Troubleshooting TCP/IP Networks with Wireshark

Tajul Azhar bin Mohd Tajul Ariffin

who.am.i

- Lecturer, Polytechnic Mersing, Johor
- CTO, Cyber Range Academy
- Student – Ph.D – second semester on Information Security – focus on Cyber Range and Incident Handling
- Speaker
  - Sharkfest US ‘19
  - Elastic Malaysia User Community
  - Durian Conference ‘17
  - Joomla! Day Malaysia 2016-2017
  - Malaysia Open Source Conference (MOSC)
  - AWS Cloud – AWS Educate
- Certified Security Application Programming (CSAP) – Global ACE
- Application Security Forensic – Ofisgate Academy

Objectives

- Top 10 reasons for network performance complaints
- Place the analyzer properly for traffic capture on a variety of network types
- Capture packets on wired and wireless networks
- Configure Wireshark for best performance and non-intrusive analysis
- Navigate through, split, and work with large traffic files
- Use time values to identify network performance problems
- Create statistical charts and graphs to pinpoint performance issues
- Filter out traffic for more efficient troubleshooting and analysis
- Customize Wireshark coloring to focus on network problems faster
- Use Wireshark's Expert System to understand various traffic problems
- Use the TCP/IP Resolution Flowchart to identify possible communication faults
- Analyze normal/abnormal Domain Name System (DNS) traffic
- Analyze normal/abnormal Address Resolution Protocol (ARP) traffic
- Analyze normal/abnormal Internet Protocol v4 (IPv4) traffic

TOGETHER WE TRANSFORM

Session 1 - Introduction to Network Analysis and Wireshark

- TCP/IP Analysis Checklist
- Top Causes of Performance Problems
- Get the Latest Version of Wireshark
- Capturing Traffic
- Opening Trace Files
- Processing Packets
- The GT Interface Overview
- Using Linked Panes
- The Icon Toolbar
- Master the Intelligent Scrollbar
- The Changing Status Bar
- Right-Click Functionality
- General Analyst Resources

Session 2 - Learn Capture Methods and Use Capture Filters

- Analyze Switched Networks
- Walk-Through a Sample SPAN Configuration
- Analyze Full-Duplex Links with a Network TAP
- Analyze Wireless Networks
- USB Capture
- Initial Analyzing Placement
- Remote Capture Techniques
- Available Capture Interfaces
- Save Directly to Disk
- Capture File Configurations
- Limit Your Capture with Capture Filters
- Examine Key Capture Filters

Session 3 - Customize for Efficiency: Configure Your Global Preferences

- First Step: Create a Troubleshooting Profile
- Customize the User Interface
- Add Custom Columns for the Packet List Pane
- Set Your Global Capture Preferences
- Define Name Resolution Preferences
- Configure Individual Protocol Preferences

Session 3 - Navigate Quickly and Focus Faster with Coloring Techniques

- Move Around Quickly: Navigation Techniques
- Find a Packet Based on Various Characteristics
- Build Permanent Coloring Rules
- Identify a Coloring Source
- Use the Intelligent Scrollbar with Custom Coloring Rules
- Apply Temporary Coloring
- Mark Packets of Interest

Session 4 - Spot Network and Application Issues with Time Values and Summaries

- Examine the Delta Time (End-of-Packet to End-of-Packet)
- Set a Time Reference
- Compare Timestamp Values
- Compare Timestamps of Filtered Traffic
- Enable and Use TCP Conversation Timestamps
- Compare TCP Conversation Timestamp Values
- Determine the Initial Round Trip Time (iRTT)
- Troubleshooting Example Using Time
- Analyze Delay Types

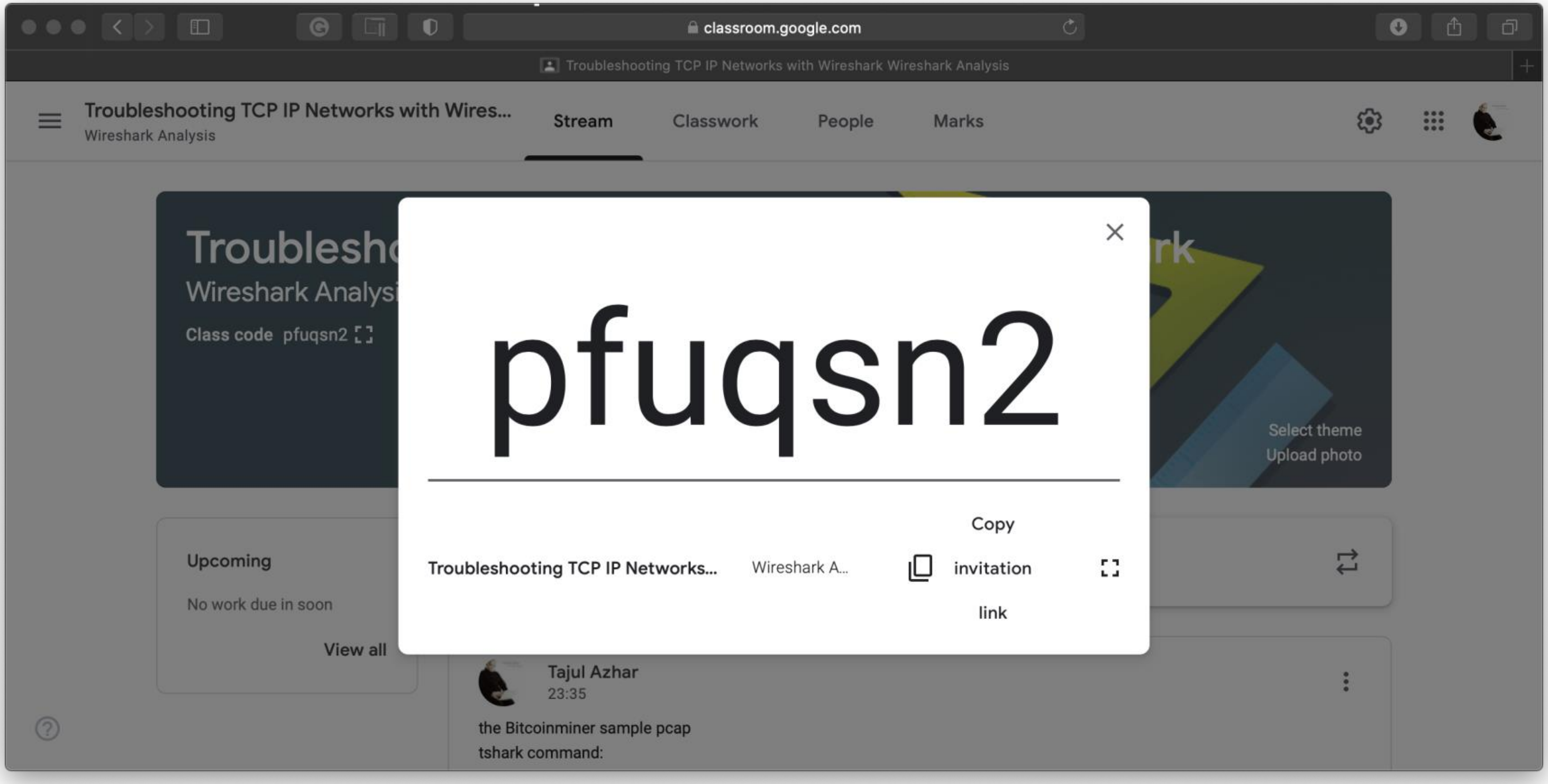
Session 5: Troubleshooting

- Effectively Use Command-Line Tools
- TCP/IP Communications and Resolutions Overview
- Analysing traffics
  - DNS
  - ARP
  - IPv4
  - ICMP
  - UDP
  - TCP
  - HTTP
  - TLS-Encrypted Traffic (HTTPS)
  - File Transfer Protocol (FTP)
- **Graph Traffic Characteristics**
  - Build Advanced IO Graphs
  - Graph Round Trip Times
  - Graph TCP Throughput
  - Find Problems Using TCP Time-Sequence Graphs

Let’s go: know your packet(s)

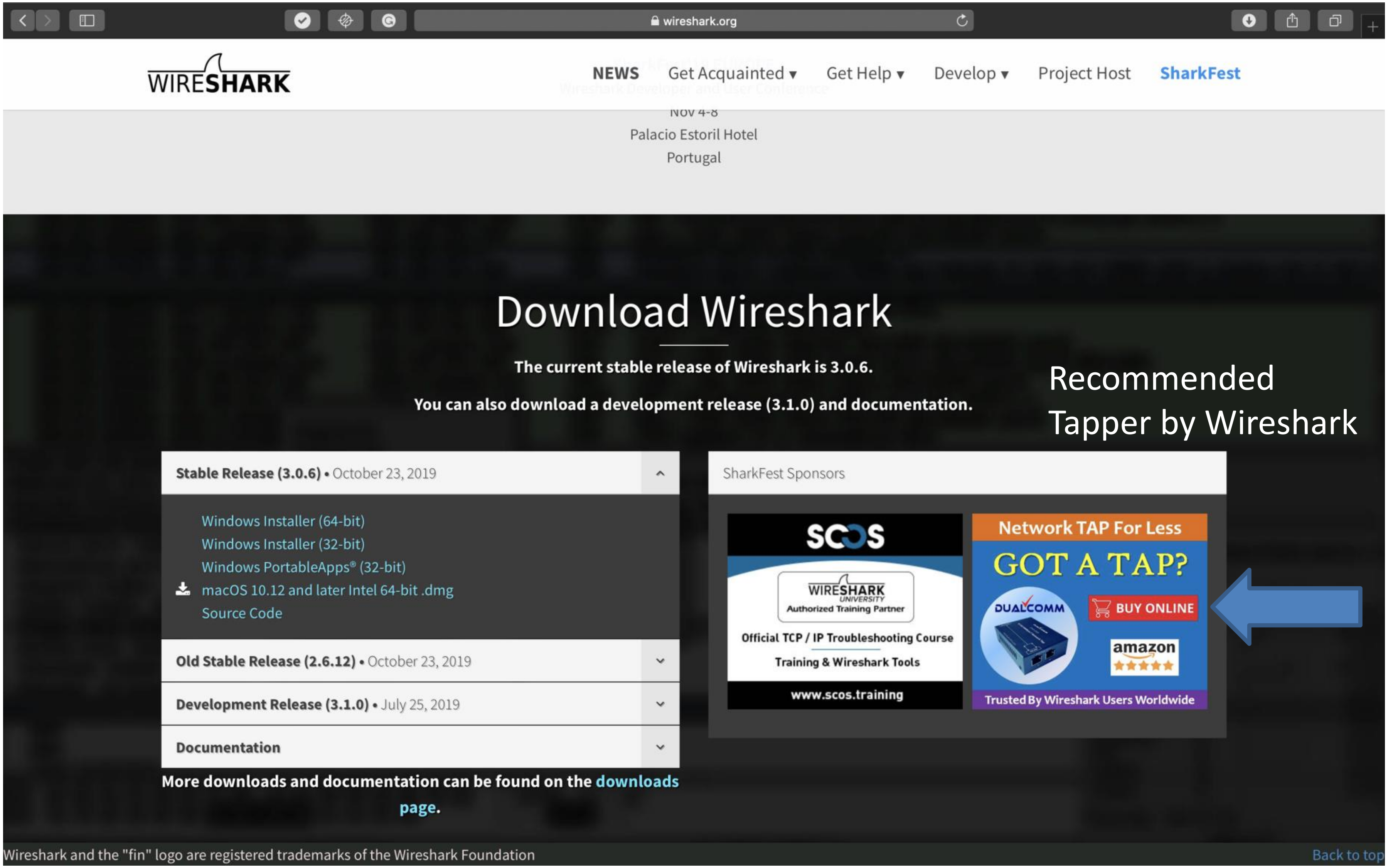
- PCAP or it didn’t happen
- Wireshark official website: https://www.wireshark.org
- In May of 2006, Gerald Combs (the original author of Ethereal) went to work for CACE Technologies (best known for WinPcap). Unfortunately, he had to leave the Ethereal trademarks behind.

Sharing platform



Digital Ocean Cloud Environment

- Use this code to get USD 100: https://m.do.co/c/141cb910b568
- A few testing and exercise(s) will use this infrastructure



Recommended Tapper by Wireshark

- Original press release
- Source: https://www.prweb.com/releases/2006/06/prweb396098.htm

Creator of Ethereal® Joins the WinPcap team; Wireshark is Born

Gerald Combs, creator of Ethereal®, has joined CACE Technologies (http://www.cacetech.com). He will be working with Loris Degioanni and Gianluca Varenni, the creators of the WinPcap packet capture library (http://www.winpcap.org). As his first venture in this new alliance, Gerald has created the Wireshark network protocol analyzer, a successor to Ethereal®.

DAVIS, CA (PRWEB) JUNE 8, 2006

We are proud to announce that Gerald Combs, creator of Ethereal®, has joined CACE Technologies (http://www.cacetech.com). He will be working with Loris Degioanni and Gianluca Varenni, the creators of the WinPcap packet capture library (http://www.winpcap.org), forming a world-class team of network analysis experts. As his first venture in this new alliance, Gerald has created the Wireshark network protocol analyzer, a successor to Ethereal®.

Wireshark's home is http://www.wireshark.org. Enhanced and improved, Wireshark is the ultimate tool of choice for network troubleshooting, protocol development, and education worldwide. The unique partnership of Wireshark and WinPcap brings a new synergy, power, and benefits to the open-source community and industry. The upcoming version of Wireshark will be 0.99.1. A pre-release version is available right now at http://www.wireshark.org.

"I am indebted to core development team of Ethereal® for joining me to work on Wireshark. With their help and contributions from the user community, we're set to continue our success in building the world's leading open-source network protocol analyzer. We have lots of new and exciting things planned for Wireshark! I'm also really excited about joining CACE. Loris and Gianluca are well respected in the community, and it will be great to work with them. As an added bonus, Davis is a great place for my wife and me to raise our daughter," said Gerald.

"We're thrilled to welcome Gerald to CACE Technologies and expect to do great things together. The sky's the limit," said Loris.

ABOUT CACE Technologies - CACE Technologies, http://www.cacetech.com, is an innovative and dynamic company specialized in low-level networking solutions. We are experts in Windows and Linux device driver and network monitoring tools development.

CONTACT: John Bruno, CACE Technologies, john.bruno @ cacetech.com

###

Installation of wireshark

- Wireshark can be obtained from here: https://www.wireshark.org/index.html#download
- Current stable release (as of today; 13 November 2019) : 3.0.6
- Old stable release: 2.6.12
- Development release: 3.1.0

Packet Analysis with Wireshark

1. What is Wireshark?
2. How does it work?
3. A brief overview of the TCP/IP model
4. An introduction to packet analysis
5. Why use Wireshark?
6. Understanding the GUI of Wireshark
7. The first packet capture

Introduction to Wireshark

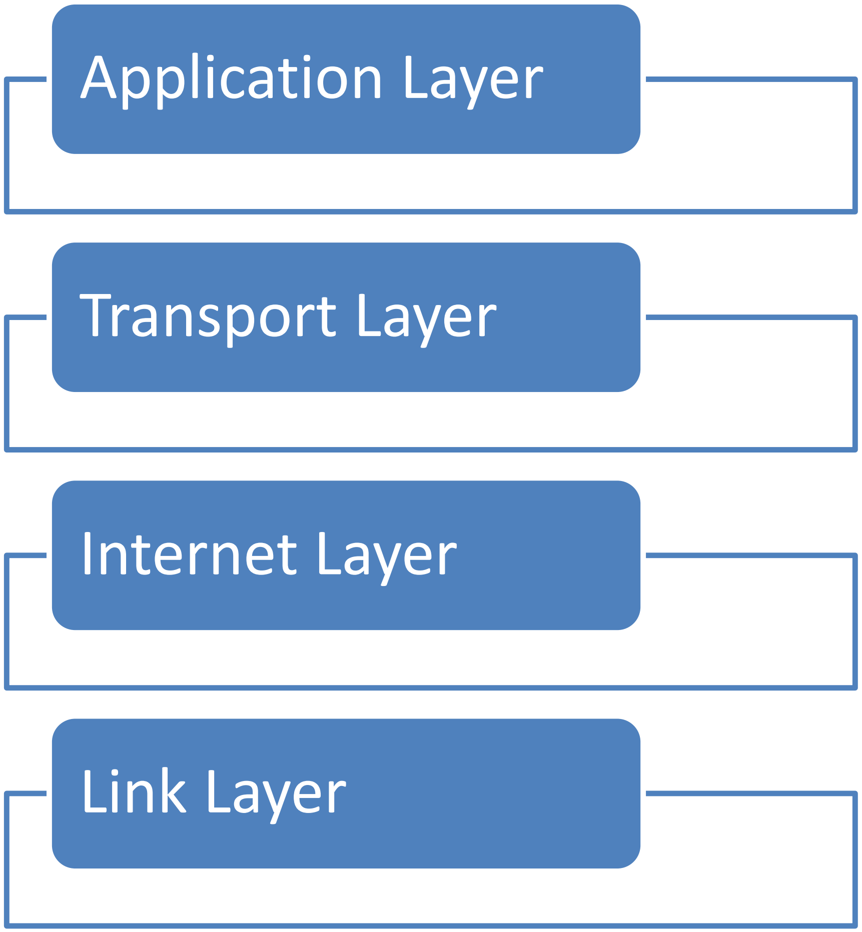
- Wireshark is one of the most advanced packet capturing software, which makes the life of system/network administrators easy and proves its usefulness among the groups of security evangelists.
- Wireshark is also called a protocol analyzer, which helps IT professionals in debugging network-level problems.
- This tool can be of great use to optimize network performance.
- Wireshark runs around dissecting network-level packets and showing packet details to concerned users as per their requirement.
- If you are one of those who deals with packet-level networking everyday, then Wireshark is for you and can be used for multiple troubleshooting purposes.

A brief overview of the TCP/IP model

- The TCP/IP model was originally known as the DoD model, and the project was regulated by United States Department of Defense.
- The TCP/IP model takes care of every aspect of every packet's life cycle, namely, how a packet is generated, how a single packet gets attached with a required set of information (PDU), how a packet is transmitted, how it comes to life, how it is routed through to intermediary nodes to the destination, how it is integrated back with other packets to get the whole information out, and so on.

The layers in the TCP/IP model

- The TCP/IP model comprises four layers, as shown in the following diagram.
- Each layer uses a different set of protocols allocated to it.
- Every protocol has specific designated roles, and all of them are designed in such a way that they comply with industry standards.



Application Layer

The Application layer also keeps track of user web sessions, which users are connected, and uses a set of protocols, which helps the application layer interface to the other layers in the TCP/IP model. Some popular protocols that we will cover in this book are as follows:

- The Hyper Text Transfer Protocol (HTTP)
- The File Transfer Protocol (FTP)
- The Simple Network Management Protocol (SNMP)
- The Simple Mail Transfer Protocol (SMTP)

Transport Layer

- The sole purpose of this layer is to create sockets over which the two hosts can communicate (you might already know about the importance of network sockets) which is essential to create an individual connection between two devices.
- There can be more than one connection between two hosts at the same instance. IP addresses and port numbers together make this possible. An IP address is required when we talk about WAN-based communication (in LAN-based communication, the actual data transfer happens over MAC addresses), and these days, a single system can communicate with more than one device over multiple channels which is possible with the help of port numbers. Apart from the restricted range of port numbers, every system is free to designate a random port for their communication.

- There can be more than one connection between two hosts at the same instance. IP addresses and port numbers together make this possible. An IP address is required when we talk about WAN-based communication (in LAN-based communication, the actual data transfer happens over MAC addresses), and these days, a single system can communicate with more than one device over multiple channels which is possible with the help of port numbers. Apart from the restricted range of port numbers, every system is free to designate a random port for their communication.

..TCP vs UDP

TCP

This is a **connection-oriented protocol,** often called a reliable protocol. Here, firstly, a dedicated channel is created between two hosts and then data is transferred. Then, the sender sends equally partitioned chunks, over the dedicated channel, and then, the receiver sends the acknowledgement for every chunk received. Most commonly, the sender waits for a particular time after which it sends the same chunk again for assurance. For example, if you are downloading something, TCP is the one that takes care and makes sure that every bit is transferred successfully.

UDP

This is a **connection-less protocol** and is often termed an unreliable form of communication. It is simple though because there is no dedicated channel created, and the sender is just concerned with sending chunks of data to the destination, whether it is received or not. This form of communication actually does not hamper the communication quality; the sole purpose of transferring the bits from a sender to receiver is fulfilled. For example, if you are playing a LAN-based game, the loss of a few bytes is not going to disrupt your gaming experience, and as a result, the user experience is not harmed.

Internet Layer

- concerned with the back and forth movement of data.
- The primary protocol that works is the **IP** (**Internet Protocol**) protocol, and it is the most important protocol of this layer.
- The IP provides the routing functionality due to which a certain packet can get to it's destination.
- Other protocols included in this layer are ICMP and IGMP.

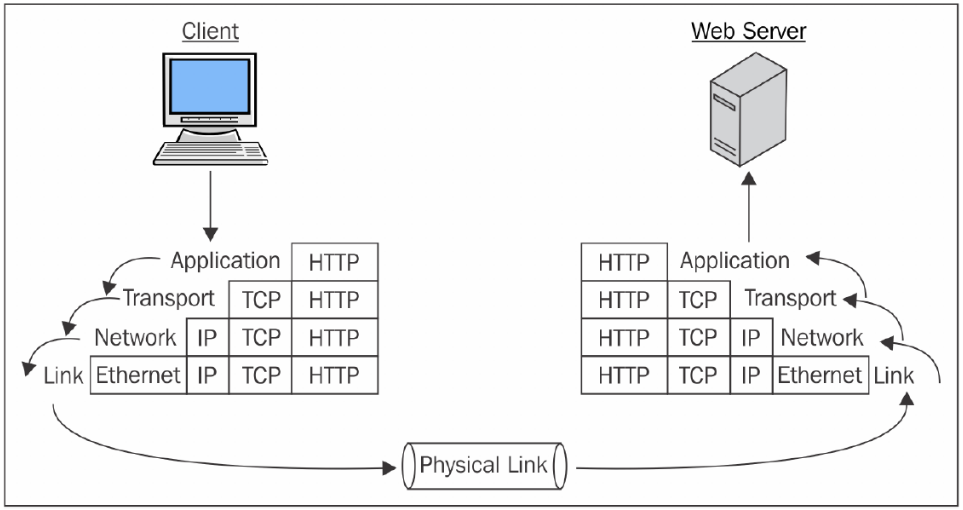
Link Layer

- The last layer is the **Link Layer** (often termed as the Network Interface Layer) that is close to the network hardware.
- There are no protocols specified in this layer by TCP/IP; however, several protocols are implemented, such as **Address Resolution Protocol** (**ARP**) and **Point to Point** (**PPP**).
- This layer is concerned with how a bit of information travels inside the real wires.
- It establishes and terminates the connection and also converts signals from analog to digital and vice versa.
- Devices such as bridges and switches operate in this layer.

Protocol Data Unit (PDU)

- The combination of an IP address and a MAC address for both the client and server is the core of the communication process, where the IP address is assigned to the device by the gateway or assigned statically, and the MAC address comes from the **Network Interface Card** (**NIC**), which should be present in every device that communicates with other hosts.
- As data progresses from the Application layer to the Link Layer, several bits of information are attached to the data bits in the form of headers or footers, which allow different layers of the TCP/IP model to coordinate with each other.
- The process of adding these extra bits is called data encapsulation, and in this process, a **Protocol data unit** (**PDU**) is created at the end of the networking model.

Data Encapsulation



- PDU consists of the information being sent along with the different protocol information that gets attached as part of the header or footer.
- By the time PDU reaches the bottom-most layer, it is embedded with all the required information required for the real transfer.
- Once it reaches the destination, the embedded header and footer PDU elements are ripped off one by one as it passes through each and every layer of the TCP/IP model as it progresses upward in the model.
- The figure above depicts the process of encapsulation

An introduction to packet analysis with Wireshark

- Packet analysis (also known as packet sniffing or protocol analyzing) is used to intercept and capture live data as it travels over the network (Ethernet or Wi-Fi) in order to understand what is happening in the network.
- Packet analysis is done by protocol analyzers such as Wireshark available on the Internet.
- Some of these are free and some are paid for commercial use.
- Numerous problems can happen in today's world of networking; for this, we need to be geared up all the time with the latest set of tools that can avail us of the ease of troubleshooting in any situation.
- Each of these problems will start from the packet level and can gradually grow up to a high network downtime.
- Even the best of protocols and services running on a system can go bad and behave maliciously.
- To get to the root of the problem, we need to look into the packet level to understand it better.
- If you need to maintain your network, then you definitely need to look into the packet level.

Packet analysis

- To **analyze** network problems by looking into the packets and their specific details so that you can get a better hold over your network.
- To detect network intrusion attempts and whether there are any malicious users who are trying to get into your network, or they have already got access to something in your network.
- To **detect** network misuse by internal or external users by establishing firewall rules in your security appliance and then monitoring each of these rules through Wireshark.
- To isolate exploited systems so that the affected system doesn't become a pivot point for your network for malicious users.
- To **monitor** data in motion once it travels live in your network to have better control over the allowed and restricted categories of data. For instance, say you want to create a rule for your firewall that will block the access to Bit Torrent sites. Blocking access to them can be done from your manageable router, but knowing from where the request was originated can be easily audited through Wireshark.
- To **gather** and report network statistics by filtering the most specific packets as per your requirements and then creating specific capture filters for your perusal that can help you in the long run.
- **Learning who is on the network and what they are doing**, is there something they are not allowed to do, and is there anyone who is trying to bypass the network restrictions. All of these simple day-to-day tasks can be achieved easily through Wireshark.
- To **debug** client/server communications so that all the request and replies communicated between the peers on our network can be audited to maintain the integrity of your network.
- **To look for applications that are sitting in the corner of your own network and eating the bandwidth**. They might be making your network insecure or making it visible to the public network. Through this unnoticed application, different forms of network traffic can enter without any restrictions.
- To **debug network protocol implementations** and any kind of anomalies present due to various misconfigurations in the current running devices.

MAMPU TOGETHER WE TRANSFORM

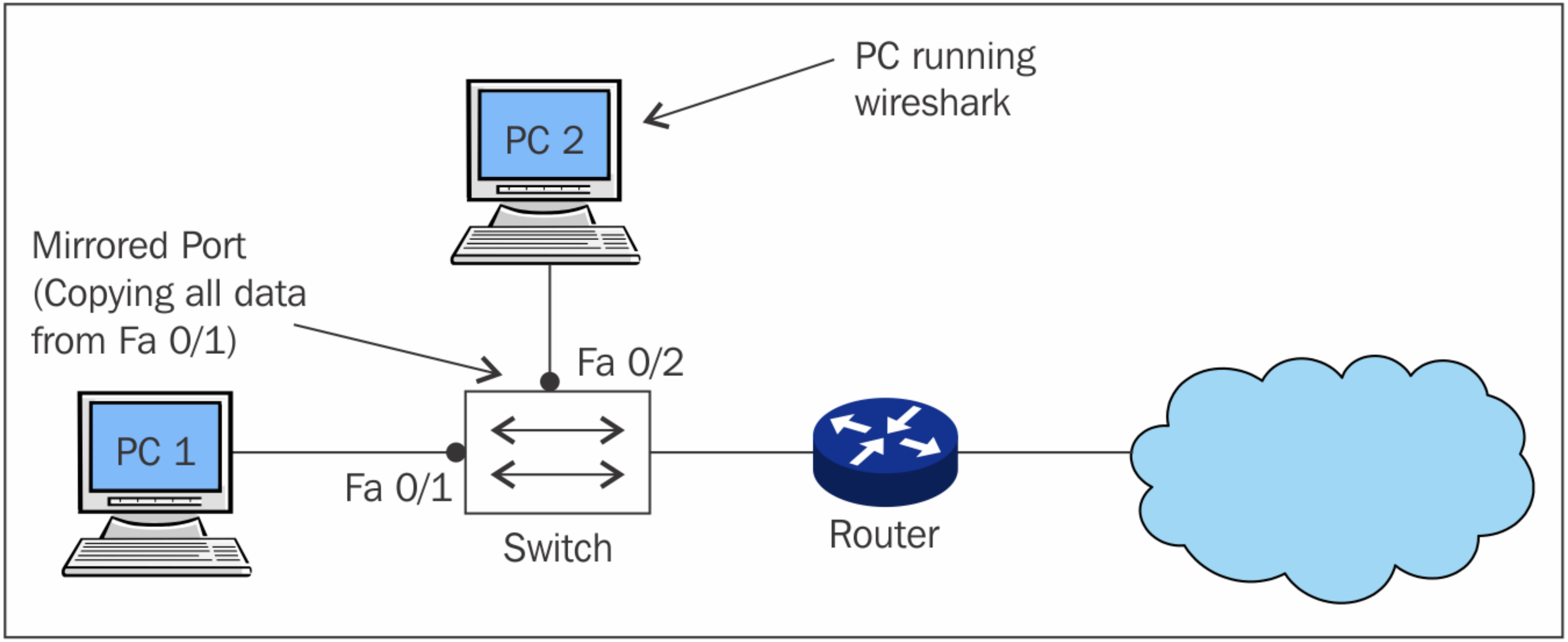
Capturing methodologies

- **Hub-based** networks are the easiest ones to sniff out because you've the freedom to place the sniffer at any place you want, as hubs broadcast each and every packet to the entire network they are a part of. So, we don't have to worry about the placement. However, hubs have one weakness that can drastically decrease network performance due to the collision of packets. Because hubs do not have any priority-based system for device that send packets, whoever wants to send them can just initiate the connection with the **HUB** (central device) and start transmitting the packets. Often, more than one devices start sending packets at the same instance. Now, as a result, the collision of the packets will happen, and the sending side will be informed to resend the previous packet. As a consequence, things such as traffic congestion and improper bandwidth utilization can be experienced.

The switched environment

- In **port mirroring**, once you have the command-line configuration console or web-based interface to mage you're the access point (router/switch), then we can easily configure port mirroring.
- Let's make it simpler for you with a logical illustration. For instance, let's assume that we have a 24-ports switch and 8 PCs which (PC-1 to PC-8) are connected. We are still left with more than 15 ports. Place your sniffer in any of those free ports and then **configure port mirroring**, which will copy all the traffic from whatever device we want to the port of our choice, where our protocol analyzer sits, which can see the whole bunch of data traveling through the mirrored port.

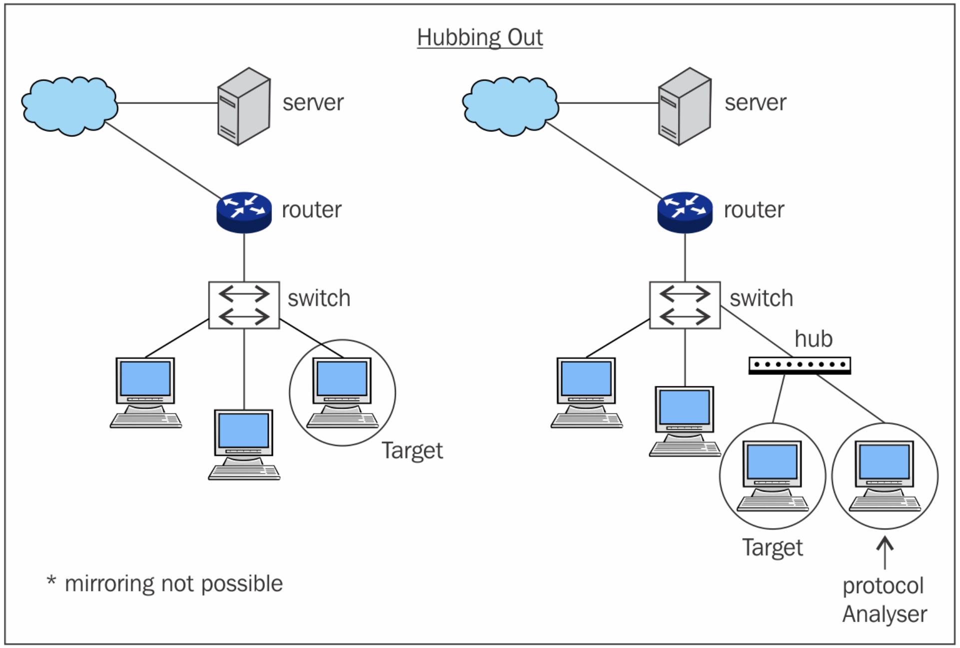
Port mirroring



Hubbing out

- **Hubbing out** is feasible when your switch doesn't support port mirroring. To use the technique, you have to actually plug the target PC out of the switched network, then plug your hub to the switch, and then connect you analyzer and target device to the switch so that becomes the part of the same network.

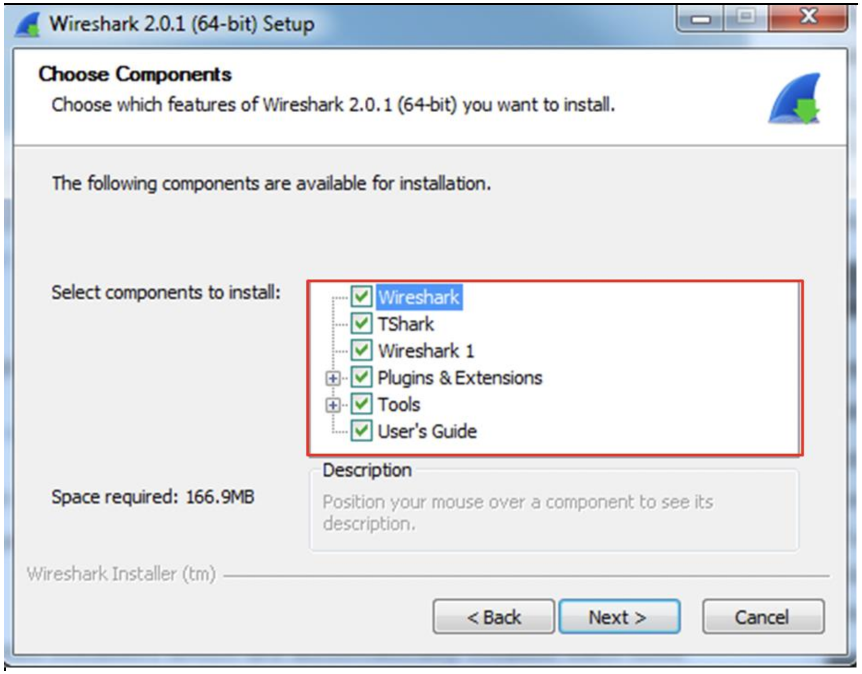
Hubbing out illustration



- Now, the protocol analyzer and the target are part of the same broadcast domain. Your analyzer will easily capture every packet destined to target or originated from the target. But make sure that the target is aware about the data loss that can happen while you try to create hubbing out for analysis. The image above will make it easier for us to understand the concept precisely.

Getting ready

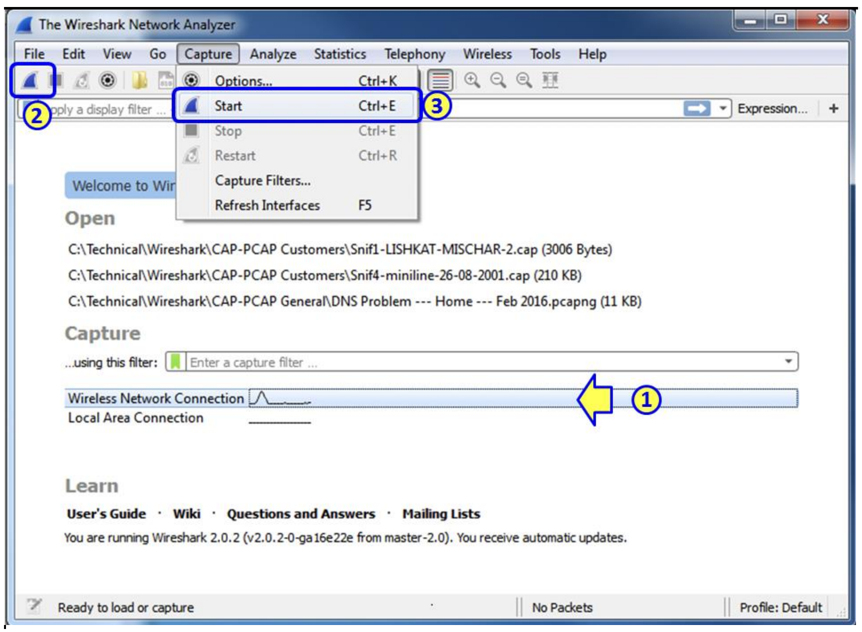
- Each Wireshark Windows package comes with the latest stable release of Winpcap, which is required for live packet capture. The Winpcap driver is a Windows version of the UNIX libpcap library for traffic capture.
- During the installation, you will get the packages installation window, presented in the preceding picture.



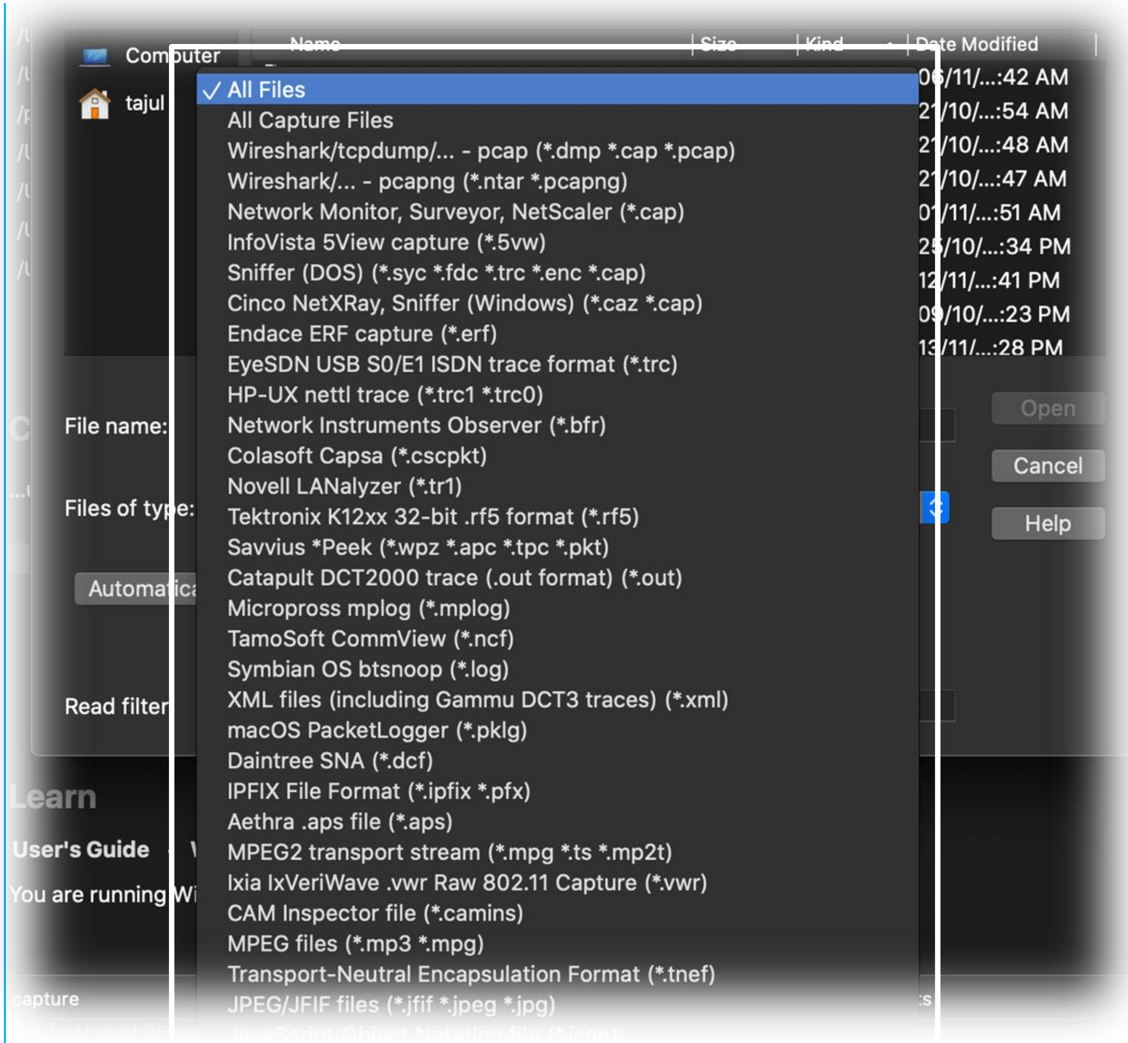
- Usually in these setup windows we simply check all and install it. In this case we have some interesting things.
- **Wireshark** - this is the Wireshark version 2 software
  TShark - a command-line protocol analyzer
- *CLI commands*
- **Plugins and extensions:**
  - **Dissector Plugins** - Plugins with some extended dissections.
  - **Tree Statistics Plugins** - Extended statistics.
- **Mate - Meta-Analysis and Tracing Engine** - User configurable extension(s) of the display filter engine. Further discussion in the MATE tool is in Appendix 3,*Meta- Analysis and Tracing Engine*
- **SNMP MIBs** - For a more detailed SNMP dissection.

- **Tools:**
  - **Editcap** - Reads a capture file and writes some or all of the packets into another capture file.
  - **Text2Pcap** - Reads in an ASCII hex dump and writes the data into a pcap capture file.
  - **Reordercap** - Reorders a capture file by timestamp.
  - **Mergecap** - Combines multiple saved capture files into a single output file.
  - **Capinfos** - Provides information on capture files. **Rawshark** - Raw packet filter.

Capturing interfaces: single interface



- 1: select the interface
- 2: choose the fin symbol to start capturing



Opening trace files

- We can open the existing trace files based on the format shows on the image.
- But most popular files are pcap, pcapng, cap.

Practice I: open a trace files and saving the captured files

- Scenario I.

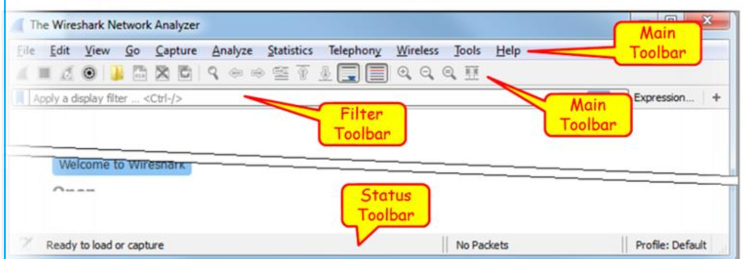
Sniff the WiFi traffic with your Wireshark

- Scenario II
- Saving your the capture file

Know your Wireshark

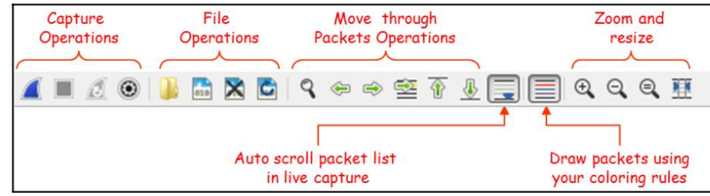
he main toolbar

he main toolbar provides quick access to frequently used items from the menu. This oolbar can be hidden using the **View** menu.



he four left-most symbols are for capture operations, then you have symbols for file perations, "go to packet" operations, auto-scroll, draw packets using coloring rules, zoom d resize.

Display Filter Toolbar



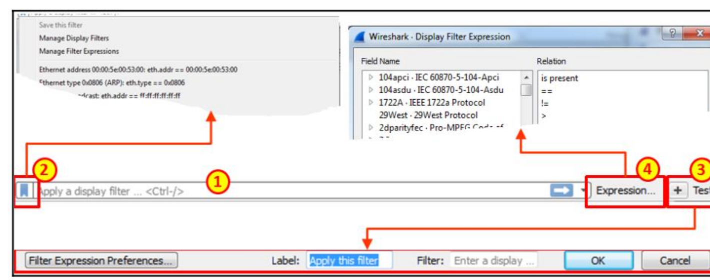
In the display filter toolbar we can:

1. Type in a display filter string, with auto complete while showing us previously configured filters
2. Manage filter expressions that allows you to bring up filter construction dialog for filter construction assistance.
3. Configure a new filter and add it to the preferences.
4. Use filter predefined expressions, and choose a filter.

Enhanced description about display filters provided in Chapter 4, *advanced configuration of display filters*.

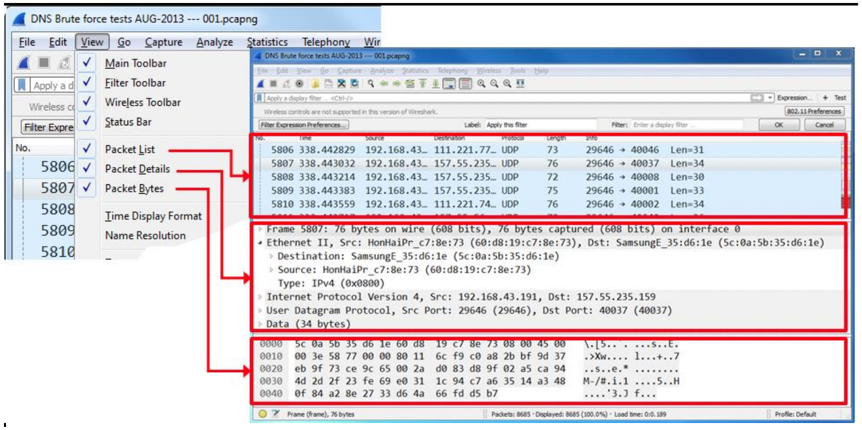
Status Bar

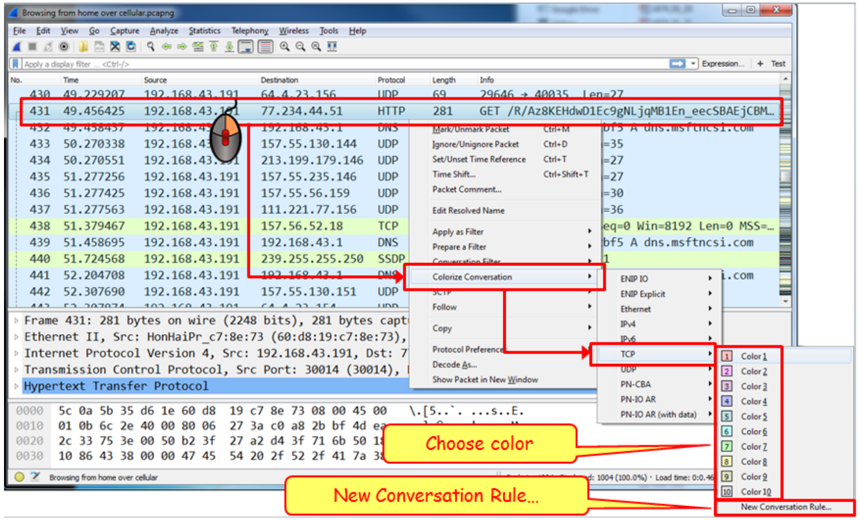
In the status bar, at the lower side of the Wireshark window, you can see the following data.



Practise 2: configure the toolbar

- Scenario 1: setting up Name resolution
- Scenario 2: Colorize packet list





- We will include some GeoIP files to translate IP to country with Wireshark
- Reference: https://wiki.wireshark.org/HowToUseGeoIP

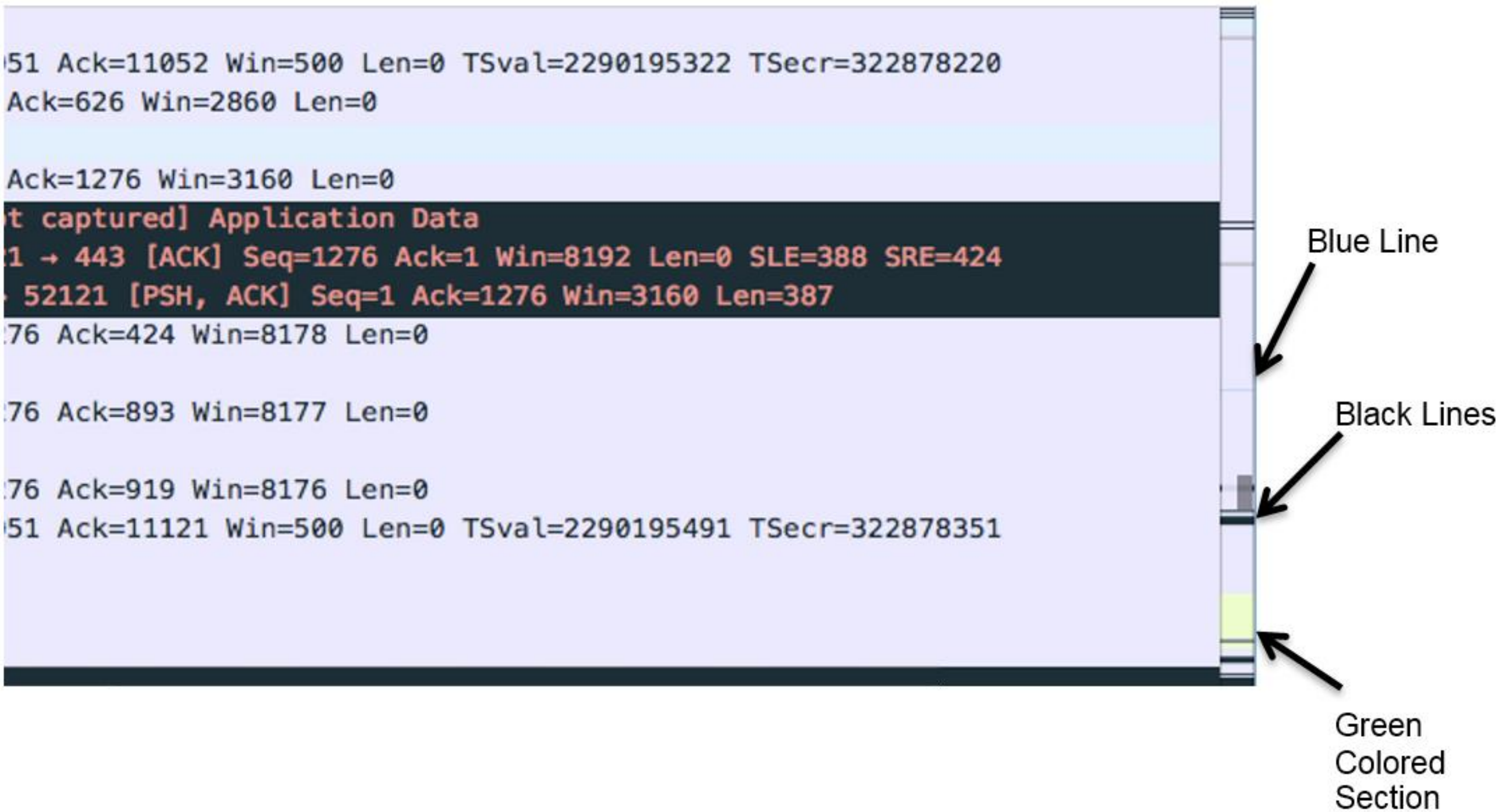
The intelligent scroll bar

- This is one of the features launched since version 2 release, and you might have already noticed some colored sections/lines in the scroll bar area.
- If not, then go back to any of the capture files you have, slowly scroll up and down, and observe the coloring pattern in the scroll bar area.
- Any guesses what difference it would make in the analysis process?
- Let's understand this with an example.

Understanding the intelligent scrollbar

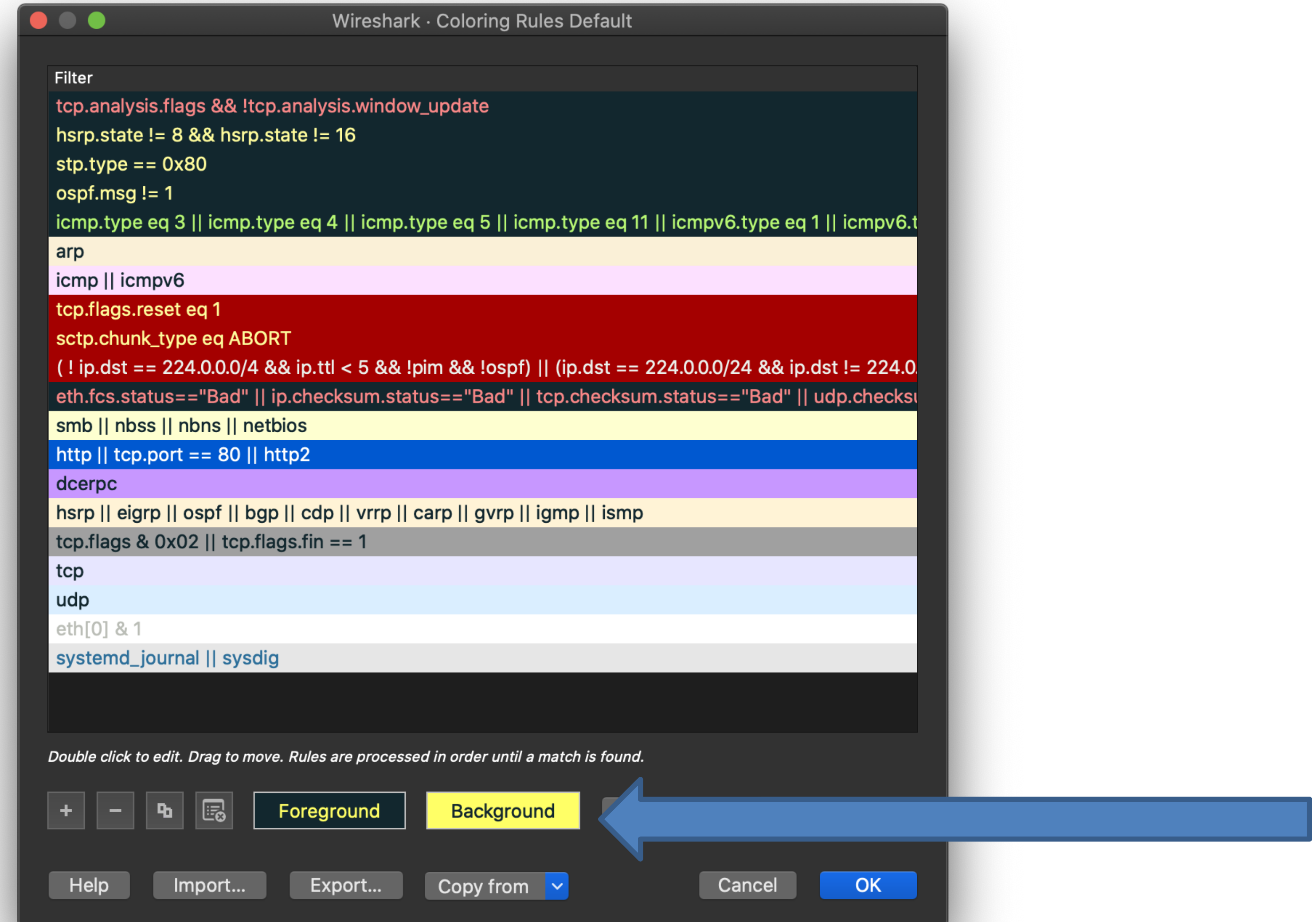
- We will use a previously captured file for demonstration purpose, which has HTTP and HTTPS packets along with some retransmission and duplicate frames. There is no difference that you can figure out at first glance, but as soon as you start scrolling, the coloring pattern will be shown in the scroll bar area.
- This pattern is based on the coloring rules that you have in your application. For example, as per the default coloring rules, duplicate and retransmission packets are usually seen with a black background and a red foreground, and HTTP packets are shown with a green background and a black foreground. Now, let's verify this in the application itself.

Intelligent Scrollbar in action



- In the same way that the blue line indicates the selected packet, the black lines denote the duplicate ACKs and retransmissions, and the green-colored section indicates that at the bottom of the capture file, we have some HTTP packets listed. By just observing the coloring pattern in the scroll bar area, we can figure out what sort of packets we have ahead, and most importantly, navigating to a certain section of packets you are looking for is now much easier and faster.
- We already discussed customizing the coloring rules in previous chapters; let's take one more example of the same capture file, and this time, I want to customize the HTTP packet coloring rule. We will change the green background color to yellow. Let's see what difference it would make in the scroll bar area in the following screenshot:

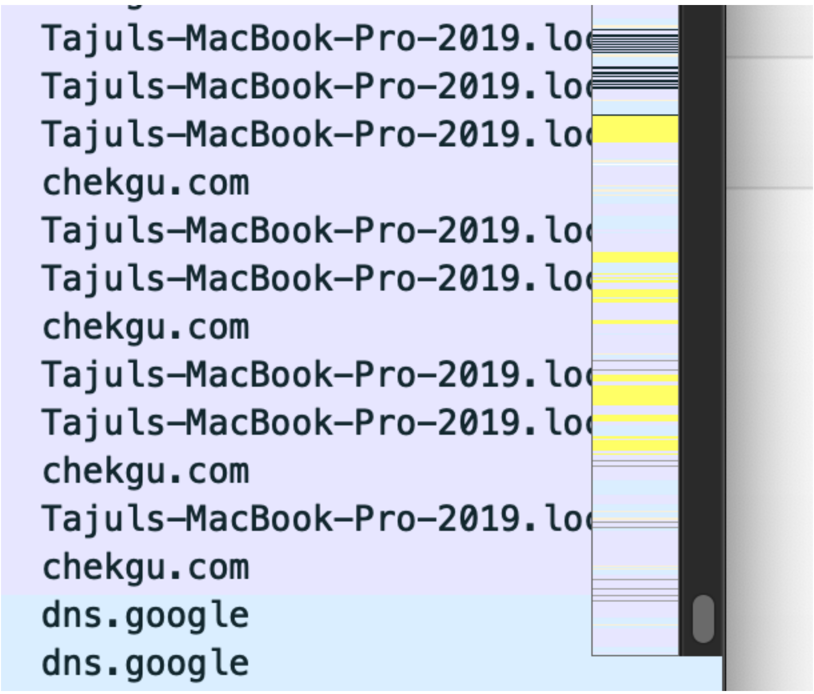
Practise 3: Changing the http coloring rules to yellow



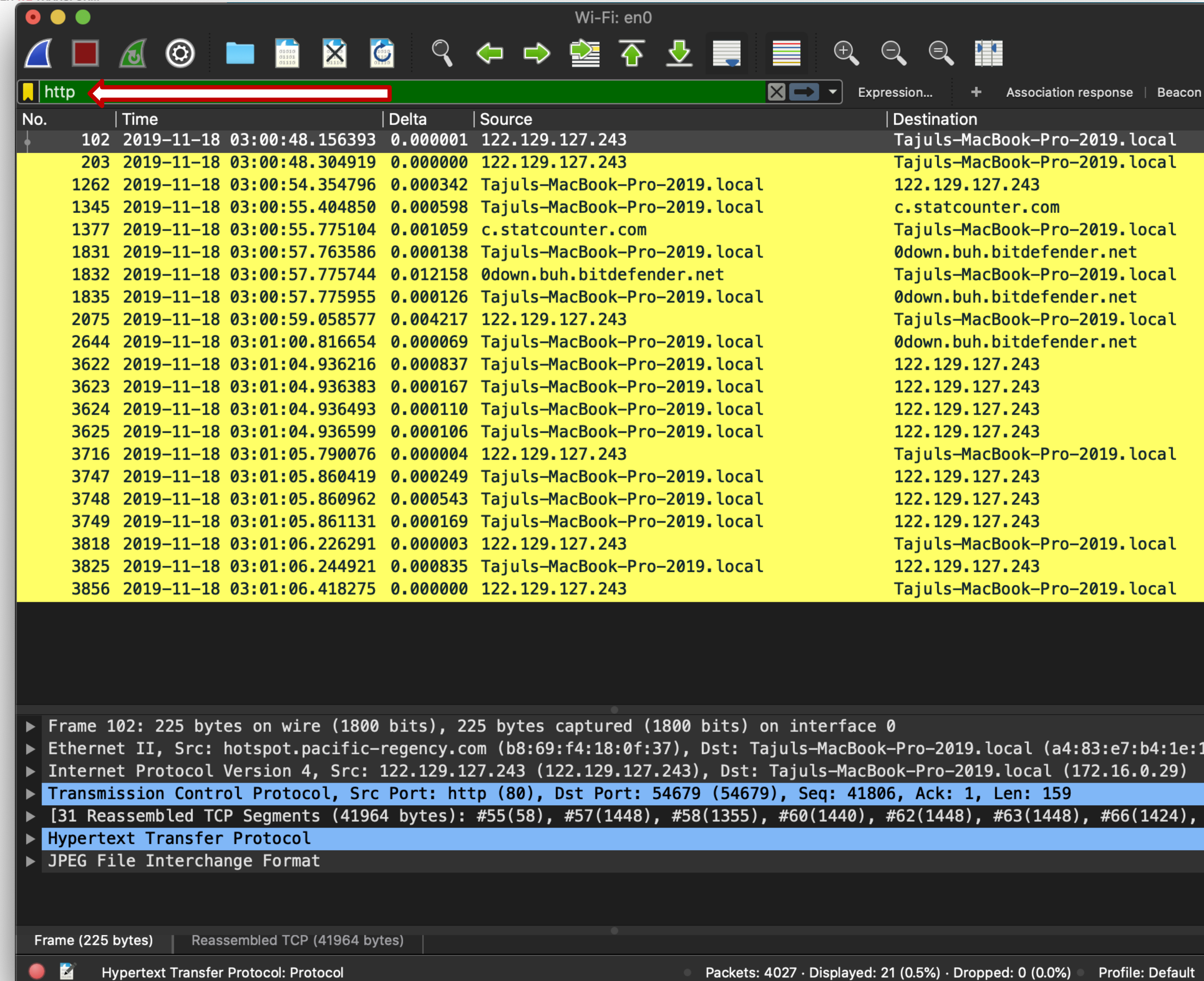
Change to yellow

- To access the coloring rules, you need to click on **View** from the menu bar and then choose **Coloring Rules** at the bottommost corner, which will show you the dialog where all coloring rules will be listed. Try changing the HTTP coloring rule to yellow. Once this has been done, close the dialog and reopen the capture file in order to see the change.
- Now, try scrolling the same file, and I hope you will see the difference in the coloring pattern in the scroll bar and your list pane too, where all HTTP packets are colored with a yellow background.

Intelligent scrollbar for http



Yellow colored
HTTP section


Wi-Fi: en0
http
Expression...
Association response
Beacon
No.
Time
Delta
Source
Destination
102 2019-11-18 03:00:48.156393 0.000001 122.129.127.243 Tajuls-MacBook-Pro-2019.local
203 2019-11-18 03:00:48.304919 0.000000 122.129.127.243 Tajuls-MacBook-Pro-2019.local
1262 2019-11-18 03:00:54.354796 0.000342 Tajuls-MacBook-Pro-2019.local 122.129.127.243
1345 2019-11-18 03:00:55.404850 0.000598 Tajuls-MacBook-Pro-2019.local c.statcounter.com
1377 2019-11-18 03:00:55.775104 0.001059 c.statcounter.com Tajuls-MacBook-Pro-2019.local
1831 2019-11-18 03:00:57.763586 0.000138 Tajuls-MacBook-Pro-2019.local 0down.buh.bitdefender.net
1832 2019-11-18 03:00:57.775744 0.012158 0down.buh.bitdefender.net Tajuls-MacBook-Pro-2019.local
1835 2019-11-18 03:00:57.775955 0.000126 Tajuls-MacBook-Pro-2019.local 0down.buh.bitdefender.net
2075 2019-11-18 03:00:59.058577 0.004217 122.129.127.243 Tajuls-MacBook-Pro-2019.local
2644 2019-11-18 03:01:00.816654 0.000069 Tajuls-MacBook-Pro-2019.local 0down.buh.bitdefender.net
3622 2019-11-18 03:01:04.936216 0.000837 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3623 2019-11-18 03:01:04.936383 0.000167 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3624 2019-11-18 03:01:04.936493 0.000110 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3625 2019-11-18 03:01:04.936599 0.000106 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3716 2019-11-18 03:01:05.790076 0.000004 122.129.127.243 Tajuls-MacBook-Pro-2019.local
3747 2019-11-18 03:01:05.860419 0.000249 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3748 2019-11-18 03:01:05.860962 0.000543 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3749 2019-11-18 03:01:05.861131 0.000169 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3818 2019-11-18 03:01:06.226291 0.000003 122.129.127.243 Tajuls-MacBook-Pro-2019.local
3825 2019-11-18 03:01:06.244921 0.000835 Tajuls-MacBook-Pro-2019.local 122.129.127.243
3856 2019-11-18 03:01:06.418275 0.000000 122.129.127.243 Tajuls-MacBook-Pro-2019.local
Frame 102: 225 bytes on wire (1800 bits), 225 bytes captured (1800 bits) on interface 0
Ethernet II, Src: hotspot.pacific-regency.com (b8:69:f4:18:0f:37), Dst: Tajuls-MacBook-Pro-2019.local (a4:83:e7:b4:1e:1
Internet Protocol Version 4, Src: 122.129.127.243 (122.129.127.243), Dst: Tajuls-MacBook-Pro-2019.local (172.16.0.29)
Transmission Control Protocol, Src Port: http (80), Dst Port: 54679 (54679), Seq: 41806, Ack: 1, Len: 159
[31 Reassembled TCP Segments (41964 bytes): #55(58), #57(1448), #58(1355), #60(1440), #62(1448), #63(1448), #66(1424),
Hypertext Transfer Protocol
JPEG File Interchange Format
Frame (225 bytes)
Reassembled TCP (41964 bytes)
Hypertext Transfer Protocol: Protocol
Packets: 4027 · Displayed: 21 (0.5%) · Dropped: 0 (0.0%)
Profile: Default

Display Filter comparison operators

C-like Syntax	Shortcut	Description	Example
>	**gt**	Greater than	`frame.len > 64`
<	**lt**	Less than	`frame.len < 1500`
>=	**ge**	Greater than or equal to	`frame.len >= 64`
<=	**le**	Less than or equal to	`frame.len <= 1500`
	is present	A parameter is present	`http.response`
	contains	Contains a string	`http.host contains cisco`
	matches	A string matches the condition	`http.host matches www.cisco.com`

Learn Capture Methods and Use Capture Filters

SESSION 2

Supported filtering for 802.11 radiotap

2,248 PROTOCOLS
185,882 FIELDS
SOURCE: WIRESHARK

Wireshark · Supported Protocols

Name	Filter	Type	Description
29West	29west		29West Protocol
▸ 2dparityfec	2dparityfec		Pro-MPEG Code of Practice #3 release 2 FEC Protocol
▸ 3COMXNS	3comxns		3Com XNS Encapsulation
▸ 3GPP2 A11	a11		3GPP2 A11
▸ 6LoWPAN	6lowpan		IPv6 over Low power Wireless Personal Area Networks
▸ 802.11 RSNA EAPOL	wlan_rsna_eapol		IEEE 802.11 RSNA EAPOL key
▸ 802.11 Radio	wlan_radio		802.11 radio information
▾ 802.11 Radiotap	radiotap		IEEE 802.11 Radiotap Capture header
# of HE-SIG-B Symbols or # o...	radiotap.he_mu.sig_b_s...	Unsigned integer, 2 bytes	
# of HE-SIG-B Symbols or # o...	radiotap.he_mu.sig_b_s...	Unsigned integer, 2 bytes	
# of HE-SIG-B Symbols/MU-...	radiotap.he_mu.symbol...	Boolean	
0 Length PSDU	radiotap.present.0_len...	Boolean	Specifies whether or not the 0-Length PSDU field is present
2 GHz spectrum	radiotap.xchannel.flags...	Boolean	Channel Flags 2 GHz spectrum
2 GHz spectrum	radiotap.channel.flags....	Boolean	Channel Flags 2 GHz spectrum
32-bit counter	radiotap.timestamp.fla...	Boolean	
5 GHz spectrum	radiotap.xchannel.flags...	Boolean	Channel Flags 5 GHz spectrum
5 GHz spectrum	radiotap.channel.flags....	Boolean	Channel Flags 5 GHz spectrum
802.11 FCS	radiotap.fcs	Unsigned integer, 4 bytes	Frame check sequence of this frame
A-MPDU Status	radiotap.present.ampdu	Boolean	Specifies if the A-MPDU status field is present
A-MPDU flags	radiotap.ampdu.flags	Unsigned integer, 2 bytes	A-MPDU status flags
A-MPDU reference number	radiotap.ampdu.refere...	Unsigned integer, 4 bytes	
A-MPDU status	radiotap.ampdu	Label	
A-MPDU subframe delimiter ...	radiotap.ampdu.delim_...	Unsigned integer, 1 byte	
Antenna	radiotap.antenna	Unsigned integer, 4 bytes	Antenna number this frame was sent/received over (starting at 0)
Antenna	radiotap.present.anten...	Boolean	Specifies if the antenna number field is present
Antenna noise	radiotap.dbm_antnoise	Signed integer, 1 byte	RF noise power at the antenna expressed as decibels from one milliwatt
Antenna signal	radiotap.dbm_antsignal	Signed integer, 1 byte	RF signal power at the antenna expressed as decibels from one milliwatt
BSS Color	radiotap.he.data_3.bss...	Unsigned integer, 2 bytes	
BSS Color known	radiotap.he.data_1.bss_...	Boolean	
BSS Color unknown	radiotap.he.data_3.bss...	Unsigned integer, 2 bytes	
Bad FCS	radiotap.fcs_bad	Boolean	Specifies if this frame has a bad frame check sequence
Bad FCS	radiotap.flags.badfcs	Boolean	Frame received with bad FCS
Bad PLCP	radiotap.rxflags.badplcp	Boolean	Frame with bad PLCP
Bandwidth	radiotap.vht.bw	Unsigned integer, 1 byte	
Bandwidth	radiotap.mcs.bw	Unsigned integer, 1 byte	
Bandwidth	radiotap.mcs.have_bw	Boolean	Bandwidth information present
Beam Change	radiotap.he.data_3.bea...	Unsigned integer, 2 bytes	
Beam Change known	radiotap.he.data_1.bea...	Boolean	
Beam Change unknown	radiotap.he.data_3.bea...	Unsigned integer, 2 bytes	
Beamformed	radiotap.vht.beamform...	Boolean	

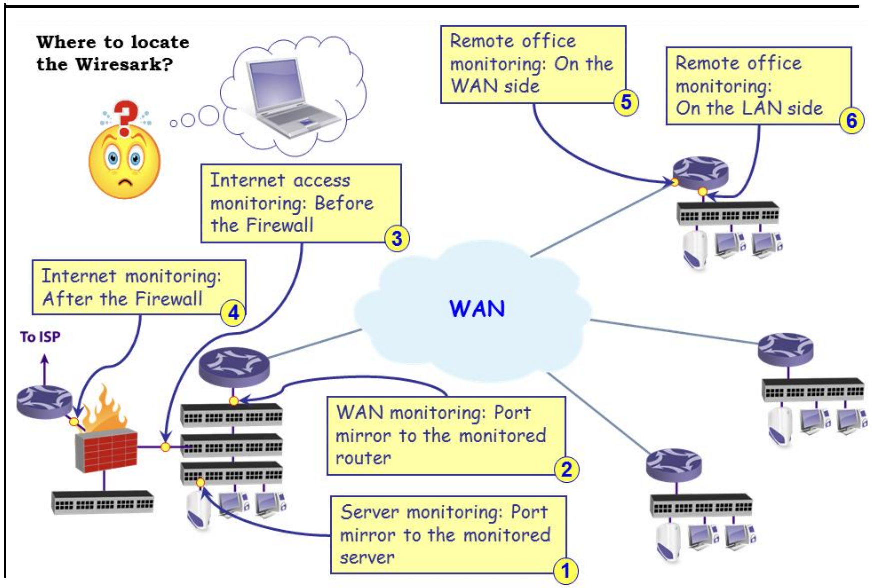
Search:

2,248 protocols, 185,882 fields.

Close

@seTajul

Locating Wireshark

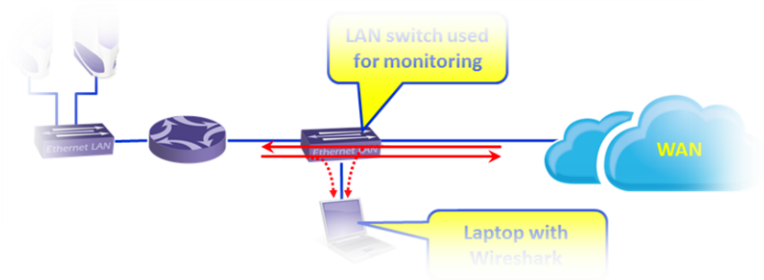


Depending on your Use Case, Wireshark can exist in any parts on the network

- Port mirror – Switch
- Port Mirror – Router
- Before / after the Firewall
- Office
  - Monitoring passively
  - On the LAN side

Monitoring firewall

- You can use monitoring Switch to get the packet from Firewall.
- If from the Internal port, traffic coming from the Internal connection and with external port, traffic received from the external firewall.



TAPS and Hubs

- TAPS: Test Access Point
- Taps can forward any errors that happens in the networks
- Whereas, Switch is more expensive compare to taps.
- Recommended taps are

- Hub: get the traffic parallel from your connection
- Hubs sometime cannot receive a huge traffic.

Capturing remote communication devices

- tshark
- sshdump
- tcpdump or ;
- installing pcap driver from your remote machine

Using capture filters

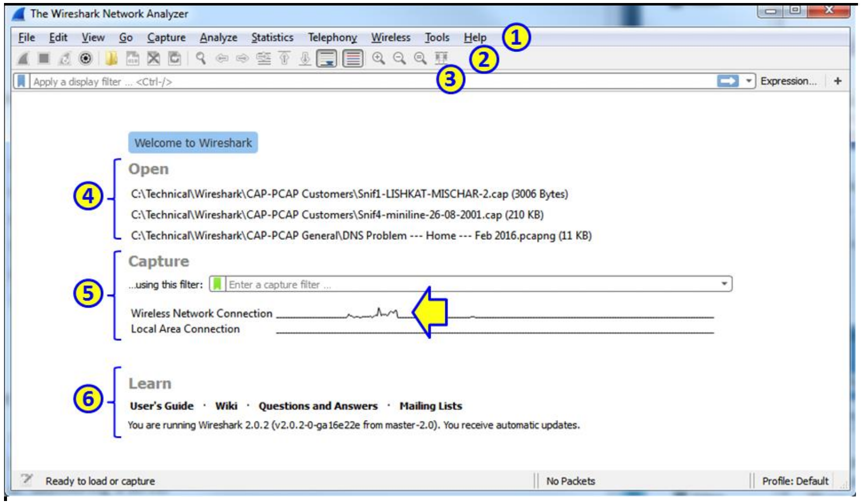
- In this chapter we will cover the following domains:
  - Configuring captures filters
  - Configuring Ethernet filters
  - Configuring hosts and networks filters
  - Configuring TCP/UDP and port filters
  - Configuring byte-offset and payload filters

- An overview of the capture filter syntax can be found in the User's Guide. A complete reference can be found in the expression section of the pcap-filter(7) manual page.
- Wireshark uses the same syntax for capture filters as tcpdump, WinDump, Analyzer, and any other program that uses the libpcap/WinPcap library.
- If you need a capture filter for a specific protocol, have a look for it at the ProtocolReference.

Available capture interface(S)

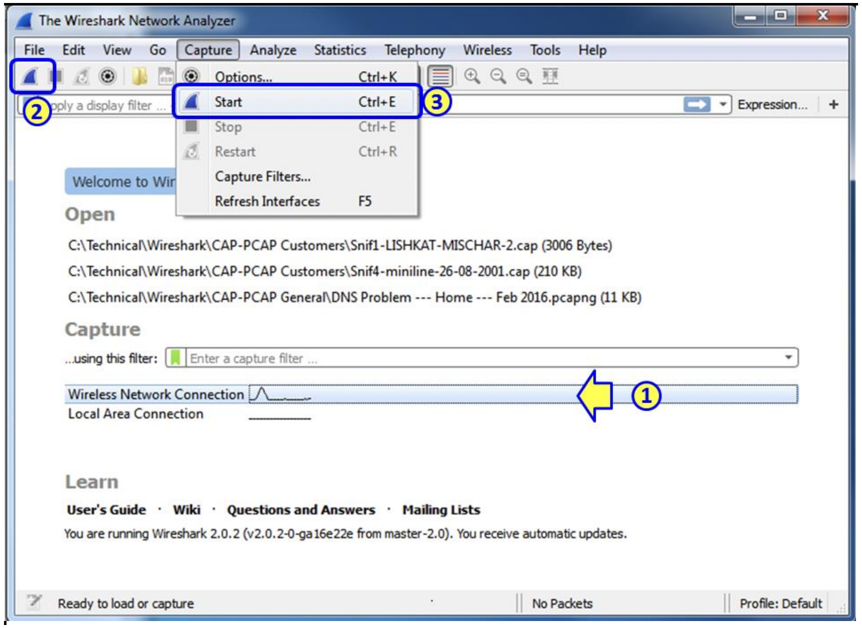
In the start window, you will see the following sections:

1. The **main menus**, with file, edit and view operations, capture, statistics and various tools.
2. The **main toolbar**, that provides quick access to frequently used items from the menu.
3. The **filter toolbar**, that provides access to the display filters. In the main area of the start window, we have the following items:
4. A list of files that were recently opened
5. A **Capture** part that enables us to configure a capture filter, and sows us the traffic on our computer interfaces.
6. The **Learn** part, that can take us directly to the manual pages



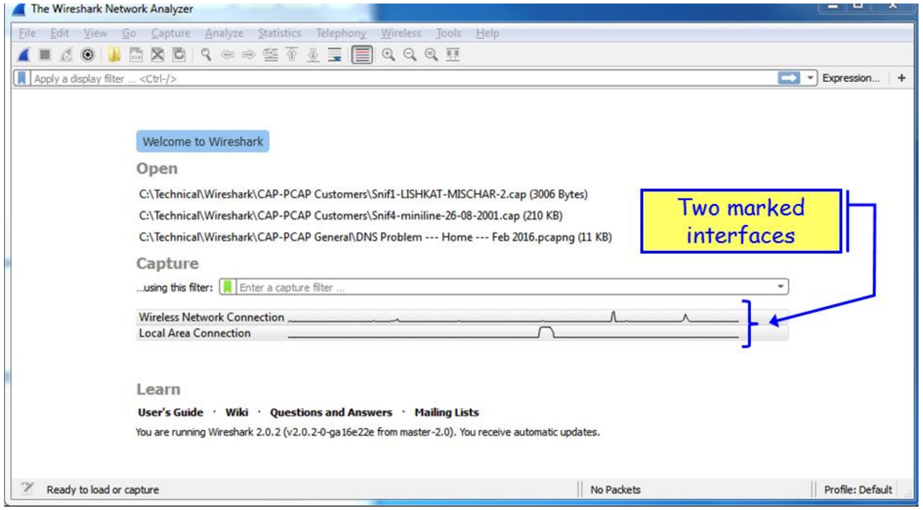
Capturing single interface

- simplest way to start a simple single-interface capture is simply to double-click the active interface (1). You can also mark the active interface and click on the capture button on the upper-left corner of the window (2), or choose **start** or **Ctrl-E** from the Capture menu (3).



Capture on multiple interfaces

- In order to start the capture on multiple interfaces, you simply use Windows **Ctrl** or **Shift** keys, and left-click to choose the interfaces you want to capture data from. In the following screenshot you see that the Wireless and the Local Area connections are picked up.



Filtering expression on the filter function

In filter expressions you configure which filter expressions will appear at the right size of the display filters bar at the top of the screen.

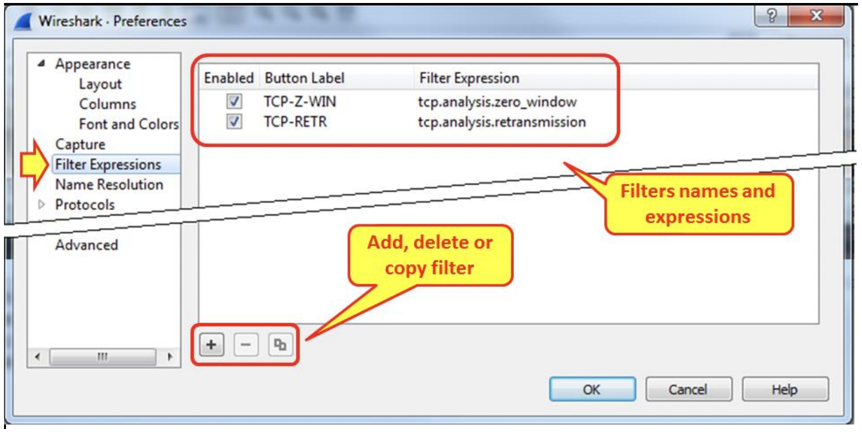
To configure display filter expressions:

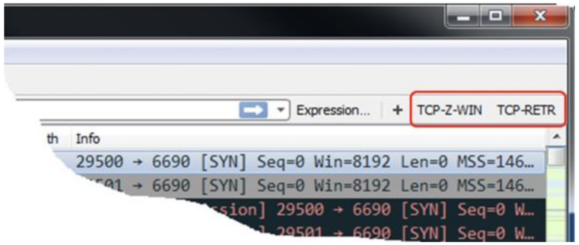
1. Click on the **Edit** menu and choose **preferences** and **Filter expressions**. The following window will come up.

2. Choose Add and configure the **button Label** and the **filter expression.**

3. As you can see in the following screenshot, the Button Label will appear at the right side of the display filters bar.

4. As you can see, the filters named **TCP-Z-WIN** and **TCP-RETR** that we have configured in the filters preferences appear at the right corner of the Wireshark.





Practice 2: capture filters

- Open your Wireshark
- On the Main toolbar, choose 'capture options'
- Source: https://wiki.wireshark.org/CaptureFilters

Capture only traffic to or from IP address 172.18.5.4:

host 172.18.5.4

Capture traffic to or from a range of IP addresses:

net 192.168.0.0/24

or

net 192.168.0.0 mask 255.255.255.0

Capture traffic from a range of IP addresses:

src net 192.168.0.0/24

or

src net 192.168.0.0 mask 255.255.255.0

C-Like Syntax	Shortcut	Description	Example
=	eq	Equal	ip.addr == 192.168.1.1 or ip.addr eq 192.168.1.1
!=	ne	Not equal	!ip.addr==192.168.1.1 or ip.addr != 192.168.1.1 or ip.addr ne 192.168.1.1
>	gt	Greater than	frame.len > 64
<	lt	Less than	frame.len < 1500
>=	ge	Greater than or equal to	frame.len >= 64
<=	le	Less than or equal to	frame.len <= 1500
	Is present	A parameter is present	http.response
	contains	Contains a string	http.host contains cisco
	matches	A string matches the condition	http.host matches www.cisco.com

Using display filters: operators

C-Like Syntax	Shortcut	Description	Example
&&	and	Logical AND	ip.src==10.0.0.1 and tcp.flags.syn==1 all SYN flags sent from IP address 10.0.0.1 practically - all connections opened (or tried to be opened) from 10.0.0.1
||	or	Logical OR	ip.addr==10.0.0.1 or ip.addr==10.0.02 All packets going in or out the two IP addresses
!	not	Logical NOT	not arp and not icmp All packets that are no ARP and not ICMP packets

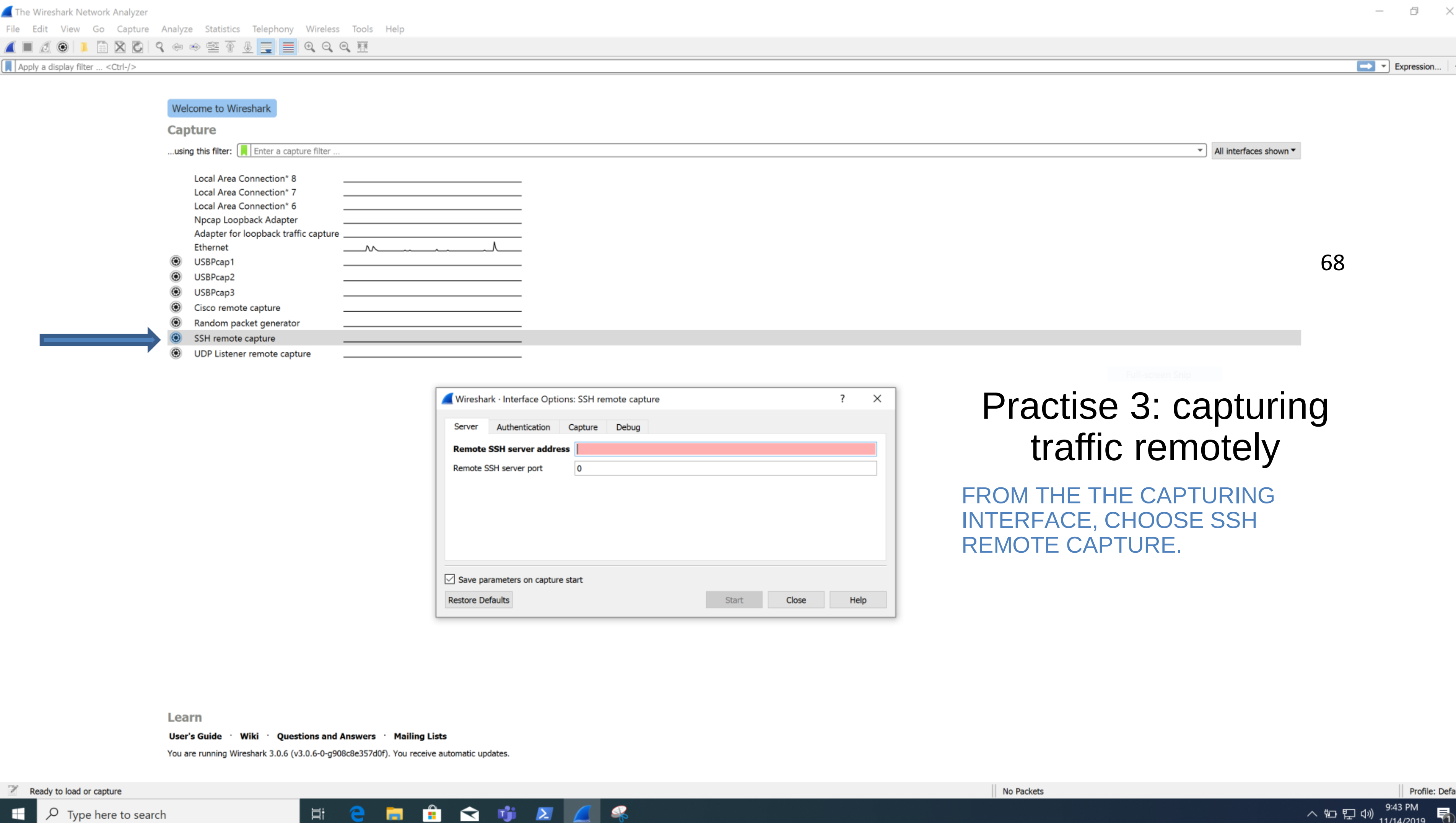
SSHDUMP

- sshdump - Provide interfaces to capture from a remote host through SSH using a remote capture binary.
- To get deep understanding, we will access remotely our server using the sshdump capabilities.
- **Sshdump** is an extcap tool that allows one to run a remote capture tool over a SSH connection. The requirement is that the capture executable must have the capabilities to capture from the wanted interface.
- The feature is functionally equivalent to run commands like

$ ssh remoteuser@remotehost -p 22222 'tcpdump -U -i IFACE -w -' > FILE & $ wireshark FILE

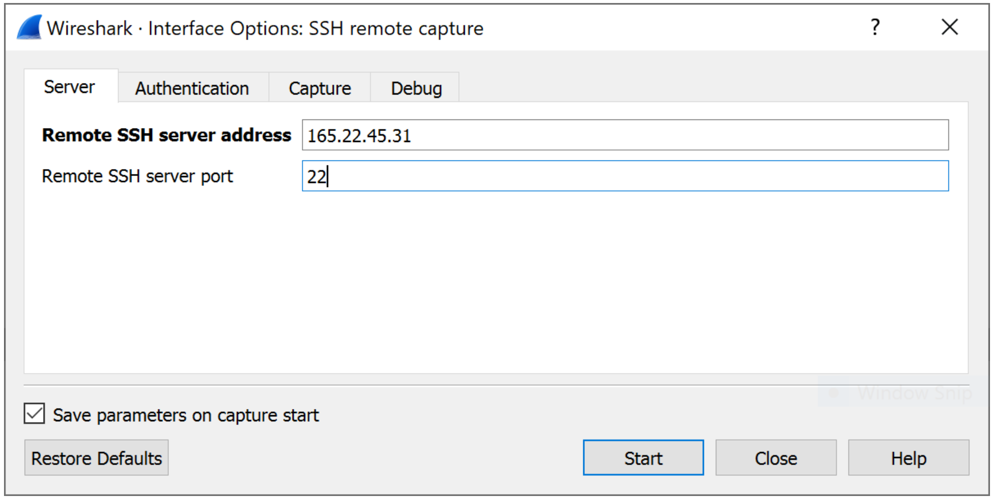
$ ssh remoteuser@remotehost '/sbin/dumpcap -i IFACE -P -w - -f "not port 22"' > FILE & $ wireshark FILE

$ ssh somehost dumpcap -P -w - -f udp | tshark -i -



Practise 3: capturing traffic remotely

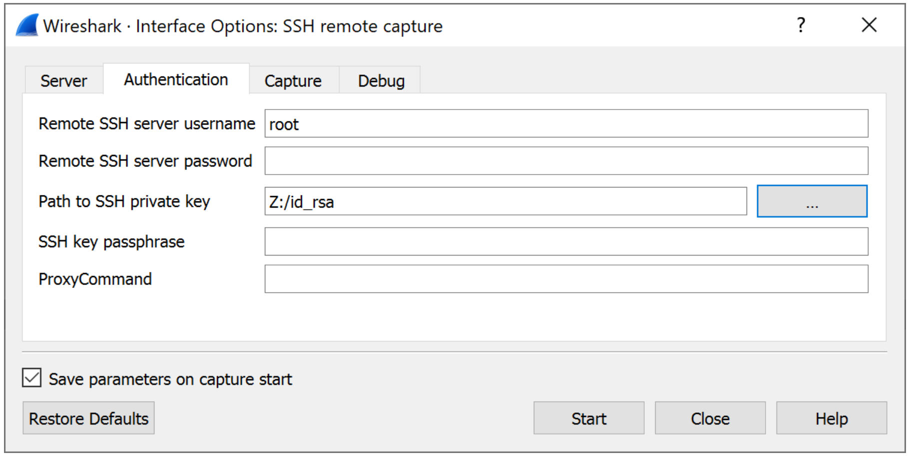
FROM THE THE CAPTURING INTERFACE, CHOOSE SSH REMOTE CAPTURE.



Adding the necessary information

ON THE 'SERVER' MENU TAB, INPUT THE OBTAINED IP FROM YOUR INSTRUCTOR.

- From the 'Authentication' tab, get the private key file if necessary.
- SSH private key is necessary if the server authentication needs the private key.



Customize for Efficiency: Configure Your Global Preferences

SESSION 3

Configuring colouring rules and navigation techniques

- Coloring rules define how Wireshark will color protocols and events in the captured data. Working with the coloring rules will help you a lot with network troubleshooting, since you are able to see different protocols in different colors, and you can also configure different colors for different events.
- Coloring rules enable you to configure new coloring rules according to various filters. It will help you to configure different coloring schemes for different scenarios and save them in different profiles. In this way you can configure coloring rules for resolving TCP issues, rules for resolving Sip and Telephony problems, and so on.

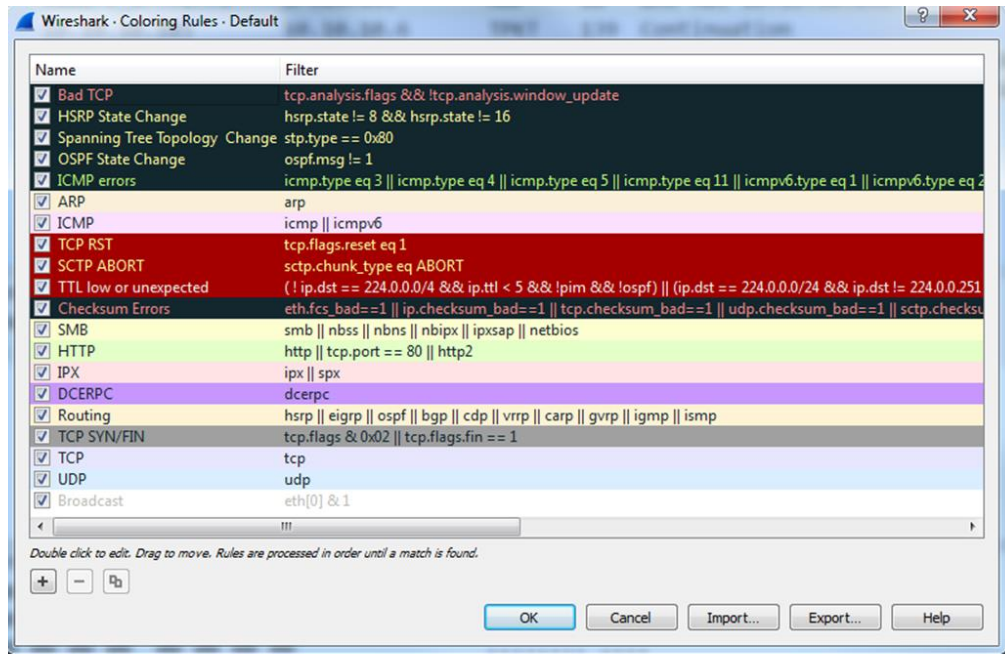
Getting ready

For starting with the coloring rules proceed as follows:

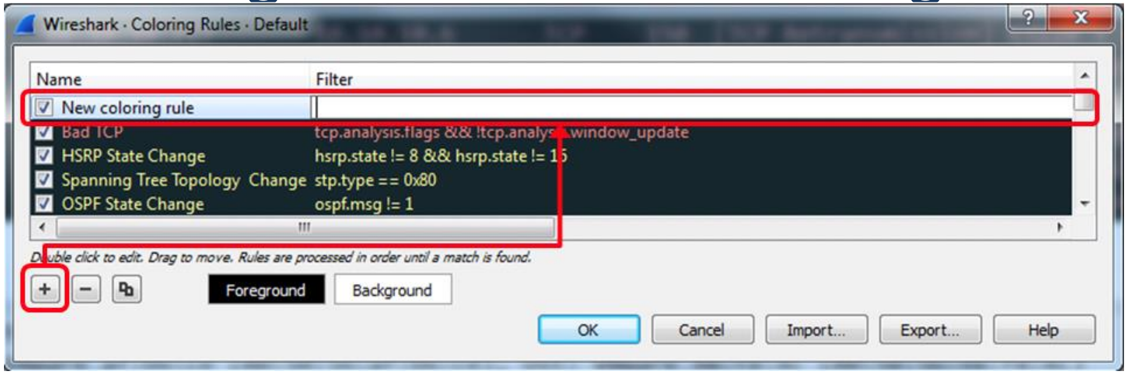
1. Go **to** the **View** menu.

2. At the lower part of the menu choose **Coloring Rules**. You will get the following window as the image on left.

In this window we see the default coloring rules that we have in Wireshark, including rules for TCP and other protocols events, routing packets and others.



Creating customize coloring rules



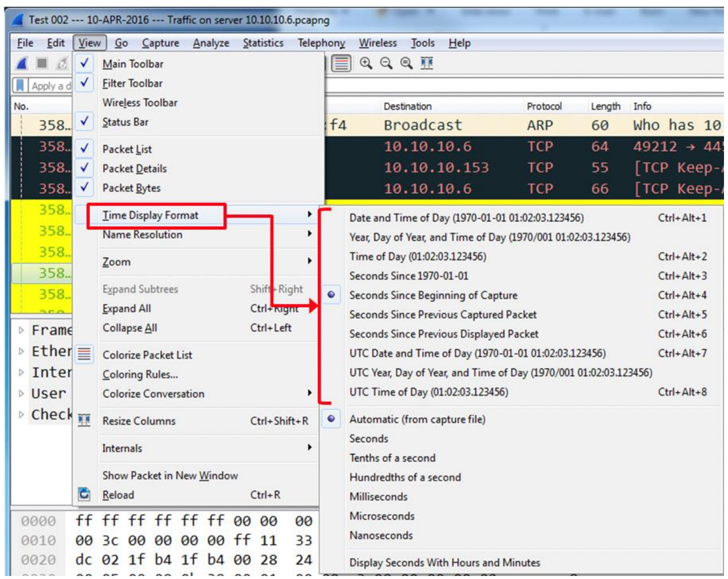
To go to the coloring rules continue as follows:

1. For a new coloring rule, click on the new tab, and you will get the following window on the right.
2. In the Name field, fill in the name of the rule. For example, fill in NTP for the Network Time Protocol.
3. In the Filter field, fill the filter string, that is what you want the rule to show (we will talk about display filters in Chapter 4, Using Display Filters).
4. Click on the Foreground Color button and choose the foreground color for the rule. This will be the foreground color of the packet in the packet list.
5. Click on the Background Color button and choose the background color for the rule. This will be the Background color of the packet in the packet list.
6. Click on the Delete button (the - sign to the left of the +) to delete coloring rule.
7. Click on the Duplicate button (to the right of the - button) if you want to edit an existing rule.
8. You can also click on Import button to import an existing coloring scheme, or
9. click on the Export rule for exporting the current scheme.

See also

- You can find various types of coloring schemes on: http://wiki.wireshark.org/ColoringRules, along with many other examples in a simple Internet search.
- To use one of the coloring rules files listed here, download it to your local machine, select **View | Coloring Rules** in Wireshark, and click the **Import...** button.

Practice 1: Using time values and summaries



- To configure the time format, go to the **View** menu, and under **Time Display Format** you will get the following window:

How to do it...

- **Date and Time of Day (the first three options)**: This will be good to configure when you troubleshoot a network with time-dependent events; for example, when you know about an event that happens in specific times, and you want to look at what happens on the network at the same time.
- **Seconds Since 1970-01-01**: Time in seconds since January 1, 1970. Epoch is an arbitrary date chosen as a reference time for a system, and January 1, 1970 was chosen for Unix and Unix-like systems.
- **Seconds Since Beginning of Capture**: The default configuration.

Continue..

- **Seconds Since Previous Captured Packet**: This is also a common feature that enables you to see time differences between packets. This can be useful when monitoring time-sensitive traffic such as TCP connections, live video streaming, VoIP calls, and so on when time differences between packets is important.
- **Seconds Since Previous Displayed Packet**: This is a useful feature, that can be used when you configure a display filter, and only a selected part of the captured data is presented (for example, a TCP stream). In this case, you will see the time difference between packets that can be important in some applications.
- **UTC Date and Time of Day**: Provides with relative UTC time. The lower part of the sub-menu provides the format of the time display. Change it only if a more accurate measurement is required.
- You can use also Ctrl + Alt + any numbered digit key on the keyboard for the various options.

Spot Network and Application Issues with Time Values and Summaries

SESSION 4

Day 4 - Spot Network and Application Issues with Time Values and Summaries

- Examine the Delta Time (End-of-Packet to End-of-Packet)
- Set a Time Reference
- Compare Timestamp Values
- Compare Timestamps of Filtered Traffic
- Enable and Use TCP Conversation Timestamps
- Compare TCP Conversation Timestamp Values
- Determine the Initial Round Trip Time (iRTT)
- Troubleshooting Example Using Time
- Analyze Delay Types

Pcapng vs pcap format

- Note that packets captured using the pcap file formats cannot define **nanosecond timestamp values**. These features are included in pcap-ng which is documented at wiki.wireshark.org/Development/PcapNg.
- For more details on the pcap file format, refer to wiki.wireshark.org/Development/LibpcapFileFormat

Troubleshooting Checklist

- **Verify** Trace File Integrity and Basic Communications
- **Focus** on Complaining User's Traffic
- **Detect** and **Prioritize** Delays
- Look for **Throughput** Issues
- Check **Miscellaneous** Traffic Characteristics
- TCP-Based Application:
  - **Determine** TCP Connection Issues/Capabilities
  - **Identify** TCP Issues
- UDP-Based Application:
  - Identify **Communication** Issues
- **Spot** Application Errors

Verify Trace File Integrity and Basic Communications

- Look for ACKed Unseen Segment (**tcp.analysis.ack_lost_segment** filter)
- Verify traffic from the complaining user's machine is visible. If not…
  - Ensure the host is running.
  - Test the host's connectivity (Can it communicate with another host?).
  - Recheck capture location and process.
  - Consider a resolution problem.
- Verify resolution process completion
  - DNS queries/successful responses (consider cache use). See Detect DNS Errors starting on
  - ARP requests/responses (consider cache use). See MAC Address Resolution – Local Target and MAC Address Resolution – Remote Target on page 97 of Troubleshooting with Wireshark, 1st Edition.

Focus on Complaining User's Traffic

- Filter **on** related traffic (such as tcp.port==80 && ip.addr==10.2.2.2). See Filter on a Host, Subnet or Conversation, Filter on an Applications Based on Port Number, Filter on Field Existence or Field Value.
- Filter **out** unrelated traffic (such as !ip.addr==239.0.0.0/8 or perhaps !bootp).
- Export related traffic to a separate trace file (**File | Export Specified Packets**).

Detect and Prioritize Delays

- Sort and identify high delta times (**Edit | Preferences | Columns | Add | Delta time displayed**).
- Sort and identify high TCP delta times (**tcp.time_delta column**).
  - If Expert Infos items are seen, examine the Errors, Warnings and Notes listings.
  - Consider "acceptable delays" (such as delays before TCP FIN or RST packets).
- Measure path latency (Round Trip Time) using delta times in TCP handshake
  - Capturing at client: measure delta from TCP SYN to SYN/ACK
  - Capturing at server: measure delta from SYN/ACK to ACK
  - Capturing in the infrastructure: measure delta from SYN to ACK

… continue

- Measure server response time
  - TCP-based application: measure from ACK to response, not request to ACK
  - Use Wireshark's response time function if possible (such as **dns.time, smb.time**, and **http.time**)

- Measure client latency
  - How long did it take for the client to make the next request?
  - Consider "acceptable delays" (such as a delay before an HTTP GET).

Look for Throughput Issues

- Build the Golden Graph (IO Graph with “Bad TCP” on Graph 2).
- Click on low throughput points to jump to problem spots in the trace file.
- Look at traffic characteristics at low throughput points.
- Consider using an Advanced IO Graph to detect delays (such as **tcp.time_delta**).

Check Miscellaneous Traffic Characteristics

- Check packet sizes during file transfer (Length column).
- Check IP DSCP for prioritization.
- Check 802.11 Retry bit setting (**wlan.fc.retry == 1**).
- Check for ICMP messages.
- Check for IP fragmentation.

TCP-Based Application: Determine TCP Connection Issues/Capabilities

- Look for unsuccessful TCP handshakes.
  - SYN, no answer
  - SYN, RST/ACK
- Examine the TCP handshake Options area.
  - Check MSS values.
  - Check for Window Scaling and Scale Factor.
  - Check for Selective ACK (SACK).
  - Check for TCP Timestamps (especially on high-speed links).

TCP-Based Application: Identify TCP Issues

- Launch the Expert Infos window.
  - Consider number of errors, warnings and notes
  - Consider impact of each item
- Check the Calculated window size field values (**tcp.window_size**).
- Examine unexpected TCP RSTs.

Spot the issue

UDP-Based Application: Identify Communication Issues

- Look for unsuccessful requests.
  - Request, no answer
- Look for repeated requests.

Spot Application Errors

- Filter for application error response codes (such as **sip.Status-Code >= 400**).

difference between capture filters and display filters?

- Capture filters are applied to traffic during the **capture process only**. Capture filters cannot be applied to existing trace files.
- Display filters can be used while capturing, but do not limit the packet you capture—display filters only limit what is visible. Display filters can be applied to existing trace files. Each filter type uses a different filter syntax.

About: iRTT

- The Wireshark initial Round Trip Time (iRTT) value is calculated when the **first two packets of a TCP handshake are seen** {SYN, SYN/ACK}. This value will remain the same for the entire TCP conversation. {tcp.analysis.initial_rtt}
- When you graph RTT in an IO graph, latency times are depicted between a data packet and the subsequent acknowledgment packet.
- You can always do your own handshake analysis and filter on {tcp.flags.syn==1} to find the start of the conversation and then set time deltas to calculate individual session RTTs.

Create Additional Time Columns

- If you want to view two or more Time columns in your Packet List pane, use **Edit | Preferences** to add a predefined Time column value or expand the **Frame** header, right click on a time field and select **Apply As Column**. Alternately, select Edit | Preferences | Columns | Add and select one of the following time-related field types:
  - **Absolute date and time**—based on the date and time of the capturing host (this is the same as the Date and Time of Day setting)
  - **Absolute time**—based on the time of the capturing host (this is the same as the Time of Day setting)
  - **Delta time (conversation)—**time from the end of one packet to the end of the next packet in a conversation
  - **Delta time displayed**—time from the end of one packet to the end of the next packet of displayed packets only (this is the same as Seconds Since Previous Displayed Packet)

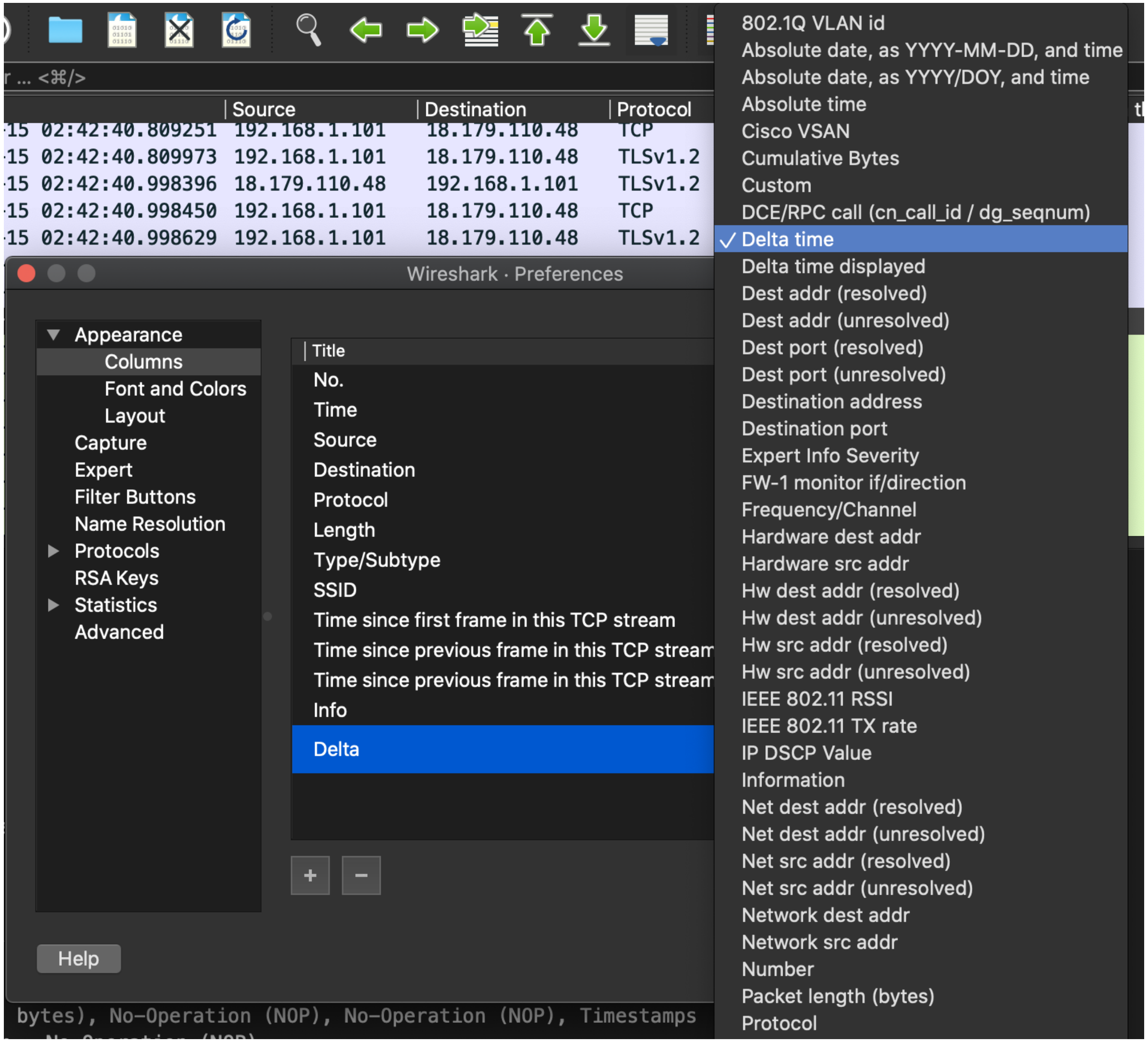
- **Relative time** - time from the first packet in the trace file (this is the same as the Seconds Since Beginning of Capture setting)
- **Relative time (conversation)** - time from the first packet in the trace file for the conversation only
- **Time (format as specified)** - this setting displays the value set using View | Time Display Format

* Using two Time columns you can easily compare the arrival packet time (Time since Beginning of Capture) to the delta time (Time since Previous Displayed Packet).

TOGETHER WE TRANSFORM

Spot Network and Application Issues with Time Values and Summaries

- With this scenario, we will use a sample PCAP file and start analyzing the issues base on the time.
- Add column to your Wireshark interface with Delta time column



Examine delta time (end-of-packet to end-of-packet)

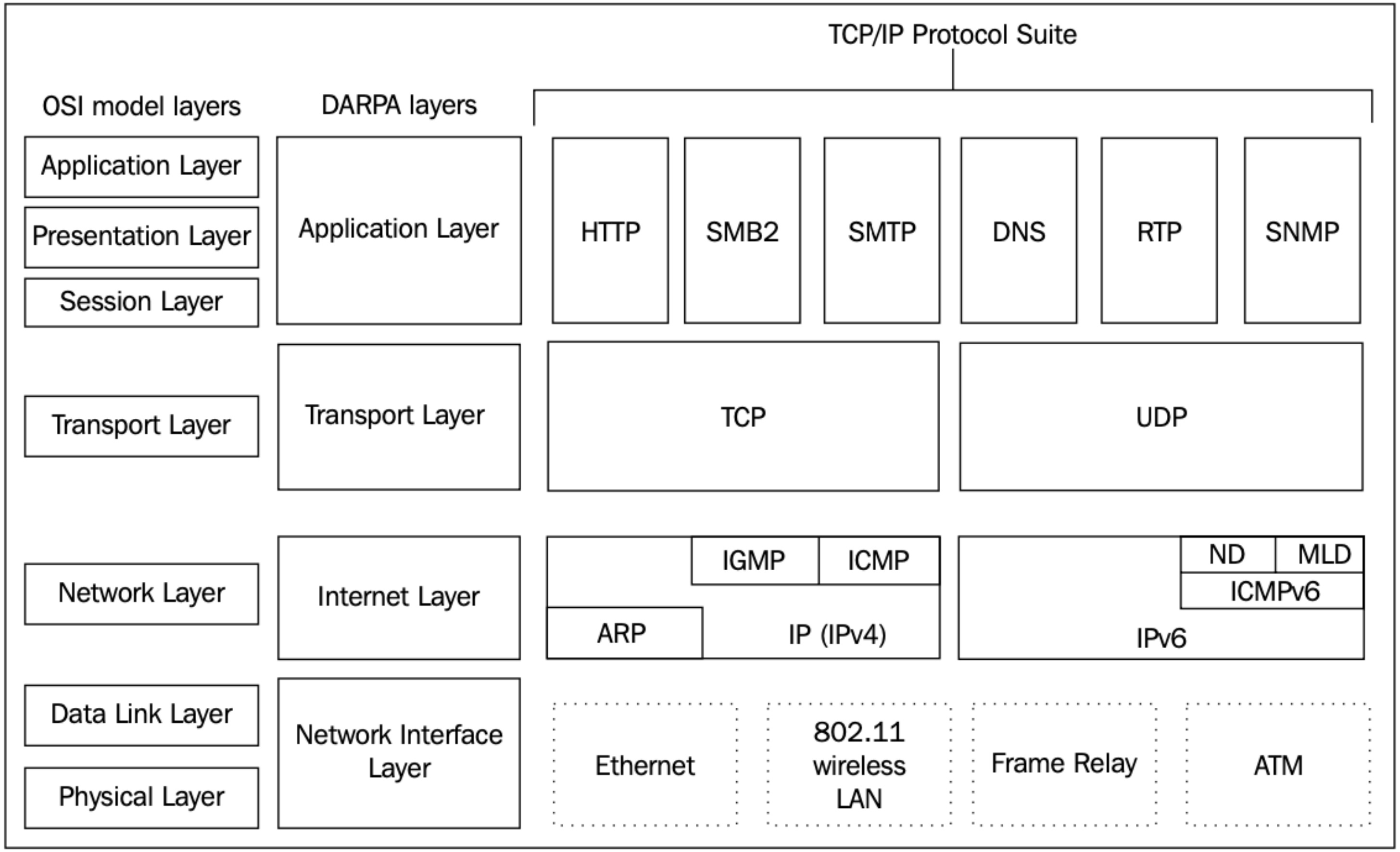
- Add delta column
- Look at Statistics -> Conversations to identify connection of interest
- Review 3-way handshake
  - iRTT
  - MSS
  - Window scale
- Adjust columns / config as needed

TCP/IP Communications and Resolutions Overview

TCP: This is a **connection-oriented protocol**, often called a reliable protocol. Here, firstly, a dedicated channel is created between two hosts and then data is transferred. Then, the sender sends equally partitioned chunks, over the dedicated channel, and then, the receiver sends the acknowledgement for every chunk received. Most commonly, the sender waits for a particular time after which it sends the same chunk again for assurance. For example, if you are downloading something, TCP is the one that takes care and makes sure that every bit is transferred successfully.

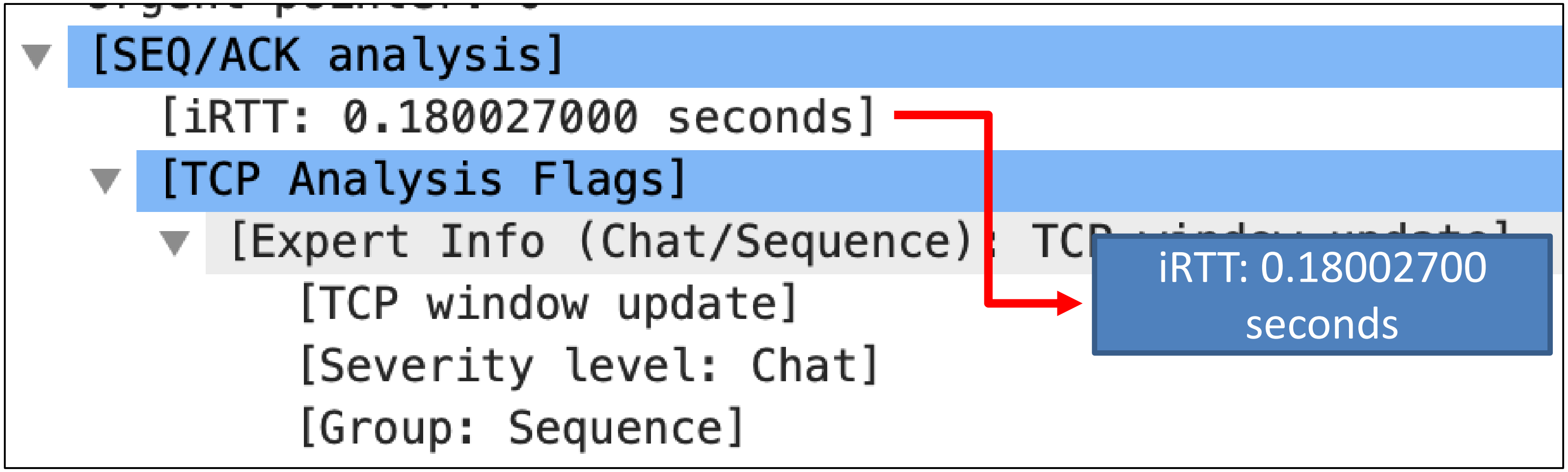
Flag field name	Description
URG (urgent)	This indicates the **Urgent Pointer** field (after the TCP header checksum) that should be examined. This flag is normally 0; the **Urgent Pointer** field is only examined if this bit is set.
ACK (acknowledgment)	This is the acknowledgment packet.
PSH (push)	This indicates whether the sending node's TCP stack should bypass any buffering and pass the data directly to the network and on to the receiving application.
RST (reset)	This is used to close the connection explicitly.
SYN (synchronize)	This is used to synchronize sequence numbers and used in a three-way TCP session initiation handshake process.
FIN (finish)	This is used when the transaction is finished. This does not mean that the connection is to be closed explicitly, but is commonly seen at the end of sessions.

The OSI and DARPA reference models



Troubleshooting and analysing network packets

SESSION 5



What is iRTT

The Wireshark initial Round Trip Time (iRTT) value is calculated when the **first two packets of a TCP handshake are seen {SYN, SYN/ACK}.** This value will remain the same for the entire TCP conversation. {**tcp.analysis.initial_rtt**}

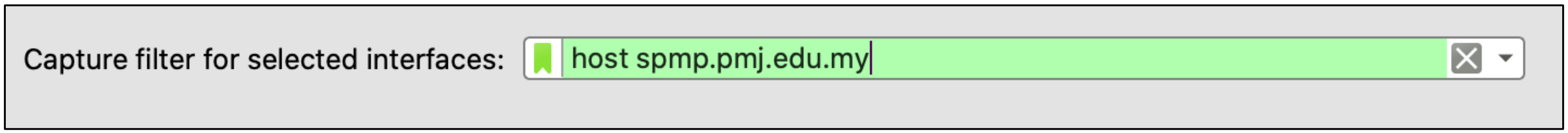
When you graph RTT in an IO graph, latency times are depicted between a data packet and the subsequent acknowledgment packet.

You can always do your own handshake analysis and filter on {tcp.flags.syn==1} to find the start of the conversation and then set time deltas to calculate individual session RTTs.

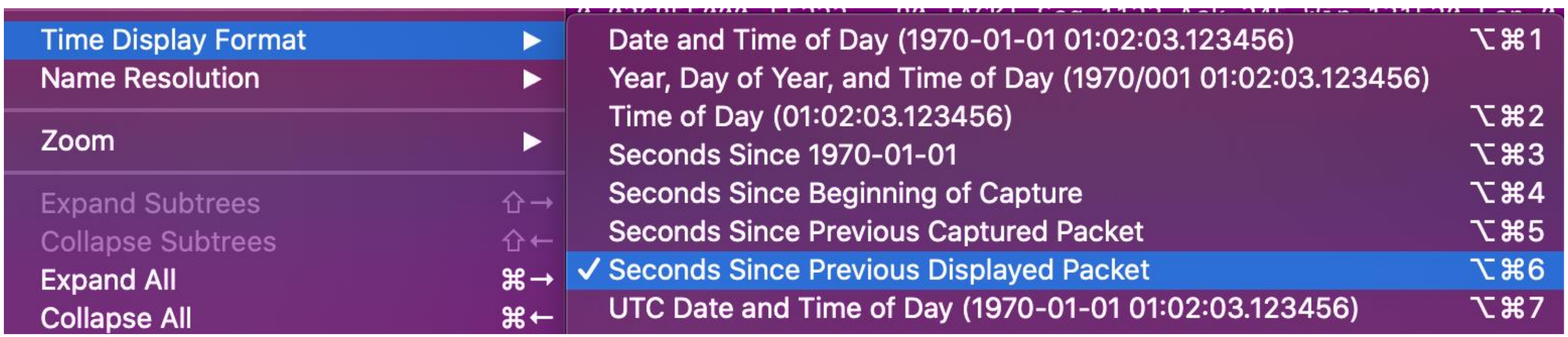
Source: https://osqa-ask.wireshark.org/questions/21813/how-is-rtt-calculated

How to do it?

1. Start with capturing filter.



2. Set time to **Seconds Since Previous Displayed Packet**.



.. continue

```
▼ [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 13]
    [The RTT to ACK the segment was: 0.036786000 seconds]
    [iRTT: 0.036855000 seconds]
```

4. From the **Packet Details**, find the **SEQ/ACK analysis** from **Transmission Control Protocol** and look at the **iRTT detail**.

5. Add column to the Packet List pane.

Getting the sample

- We will analyzing a sample PCAP based on the official wireshark sample capture.
- Download the pcap here: https://wiki.wireshark.org/SampleCaptures
- ARP Request: https://wiki.wireshark.org/SampleCaptures?action=AttachFile&do=get&target=rarp_request.cap
- ICMP: https://wiki.wireshark.org/SampleCaptures?action=AttachFile&do=get&target=ipv4frags.pcap
- TCP and JPeG: https://wiki.wireshark.org/SampleCaptures?action=AttachFile&do=get&target=http_with_jpegs.cap.gz

Measure Slow DNS Response Time

- Open dns-slow.pcapng. Select **View | Time Display Format | Seconds since Previous Displayed Packet**.
- How much time elapsed between the first and second DNS query for www.ncmec.org? **You should see1.000620 seconds**.
- How much time elapsed between the first and second DNS response for www.ncmec.org? Right click on the first DNS response and set a time reference to measure this value. (By the time the second DNS response arrived, the client had closed the listening port for the DNS response –that's why the client sent an ICMP Destination Unreachable/Port Unreachable response. You should see 0.184489 seconds between the first and second DNS response packet.
- How much time did it take for the server to answer the DNS query in packet 98? You should see **.207250** seconds elapsed between the DNS query in packet 98 and the DNS response in packet 107.

Measure a High Latency Path

- Open http-download-good.pcapng. Reset the Time column to **Seconds since Previous Displayed Packet**. What is the latency time between the first and second packets of the TCP handshake (packets1 and 2)? You should see 0.179989 seconds.
- Sort the **Time column**. What is the largest time delay in the trace file? You should see 2.753091 seconds is the largest time delay in the trace file.
- Sort by the Number (No.) column. What happened around the largest time delay in the trace file? You should see a TCP window update process occurred at this time.

tshark

NAME

tshark – Dump and analyze network traffic

SYNOPSIS

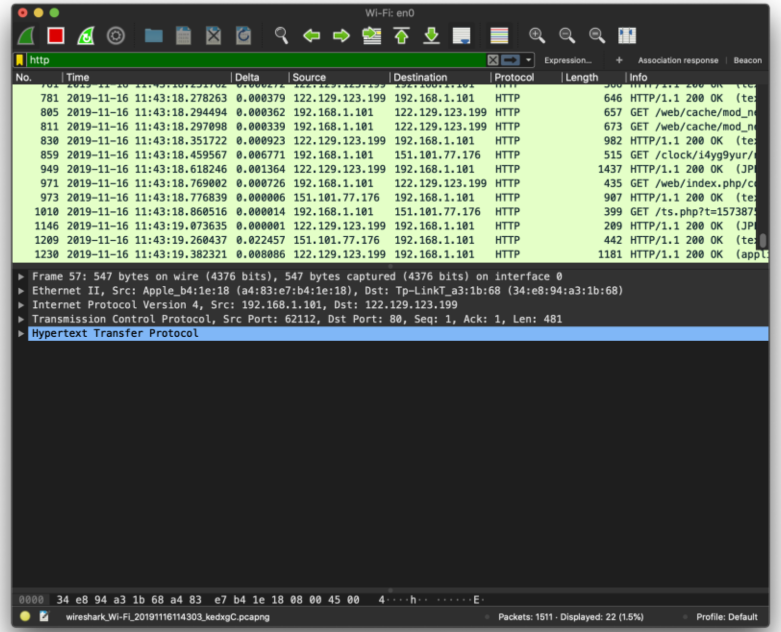
tshark [**-2**] [**-a** <capture autostop condition>] ... [**-b** <capture ring buffer option>] ... [**-B** <capture buffer size>] [**-c** <capture packet count>] [**-C** <configuration profile>] [**-d** <layer type>==<selector>,<decode-as protocol>] [**-D**] [**-e** <field>] [**-E** <field print option>] [**-f** <capture filter>] [**-F** <file format>] [**-g**] [**-h**] [**-H** <input hosts file>] [**-i** <capture interface>|-] [**-j** <protocol match filter>] [**-I**] [**-K** <keytab>] [**-l**] [**-L**] [**-n**] [**-N** <name resolving flags>] [**-o** <preference setting>] ... [**-O** <protocols>] [**-p**] [**-P**] [**-q**] [**-Q**] [**-r** <infile>] [**-R** <Read filter>] [**-s** <capture snaplen>] [**-S** <separator>] [**-t** a|ad|adoy|d|dd|e|r|u|ud|udoy] [**-T** ek|fields|json|jsonraw|pdml|ps|psml|tabs|text] [**-u** <seconds type>] [**-U** <tap_name>] [**-v**] [**-V**] [**-w** <outfile>|-] [**-W** <file format option>] [**-x**] [**-X** <eXtension option>] [**-y** <capture link type>] [**-Y** <displaY filter>] [**-M** <auto session reset>] [**-z** <statistics>] [**--capture-comment** <comment>] [**--list-time-stamp-types**] [**--time-stamp-type** <type>] [**--color**] [**--no-duplicate-keys**] [**--export-objects** <protocol>,<destdir>] [**--enable-protocol** <proto_name>] [**--disable-protocol** <proto_name>] [**--enable-heuristic** <short_name>] [**--disable-heuristic** <short_name>] [<filter>]

tshark -G [<report type>] [--elastic-mapping-filter <protocols>]

Source: https://www.wireshark.org/docs/man-pages/tshark.html

Analyze http traffic

1. Open your internet browser
2. Click on "Capture > Interfaces". A pop up window will show up.
3. You probably want to capture traffic that goes through your ethernet driver. Click on the Start button to start capturing traffic via this interface.
4. Visit the URL that you wanted to capture the traffic from. For example, open http://www.polimas.edu.my
5. Go back to your Wireshark screen and press Ctrl + E to stop capturing.
6. On the filter pane, input ‘http’ to see the http traffic



Analyzing File Transfer Protocol (FTP) Traffic

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs. The attempt in this specification is to satisfy the diverse needs of users of maxi-hosts, mini-hosts, personal workstations, and TACs, with a simple, and easily implemented protocol design. This paper assumes knowledge of the Transmission Control Protocol (TCP) [2] and the Telnet Protocol [3]. These documents are contained in the ARPA-Internet protocol handbook [1].

Source: https://tools.ietf.org/html/rfc959

Analyzing ftp problems

- **File Transfer Protocol** (**FTP**) is a protocol created for transferring files over TCP/IP across a network. FTP is a protocol that runs over TCP ports 20 and 21 for the data and control connections (FTP commands) respectively
- FTP has two modes of operation
  - **Active mode** (**ACTV**): In this mode, the client initiates a control connection to the server, and the server initiates a data connection to the client
  - **Passive mode** (**PASV**): In this mode, the client initiates the control and data connections to the server

Both types of connections can be implemented, and they will be explained later in this recipe in the *How it works...* section.

Getting ready

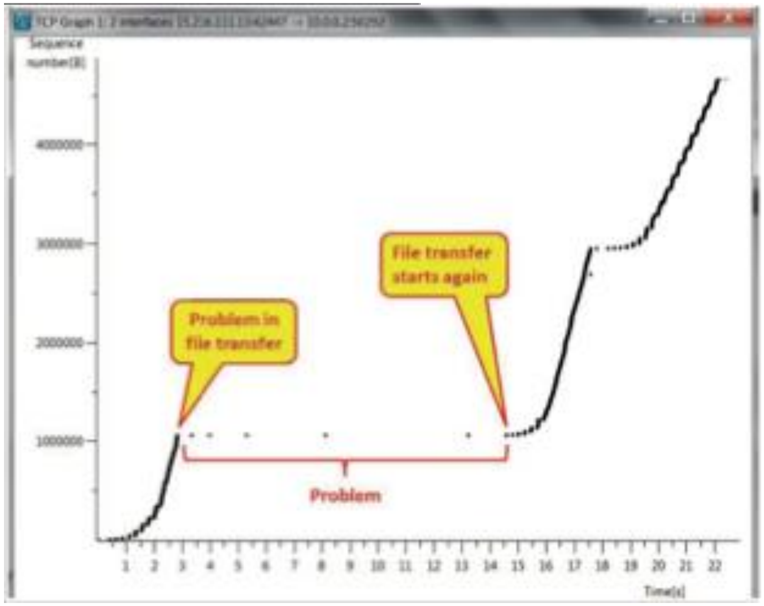
- When working with FTP, if you suspect any connectivity or slow response problems, configure port mirror to one of the following:
  - The FTP server port
  - The client port
  - A link that the traffic crosses
- If required, configure a capture or display filter.

If required, configure a capture or display filter.

checking FTP performance problems

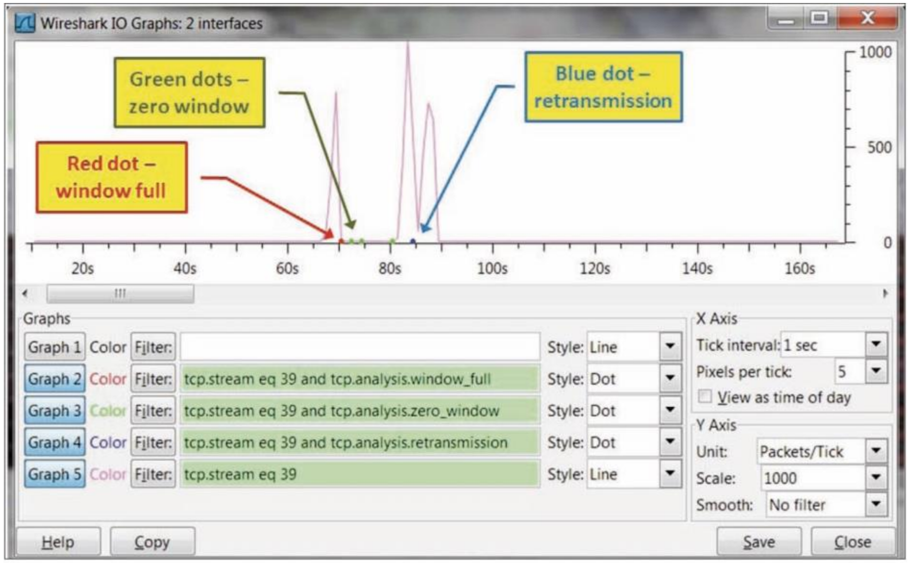
1. First, check for any Ethernet, IP, or TCP problems. In many cases, slow responses happen due to networking problems and not necessarily due to application problems.
2. Check for TCP retransmissions and duplicate ACKs. Check if they are on the entire traffic or only on the FTP connection.
3. If you get it on various connections, it is probably due to a slow network that influences the entire traffic.
4. If you get it only on FTP connections to the same server or client, it can be due to a slow server or client.
5. When you are copying a single file in an FTP file transfer, you should get a straight line in the IO graph and a straight gradient in the TCP stream graph (time-sequence).

- In the following screenshot, we can see what a bad FTP looks like in the TCP stream graph (time-sequence):



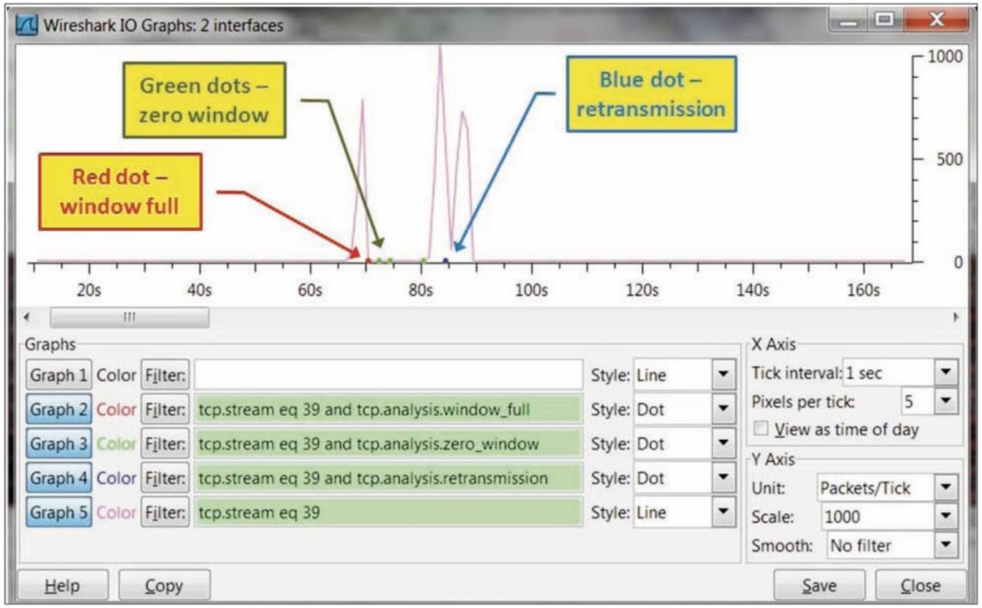
Bad FTP in tcp stream graph

- In the screenshot on the left, we can see



IO graph for bad tcp

- Following screenshot shows the IO Graph (configured with filters)



What does TCP Zero Window mean?

- Zero Window is something to investigate.
- TCP Zero Window is when the Window size in a machine remains at zero for a specified amount of time.
- TCP Window size is the amount of information that a machine can receive during a TCP session and still be able to process the data. Think of it like a TCP receive buffer. When a machine initiates a TCP connection to a server, it will let the server know how much data it can receive by the Window Size.
- In many Windows machines, this value is around 64512 bytes. As the TCP session is initiated and the server begins sending data, the client will decrement it's Window Size as this buffer fills. At the same time, the client is processing the data in the buffer, and is emptying it, making room for more data. Through TCP ACK frames, the client informs the server of how much room is in this buffer. If the TCP Window Size goes down to 0, the client will not be able to receive any more data until it processes and opens the buffer up again. In this case, Protocol Expert will alert a "Zero Window" in Expert View.
- Troubleshooting a Zero Window For one reason or another, the machine alerting the Zero Window will not receive any more data from the host. It could be that the machine is running too many processes at that moment, and its processor is maxed. Or it could be that there is an error in the TCP receiver, like a Windows registry misconfiguration. Try to determine what the client was doing when the TCP Zero Window happened.

Source: flukenetworks.com

Configuring Ethernet, ARP, host, and network filters

- In this recipe we will discuss how to configure filters of layers 2 and 3, that is, Ethernet- and IP-based filters respectively. We will also discuss **Address Resolution Protocol** (**ARP**) filters.
- In layer 2 we will configure Ethernet-based filters, while in layer 3 we will configure IP-based filters. In Ethernet we have filters based on the Ethernet frame and the MAC address, while in IP we have filters based on the IP packet and address.
- The common frame delta filters are as follows:
  - frame.time_delta: This is used for the time delta between the current and previously captured frames; this will be used in statistical graphs displayed in *Chapter 5*, *Using Advanced Statistics Tools*
  - frame.time_delta_displayed: This is used for the time delta between current and previously displayed frames;

- Since the time between frames can influence TCP performance significantly, we will use the frame.time_delta parameters in statistical graphs for monitoring TCP performance.
- The common layer 2 (Ethernet) filters are as follows:
  - eth.addr == <MAC Address>: This is used to display a specific MAC address
  - eth.src == <MAC Address>: This is used to get the source MAC address
  - eth.dst == <MAC Address>: This is used to get the destination MAC address
  - eth.type == <Protocol Type (Hexa)>: This is used to get the Ethernet protocol types

- The common ARP filters are as follows:
  - arp.opcode == <value>: This is used for ARP requests/responses
  - arp.src.hw_mac == <MAC Address>: This is used to capture the ARP address of the sender
- The common layer 3 (IP) filters are as follows:
  - ip.addr == <IP Address>: This is used to get the source or destination IP address
  - ip.src == <IP Address>: This is used to get the source IP address
  - ip.dst == <IP Address>: This is used to get the destination IP address
  - ip.ttl == <value>, ip.ttl < value>, or ip.ttl > <value>: This is used to get IP TTL (Time To Live) values
  - ip.len = <value>, ip.len > <value>, or ip.len < <value>: This is used to get IP packet length values
  - ip.version == <4/6>: This is used to get the IP protocol version (Version 4 or Version 6)

Here we will see some common examples for layer 2 and 3 filters.

Address format	Syntax	Example

Ethernet (MAC) address	`eth.addr == xx:xx:xx:xx:xx:xx`
	Here, x = 0 to f.
	`eth.addr == xx-xx-xx-xx-xx-xx`
	Here, x = 0 to f.
	`eth.addr == xxxx.xxxx.xxxx`
	Here x = 0 to f.
		`eth.addr == 00:50:7f:cd:d5:38`
		`eth.addr == 00-50-7f-cd-d5-38`
		`eth.addr == 0050.7fcd.d538`

Broadcast MAC address	`Eth.addr == ffff.ffff.ffff`	
IPv4 host address	`ip.addr == x.x.x.x` Here, x = 0 to 255.	`Ip.addr == 192.168.1.1`
IPv4 network address	`ip.addr == x.x.x.x/y` Here x = 0 to 255, y = 0 to 32.	`ip.addr == 192.168.200.0/24` This covers all the addresses in the network 192.168.200.0 mask 255.255.255.0.

IPv6 host address	`ipv6.addr == x:x:x:x:x:x:x:x`
	`ipv6.addr == x::x:x:x:x`
	Here in the format of nnnn, n = 0 to f (Hex).
		`ipv6.addr == fe80::85ab:dc2e:ab12:e6c7`

IPv6 network address	`ipv6.addr == x::/y` Here x = 0 to f (Hex) and y = 0 to 128.	`ipv6.addr == fe80::/16` This covers all the addresses that start with the 16 bits fe80.

The table refers to ip.addr and ipv6.addr filter strings. The value for any field that has an IP address value can be written the same way.

ethernet and ARP filters

Ethernet filters

- These are classified into two categories:
  - To display only packets sent from or to specific MAC addresses, use something like these: eth.src == 10:0b:a9:33:64:18 and eth.dst == 10:0b:a9:33:64:18
  - To display only broadcasts, use Eth.dst == ffff.ffff.ffff

ARP filters

- These are classified into two categories:
  - To display only ARP requests, use arp.opcode == 1
  - To display only ARP responses, use arp.opcode == 2

1. To display only packets from a **specific IP address**, use something like this: ip.src==192.168.1.4
2. To display only packets that are **not from a specific address**, use something like this: !ip.src==192.168.1.4
3. To display only packets between **two hosts**, use something like these: ip.addr==192.168.1.104 and ip.addr==192.168.1.100
4. To display only packets that are sent to **multicast IP addresses**, use something like this: ip.dst == 224.0.0.0/4
5. To display **only packets** coming from the network 192.168.1.0/24 (mask 255.255.255.0), use ip.src==192.168.1.0/24

Ipv6

- To display only IPv6 packets to/from specific addresses, use something like the following:
  - ipv6.addr == ::1
  - ipv6.addr == 2008:0:130F:0:0:09d0:666A:13ab
  - ipv6.addr == 2006:0:130f::9c2:876a:130b
  - ipv6.addr == ::

Complex filters

- To check for packets sent from the network 10.0.0.0/24 to a website that contains the word packt, use ip.src == 10.0.0.0/24 and http.host contains "packt"
- To check for packets sent from the network 10.0.0.0/24 to websites that end with .com, use ip.addr == 10.0.0.0/24 and http.host matches "\.com$"
- To check for all the broadcasts from the source IP address 10.0.0.0, use ip.src == 10.0.0.0/24andeth.dst == ffff.ffff.ffff
- To check for all the broadcasts that are not ARP requests, use not arp and eth. dst == ffff.ffff.ffff
- To check for all the packets that are not ICMP, use !arp || !icmp, and to check for all the packets that are not ARP, use not arp or not icmp

Domain name system (DNS)

- DNS is the system used to resolve store information about domain names including IP addresses, mail servers, and other information.

History

- DNS was invented in 1982-1983 by Paul Mockapteris and Jon Postel.

Protocol dependencies

- TCP/UDP: Typically, DNS uses TCP or UDP as its transport protocol. The well known TCP/UDP port for DNS traffic is 53.

Display Filter

- A complete list of DNS display filter fields can be found in the display filter reference
- Show only the DNS based traffic: dns

Capture Filter

- You cannot directly filter DNS protocols while capturing if they are going to or from arbitrary ports. However, DNS traffic normally goes to or from port 53, and traffic to and from that port is normally DNS traffic, so you can filter on that port number.
- Capture only traffic to and from port 53: udp.port== 53

Analyzing DNS traffic

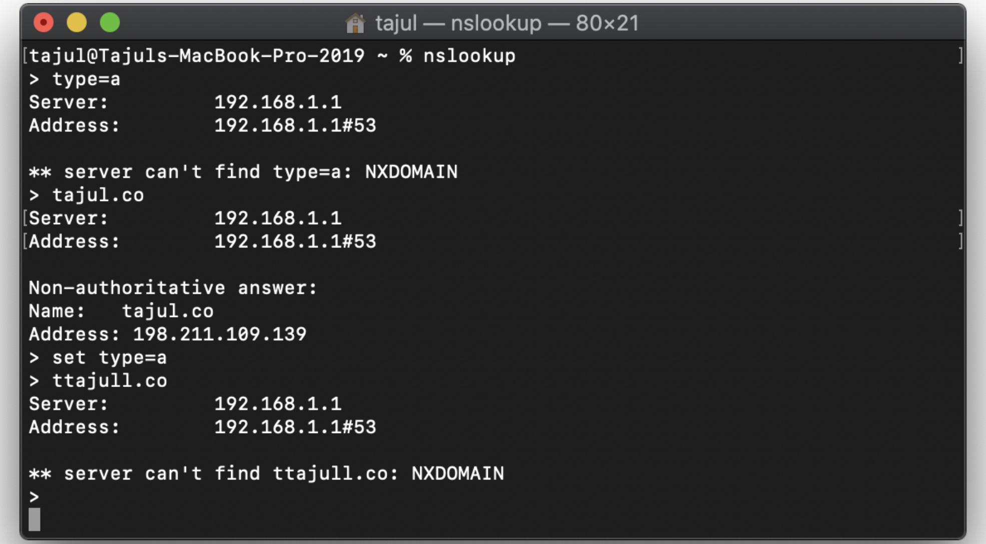
- As we know, the DNS protocol runs over a UDP or TCP. There are various response code that relate to DNS errors that range from 0 to 21. The dissectors present in Wireshark do know about response code. Using this, Wireshark is able to show you messages relevant to the error code.
- To replicate an error, I will visit a website that does not exist on the Web; hence, I will receive an error. But my gateway does not know about this, so it will try to resolve the IP address associated with that name.
- In return, we will see a DNS response containing an error. The infrastructure is the same that we used in the preceding examples.

Capturing the DNS traffic using nslookup

- You can replicate the scenario step by step with me or do it later once you finish reading. Follow these steps to replicate the scenario:
- Open Wireshark and start capturing. Let it run in the background.
- Open a terminal (Command Prompt) of whichever operating system you are using, type nslookup in it, and press *Enter*.
- Now, you'll enter the interactive mode of the nslookup tool. If you are not aware of the tool, do read about it before you proceed. There are plenty of documents available for the tool. Refer to the following screenshot:

Looking the unknown domain

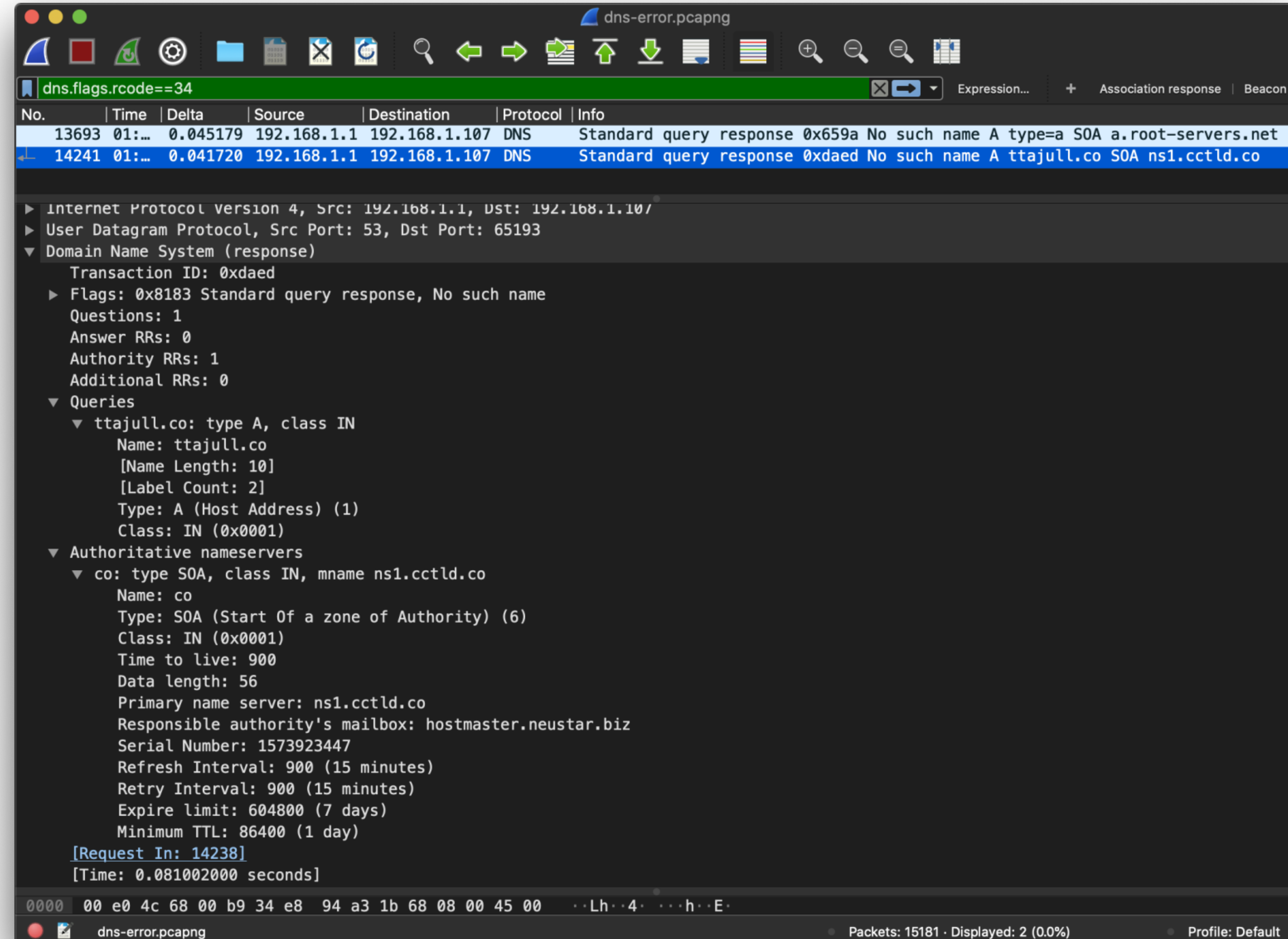
- To generate DNS error response code, just type any domain name and press *Enter*. Before you specify a domain change the type of query to A by using the set type=a command and then give the domain you want.
- First, we can try the same for a domain that exists, such as tajul.co Then, you can try it for the nonexistent domain. e.g: ttajull.co.
- The preceding screenshot shows the various IP addresses that are associated with the tajul.co domain. The domain already exists. That's why we are able to see the reply.



- I typed my name in place of the domain name and pressed *Enter*, and this is what I saw because there was no domain with that name. The DNS server was not able to resolve an IP address, hence resulting in the reply server can't find.
- Now, you can go back to Wireshark and stop the capture process. We will now start analyzing error code.
- The best option would be to segregate the DNS error response code from the normal frames in the trace file that we have. To achieve this, apply the dns. flags.rcode ==3 display filter, which means that the shown DNS response frame with error code 3 is for nonexistent domains. For more information on DNS error code, visit https://tools.ietf.org/html/rfc 2929.

Looking the error for dns request

- Once you have applied the preceding display filter, you will only see relevant packets matching your filter expression.
- Filter capture for DNS error code: dns.flags.rcode==3


dns-error.pcapng
dns.flags.rcode==34
Expression...
Association response
Beacon
No. | Time | Delta | Source | Destination | Protocol | Info
13693 01:… 0.045179 192.168.1.1 192.168.1.107 DNS Standard query response 0x659a No such name A type=a SOA a.root-servers.net
14241 01:… 0.041720 192.168.1.1 192.168.1.107 DNS Standard query response 0xdaed No such name A ttajull.co SOA ns1.cctld.co
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.107
User Datagram Protocol, Src Port: 53, Dst Port: 65193
Domain Name System (response)
Transaction ID: 0xdaed
Flags: 0x8183 Standard query response, No such name
Questions: 1
Answer RRs: 0
Authority RRs: 1
Additional RRs: 0
Queries
ttajull.co: type A, class IN
Name: ttajull.co
[Name Length: 10]
[Label Count: 2]
Type: A (Host Address) (1)
Class: IN (0x0001)
Authoritative nameservers
co: type SOA, class IN, mname ns1.cctld.co
Name: co
Type: SOA (Start Of a zone of Authority) (6)
Class: IN (0x0001)
Time to live: 900
Data length: 56
Primary name server: ns1.cctld.co
Responsible authority's mailbox: hostmaster.neustar.biz
Serial Number: 1573923447
Refresh Interval: 900 (15 minutes)
Retry Interval: 900 (15 minutes)
Expire limit: 604800 (7 days)
Minimum TTL: 86400 (1 day)
[Request In: 14238]
[Time: 0.081002000 seconds]
0000 00 e0 4c 68 00 b9 34 e8 94 a3 1b 68 08 00 45 00 ··Lh··4· ···h··E·
dns-error.pcapng
Packets: 15181 · Displayed: 2 (0.0%)
Profile: Default

Analyzing ARP traffic

- The Address Resolution Protocol is used to dynamically discover the mapping between a layer 3 (protocol) and a layer 2 (hardware) address. A typical use is the mapping of an IP address (e.g. 192.168.0.10) to the underlying Ethernet address (e.g. 01:02:03:04:05:06). You will often see ARP packets at the beginning of a conversation, as ARP is the way these addresses are discovered.
- ARP can be used for Ethernet and other LANs, ATM, and a lot of other underlying physical addresses (the list of hardware types in the ADDRESS RESOLUTION PROTOCOL PARAMETERS document at the IANA Web site includes at least 33 hardware types).

… continue

- ARP is used to dynamically build and maintain a mapping database between link local layer 2 addresses and layer 3 addresses. In the common case this table is for mapping Ethernet to IP addresses. This database is called the ARP_Table. Dynamic entries in this table are often cached with a timeout of up to 15 minutes, which means that once a host has ARPed for an IP address it will remember this for the next 15 minutes before it gets time to ARP for that address again.
- A peculiarity of ARP is that since it tries to reduce/limit the amount of network traffic used for ARP a host MUST use all available information in any ARP packet that is received to update its ARP_Table. Thus sometimes a host sends out ARP packets NOT in order to discover a mapping but to use this side effect of ARP and preload the ARP table of a different host with an entry. These special ARP packets are referred to as Gratuitous_ARPs and Wireshark will detect and flag the most common versions of such ARPs in the packet summary pane.

Gratuitous ARP

- Gratuitous_ARPs are more important than one would normally suspect when analyzing captures. So don't just ignore them or filter out ARP from your capture immediately. Consider that a normal host will always send out a Gratuitous_ARP the first thing it does after the link goes up or the interface gets enabled, which means that almost every time we see a Gratuitous_ARP on the network, that host that sent it has just had a link bounce or had its interface disabled/enabled. This is very useful information when troubleshooting networks. Remember though that you can only see these Gratuitous_ARPs (or any other ARPs for that matter) if your capture device is in the same Broadcast Domain as the host that originates the ARP packet.
- Several viruses send a lot of ARP traffic in an attempt to discover hosts to infect; see the ArpFlooding page.

Arp flooding : source- https://wiki.wireshark.org/ArpFlooding

- If you see a lot of ARP traffic from a single machine, looking for MAC addresses for many of the IP addresses on your local network, there might be a virus on your network that's scanning your network for machines to infect. It's been claimed that the Wootbot virus does this.
- This is not limited to Wootbot - i have observed during nachi outbreak networks very flooded with random arp and icmp requests which was very hard on L2/L3 devices - *Anith Anand*
- -updated 6th Mar 05 (NetworkFlooding)
- It is not just worms and viruses that can bring down the network or firewall - recently i was troubleshooting slow production network problem for a large organisation - intially i suspected it as some kinda virus outbreak or ddos attacks ..however thanks to wireshark - when i port spanned the firewall interfaces i noticed as much as 300,000 packets per min (5000 udp packets per second) in addition to the regular traffic was traversing through firewall (checkpoint) on single interface (double it for exit interface) which made it bleed badly - even simple ping across f/w interface will timeout during this event - the above traffic was created by faulty (or mis configured) syslog listenter service which was pumping those error messages - however i should also thank "pathping" utility found in XP as it helped me in zooming into the problem by providing RTT and Packet Drop rate across network use pathping with -n option to make it work for you faster - "Anith Anand"

Protocol dependencies

Layer 2 protocols:

- ATM: ARP can use ATM as its transport mechanism.
- Ethernet: ARP can use Ethernet as its transport mechanism. The assigned Ethernet type for ARP traffic is 0x0806.
- Other LANs: ARP can also be used on Token Ring, FDDI, and IEEE 802.11; the same assigned type is used.

Layer 3 protocols:

- IP: ARP can map IP addresses to layer 2 addresses.

filter

Display filter

- A complete list of ARP display filter fields can be found in the display filter reference
- Show only the ARP based traffic: arp

Capture filter

- You can filter ARP protocols while capturing.
- Capture only the ARP based traffic: arp

or:

- ether proto \arp

Capturing only ARP packets is rarely used, as you won't capture any IP or other packets. However, it can be useful as part of a larger filter string.

FIELD NAME	DESCRIPTION	TYPE	VERSIONS
arp.dst.atm_num_e164	Target ATM number (E.164)	Character string	1.0.0 to 3.0.6
arp.dst.atm_num_nsap	Target ATM number (NSAP)	Sequence of bytes	1.0.0 to 3.0.6
arp.dst.atm_subaddr	Target ATM subaddress	Sequence of bytes	1.0.0 to 3.0.6
arp.dst.drarp_error_status	DRARP error status	Unsigned integer, 2 bytes	1.8.0 to 3.0.6
arp.dst.hlen	Target ATM number length	Unsigned integer, 1 byte	1.0.0 to 3.0.6
arp.dst.htype	Target ATM number type	Boolean	1.0.0 to 3.0.6
arp.dst.hw	Target hardware address	Sequence of bytes	1.0.0 to 3.0.6
arp.dst.hw_ax25	Target AX.25 address	AX.25 address	1.10.0 to 3.0.6
arp.dst.hw_mac	Target MAC address	Ethernet or other MAC address	1.0.0 to 3.0.6
arp.dst.pln	Target protocol size	Unsigned integer, 1 byte	1.0.0 to 3.0.6
arp.dst.proto	Target protocol address	Sequence of bytes	1.0.0 to 3.0.6
arp.dst.proto_ipv4	Target IP address	IPv4 address	1.0.0 to 3.0.6
arp.dst.slen	Target ATM subaddress length	Unsigned integer, 1 byte	1.0.0 to 3.0.6
arp.dst.stype	Target ATM subaddress type	Boolean	1.0.0 to 3.0.6
arp.duplicate-address-detected	Duplicate IP address configured	Label	1.0.0 to 3.0.6
arp.duplicate-address-frame	Frame showing earlier use of IP address	Frame number	1.0.0 to 3.0.6
arp.hw.size	Hardware size	Unsigned integer, 1 byte	1.0.0 to 3.0.6
arp.hw.type	Hardware type	Unsigned integer, 2 bytes	1.0.0 to 3.0.6
arp.isgratuitous	Is gratuitous	Boolean	1.2.0 to 3.0.6
arp.opcode	Opcode	Unsigned integer, 2 bytes	1.0.0 to 3.0.6
arp.packet-storm-detected	ARP packet storm detected	Label	1.0.0 to 3.0.6
arp.proto.size	Protocol size	Unsigned integer, 1 byte	1.0.0 to 3.0.6
arp.proto.type	Protocol type	Unsigned integer, 2 bytes	1.0.0 to 3.0.6
arp.seconds-since-duplicate-address-frame	Seconds since earlier frame seen	Unsigned integer, 4 bytes	1.0.0 to 3.0.6
arp.src.atm_afi	AFI	Unsigned integer, 1 byte	2.0.0 to 3.0.6
arp.src.atm_afi.unknown	Unknown AFI	Label	2.0.0 to 3.0.6
arp.src.atm_data_country_code	Data Country Code	Unsigned integer, 2 bytes	2.0.0 to 3.0.6
arp.src.atm_data_country_code_group	Data Country Code (group)	Unsigned integer, 2 bytes	2.0.0 to 3.0.6
arp.src.atm_e.164_isdn	E.164 ISDN	Sequence of bytes	2.0.0 to 3.0.6
arp.src.atm_e.164_isdn_group	E.164 ISDN	Sequence of bytes	2.0.0 to 3.0.6
arp.src.atm_end_system_identifier	End System Identifier	Sequence of bytes	2.0.0 to 3.0.6

Display Filter Reference: Address Resolution Protocol

10 keys of troubleshooting steps

- Baseline "NormalTraffic
- Use Color
- Look Who's Talking: Examine Conversations and Endpoints
- Focus by Filtering
- Create Basic IO Graphs
- Examine the Expert System
- Follow the Streams
- Graph Bandwidth Use, Round Trip Time, and TCP Time/Sequence Information
- Watch Refusals and Redirections
- Examine Delta Time Values

Internet Control Message protocol

- Used by network devices such as routers to send error messages indicating that a requested services is not available, or a host or network router could not be reach.
- A control protocol
- Although it is transported as IP datagrams, it does not carry application data – instead , it carries information about the status of the network itself.

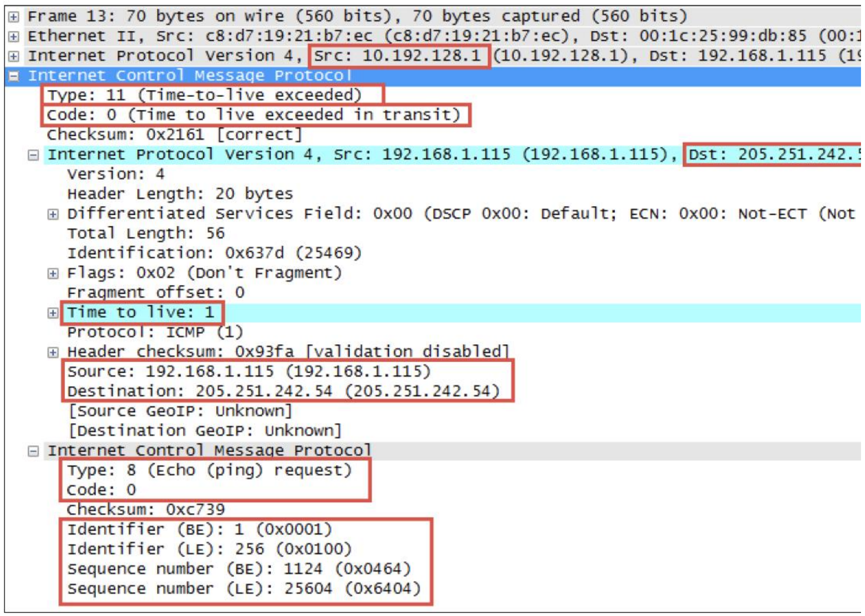
ICMP pings

- One of the most well-known uses of ICMP is to ping, wherein a device sends an ICMP echo request (**Type 8, Code 0**) packet to a distant host (via that host's IP address), which will (if the ICMP service isn't disabled or blocked by an intermediate firewall) respond with an ICMP echo reply (**Type 0, Code 0**) packet. Pings are used to determine whether the target host is available and can be reached over the network. By measuring the time that expires between ping requests and replies, we know the **round trip time (RTT)** delay time over the network path.

ICMP traceroutes

- A variation of ping functionality is used to perform a traceroute (also known as traceroute), which is a list of the IP addresses of the router interfaces that packets traverse to get from a sending device to a target host or device. The traceroutes are used to determine or confirm the network path taken from a sending device to a target host or device.
- A traceroute is accomplished by sending the ICMP echo request packets to a distant host just as in a normal ping, but with modifications to the Time-to-Live (TTL) field in the IP header of each packet. The traceroute function takes advantage of the fact that each router in a network path decrements the TTL value in a packet by 1, so as the packet traverses, the routers in a path and the TTL value will decrease accordingly along the way. If a router receives a packet with a TTL value of 1, it will send an ICMP TTL exceeded in transit (Type 11, Code 0) error message back to the sender (along with a copy of the request packet it received) and otherwise discard (not forward) the packet.
- The traceroute works by sequentially setting the TTL in multiple ICMP request packets to 1, then to 2, then 3, and so on, which results in each router in the network path sending TTL exceeded error messages back to the sender. Since these returned messages are sent by the in-path router using the IP address of the interface where the ICMP packet was received, the traceroute utility can build and display a progressive list of router interface IP addresses in the path and the RTT delay to each router

ICMP PACKET ANALYSIS

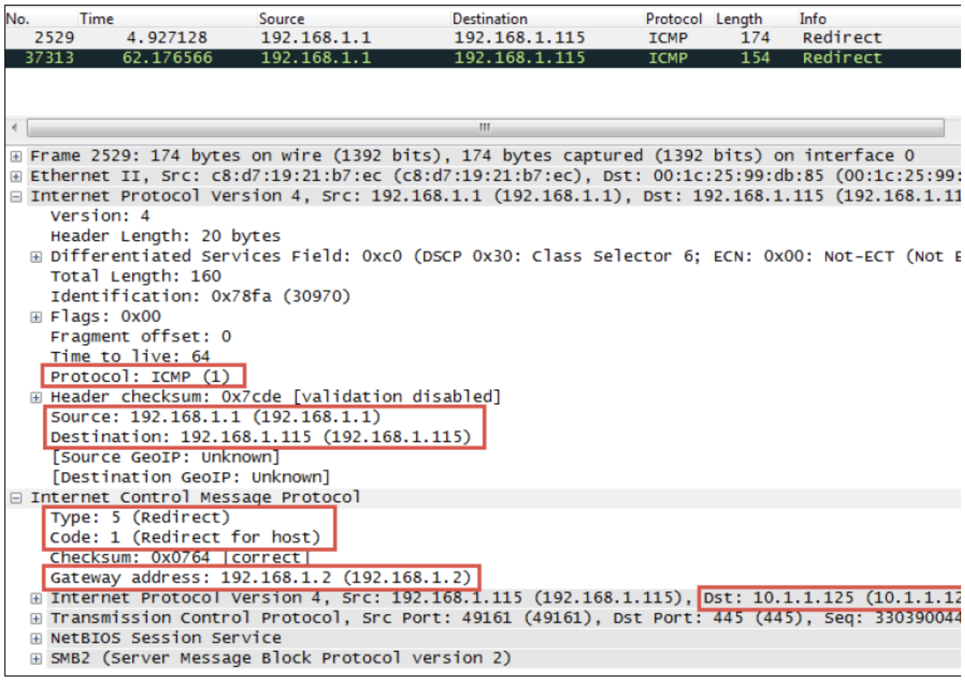


- The Wireshark packet details fields for the ICMP packet illustrated in the following screenshot depict a Time-to-live exceeded message as seen in a typical traceroute capture.
- The following points are significant to analyze this packet:
  - The source IP address seen in the IPv4 header summary is 10.192.128.1, which is the IP address of the router interface sending the ICMP message to the originator, 192.168.1.115
  - The ICMP packet is Type 11, Code 0 (TTL exceeded in transit)

…CONTINUE (ICMP ANALYSIS)

- The second set of IPv4 and ICMP headers that follow the first IPv4 and ICMP headers are copies of the original packet transmitted by the sender. This copy is returned to allow determination of the packet that caused the ICMP message. The significant points in the packet details of this ICMP message copy include:
  - The target destination IP address, where the echo request packet was intended to be sent (and would have been if the TTL value hadn't been altered) is 205.251.242.51.
  - The TTL value was 1 when this packet reached the 10.192.128.1 router interface. This packet cannot be forwarded, resulting in the TTL exceeded message being sent back to the sender.
  - The original ICMP packet was a Type 8, Code 0 echo request message.
  - The Header Data section of the ICMP packet for the echo requests and replies will include a 16-bit identifier and 16-bit sequence number, which are used to match echo replies to their requests.

ICMP REDIRECT



Another common use of ICMP is to redirect a client to use a different default gateway (router) to reach a host or network than the gateway it originally tried to use. In the **ICMP Redirect** packet depicted in the following screenshot, a number of packet fields should be noted:

- The source IP address of the ICMP redirect packet is 192.168.1.1, which was the client's default gateway; this is the router sending the redirect packet back to the client
- The ICMP **Type** is **5** (**Redirect**) and **Code** is **1** (**Redirect for host**)
- The gateway IP address that the router 192.168.1.1 is telling the client to use to reach the desired target host is 192.168.1.2
- The IP address of the target host was 10.1.1.125

WIRESHARK ICMP FILTERS

- Capture filters(s): icmp
- Display filter(s):icmp.type==5 && icmp.code==1 (host redirects)

Type	Code	Description
0	0	This indicates echo reply (ping)
3	0	This indicates destination network unreachable
3	1	This indicates destination host unreachable
3	4	This indicates fragmentation required and do not fragment bit set
3	6	This indicates destination network unknown
3	7	This indicates destination host unknown
5	0	This indicates redirect datagram for the network
5	1	This indicates redirect datagram for the host
8	0	This indicates echo request (ping)
11	0	This indicates TTL expired in transit (seen in traceroutes)

ICMP control message types

Analyzing udp traffic

- The UDP layer provides datagram based connectionless transport layer (layer 4) functionality in the InternetProtocolFamily.
- UDP is only a thin layer, and provides not much more than the described UDP port multiplexing. Just like IP, UDP doesn't provide any mechanism to detect PacketLoss, DuplicatePackets, and the like. There are a lot of protocols on top of UDP, including: BOOTP, DNS, NTP, SNMP, ...

Protocol dependencies

- IP: Typically, UDP uses IP as its underlying protocol. The assigned protocol number for UDP on IP is 17.

.. continue

- The UDP dissector is fully functional. There are two statistical menu items for UDP available: *Statistics/Endpoints* which contains a tab showing all UDP endpoints (combination of IP address and UDP port) and *Statistics/Conversations*, which contains a tab showing all UDP conversations (combination of two endpoints).

Display Filter

- A complete list of UDP display filter fields can be found in the display filter reference
- Show only the UDP based traffic: udp

Capture Filter

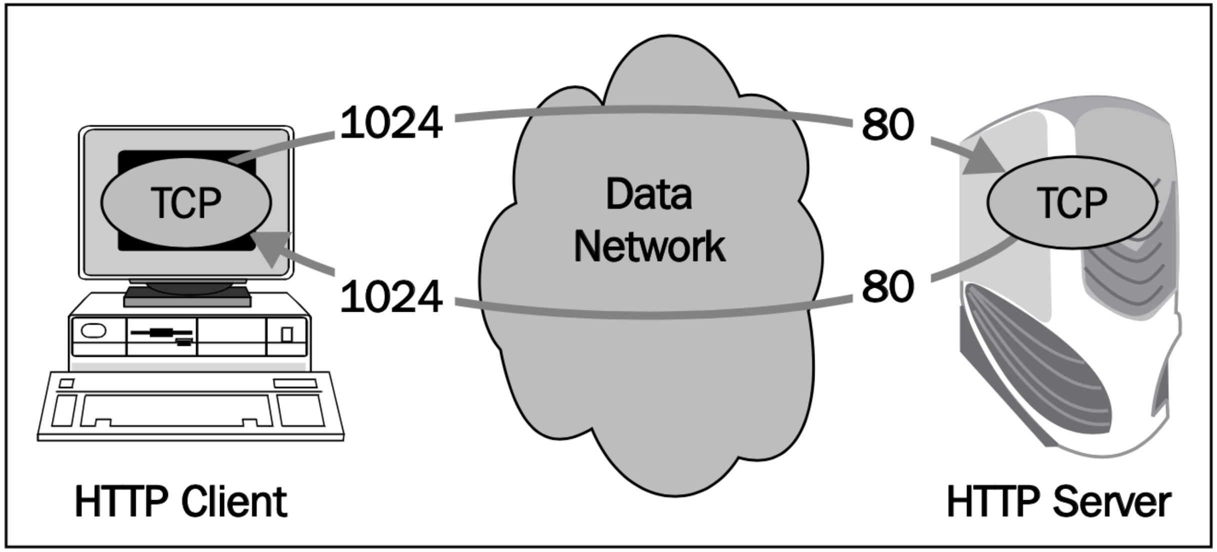
- Capture only the UDP based traffic: udp

Analyze TCP protocols

- The goal of **Transmission Control Protocol** (**TCP**) and **User Datagram Protocol** (**UDP**) is to pass information between end applications, for example, from a web client to a web server, mail client to a mail server, and so on. This is done by providing identification to end applications and forwarding packets between them. These identifications are called port numbers, and a port number with its IP address is called a socket. In the following diagram you can see what happens when you open a connection from your browser to a web server. The web server listens on port 80 and you will open a connection, for example, from port 1024.
- So, the server is listening to requests on port 80 and will send responses to you on port 1024.

TCP handshake

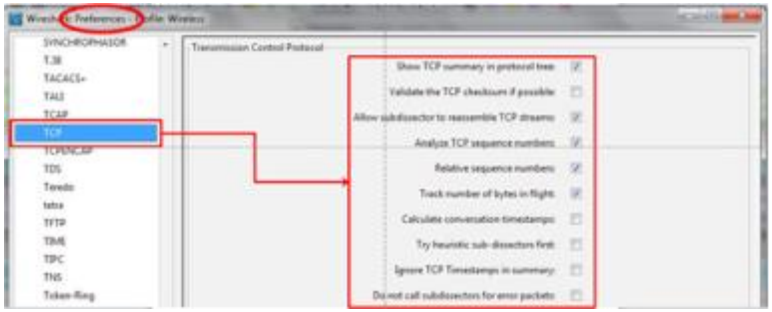
- So, the server is listening to requests on port 80 and will send responses to you on port 1024.



Configuring TCP and UDP preferences for troubleshooting

- In most cases you can use the default Wireshark parameters for TCP and UDP network analysis, but there are also some changes that can be configured. The changes will be configured in the **Preferences** window.
- Getting ready
- For TCP or UDP configuration:
- Start Wireshark, and from the **Edit** menu, choose **Statistics**.
- Under **Protocols**, choose **TCP** or **UDP**.

- By default only the first parameter is set. In most cases it is enough.

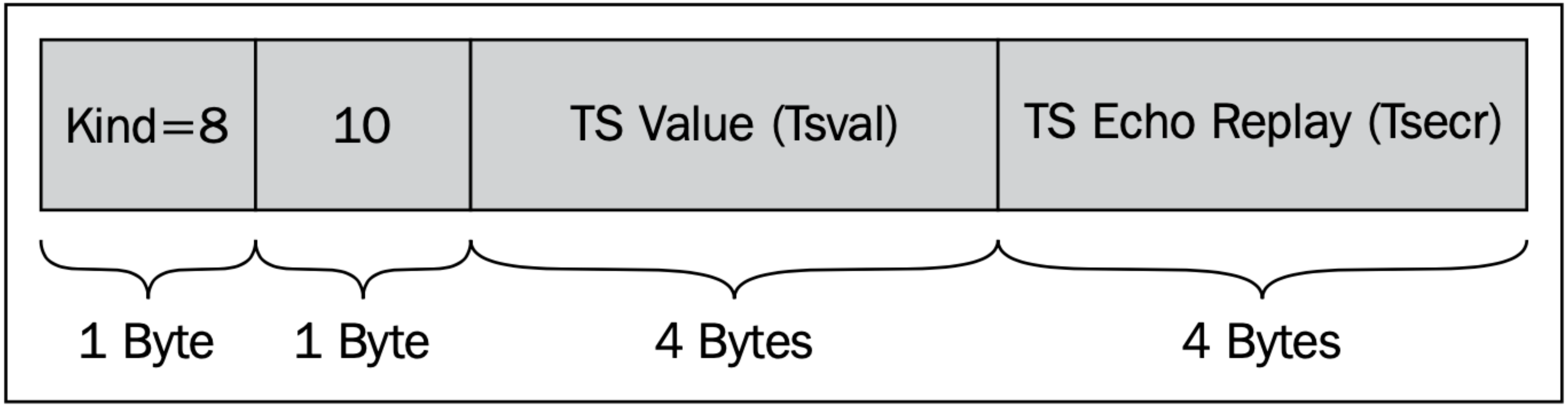


You can configure the following parameters in TCP:

- **Show TCP summary in protocol tree**: Mark this button if you want the TCP summary line to be shown in the protocol tree (set by default).
- **Validate the TCP checksum if possible**: This feature can slow down performance. In most cases it is not required.
- **Allow subdissector to reassemble TCP streams**: This option is for stream analysis (set by default).
- **Analyze TCP sequence numbers**: When this is set, Wireshark analyzes sequence numbers and track phenomena such as retransmission, duplicate ACKs, and so on, which is one of the important features of Wireshark.
- **Relative sequence numbers**: When this is set, Wireshark will show you every TCP connection that starts from Seq=0.
- **Track number of bytes in flight**: This setting enables Wireshark to track the number of unacknowledged bytes flowing on the network (set by default).
- **Calculate conversation timestamps**: This feature enables the calculations of TCP timestamps option.
- **Try heuristic sub-dissectors first**: Try to decode a packet using heuristic method before using a sub-dissector registered to the specific port.
- **Ignore TCP Timestamps in summary**: Ignore the timestamp option in the TCP header. f **Do not call subdissector for error packets**: This option does not analyze erroneous TCP packets.

TCP preferences

- Referring to **relative sequence numbers**, when you look at a TCP connection you see that it always starts with sequence numbers equal to zero. These are the relative numbers that are normalized to zero by Wireshark. The real numbers are numbers between 0 and 232, picked by the TCP process, which are difficult to follow. The TCP standard does not set any rule for picking this number.
- The **calculating conversations timestamps** refers to the timestamp option of the TCP packet. The TCP timestamps option carries two 4-byte timestamp fields, as seen in the diagram:



Finding the root cause

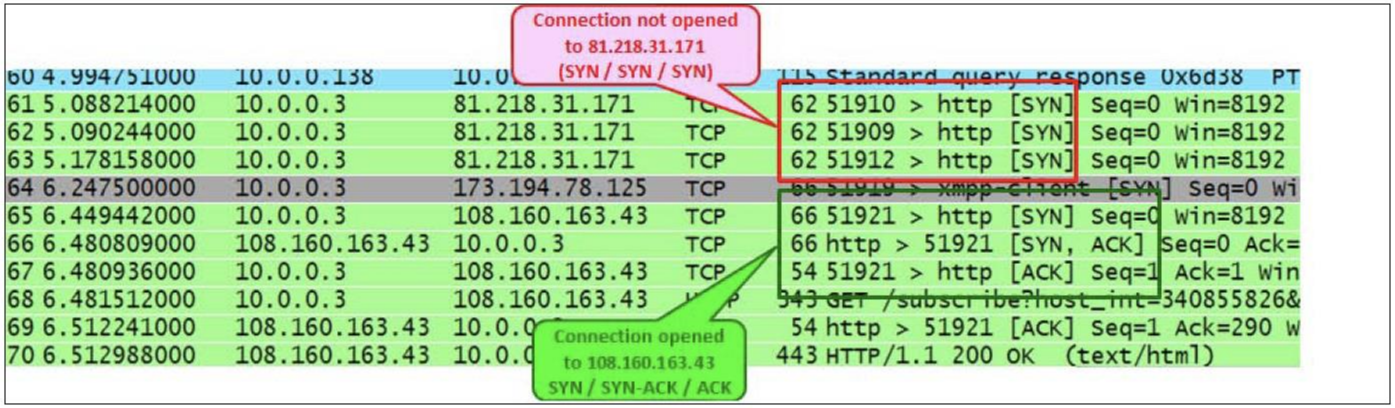
- If you experience one of the following problems, use Wireshark in order to find out what is the reason for it.
- These problems can be of many types. Of these:
  - You try to run an application and it does not work. You try to browse the Internet and you don't get any response.
  - You try to use your mail but you don't have a connection to the mail server.
  - Problems can be due to simple reasons, such as the server being down, the application is not running on the server, or the network is down somewhere on the way to the server.
  - Problems can be also due to more complicated reasons, such as DNS problems, insufficient memory on the server that does not enable you to connect (due to high memory consumption by an application, for example), duplicate IPs, and many others.
- In this recipe we focus on these GO/NO-GO problems that are usually quite easy to solve.

Getting ready

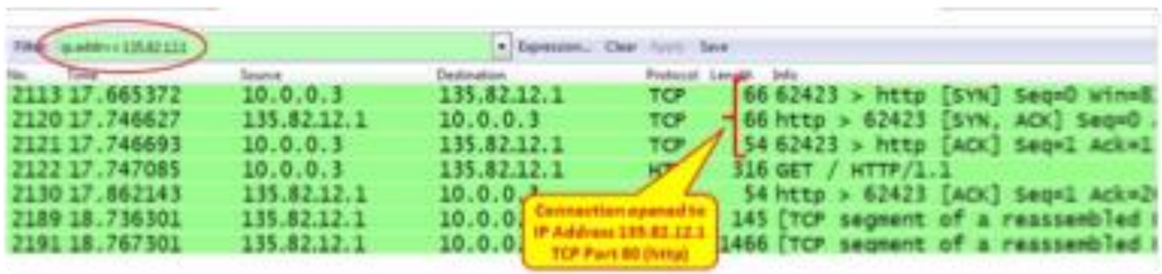
- Here you will see some indicators and what you can see when you use Wireshark for debugging TCP connectivity problems. Usually these problems result in trying to run an application and getting no results.
- When you try to run an application, for example, a database client, a mail client, watching cameras servers, and so on, and you don't get any output, follow these steps: Verify that the server and applications are running.
- Verify if your client is running, you have an IP address configured (manually or by DHCP), and you are connected to the network.
- Ping the server and verify you have connectivity to it.

How to do it..

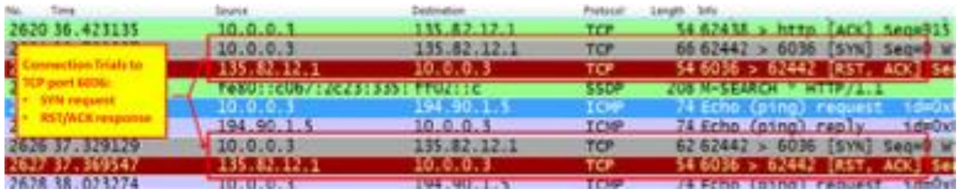
- In some cases, you will not have Ping to the server, but still have connectivity to the application. This can happen because a firewall is blocking the ICMP messages, so if you don't have Ping to a destination it doesn't necessarily mean that something is wrong. The firewall can be a dedicated device in the network or a windows (or Linux/UNIX) firewall installed on the end device.



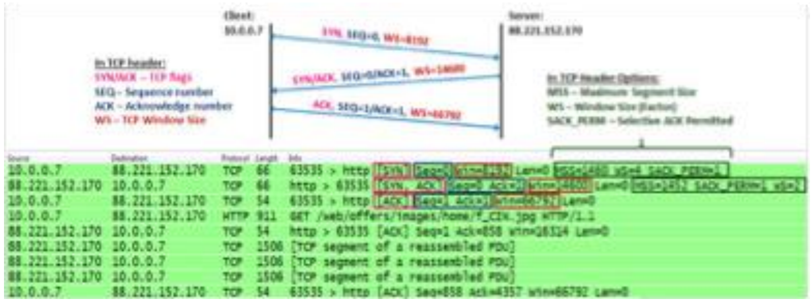
- In the capture file, look for one of the following patterns:
  - Triple SYN messages with no response (in the following screenshot)
  - SYN messages with a reset (RST) response
- In both cases it can be that a firewall is blocking the specific application or the application is not running.
- In the following screenshot, we see a simple case in which we simply don't get access to web server 81.218.31.171 (packets 61, 62, and 63). It can be because it is not permitted by a firewall, or simply because there is a problem with the server. We can also see that we have a connection to another website (108.160. 163.43, packets 65, 66, and 67), so the connection problem is only to 81.218.31.171.



- In the next screenshot we see a slightly more complex case of the same situation. In this case, we've had a cameras server that the customer wanted to log in to and watch the cameras on a remote site. The camera's server had the IP address 135.82.12.1 and the problem was that the customer was able to get the main web page of the server with the login window, but couldn't log into the system. In the following screenshot, we can see that we open a connection to the IP address 135.82.12.1. We see that a TCP connection is opened to the HTTP server, and at first it looks like there are no connectivity problems:



- The problems arise when we filter all traffic to the IP address 135.82.12.1, that is, the cameras server.
- Here we see that when we try to connect to TCP port 6036, we get an RST/ACK response, which can be:
  - A firewall that blocks port 6036 (that was the case here)
  - When port address translation (PAT) is configured, and we translate only port 80 and not 6036
  - The authentication of the username and password were done on TCP port 6036, the firewall allowed only port 80, the authentication was blocked, and the application didn't work
- To summarize, when you don't have connectivity to a server, check the server and the client if all TCP/UDP ports are forwarded throughout the network, and if you have any ports that you don't know about.



- Starting a TCP connection, as seen in the next screenshot, happens in three steps:

1. The TCP process on the client side sends an SYN packet. This is a packet with the SYN flag set to 1. In this packet the client:

  – Specifies its initial sequence number. This is the number of the first byte that the client sends to the server.
  – Specifies its window size. This is the buffer the clients allocate to the process (the place in the client's RAM).
  – Sets the options that will be used by it: MSS, Selective ACK, and so on.

2. When the server receives the request to establish a connection, the server:

  – Sends an SYN/ACK packet to the client, confirming the acceptance of the SYN request.
  – Specifies the server's initial sequence number. This is the number of the first byte that the server sends to the client.
  – Specifies the server's window size. This is the buffer size that the server allocates to the process (the place in the server's RAM).
  – Responds to the options requested and sets the options on the server side.

3. When receiving the server's SYN/ACK, the client:

- Sends an ACK packet to the server, confirming the acceptance of the SYN/ ACK packet from the server.
- Specifies the client's window size. This is the buffer size that the client allocates to the process. Although this parameter was defined in the first packet (the SYN packet), the server will refer to this one since it is the latest window size received by the server.

summary

- In the options field of the TCP header, we have the following main options:
  - **Maximum Segment Size (MSS)**: This is the maximum size of the TCP datagram, that is, the number of bytes from the beginning of the TCP header to the end of the entire packet.
  - **Windows Size (WSopt)**: This factor is multiplied with the Window Size field in the TCP header to notify the receiver on a larger size buffer. Since the maximum window size in the header is 64 KB, a factor of 4 gives us 64 KB multiplied by 4, that is, a 256 KB window size.
  - **SACK**: Selective ACK is an option that enables the two parties of a connection to acknowledge specific packets, so when a single packet is lost, only this packet will be sent again. Both parties of the connection have to agree on SACK in the connection establishment.
  - **Timestamps options (TSopt)**: This parameter was explained earlier in this chapter, and refers to measurement of the delay between client and the server.
- By this stage, both sides:
  - Agree to establish a connection
  - Know the other side's initial sequence number
  - Know the other side's window size

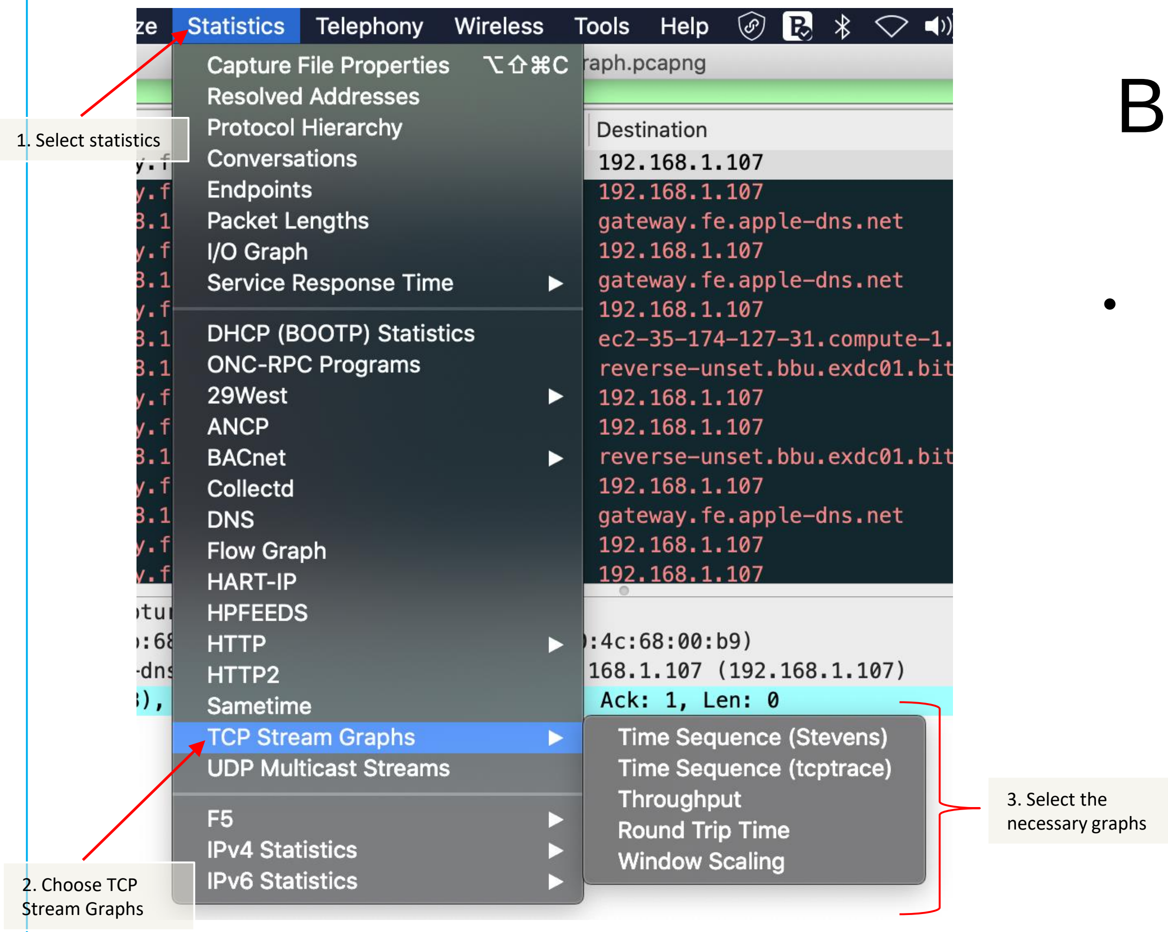
GRAPH TRAFFIC CHARACTERISTIC

- Build advanced IO graph
- Graph round trip times
- Graph TCP throughput
- Find problems using TCP Time-sequence graph

Tcp stream graph

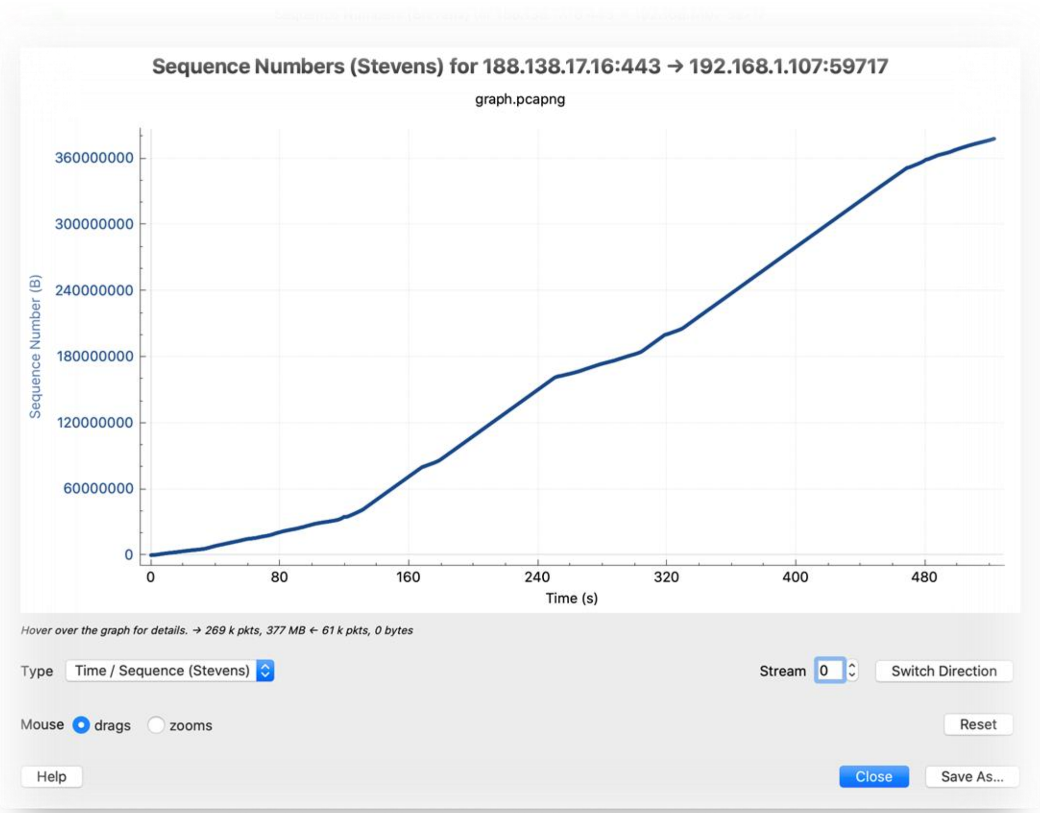
- Wireshark provides TCP StreamGraphs to analyze several key data transport metrics, including:
  - **Round-trip time**: This graphs the RTT from a data packet to thecorresponding ACK packet.
  - **Throughput**: These are plots throughput in bytes per second.
  - **Time/sequence (Stephen's-style)**: This visualizes the TCP-based packet sequence numbers (and the number of bytes transferred) over time. An ideal graph flows from bottom-left to upper-right in a smooth fashion.
  - **Time/sequence (tcptrace):** This is similar to the Stephen's graph, but provides more information. The data packets are represented with an I-bar display, where the taller the I-bar, the more data is being sent. A gray bar is also displayed that represents the receive window size. When the gray bar moves closer to the I-bars, the receive window size decreases.
  - **Window Scaling**: This plots the receive window size.

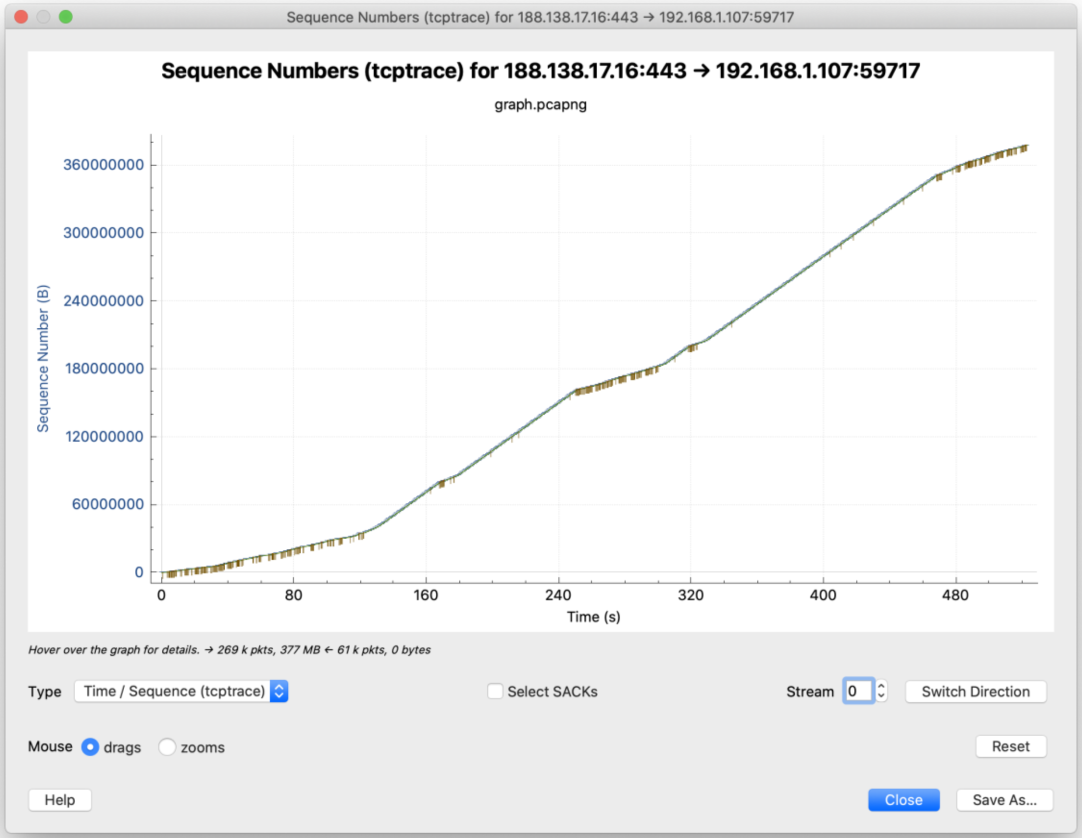
Building tcp stream graphs



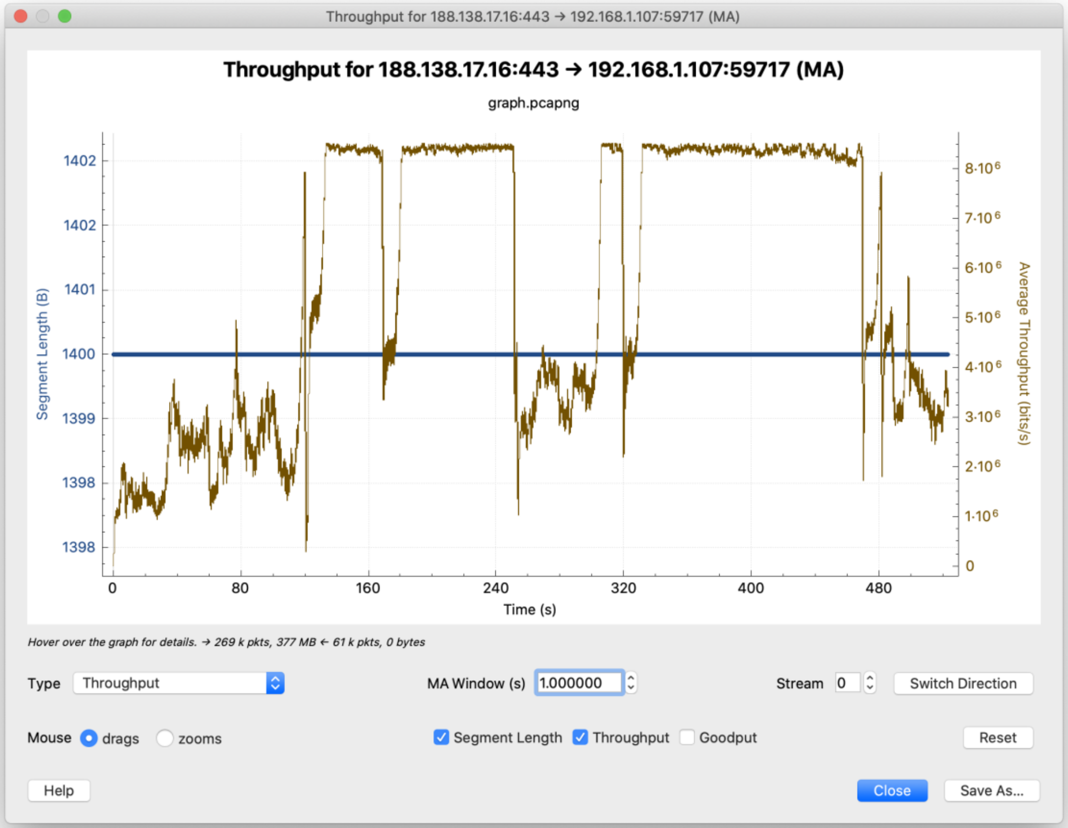
- These analysis graphs can be utilized by selecting one of the packets in a TCP stream in the **Packet List** pane and selecting **TCP StreamGraph** from the **Statistics** menu and then one of the options such as the **Time-Sequence Graph (tcptrace)**.

Sequence numbers (stevens)

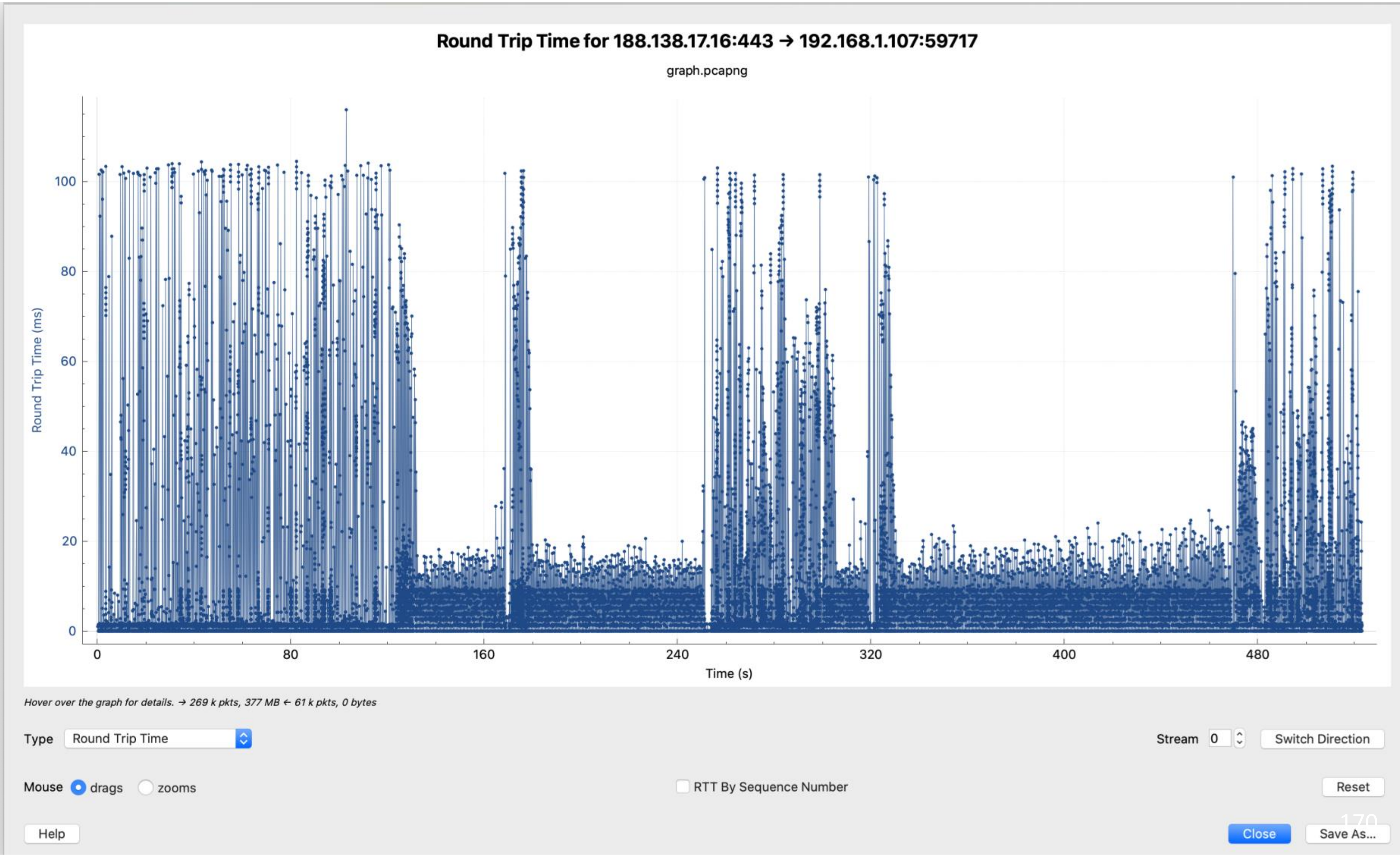


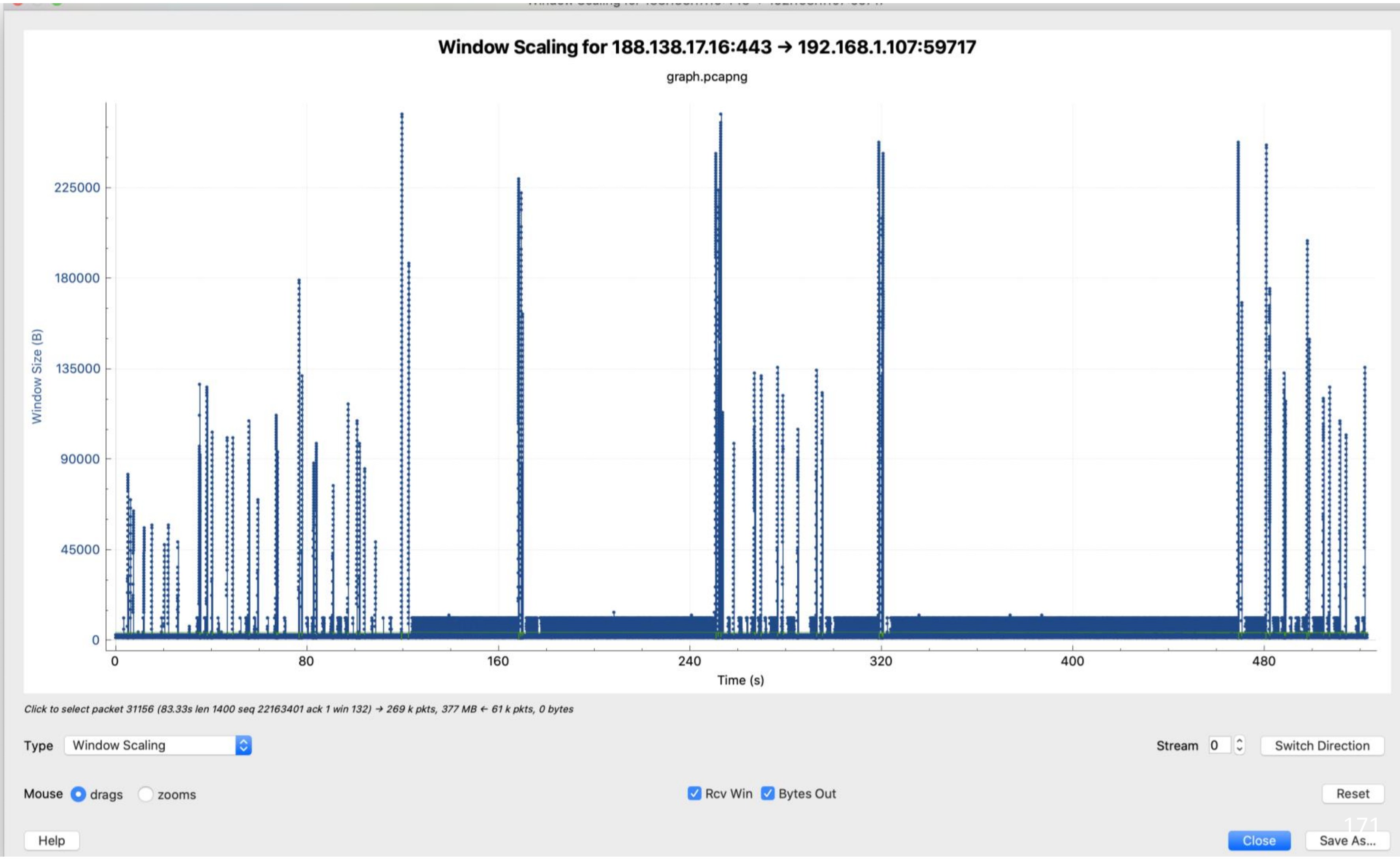


Sequence numbers (tcptrace)



Througput graph

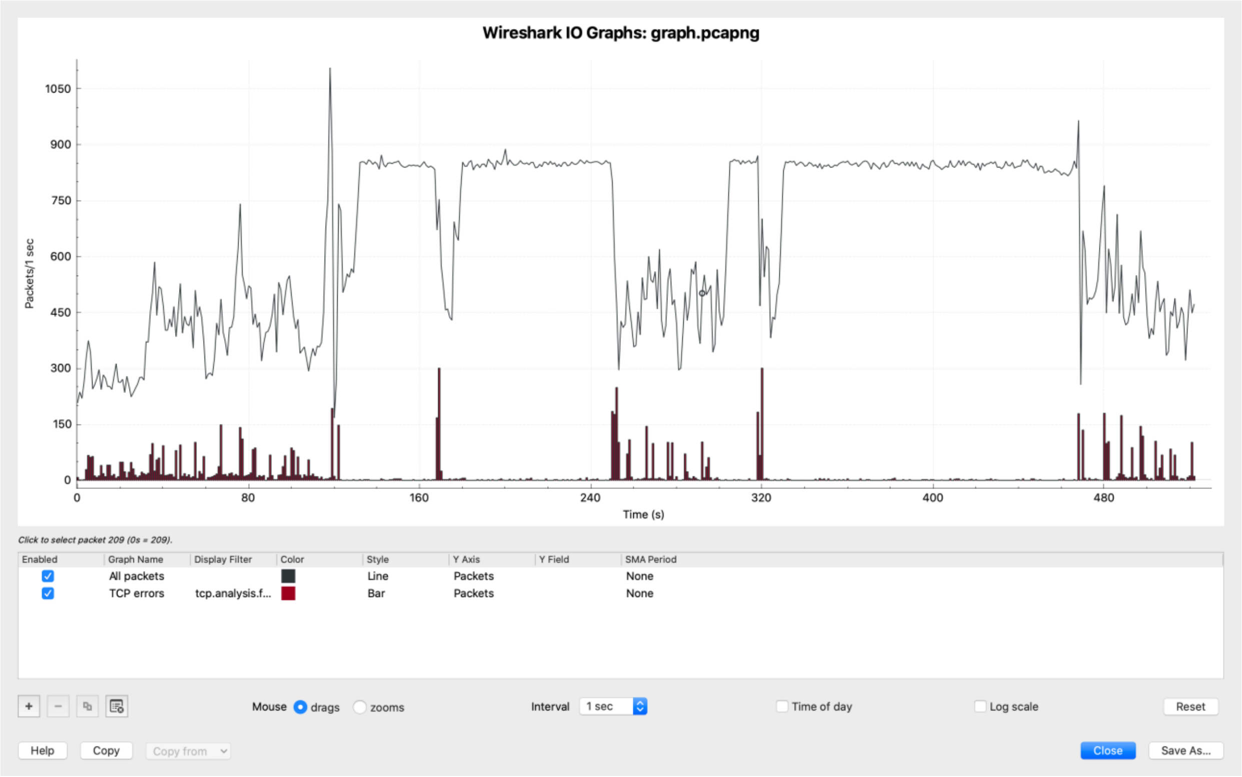

Round Trip Time for 188.138.17.16:443 → 192.168.1.107:59717
graph.pcapng
Round Trip Time (ms)
100
80
60
40
20
0
0
80
160
240
320
400
480
Time (s)
Hover over the graph for details. → 269 k pkts, 377 MB ← 61 k pkts, 0 bytes
Type
Round Trip Time
Stream
0
Switch Direction
Mouse
drags
zooms
RTT By Sequence Number
Reset
Help
Close
Save As...


Window Scaling for 188.138.17.16:443 → 192.168.1.107:59717
graph.pcapng
Window Size (B)
225000
180000
135000
90000
45000
0
0
80
160
240
320
400
480
Time (s)
Click to select packet 31156 (83.33s len 1400 seq 22163401 ack 1 win 132) → 269 k pkts, 377 MB ← 61 k pkts, 0 bytes
Type
Window Scaling
Stream
0
Switch Direction
Mouse
drags
zooms
Rcv Win
Bytes Out
Reset
Help
Close
Save As...

IO graph

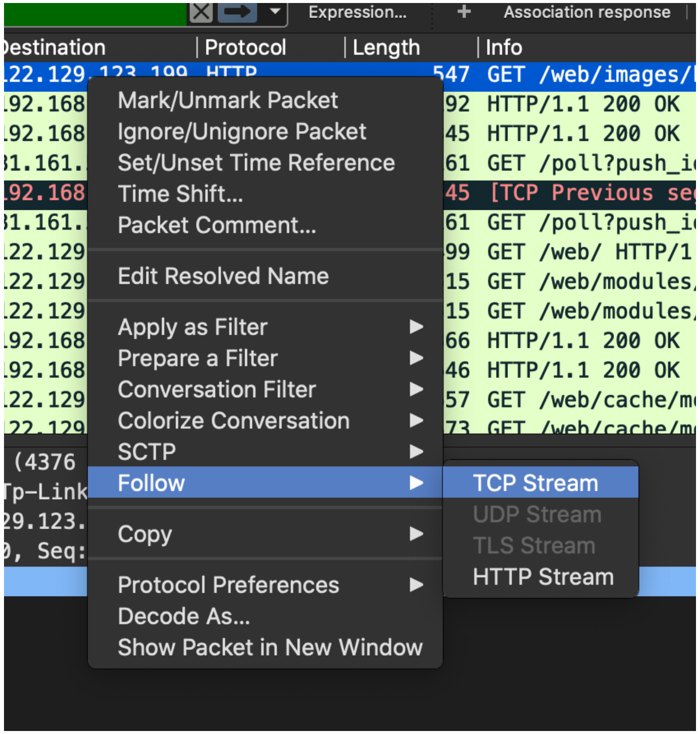
You can also analyze a the effects of TCP issues on network throughput by applying TCP analysis display filter strings to Wireshark's IO Graph, such as:

tcp.analysis.flags && !tcp.analysis.window_update

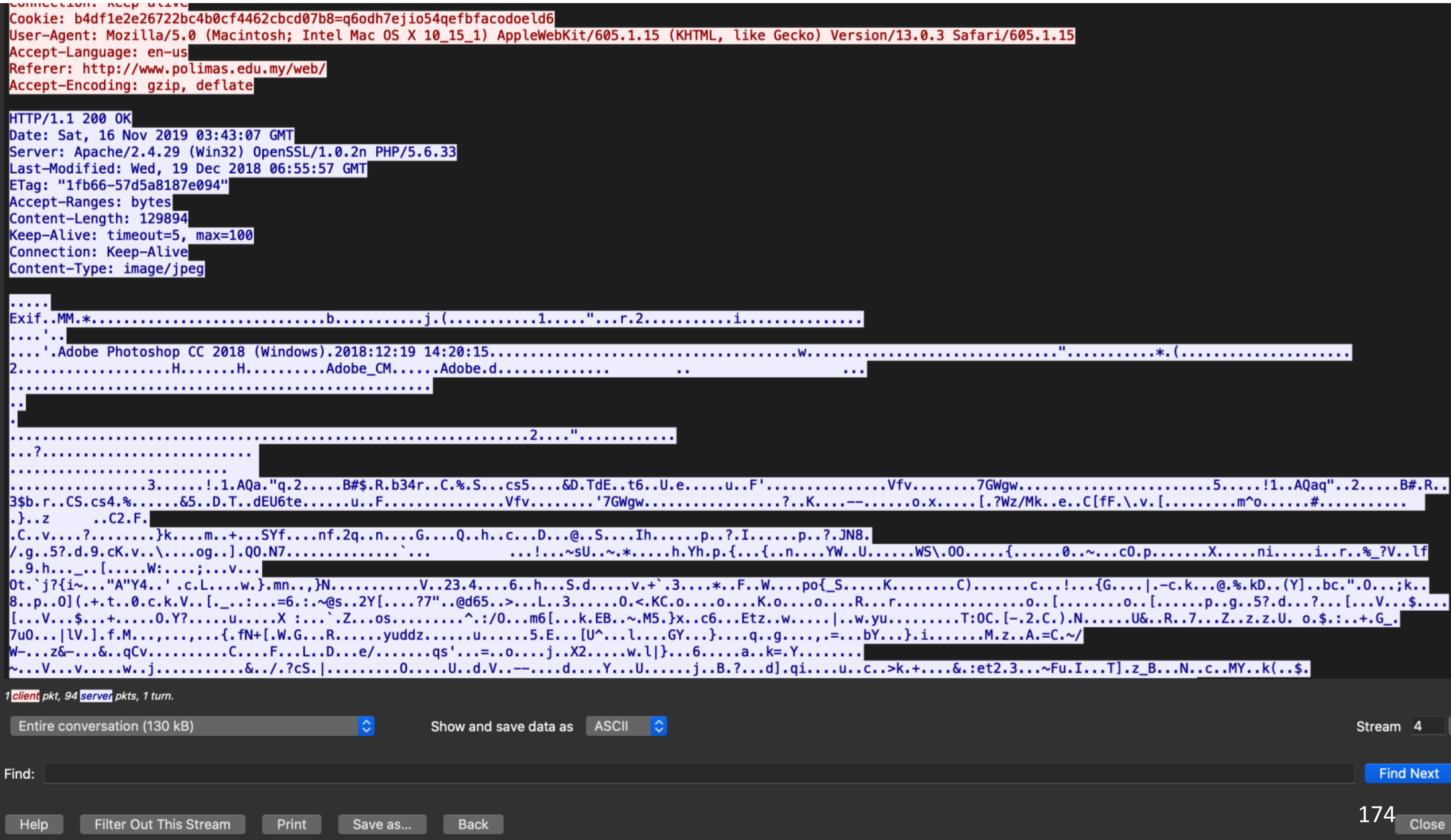


Anylyzing http traffic

- Next, right click on the first pcap and choose ‘Follow’ and ‘TCP Stream’
- Analyze and discuss the result with your instructor
- (see the result on the next page)



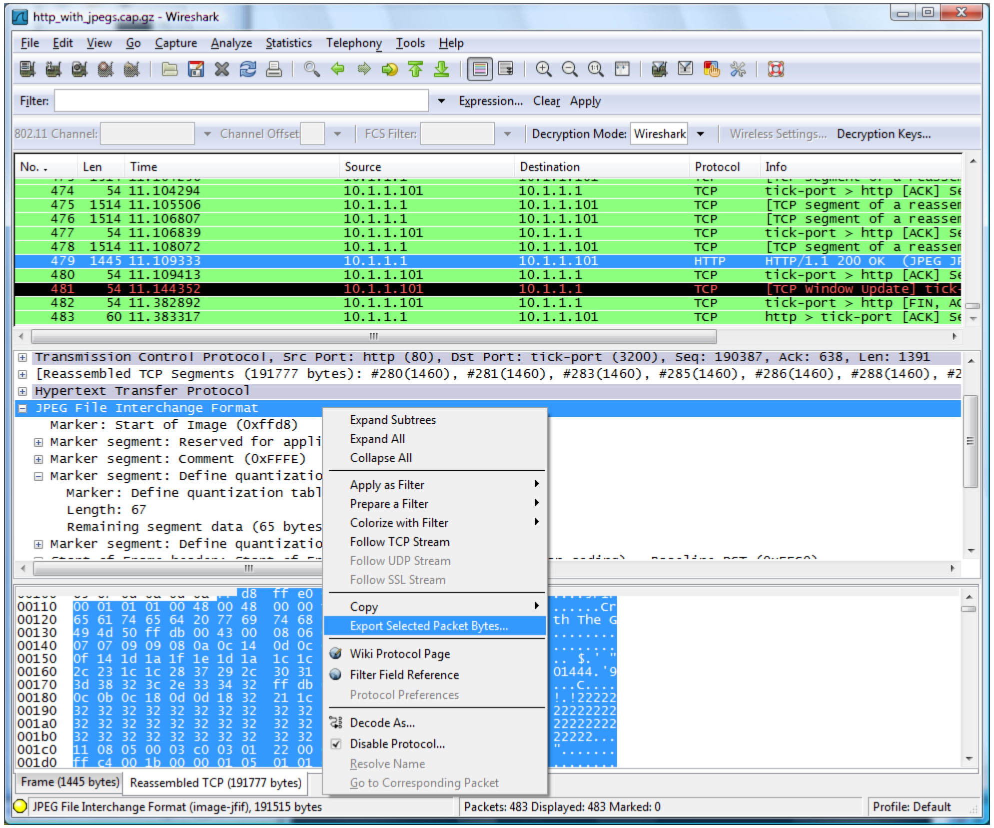
TOGETHER WE TRANSFORM



Reassembled http packets

This is an example of how to reassemble a HTTP stream and to extract and save to a file a JPEG image from inside a HTTP PDU.

1. First download the example capture SampleCaptures/http_with_jpegs.cap.gz from the SampleCaptures page.
2. Then enable all three preferences above.
3. Then select packet #479 (ctrl+G and input 479) and click on the JPEG protocol to select it.
4. Then just right click on the JPG protocol and select "Export Selected Bytes" and save it to a file. If everything worked, you will now have a nice JPEG of the Dolphin Show at SeaWorld in SurfersParadise to view for your enjoyment.



SSL/TLS Version History

1994
SSLv2
Netscape Navigator

1995
SSLv3
resolved serious v2 issues

1999
TLSv1.0
RFC 2246

2006
TLSv1.1
RFC 4346

2008
TLSv1.2
RFC 5246

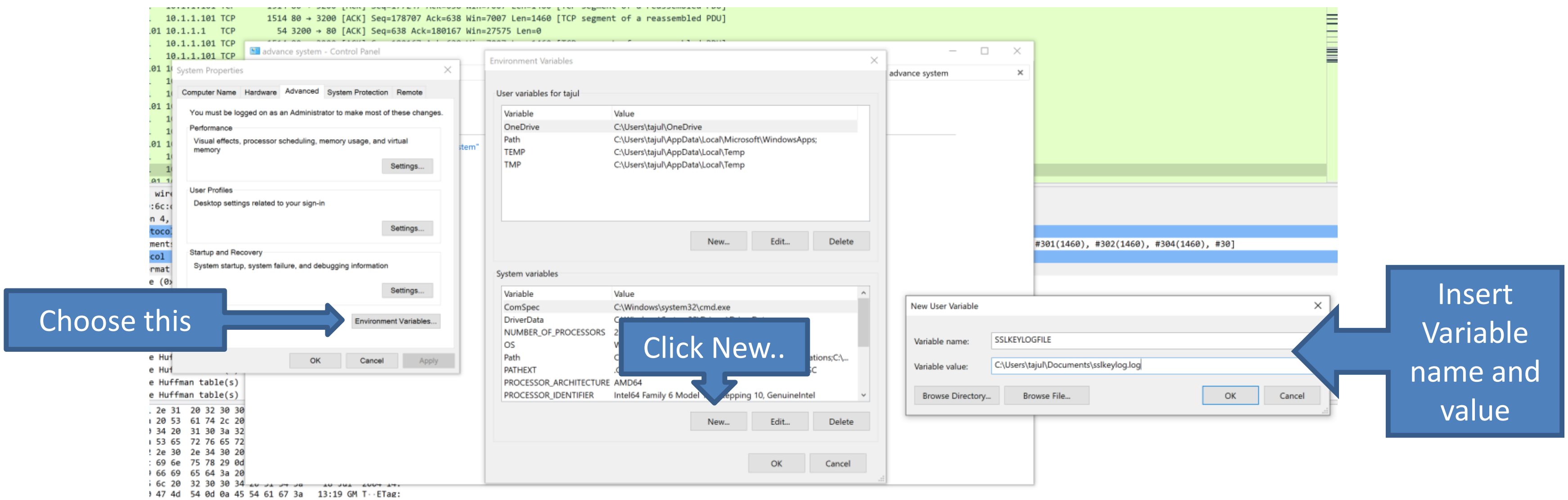
2013
Microsoft/Apple
Enable support for
TLSv1.2

2018
TLSv1.3
Prop. Standard

Courtesy of Kary Rogers, Sharkfest'19 US

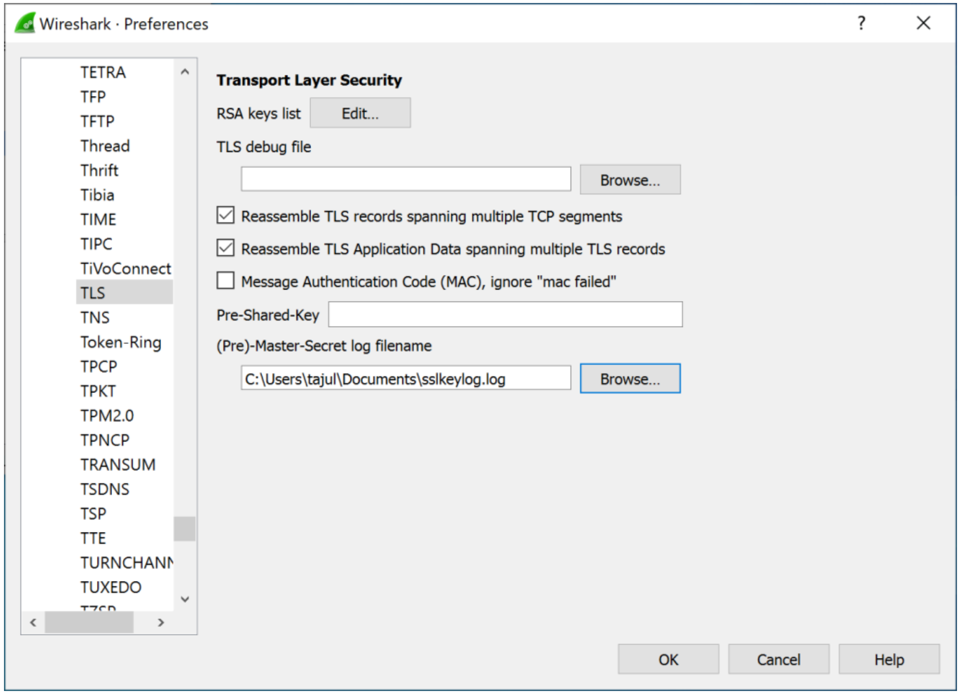
Analyze the TLS-Encrypted Traffic (HTTPS)

- Firefox and Chrome both support logging the symmetric session key used to encrypt TLS traffic to a file.
- For this example we will create a SSLKEYLOG file that later will be used by Wireshark to decrypt the encrypted traffic.

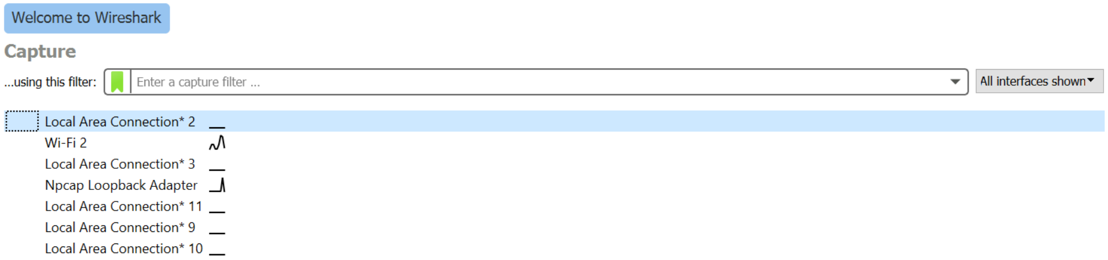


Performing traffic decryption

1. If you want to decrypt TLS traffic, you first need to capture it. For this reason, it's important to have Wireshark up and running before beginning your Web browsing session.
2. Before we start the capture, we should prepare it for decrypting TLS traffic. To do this, click on Edit → Preferences. Select Protocols in the left-hand pane and scroll down to TLS. At this point, you should see something similar to the screen on the right
3. At the bottom of this screen, there is a field for (Pre)-Master-Secret log filename. As shown above, you need to set this value to the same location as the SSLKEYLOGFILE for your browser. When done, click OK.

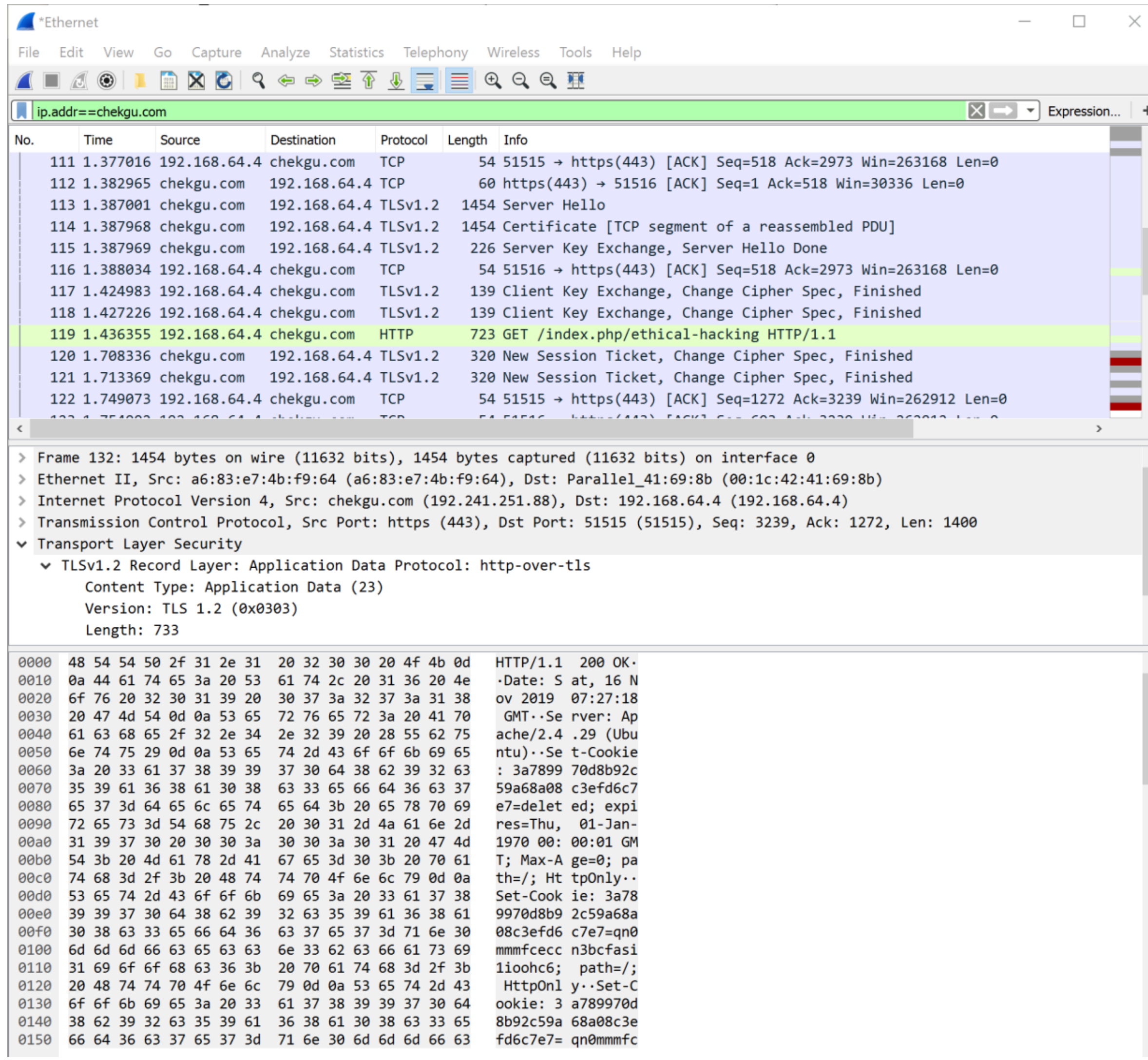


- Now on the main screen of Wireshark, it will show a list of possible adapters to capture from.
- Clicking on an adapter will start capturing traffic on it.
- At this point, you're ready to create some TLS-encrypted traffic. Go to Chrome or Firefox and browse to a site that uses HTTPS (we used Facebook for this example). Once it's loaded, return to Wireshark and stop the capture (red square).
- Looking through the capture, you'll probably see a lot of traffic.



Decrypted tls traffic

▪This is what it looks like when you switch to the “Decrypted SSL Data” tab. Note that we can now see the request information in plain-text! Success!



Magic Byte

- Magic bytes are the bytes in the header of a file which identify the file type.
- There are many common magic bytes you should become familiar with:
  - Windows Executable = \x4D \x5A –“MZ”
  - *nix Executable Format = \x7F \x45 \x4C \x46 –“.ELF”
  - PDF = \x25 \x50 \x44 \x46 –“%PDF”
  - Flash = \x43 \x57 \x53 –“CWS”
  - ZIP = \x50 \x5B \x03 \x04 –“PK..”
  - JPEG (Raw) = \xFF\xD8 \xFF\xDB–“ÿØÿÛ”
  - Many, many more…

Good luck!

THE END