/learning/wireshark-essential-training/quiz/urn:l...
Clean energy soluti...
LEARNING
Browse
Search
My Learning
Notifications
Me
Contents
Wireshark Essential Training
Chapter Quiz
4,873
36,796
Question 1 of 5
____ makes it possible for Web browsers to find Web servers and for email clients to find their respective servers, since it converts host names to an IP address.
ARP
DHCP
SPX
DNS
Correct
Next question
OSI MODEL and WireShark
Packet Captures - PacketLife.net
notion.so/OSI-MODEL-and-WireShark-95966fe323a54910b5...
Cybersecurity / OSI MODEL and WireShark
Share
hand corner, we see it's blue. And what does that color indicate? Well, let's just open it up and we see that it indicates chat. Now chat provides information about typical workflow, such as TCP window update or connection finish. Nothing really bad, just information about what's happening. All right, I'll close this. Now, one other thing is, in order for the host not to be overwhelmed, we have the windowing feature. So as you can take a look down below, and I'll going to expand that TCP header and make sure I'm on frame five. And I want you to take a look at the window size value. It's 229. But there is scaling, as you can see. The scaling factor. So there is the calculated window size, and that means that's kind of the true value of the window size. So the server can send up to that amount of data before it overwhelms the host. Now I'll right click and we can say apply as a column. And as you can see here, and I'll pull this down so you can see, now we have the calculated window size added to a column value. Now I want to take it off by going right click. And I'll just uncheck that and take off calculated window size and we've removed that column.
Okay, so that's the end of our challenge and I hope you had fun.
1. TFTP USES UDP AT TRANSPORT LAYER
ipv4-smtp.cap
IPv6_NDP.cap
Show all
Links
ENG
12:24


4. Deep Packet Analysis of Comm
linkedin.com/learning/wireshark-essential-training/quiz/urn:l...
Clean energy soluti...
LEARNING
Browse
Search
My Learning
Notifications
Me
Contents
Wireshark Essential Training
Chapter Quiz
4,873
36,796
Question 2 of 5
What protocol is responsible for dynamically assigning an IP address to network hosts?
SPX
DHCP
Correct
ARP
MAC
Next question
OSI MODEL and WireShark
Packet Captures - PacketLife.net
notion.so/OSI-MODEL-and-WireShark-95966fe323a54910b5...
Cybersecurity / OSI MODEL and WireShark
Share
hand corner, we see it's blue. And what does that color indicate? Well, let's just open it up and we see that it indicates chat. Now chat provides information about typical workflow, such as TCP window update or connection finish. Nothing really bad, just information about what's happening. All right, I'll close this. Now, one other thing is, in order for the host not to be overwhelmed, we have the windowing feature. So as you can take a look down below, and I'll going to expand that TCP header and make sure I'm on frame five. And I want you to take a look at the window size value. It's 229. But there is scaling, as you can see. The scaling factor. So there is the calculated window size, and that means that's kind of the true value of the window size. So the server can send up to that amount of data before it overwhelms the host. Now I'll right click and we can say apply as a column. And as you can see here, and I'll pull this down so you can see, now we have the calculated window size added to a column value. Now I want to take it off by going right click. And I'll just uncheck that and take off calculated window size and we've removed that column.
Okay, so that's the end of our challenge and I hope you had fun.
1. TFTP USES UDP AT TRANSPORT LAYER
ipv4-smtp.cap
IPv6_NDP.cap
Show all
Links
ENG
12:25


4. Deep Packet Analysis of Comm
linkedin.com/learning/wireshark-essential-training/quiz/urn:l...
Clean energy soluti...
LEARNING
Browse
Search
My Learning
Notifications
Me
Contents
Wireshark Essential Training
Chapter Quiz
4,873
36,796
Question 3 of 5
What standard network protocol is used to access directories and files, and to transfer files from one host to another over a TCP-based network?
UDP
DHCP
FTP
Correct
ICMP
Next question
OSI MODEL and WireShark
Packet Captures - PacketLife.net
notion.so/OSI-MODEL-and-WireShark-95966fe323a54910b5...
Cybersecurity / OSI MODEL and WireShark
Share
that column.
Okay, so that's the end of our challenge and I hope you had fun.
1. TFTP USES UDP AT TRANSPORT LAYER
ipv4-smtp.cap
IPv6_NDP.cap
Show all
Links
ENG
12:25


4. Deep Packet Analysis of Comm
linkedin.com/learning/wireshark-essential-training/quiz/urn:l...
Clean energy soluti...
LEARNING
Browse
Search
My Learning
Notifications
Me
Contents
Wireshark Essential Training
Chapter Quiz
4,873
36,796
Question 4 of 5
What protocol provides the ability to exchange resources, such as displaying a Web page in your browser?
TCP
IP
MAC
HTTP
Correct
Next question
OSI MODEL and WireShark
Packet Captures - PacketLife.net
notion.so/OSI-MODEL-and-WireShark-95966fe323a54910b5...
Cybersecurity / OSI MODEL and WireShark
Share
ipv4-smtp.cap
IPv6_NDP.cap
Show all
Links
ENG
12:25



OSI MODEL and WireShark 2