Monitoring by Wireshark

Basic tutorial

Ing. Pedro Escudero

Telf: 0994667184
Mail: pedro.escudero@unach.edu.ec
Web: https://www.researchgate.net/profile/Pedro-Escudero-3/research

Abril – 2025

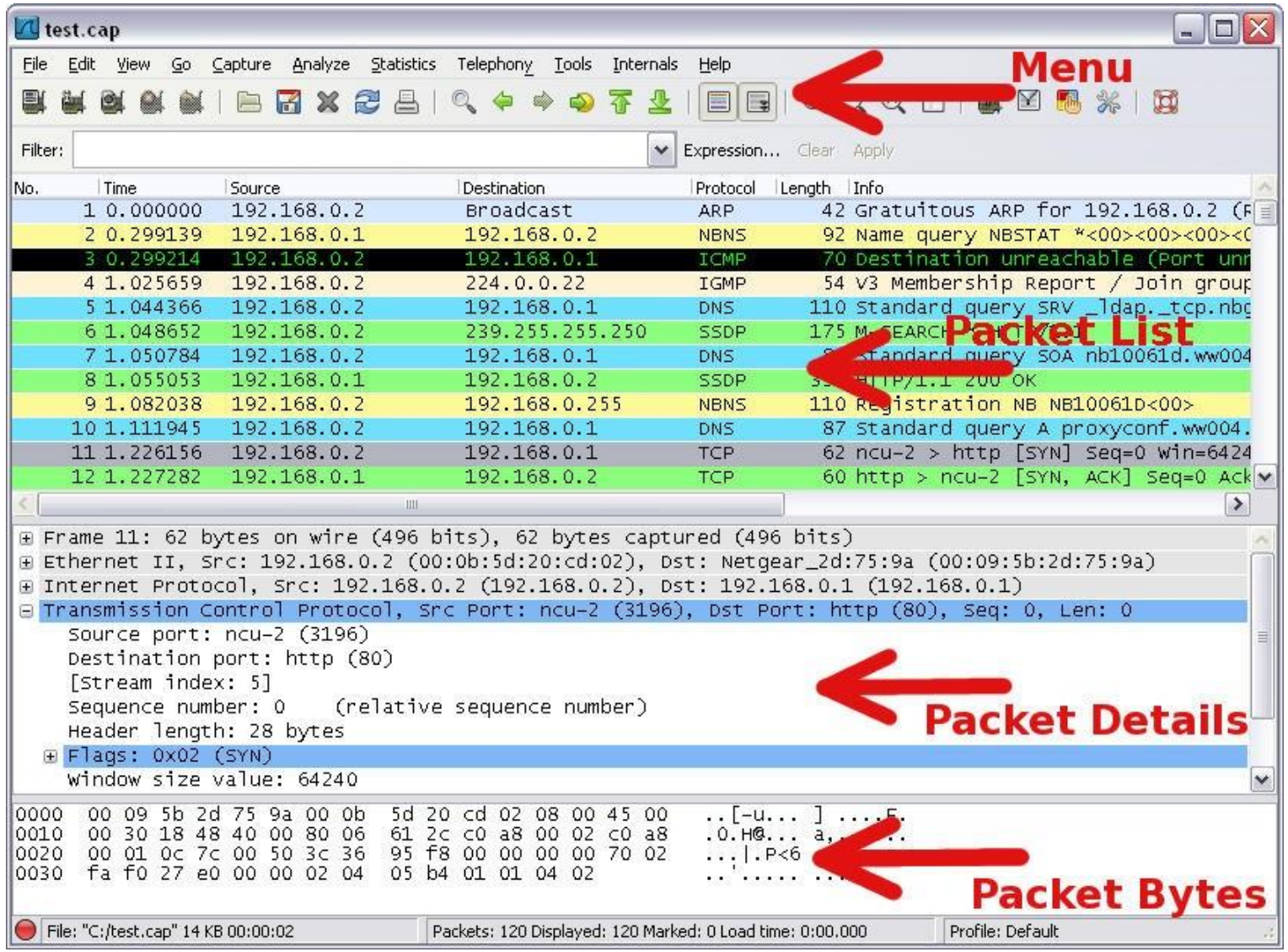
Contents

Introduction

- Network Traffic Trace
  - A recording of the network packets both received by and transmitted from a network interface.
- What is a pcap file?
  - pcap = Packet Capture
  - File format originally designed for tcpdump/libpcap.
  - Most widely used packet capture format.

- What is Wireshark?
  - A graphical network packet analyser.
  - Found at http://www.wireshark.org
  - The complete manual is located here.
- What some are it's uses?
  - Troubleshoot network problems.
  - Learn network protocol internals.
  - Debug protocol/program implementation.
  - Examine network-related security issues.

- **Basic UI**


test.cap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Menu
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.0.2 Broadcast ARP 42 Gratuitous ARP for 192.168.0.2
2 0.299139 192.168.0.1 192.168.0.2 NBNS 92 Name query NBSTAT
3 0.299214 192.168.0.2 192.168.0.1 ICMP 70 Destination unreachable (Port unr
4 1.025659 192.168.0.2 224.0.0.22 IGMP 54 V3 Membership Report / Join group
5 1.044366 192.168.0.2 192.168.0.1 DNS 110 Standard query SRV _ldap._tcp.nbg
6 1.048652 192.168.0.2 239.255.255.250 SSDP 175
Packet List
7 1.050784 192.168.0.2 192.168.0.1 DNS
8 1.055053 192.168.0.1 192.168.0.2 SSDP
9 1.082038 192.168.0.2 192.168.0.255 NBNS 110 Registration NB NB10061D<00>
10 1.111945 192.168.0.2 192.168.0.1 DNS 87 Standard query A proxyconf.ww004.
11 1.226156 192.168.0.2 192.168.0.1 TCP 62 ncu-2 > http [SYN] Seq=0 Win=6424
12 1.227282 192.168.0.1 192.168.0.2 TCP 60 http > ncu-2 [SYN, ACK] Seq=0 Ack
Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: ncu-2 (3196), Dst Port: http (80), Seq: 0, Len: 0
Source port: ncu-2 (3196)
Destination port: http (80)
[Stream index: 5]
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x02 (SYN)
Window size value: 64240
Packet Details
0000 00 09 5b 2d 75 9a 00 0b 5d 20 cd 02 08 00 45 00
0010 00 30 18 48 40 00 80 06 61 2c c0 a8 00 02 c0 a8
0020 00 01 0c 7c 00 50 3c 36 95 f8 00 00 00 00 70 02
0030 fa f0 27 e0 00 00 02 04 05 b4 01 01 04 02
Packet Bytes
File: "C:/test.cap" 14 KB 00:00:02
Packets: 120 Displayed: 120 Marked: 0 Load time: 0:00.000
Profile: Default

- **Basic UI**

.File -> Open

- Opens a packet capture file.

.View -> Time Display Format

- Change the format of the packet timestamps in the packet list pane.
- Switch between absolute and relative timestamps.
- Change level of precision.

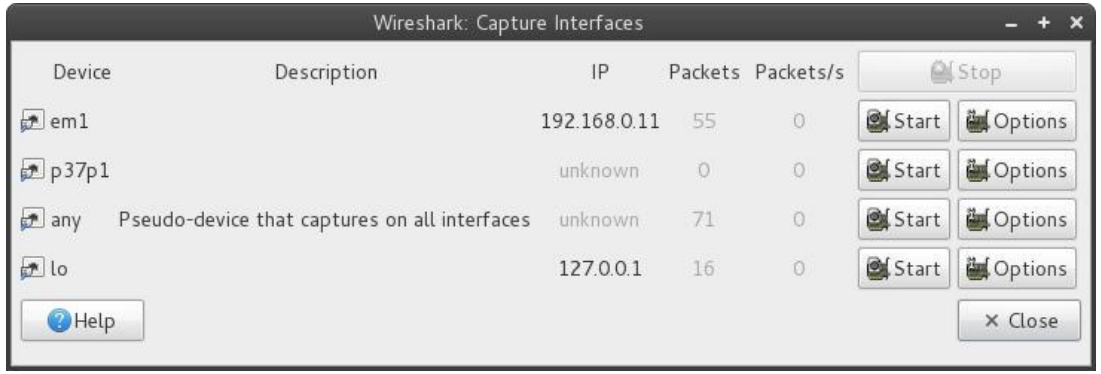
.View -> Name Resolution

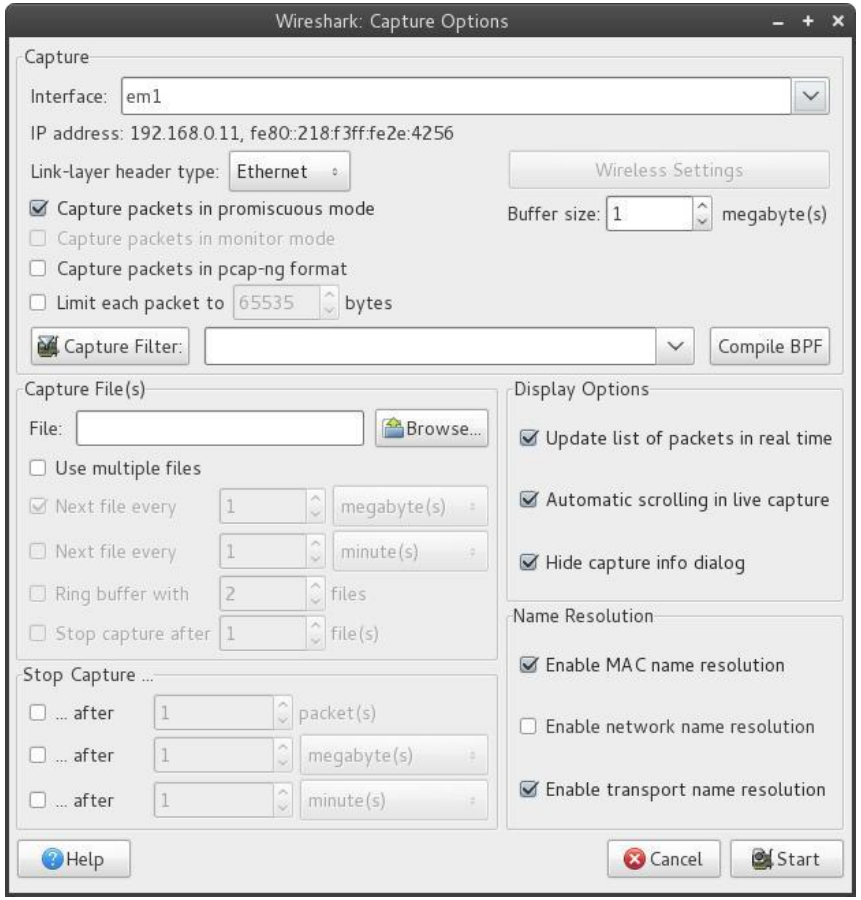
- Allow wireshark to resolve names from addresses at different protocol layers.

• Basic UI

.Capture -> Interfaces

- Available network interfaces for capture.
- Total packets per interface.
- Packet rate per interface.





.Capture -> Options

- Set various capture parameters.

.Promiscous mode

- On – record all packets reaching the interface.
- Off – record only those packets directed to the host.

• Basic UI

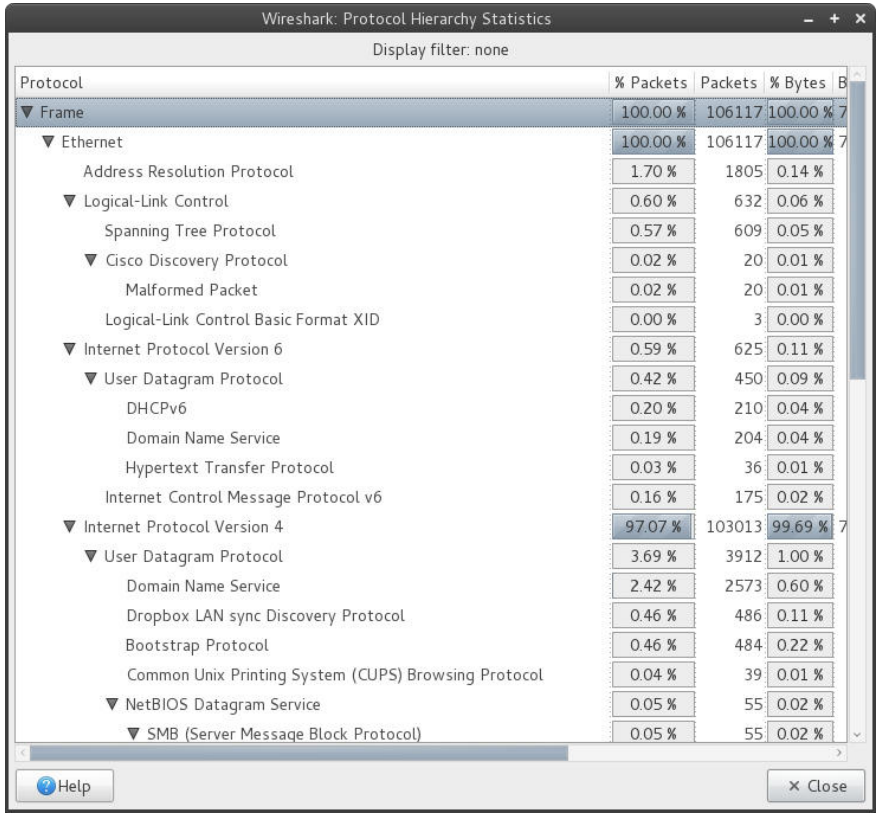
. Analyze -> Follow TCP Stream

- Applies a filter to follow a single tcp conversation within the trace.
- Displays the reassembiled data section of each packet in the conversation.
- Useful for debugging or analyzing any TCP based application layer protocol.
  - HTTP, FTP, SSH, LDAP, SMTP, etc.

• Basic UI

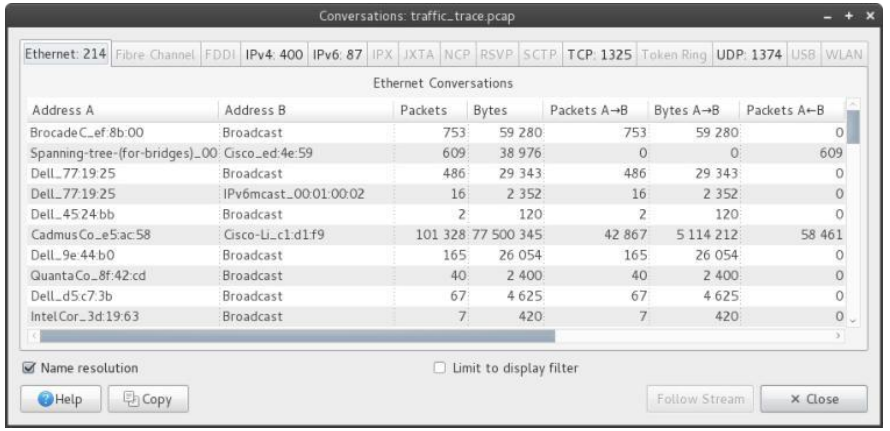
.Statistics -> Protocol Hierarchy

- Presents descriptive statistics per protocol.
- Useful for determining the types, amounts, and relative proportions of protocols within a trace.



.Statistics -> Conversations

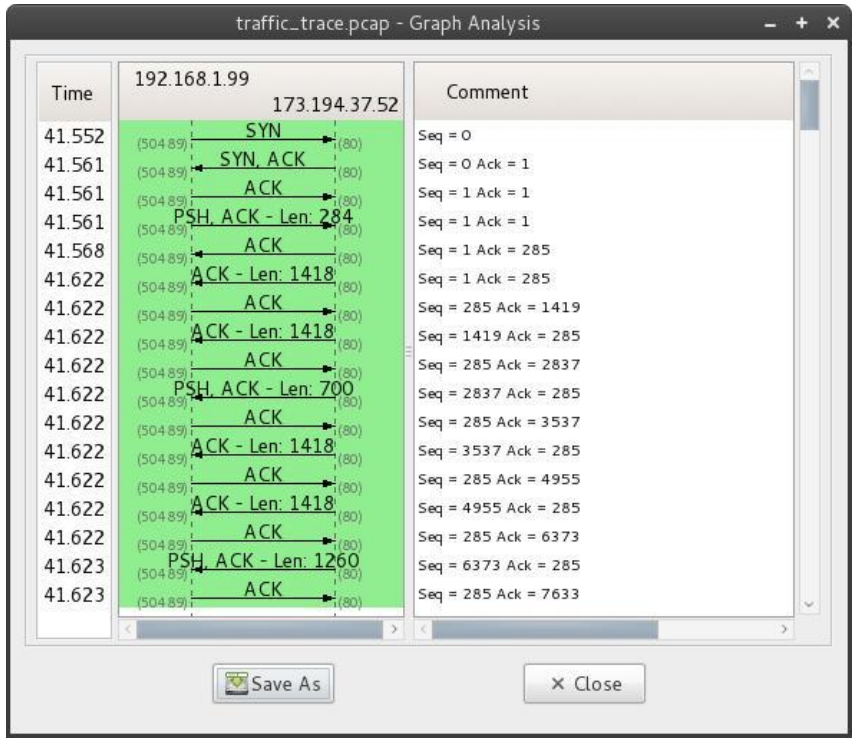
- Generates descriptive statistics about each conversation for each protocol in the trace.



• Basic UI

.Statistics -> Flow Graph

- Generates a sequence graph for the selected traffic.
- Useful for understanding seq. and ack. calculations.



- **Packet Capture**

.Interface selection

- Capture -> Interfaces
  - Select the interface from which to capture packets.
    - any – captures from all interfaces
    - lo – captures from the loopback interface (i.e. from localhost)
  - Set the desired capture parameters under the options menu.

.Start Capture

- Click the start button next to the desired interface.
- Captured traffic will be displayed in the packet list pane.

- **Packet Capture**

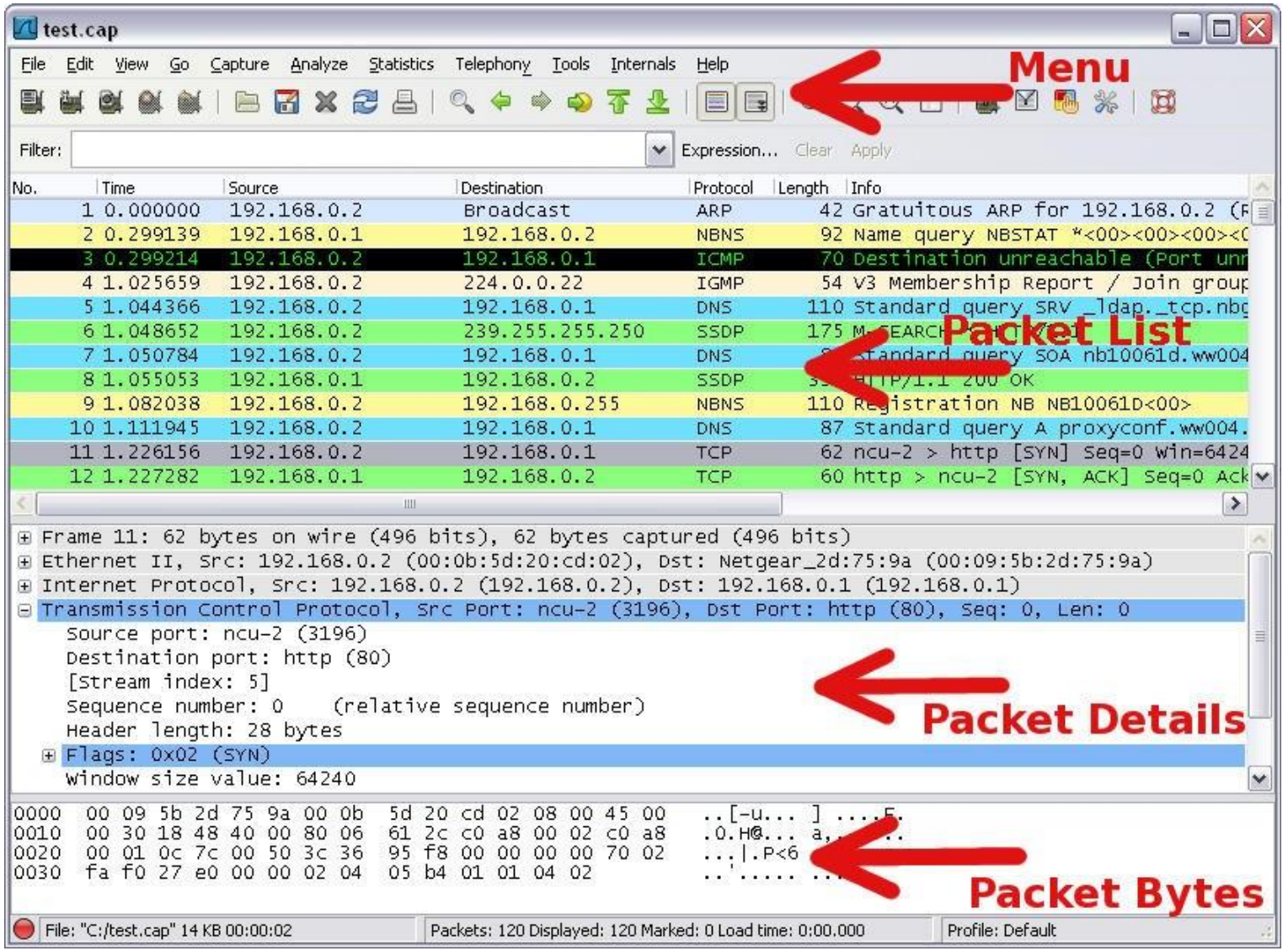
.Stop Capture

- Select Capture -> Stop

.Saving Capture

- Once the capture has been stopped select File -> Save As.
- From the save dialog you can specify file type and which packets to save via the packet range menu.

- **Trace Analysis**



• Trace Analysis

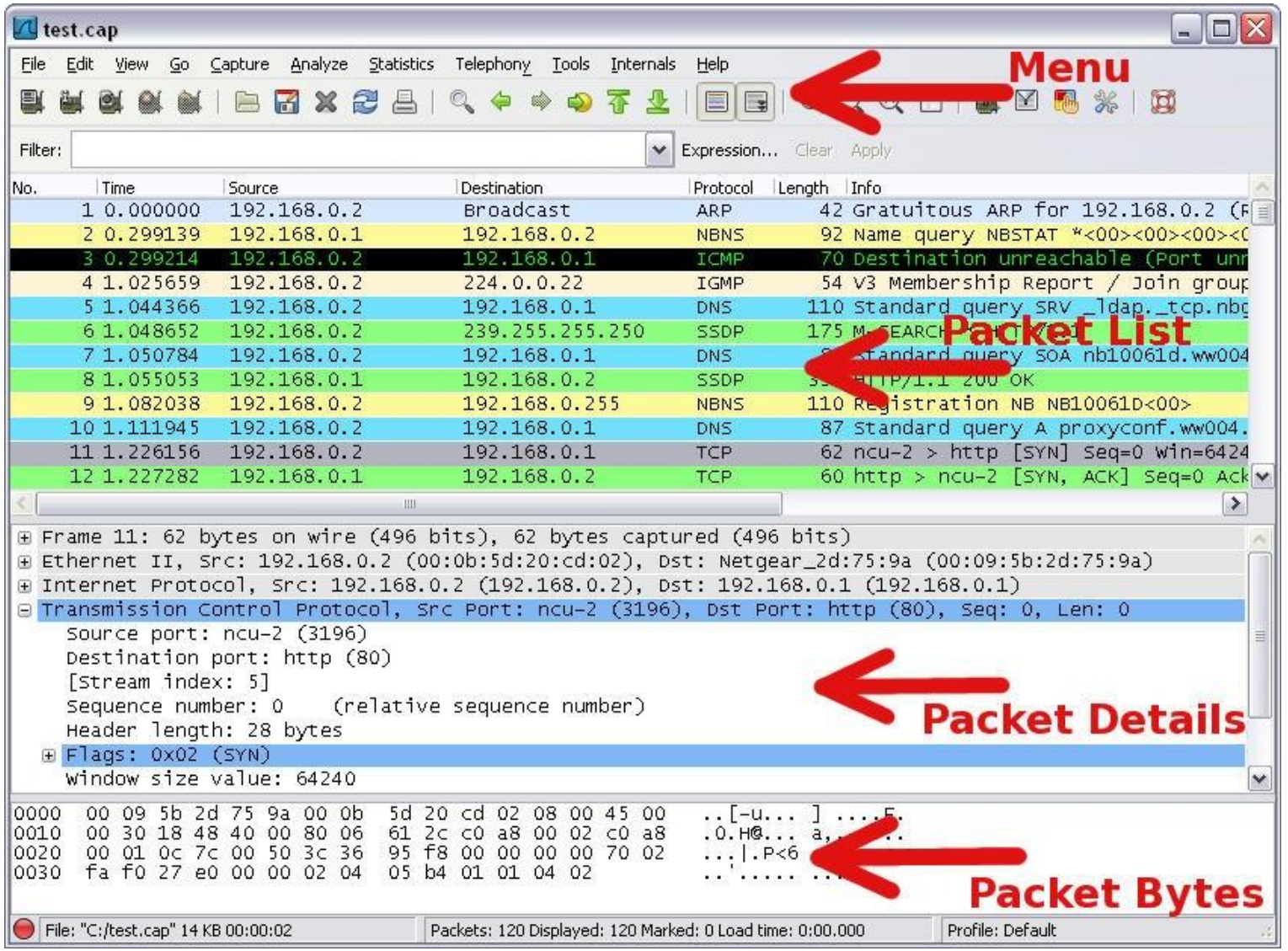
- Packet list
  - Displays all of the packets in the trace in the order they were recorded.
  - Columns
    - Time – the timestamp at which the packet crossed the interface.
    - Source – the originating host of the packet.
    - Destination – the host to which the packet was sent.
    - Protocol – the highest level protocol that Wireshark can detect.
    - Lenght – the lenght in bytes of the packet on the wire.
    - Info – an informational message pertaining to the protocol in the protocol column.

- **Trace Analysis**

.Packet list

- Default Coloring
  - Gray – TCP packets
  - Black with red letters – TCP Packets with errors
  - Green – HTTP Packets
  - Light Blue – UDP Packets
  - Pale Blue – ARP Packets
  - Lavender – ICMP Packets
  - Black with green letters – ICMP Packets with errors
- Colorings can be changed under View -> Coloring Rules

• Individual Packet Analysis



Individual Packet Analysis

- Packet Details
  - Detailed information about the currently selected packet is displayed in the packet details pane.
  - All packet layers are displayed in the tree menu.
  - Any portion of any layer can be exported via a right click and selecting Export Selected Packet Bytes
- Packet Bytes
  - Displays the raw packet bytes.
  - The selected packet layer is highlighted.

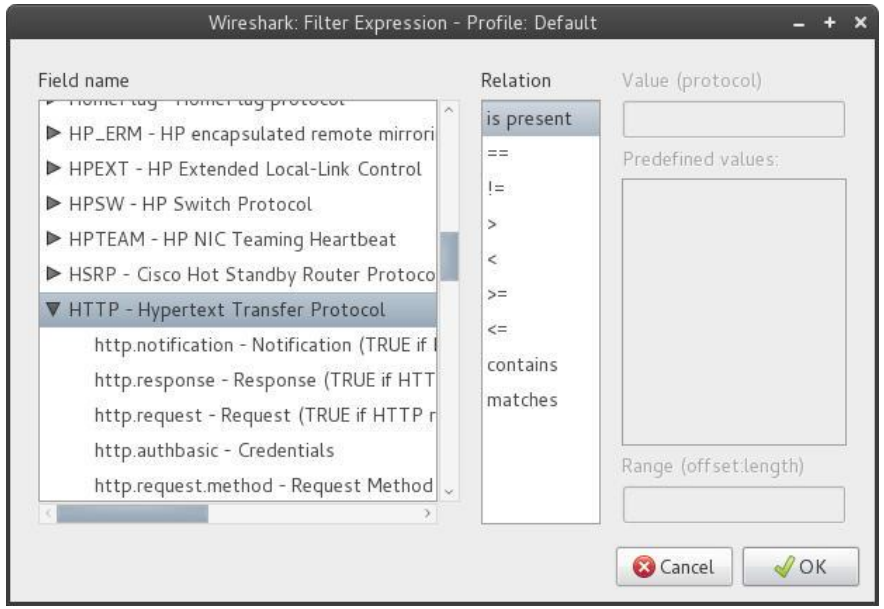
Filters

- Packets captures usually contain many packets irrelevant to the specific analysis task.
- To remove these packets from display or from the capture Wireshark provides the ability to create filters.
- Filters are evaluted against each individual packet.
- Boolean expresions dealing with packet properties.
- Supports regular expressions.
- Can either be manually constructed, composed via the Expressions menu or composed based on a selected packet's properties.

• Filters

.Expressions Menu

- Field name – selects the packet property.
- Relation – selects the boolean test.
- Predefined values – common values against which the selected packet property is tested.
- Value – Arbitrary Textual or Numeric value against which the selected packet property is tested.



- **Filters**

.Compound Filters

- Filters can be composed of multiple tests joined with boolean connectives.
  - && - logical conjuction (i.e. AND)
  - || - logical disjunction (i.e OR)
  - ! - logical negation (i.e. NOT)
- Supports the order of operations.

.Regular Expressions

- Fields can be evaluated against a regular expression using the “matches” test.
- Uses Perl regex syntax.

Filters

- Filter Text Box
  - Green – valid filter
  - Red – invalid filter
  - Yellow – may produce unexpected results
- Packet based filters
  - Filters can be constructed on the basis of individual packets by right clicking on a packet and selecting either:
    - Prepare as filter – creates a filter.
    - Apply as filter – creates a filter and applies it to the trace.
    - Follow TCP Stream – creates a filter from a TCP packet's stream number and applies it to the trace.

- **Filters**

.Filter examples

- http.request – Display all HTTP requests.
- http.request ||http.response – Display all HTTP request and responses.
- ip.addr = = 127.0.0.1 – Display all IP packets whose source or destination is localhost.
- tcp.len < 100 – Display all TCP packets whose data length is less than 100 bytes.
- http.request.uri matches “(gif)$” - Display all HTTP requests in which the uri ends with “gif”.
- dns.query.name = = “www.google.com” - Display all DNS queries for “www.google.com”.

Exercises

Exercise 1: Basic Packet Capture and Analysis

Objective: Learn to capture network traffic and understand packet structures.

1. **Step 1:** Open Wireshark and start a packet capture on the network interface that has internet connectivity.
2. **Step 2:** Browse to a website (e.g., http://example.com) to generate some traffic.
3. **Step 3:** Stop the capture after the page fully loads.

Tasks:

- Filter HTTP packets and identify the GET request for the page.
- Analyze the TCP 3-way handshake (SYN, SYN-ACK, ACK) for the connection.
- Find the server's IP address and the port used for the HTTP service.
- Take a screenshot of the packet structure and identify fields such as the source and destination IP addresses, source and destination ports, and sequence numbers.

Exercise 2: Analyzing DNS Requests

Objective: Understand how DNS requests are handled in network communication.

1. **Step 1:** Clear your DNS cache to ensure all DNS requests are fresh.
2. **Step 2:** Start Wireshark and begin capturing on your active network interface.
3. **Step 3:** In a browser, visit several websites (e.g., open Google, Yahoo, etc.) and then stop the capture.

Tasks:

- Use a filter to isolate DNS traffic (dns in the filter).
- Identify at least one DNS query for each website and find the corresponding response.
- Note the response time and TTL (time-to-live) for each DNS query.
- Examine the DNS response and identify if any DNS queries returned multiple IP addresses.

Exercise 3: Monitoring HTTP and HTTPS Traffic

Objective: Differentiate between HTTP and HTTPS traffic and understand how encryption affects packet analysis.

1. **Step 1:** Start capturing packets in Wireshark on your network interface.
2. **Step 2:** In a browser, visit a website that uses HTTP (e.g., http://neverssl.com) and another that uses HTTPS (e.g., https://example.com).
3. **Step 3:** Stop the capture.

Tasks:

- Use a filter to view only HTTP and HTTPS packets.
- Compare the packets from the HTTP and HTTPS connections. Note which fields are visible and inspectable in HTTP vs. HTTPS.
- Identify any unencrypted data in the HTTP packets, such as the contents of the GET requests.
- Try to locate the TLS handshake in the HTTPS packets. Identify fields such as the ServerHello and Certificate messages.