EXPERIMENT 1

1. Familiarization with a traffic generator and a packet analyzer in the process of basic packet synthesis and analysis

1.1 Aims

1. Connect the traffic generator to a Docker container, as shown in Fig. 1 below.
2. Create an ICMP "echo request" stream.
3. Inspect an echo-request & reply pair using Wireshark
4. Count the number of layers.
5. Identify the MAC addresses, and note the layer wherein they are found.
6. Identify the IP addresses, and note the layer wherein they are found.

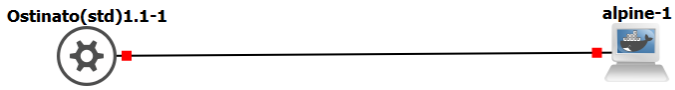

Fig. 1: Setup for first experiment

1.2 Procedure

1.2.1 Deploy an instance of the Ostinato appliance

1. Drag and drop the "Ostinato (std) 1.1" icon to the workspace. This creates an Ostinato instance from the appliance in the GNS3 library.
2. Right click the icon and select “Start” to start the Ostinato instance.
3. Double-click on the Ostinato appliance to open a VNC client (TightVNC) that connects to the VNC server embedded in the Ostinato instance. The Ostinato instance’s graphical user interface (GUI) will appear, as shown in Fig. 2.

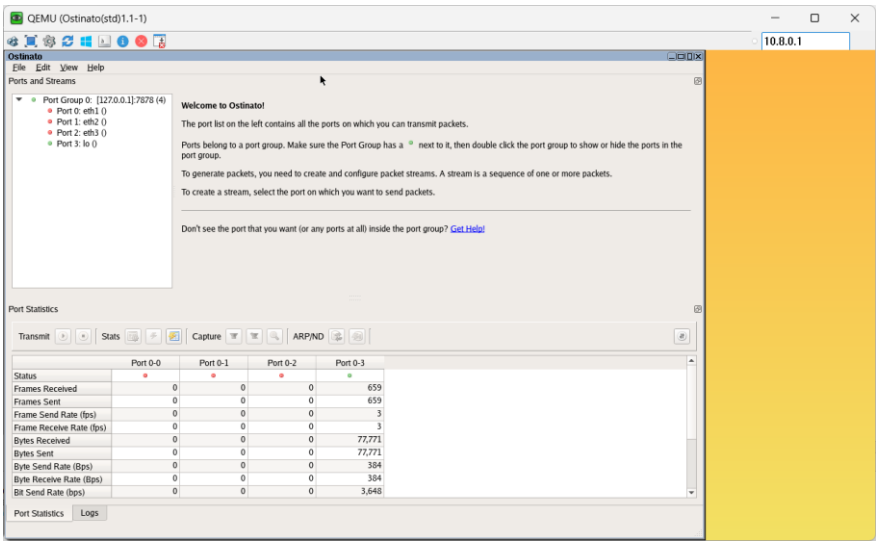

Fig. 2: Appearance of Ostinato GUI

1.2.2 Configuration of IP addresses

A network traffic generator is a flexible tool that supports the crafting of packets for injection into a test environment.

In order to set up an IP address on the adapters (by keeping defaults during installation, we are provided with four adapters), it is necessary to create a "device group", and use the device group's parameters to set up IP addresses on the ports subsumed within the device group.

The first step in creation of the device group is shown in Fig. 3; on invoking the context menu, the “new device group” option is presented.

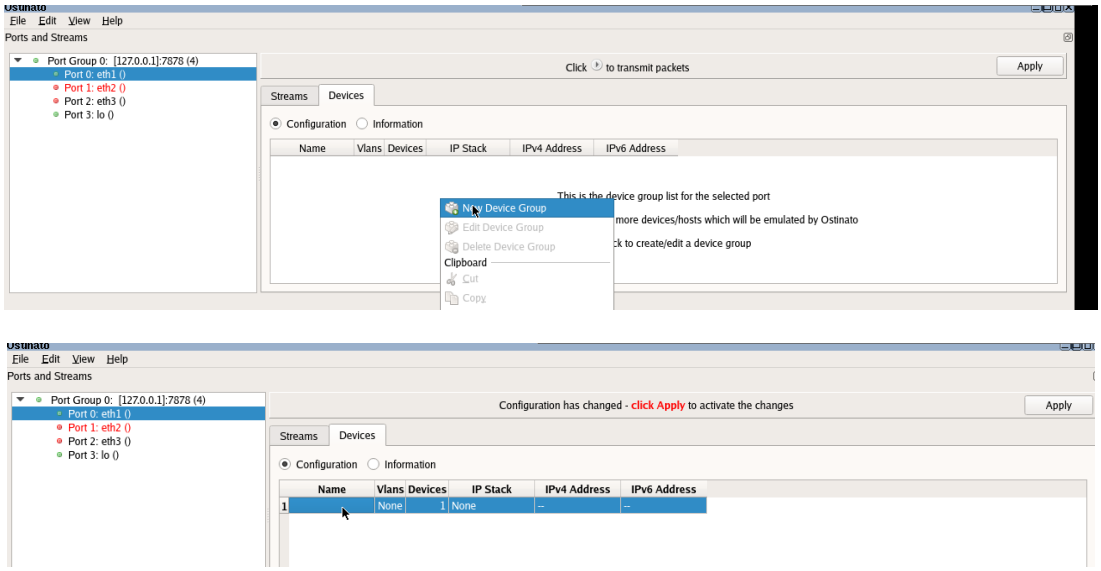

Fig. 3: Start by creating a device group bound to a port

Next: set the "Name" field, the “IP Stack” field and the “IPv4 address” field as shown and leave other fields with their defaults. The “gateway” field’s value is **redundant** and can be cleared (see Fig. 4). To end creation of the device group, click "OK".

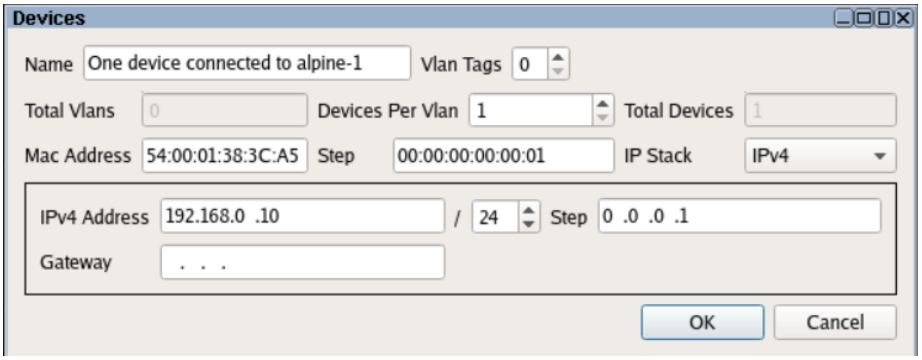

Fig. 4: Configuration of the device group

1.2.3 Docker container network interface configuration and launch

1. Drag-and-drop an instance of the Docker container (Fig. 5) onto the workspace, as follows:
    a. Context menu (right click in Windows) for instance of docker container.
    b. Select "Edit config"
    c. Uncomment the following lines:
        i. auto eth0
        ii. iface eth0 inet static
        iii. address 192.168.0.2
        iv. netmask 255.255.255.0
    d. Edit the address (in line 1(c)(iii)) to 192.168.0.1 (instead of 192.168.0.2)
2. Select the link icon and use it to set up a link between the docker container's sole Ethernet interface and port "Eth1" on the Ostinato instance.
3. Right-click the Docker container instance icon and click on "Start"

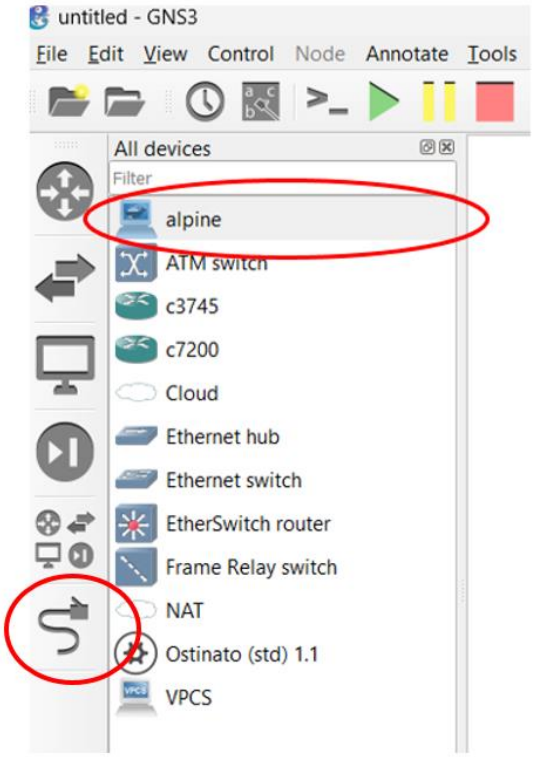

Fig. 5: The alpine flavour Docker container, and the link tool

1.2.4 Create an ICMP Echo Request stream

1. In the Ostinato GUI, highlight Eth1 and switch to the streams tab (Fig. 6).
2. Pull up the context menu (e.g., by right-click) in the empty space of the streams tab, and select "New Stream" (Fig. 7).

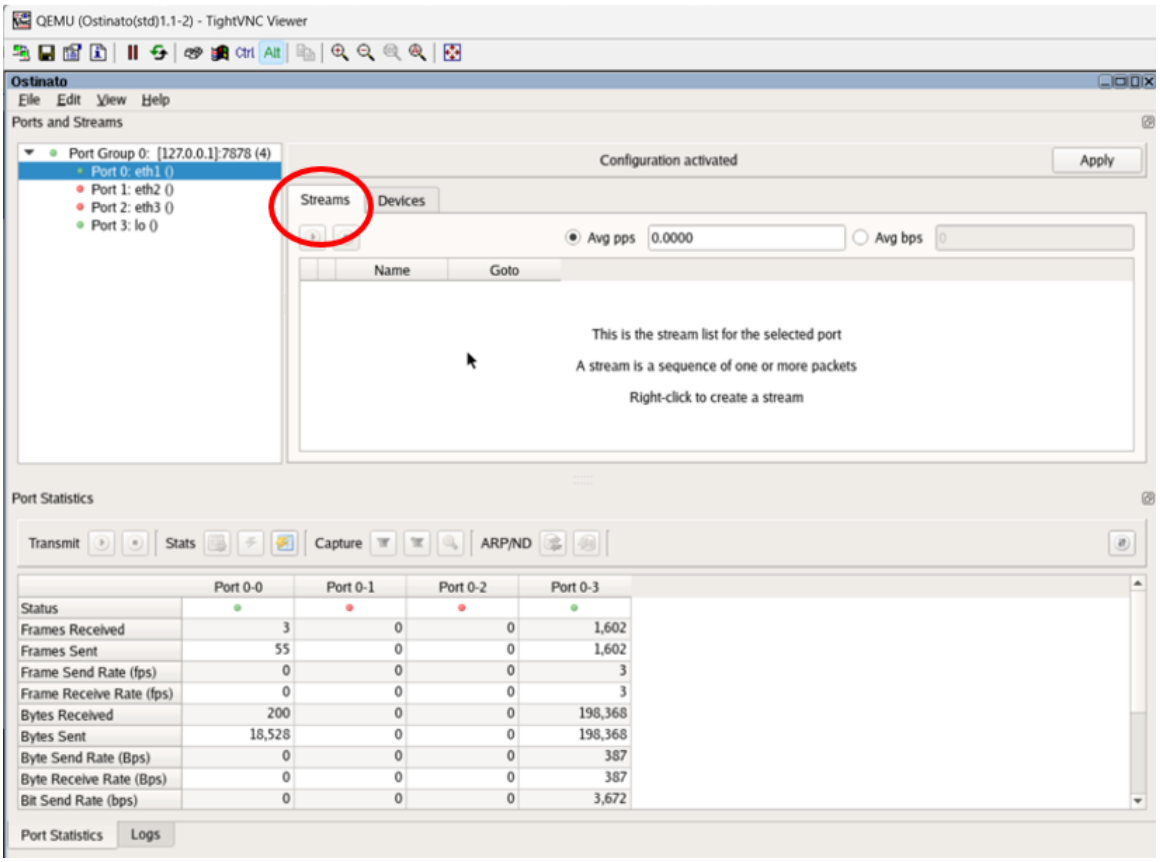

Fig. 6: First step in creation of a packet stream on the port used in the schematic

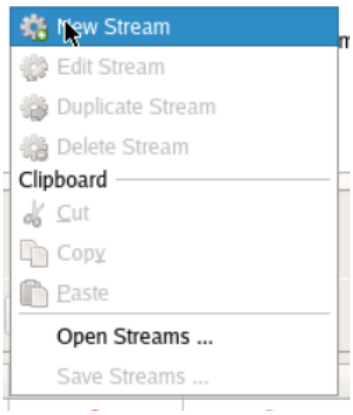

Fig. 7: Second step in creation of a packet stream on the port used in the schematic

3. In the protocol selection tab, name the stream "ICMP Echo Request Stream" and select the layer combinations shown below (Fig. 8).
4. In the protocol data tab, under the “Media Access Protocol” section, set Ostinato to use ARP to resolve IP addresses into layer 2 (MAC) addresses (Fig. 9).

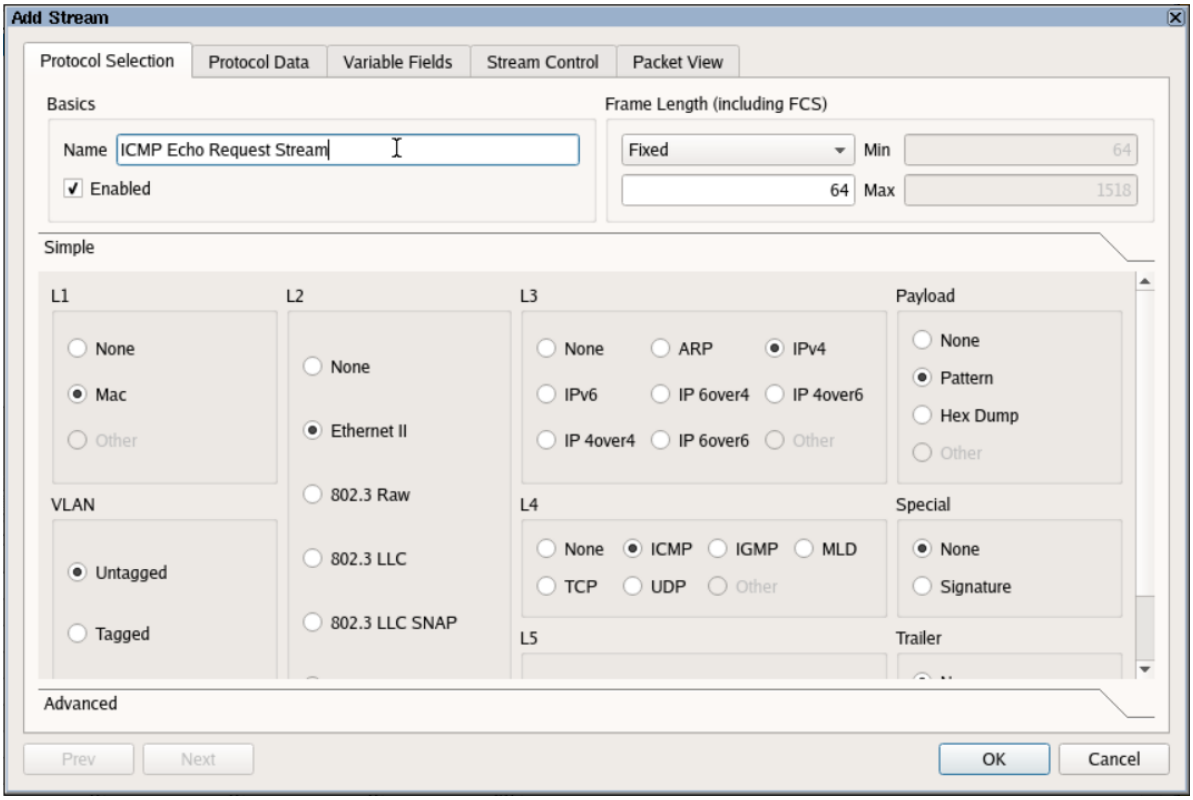

Fig. 8: Third step in creation of a packet stream on the port used in the schematic

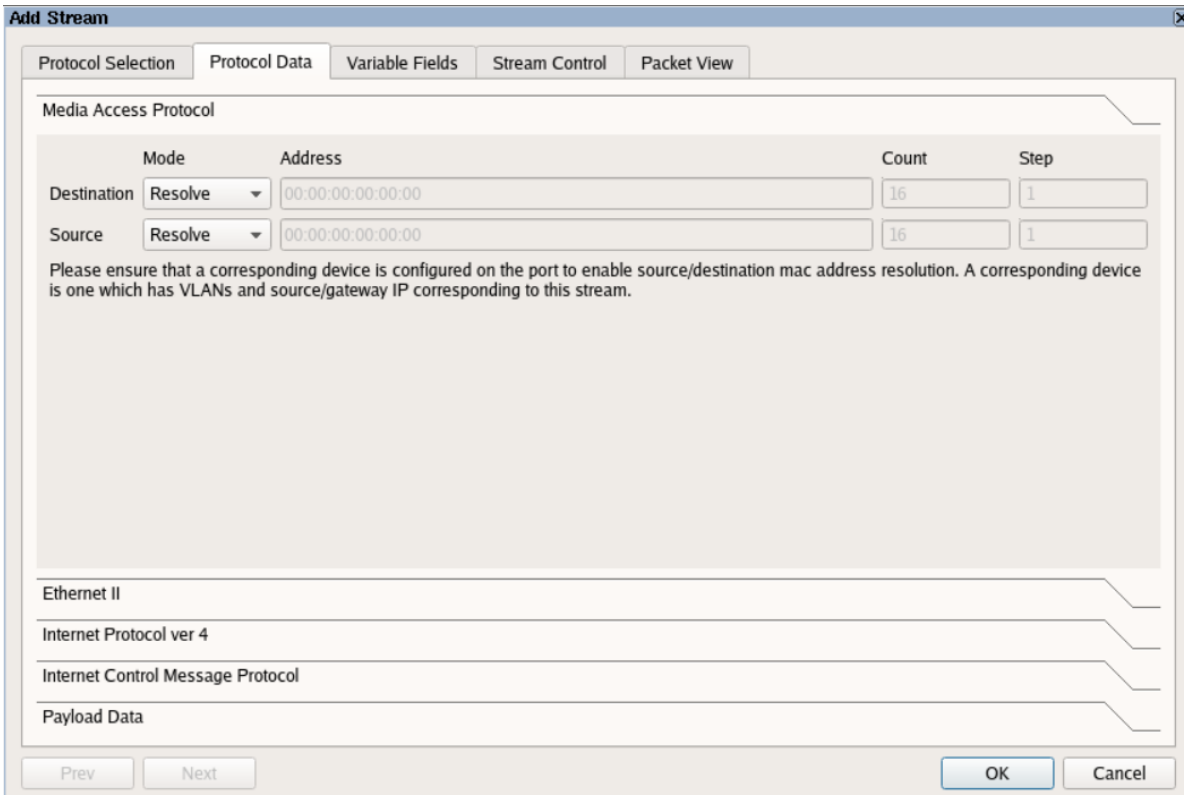

Fig. 9: Set use of ARP for both source and destination IP addresses

5. In the protocol data tab, under the “Internet Protocol ver 4” section, set Ostinato to use the source and destination layer 3 addresses shown in Fig. 10.

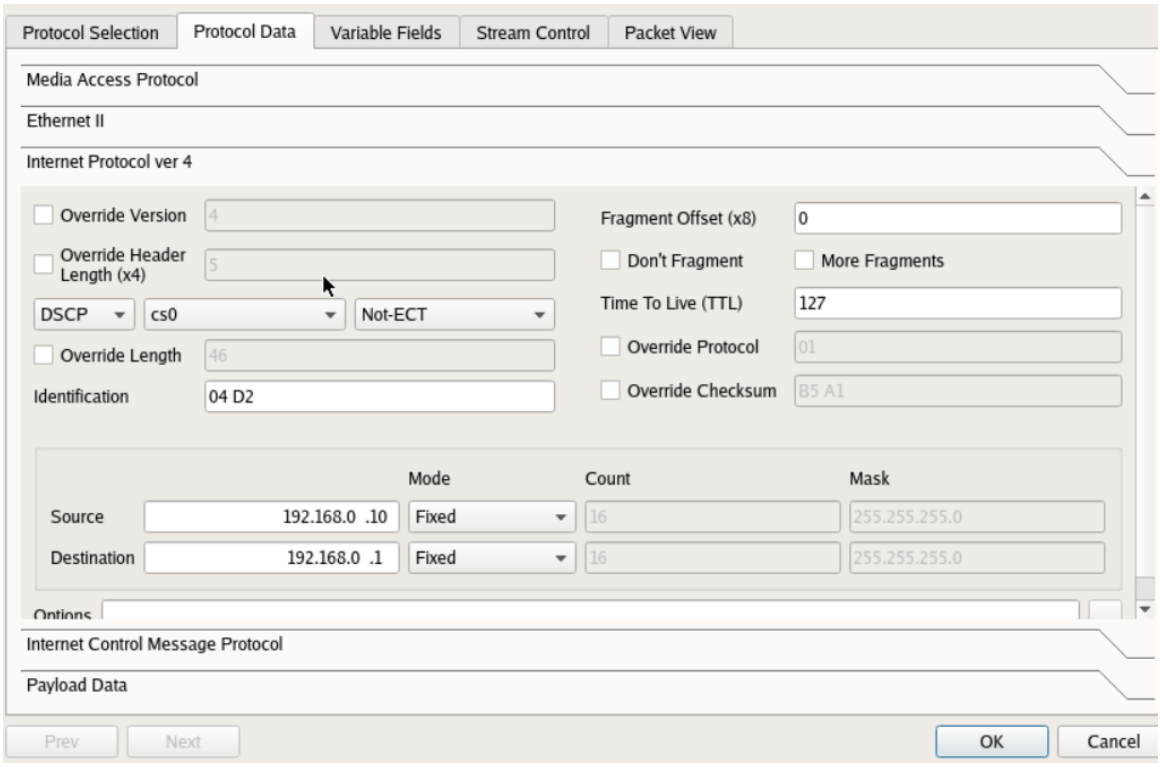

Fig. 10: Set use of ARP for both source and destination IP addresses

On exiting, click Apply.

QUESTION 1

Configuration of the device group can best be understood in terms of what is happening “under the hood” (i.e., beneath the surface, or behind the interface which the GUI provides). Describe the activities “under the hood” in terms which you have learnt in your study of the OSI 7-layer model.

10 marks

1.2.5 Packet monitoring ("sniffing")

Packets are monitored ("sniffed") using Wireshark, which was installed along with the GNS3 operating environment (OE) installation (unless GNS3's installation defaults were changed). Verify that Wireshark is installed before proceeding to the steps below.

1. Move cursor over to the link between packet generator and the Docker container (henceforth: the device under test (DUT)).

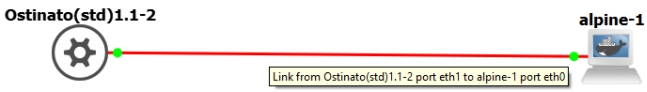

Fig. 11: Highlight the link between the packet generator and the Docker container

2. Bring up the context menu (e.g., by right-clicking if using Windows) on the link while it is highlighted red (be careful - the link loses highlight easily), and therefrom, click on "Start capture".

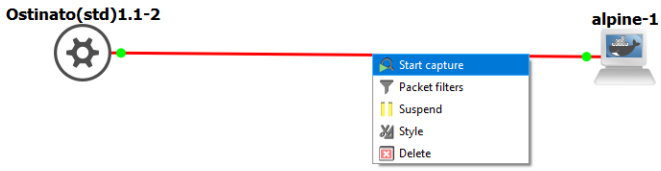

Fig. 12: Bring up the link's context menu

3. Let the defaults offered stand:

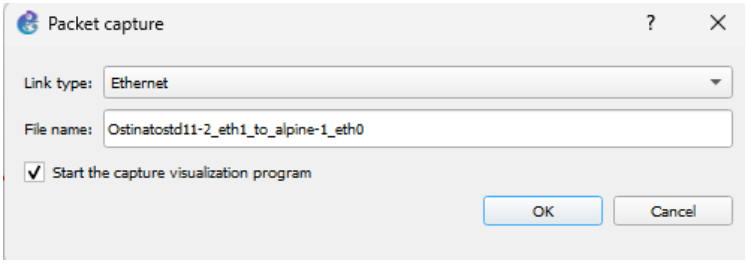

Fig. 13: Defaults, prior to starting capture

4. In the display filter bar, enter the text "icmp", to filter out all packets traversing the link except for icmp packets. After you enter the text, hit "Enter" and note that any packets displayed will disappear from view.

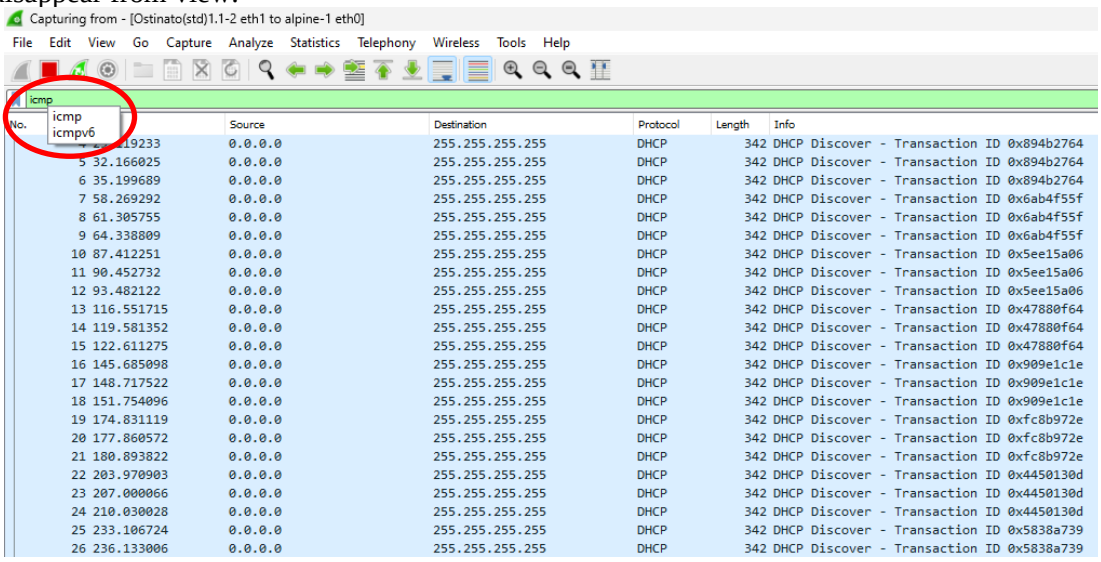

Fig. 14: Set a display filter to clear all packets except ICMP packets …

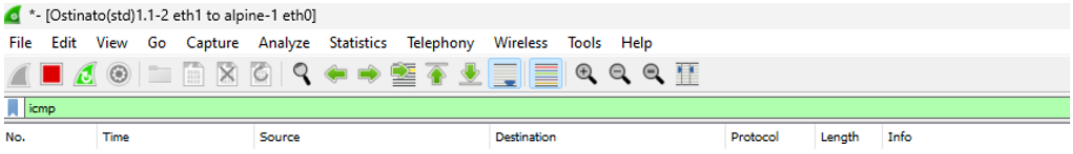

Fig. 15: … and the clutter is cleared.

5. Go to Ostinato's user interface, and pull up the streams tab on the port group. Click on the "play" button.

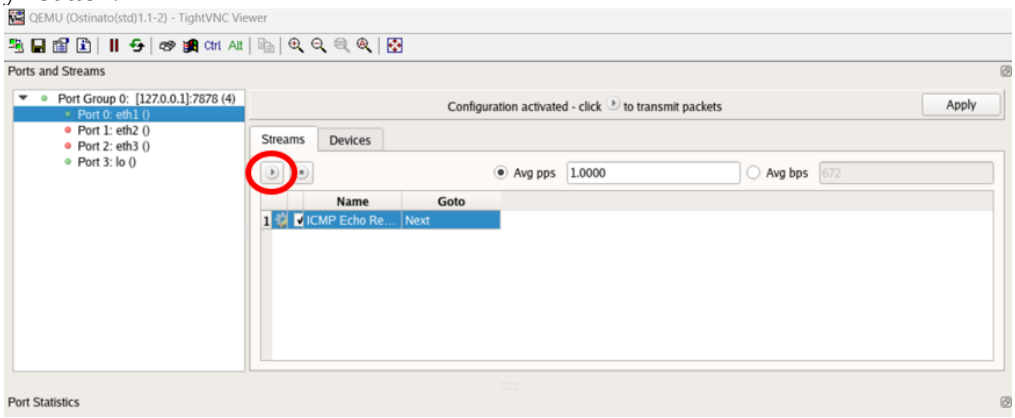

Fig. 16: Play the stream

6. Observe the Wireshark display capture.

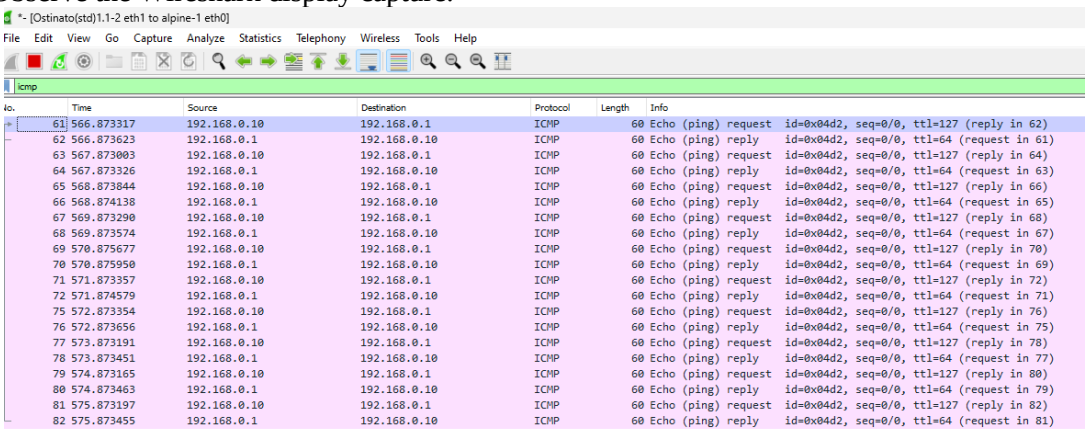

Fig. 17: Captured ICMP packets

1.3 Analysis

Fig. 18 shows the anatomy of the Wireshark user interface (UI).

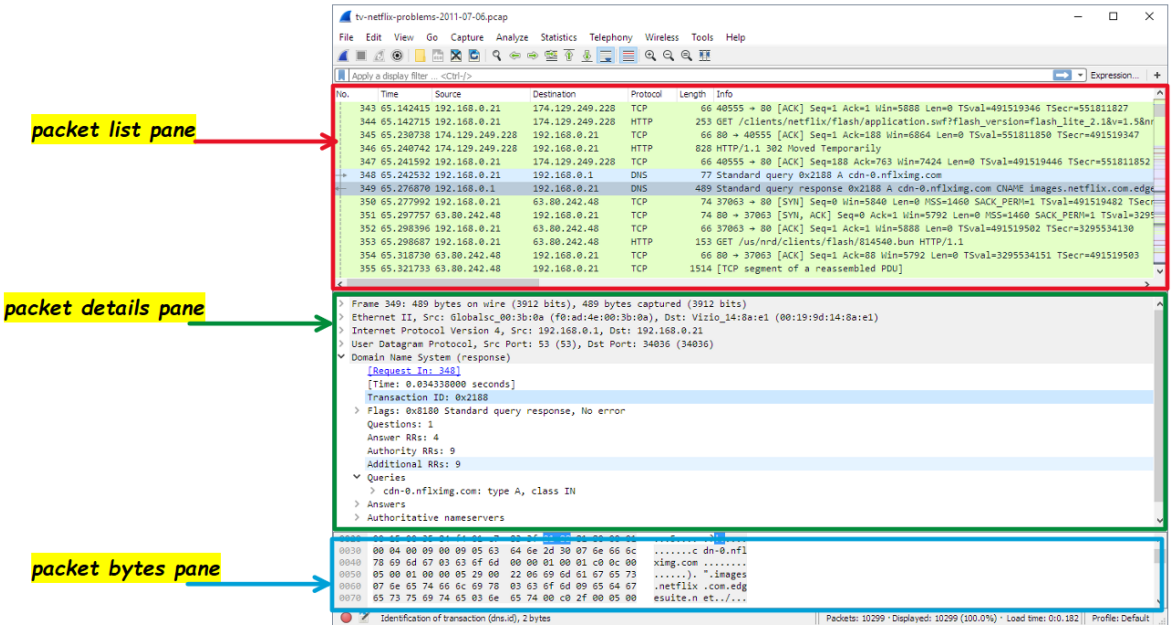

Fig. 18: Packet list pane, packet details pane and packet bytes pane in the Wireshark UI

1.3.1 Basic Measurements

QUESTION 2

2.1 Count the number of packets that you have captured and compare this with the setting in the stream configured on the packet generator.

2.2 Write down the numbers (leftmost column) of the packets that have been generated by:

- the packet generator
- the DUT

2.3 Use your answer to 2.2 to find the average inter-packet delay for the packets generated by the packet generator.

5 marks

7. Change the packet rate to 2 packet/s.
8. Right-click on the link between the packet generator and the DUT.
9. Stop the capture.
10. Return to the packet stream in the packet generator and edit it through the "stream control" tab to send 2 packets per second.
11. Restart capture and apply the "icmp" display filter.
12. Re-run the packet stream.

QUESTION 3

Repeat the exercises of question 3 with the new packet rate (2 pkt/s).

5 marks

13. Return the packet rate to 1 packet/s
14. Right-click on the link between the packet generator and the DUT.
15. Stop the capture.
16. Return to the packet stream in the packet generator and edit it through the "stream control" tab to send 1 pkt/s.

17. Restart capture and apply the "icmp" display filter.
18. Re-run the packet stream and proceed to packet inspection (section 3.2).

1.3.2 Packet inspection

1. Identify one packet sent by the packet generator and the corresponding reply sent by the DUT.
2. In the ***packet list pane***, highlight the packet sent by the packet generator. Let this be packet T.
3. In the ***packet details pane***, collapse all T's sections and list them (e.g., the packet details pane in Fig. 19 has all sections collapsed, except the "Domain Name System" section).

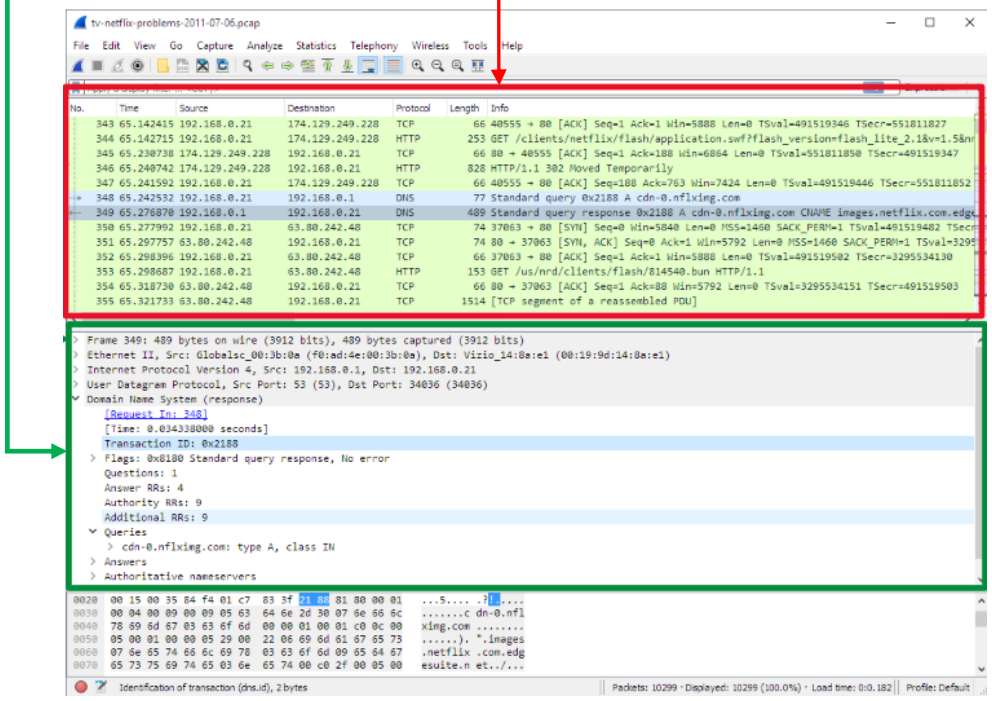

Fig. 19: Inspecting a single packet in Wireshark's UI

QUESTION 4

4.1 Expand the frame section and inspect it to find out the number of bytes captured by Wireshark from the link ("bytes on wire").
4.2 Search for the minimum length of an Ethernet packet, and state it.
4.3 How does the number of bytes captured (which you found in (4.1)) compare with the minimum length of an Ethernet packet (which you looked up in (4.2))? Can you explain the difference?
4.4 Expand the Ethernet II section and write down the source MAC address and the destination MAC address.
4.5 Look up the source MAC address in the device group configured on the packet generator at the port group.
4.6 Compare what was found in 5.4 with what was found in 5.5.
4.7 Inspect the Ethernet II section and find the field in that section which the receiver (the DUT) uses to identify the layer 3 entity which is to receive the Ethernet frame's payload.
4.8 Inspect the section below the Ethernet II section and:

- write down the source address and the destination address that you see in this underlying section;
- compare the source address and the destination address with what you have set up in the stream named "ICMP Echo Request Stream".
- Look up the field named "Total length" in this section and account for the difference between this number and the number you have found in 4.1.

15 marks

EXPERIMENT 2

2. Familiarization with some basics of Ethernet networks and with transport protocols

2.1 Aims

1. **Phases 1, 2**
    a. To investigate the role of the address resolution protocol (ARP)
    b. To compare packet transfer on a hub and a switch

2. **Phase 3:** To measure the performance of the transmission control protocol (TCP) on a simple link

2.2 Procedure

2.2.1 Procedure: topology deployment and IP address configuration

1. Deploy the instances as shown in Fig. 20.
2. Configure the IP addresses as indicated in Fig. 20.

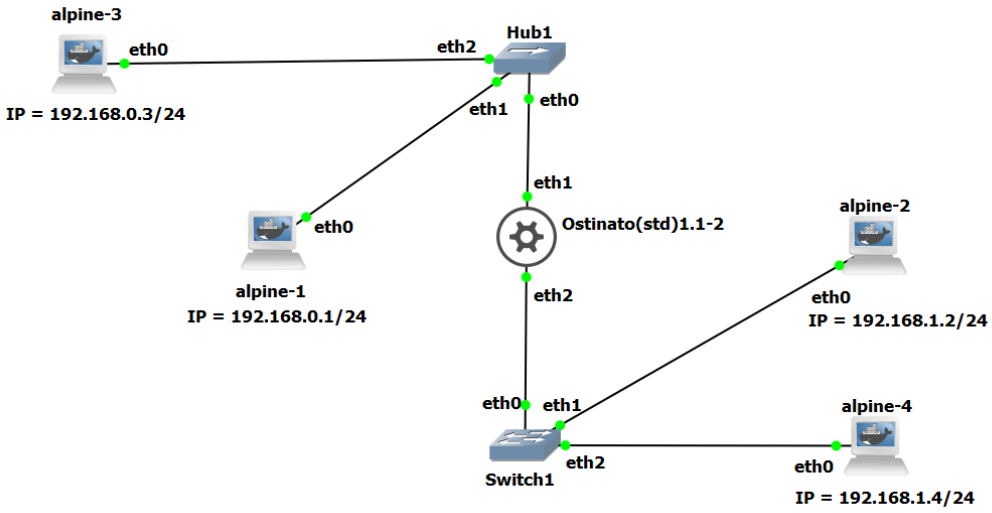

Fig. 20: Physical topology and interface IP addresses

2.2.2 Procedure: phase 1

1. First (phase 1, part 1), the address resolution protocol will be disabled. An overview follows:
    (a) ARP will be disabled in Ostinato and alpine-3's MAC address will be provided manually to Ostinato. Alpine-3's MAC address will be provided in the process of configuring the Ostinato ping stream.
    (b) When alpine-3 receives the ICMP echo request, it will not have yet run ARP to map Ostinato's IP address to its MAC address. Therefore, alpine-3 will be unable to send an ICMP echo reply back to Ostinato, as it will not have the MAC address corresponding to the ICMP packet's source IP address (Ostinato's IP address from eth1).
    (c) Next, an ARP mapping will be manually set up (command-line configuration) on alpine-3.
    (d) With this mapping, alpine-3 can match Ostinato's IP address to a MAC address.

2. Next (phase 1, part 2), ARP will be enabled.

2.2.2.1 Phase 1, part 1: Setting up a static layer 2 – layer 3 address mapping (ARP disabled)

1. After deployment, your Ostinato dashboard should show ports 0, 1 and 3 lit:

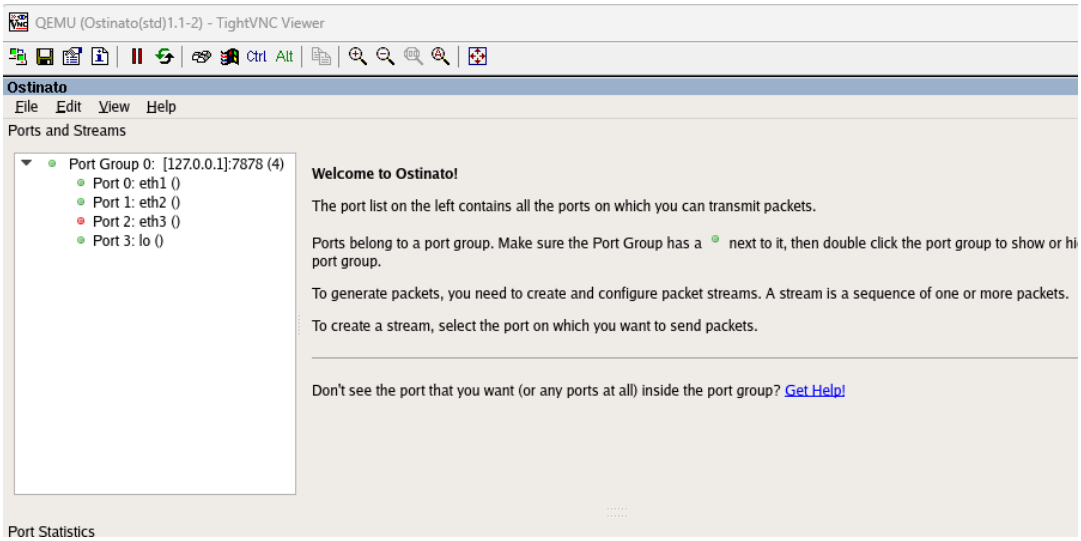

Fig. 21: Lit ports (eth1, eth2 and lo are able to send and receive packets)

2. Double-click on alpine-3 to open up a console, and run the command shown to obtain the container's MAC address (your MAC address will almost certainly differ from the one shown.

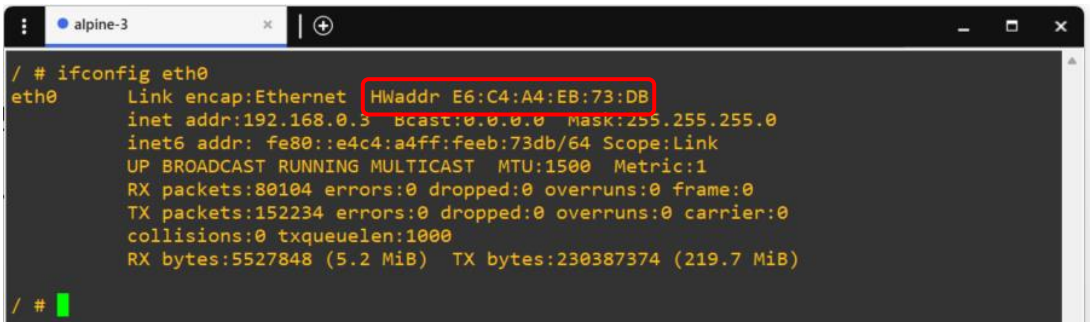

Fig. 22: Inspecting alpine-3 to obtain its layer 2 address

3. Configure a packet stream.
    a. In the protocol selection tab, set up the stream as shown in Fig. 23.

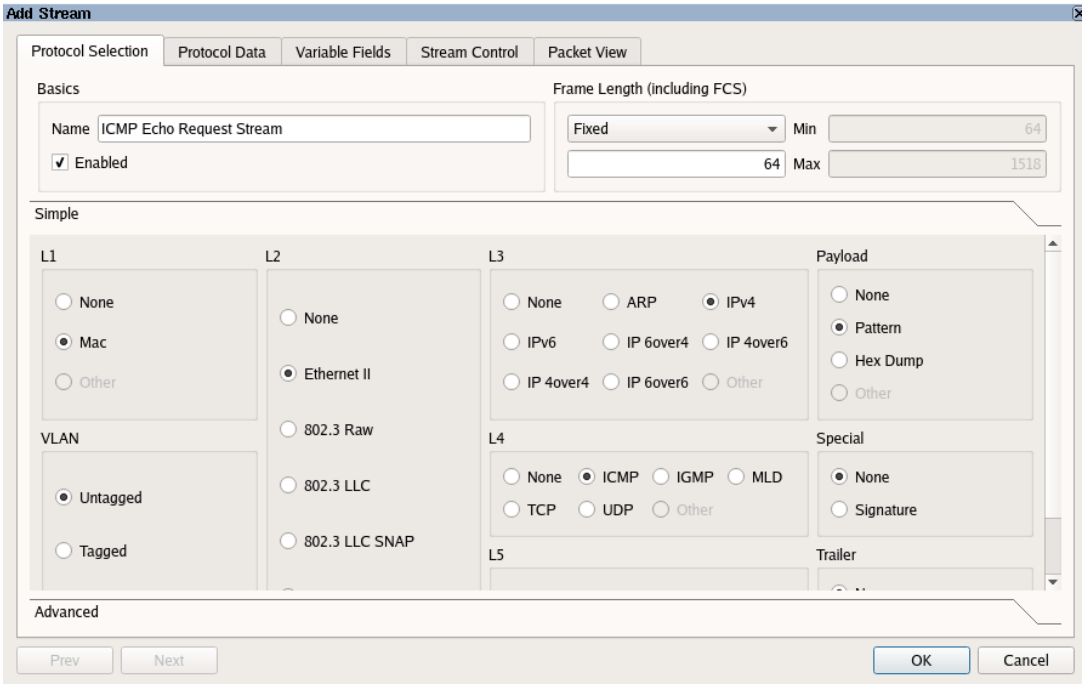

Fig. 23: Packet stream configuration – step 1

b. Under the protocol data tab, media access protocol section (see Fig. 24), specify fixed layer 2 addresses for both source and destination. As ***Destination***, use the one shown in alpine-3's console and some six-octet number for the ***Source***, e.g., 00:00:00:00:00:01 (note that the most significant octet - leftmost - of the Source address must have zero in the least significant two bits, i.e. xxxxxx00).

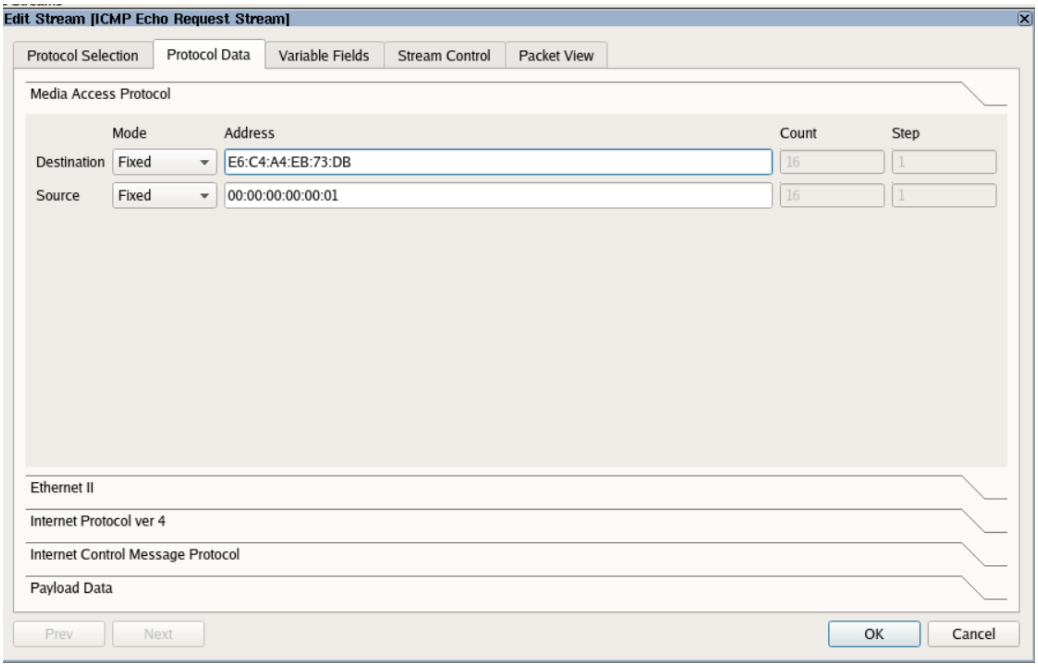

Fig. 24: Packet stream configuration – step 2

c. Under the protocol data tab, Internet Protocol ver 4 section (see Fig. 25), specify source and destination IP addresses. The source address can be chosen from the free set in the 192.168.0.0/24 range. In Fig. 25, the chosen source address is 192.168.0.10. The destination address is that which pertains to alpine-3's eth0 interface, i.e., 192.168.0.3.

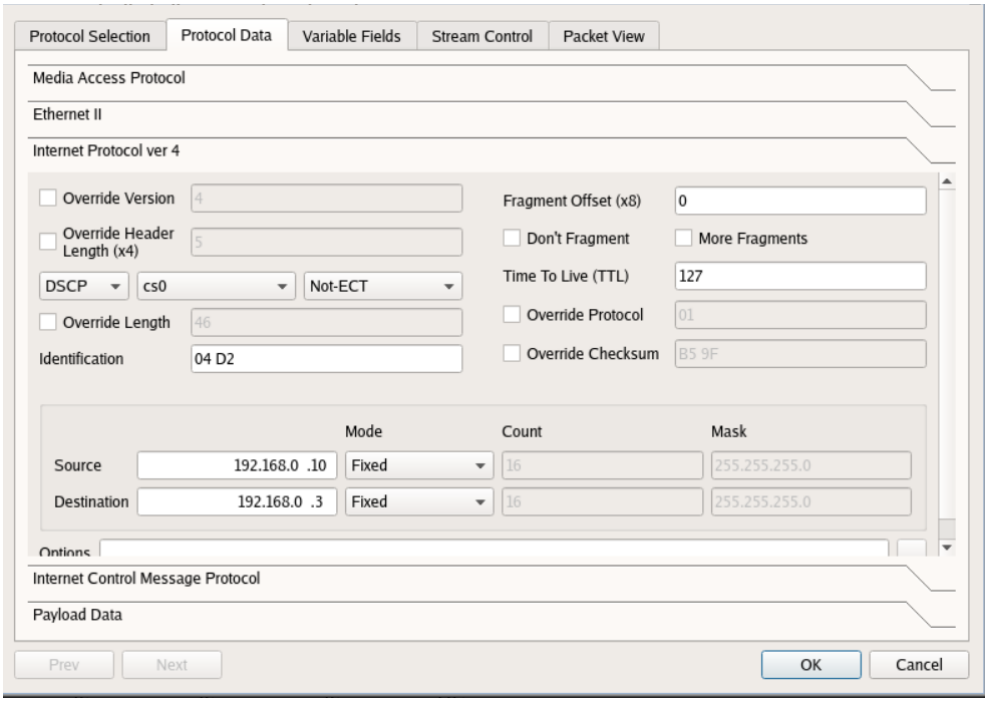

Fig. 25: Packet stream configuration – step 3

d. Under the stream control tab, specify that only one packet needs to be generated.

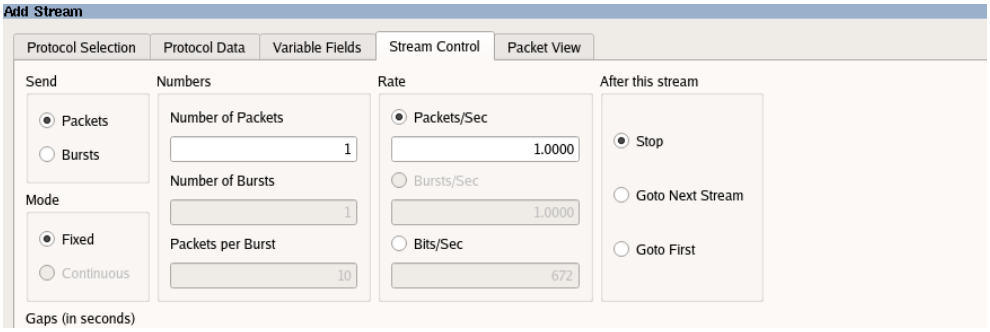

Fig. 26: Packet stream configuration – step 3

e. Implement the packet stream by clicking on "Apply" (Fig. 27).

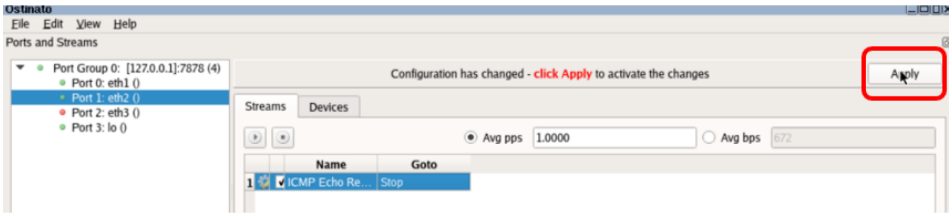

Fig. 27: Implementation of the configured packet stream

4. Start capturing on the ***hub1 - alpine-1*** cable run. Note that neither Ostinato nor alpine-3 are present on this cable (Fig. 28).

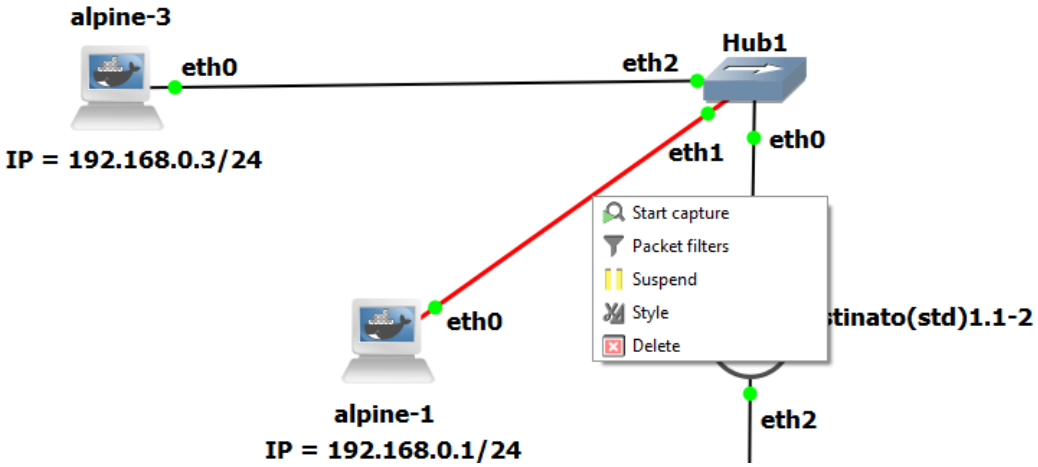

Fig. 28: Start capturing on cable run hub1 – alpine-1

5. Apply the following display filter: icmp or arp.
6. Run the stream. Note the packets captured.
7. Go to alpine-3's console again and type the following command:
   ```
   arp -s 192.168.0.10 0:0:0:0:0:1
   ```
   This command populates alpine-3's ARP cache with an entry which would otherwise be created dynamically (automatically) by an exchange of ARP packets between Ostinato and alpine-3; however, this time, through the fixed settings in step #3 (above), we have precluded the ARP exchange.
8. Repeat the procedure and note the packets captured.
9. Once the procedure is complete, delete the ARP entry for 192.168.0.10 using the command:
   ```
   arp -d 192.168.0.10
   ```

2.2.2.2 Phase 1, part 2: Dynamic ARP

The purpose of phase 1, part 2 is two-fold:

1. observe ARP's role in converting IP addresses into MAC addresses, and
2. observe the function of the hub, for contrast later with the function of the switch.

Repeat the experiment in phase 1, part 1, but this time capture packets as follows:

1. start capture on hub-1 <-> alpine-1 ***before creating the device group***. Note that Ostinato effects ARP resolution when the device group is created! We want to capture Ostinato's resolution of (layer 3) IP address 192.168.0.3 into a (layer 2) MAC address; therefore, we start capturing before creating the device group.
2. Use display filter **`icmp or arp`**.
3. Create the device group (see Fig. 29). Note that a device group was ***not*** created during part 1's operations.

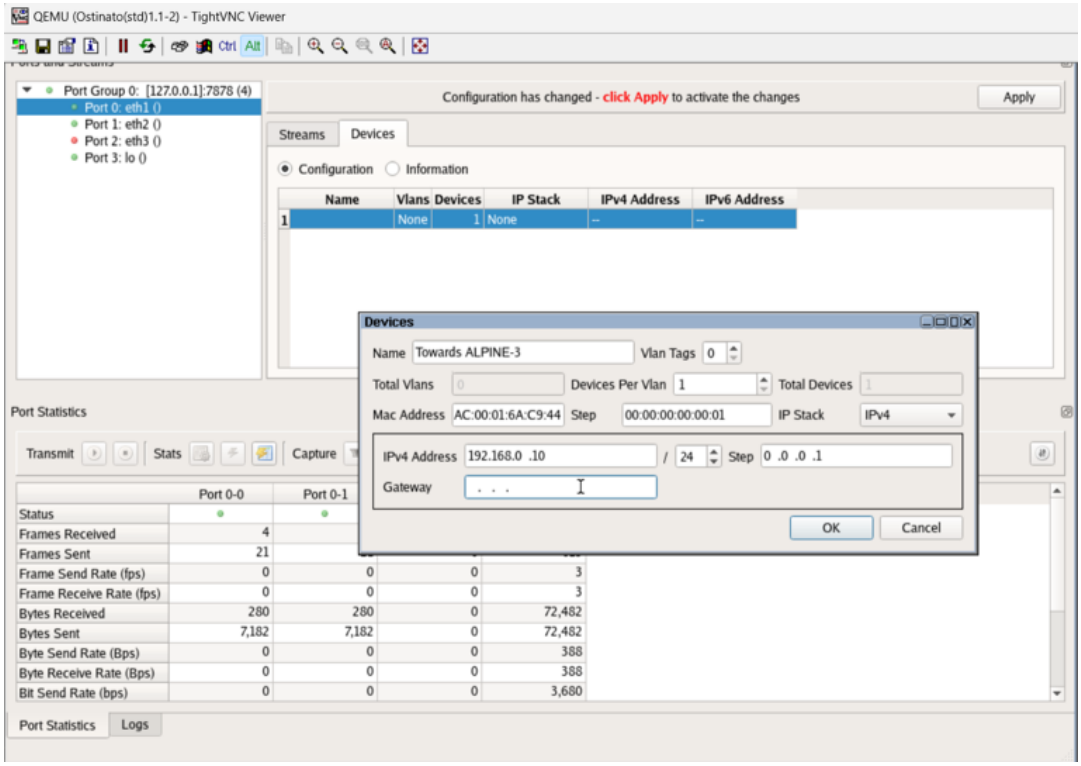

Fig. 29: Creation of device group on Ostinato port connected to hub

4. Create an ICMP Echo Request packet stream, as shown in Figs. 30(a) – (d), ***but do not apply it***. "Apply" would implement the configuration and trigger ARP resolution.

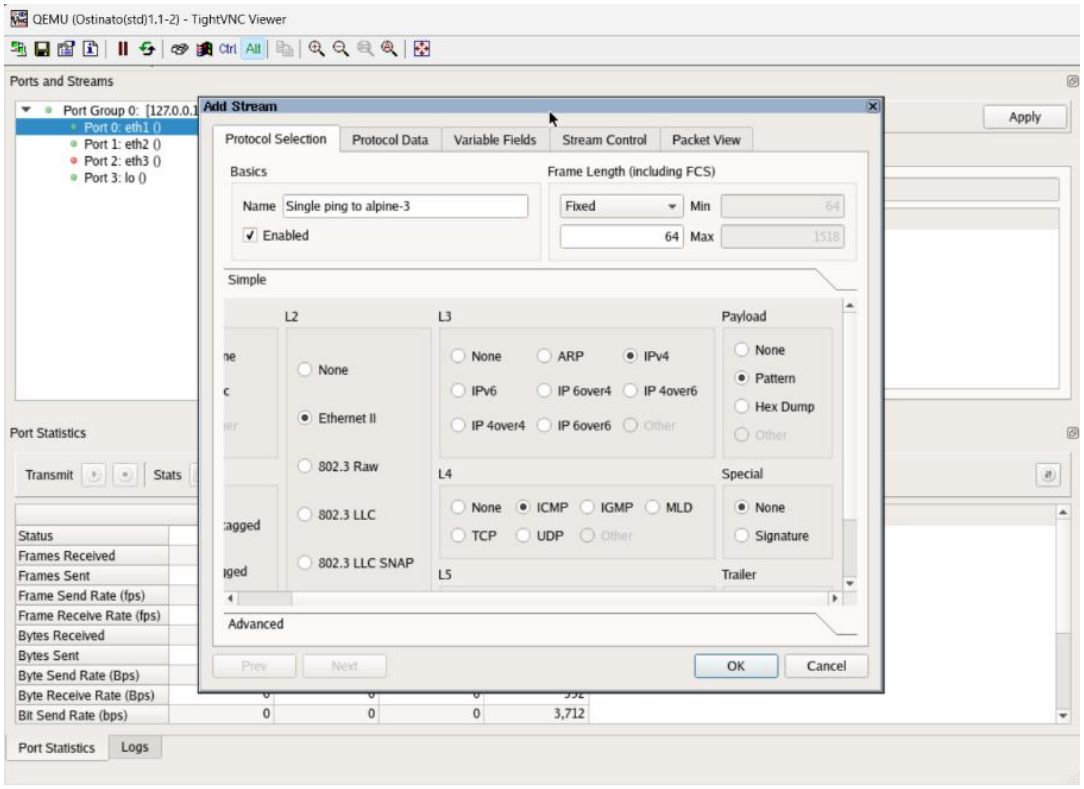

Fig. 30(a): Packet stream creation - "protocol selection" tab.

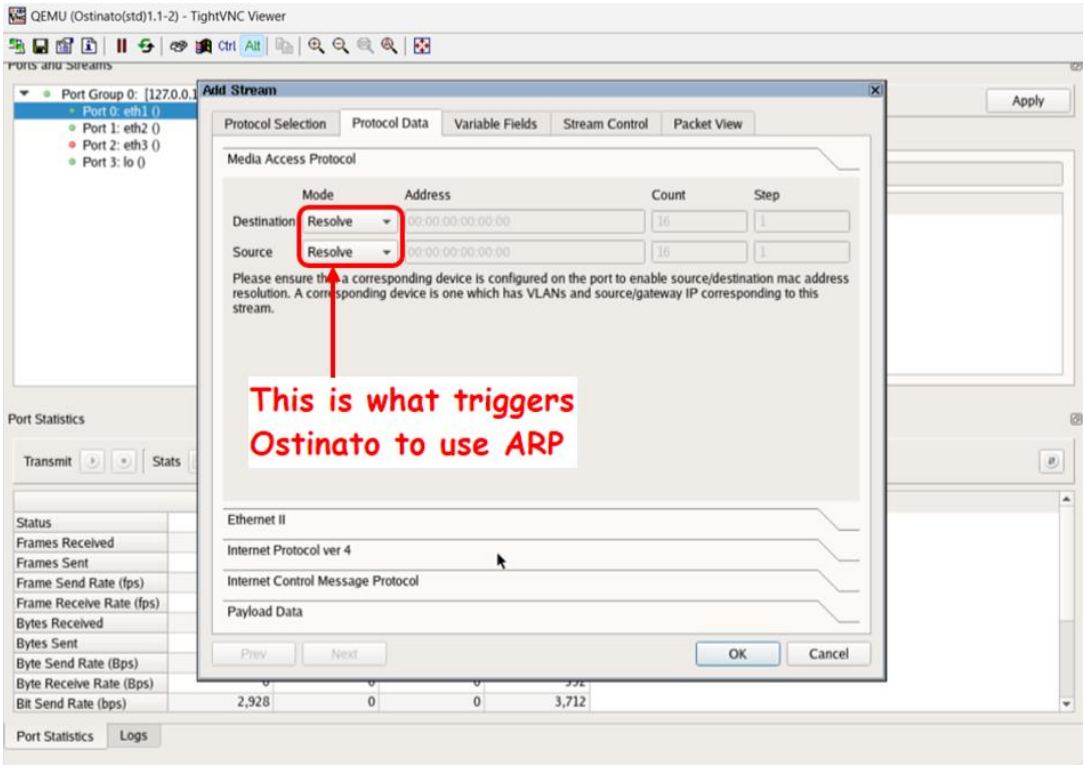

Fig. 30(b): Packet stream creation - specify protocol data – layer 2 operating mode

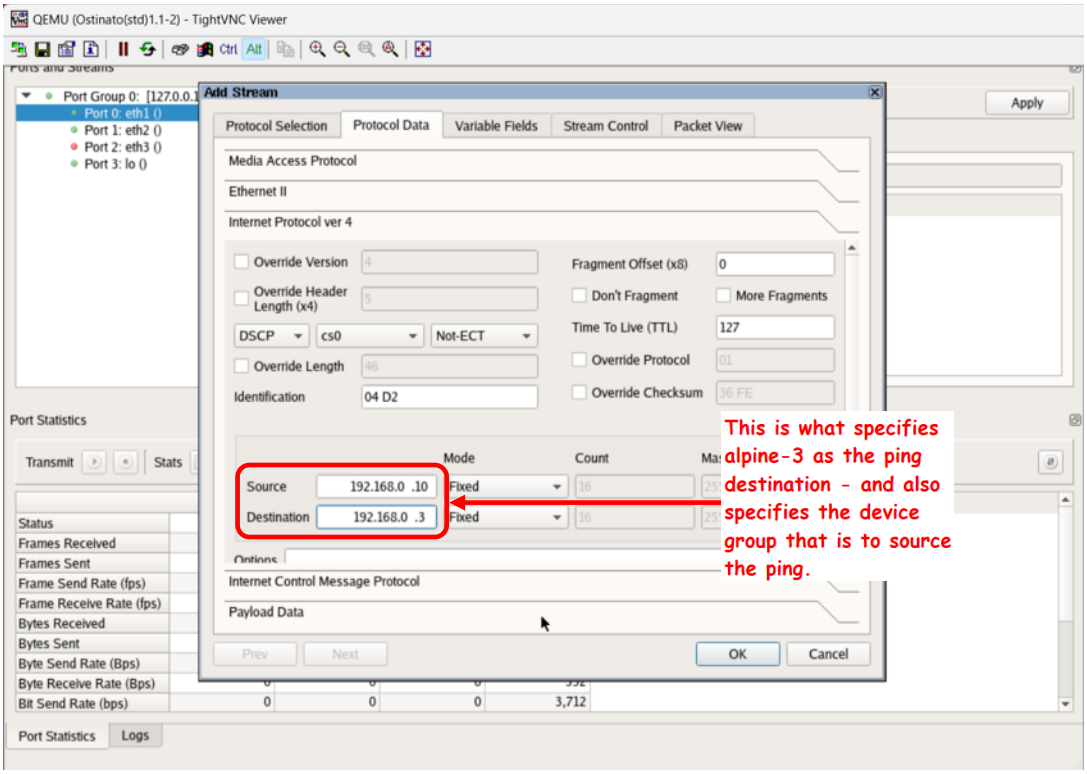

Fig. 30(c): Packet stream creation - specify protocol data - IPv4

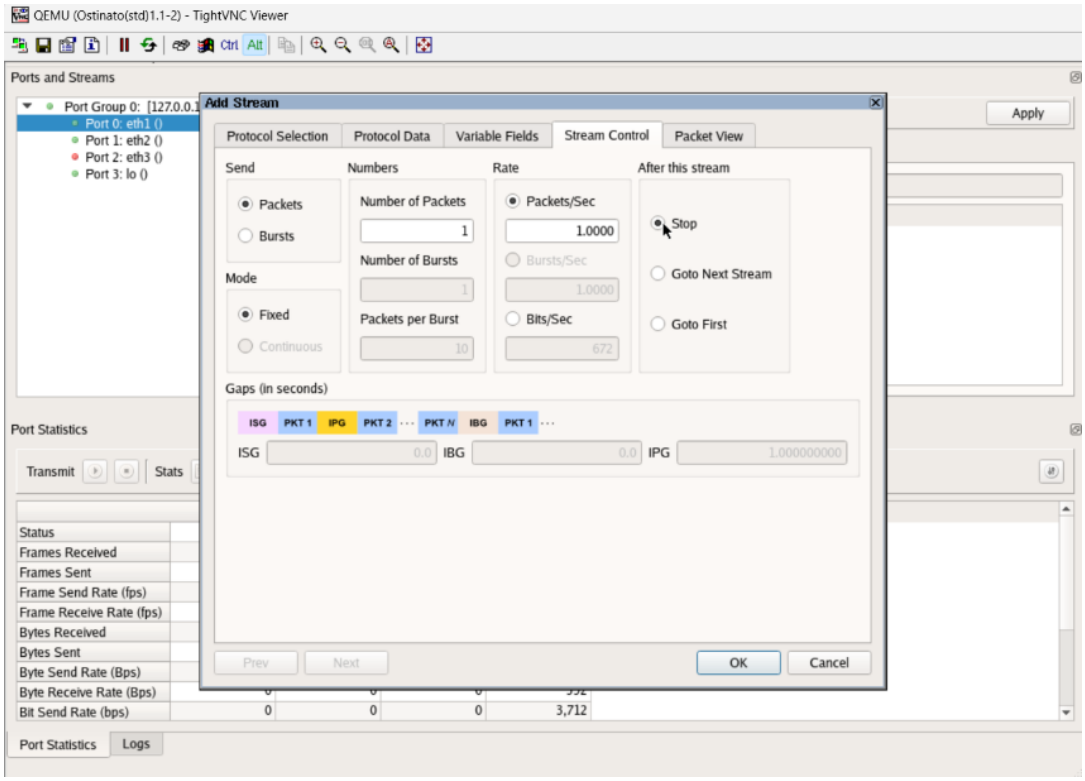

Fig. 30(d): Packet stream creation - specify stream control

5. Note Wireshark's UI before running the packet stream. With the display filter created in step 2, there are no packets captured as yet.

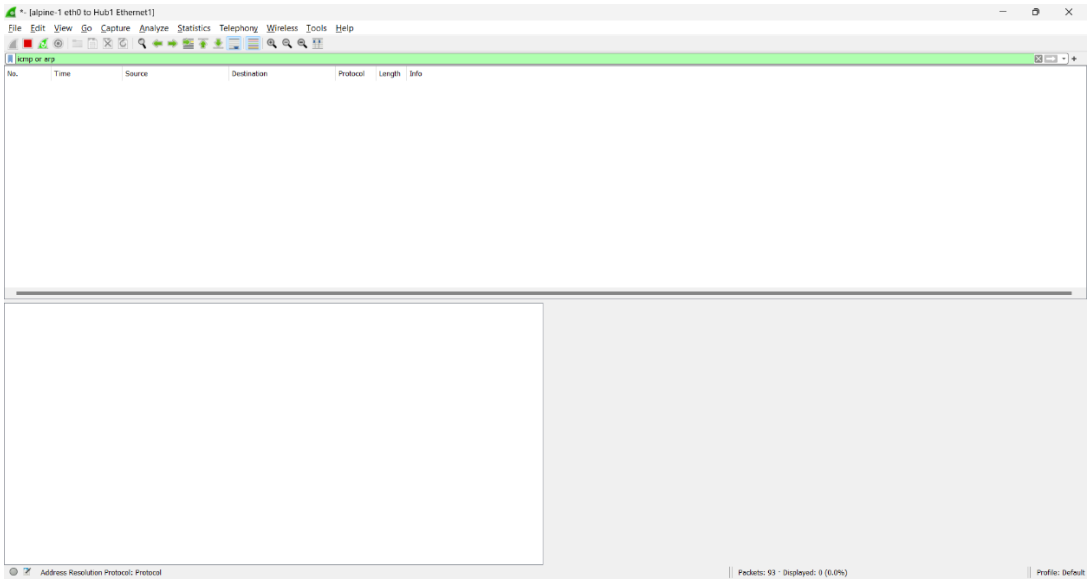

Fig. 31: No ARP or ICMP packets are visible before implementing the packet stream

6. Now, proceed to implement the packet stream (click on Apply).

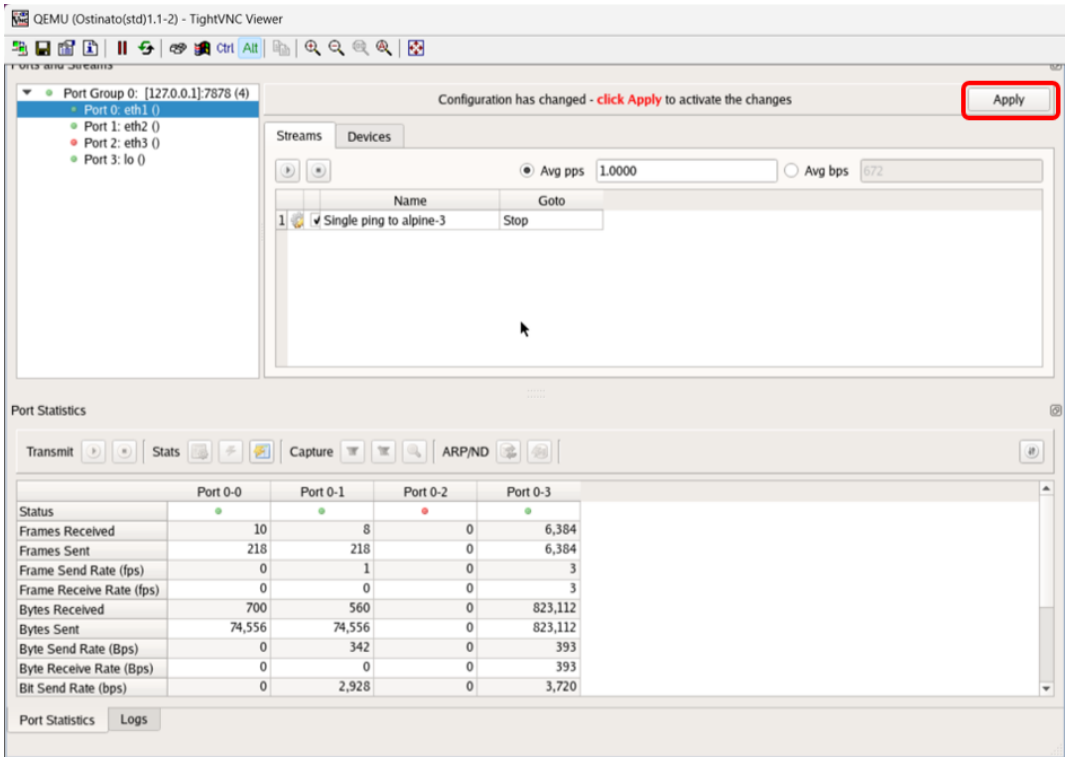

Fig. 32: Click on "Apply" to implement the packet stream and trigger ARP

7. Observe the packet list pane in the Wireshark UI: note that Ostinato has issued an ARP-request broadcast and that alpine-3 has returned with an ARP-reply unicast. Note that both the ***unicast*** packet (frame 83) is obtained on the physical cable connecting ***alpine-1 to hub-1***, as well as the ***broadcast*** packet (frame 82).

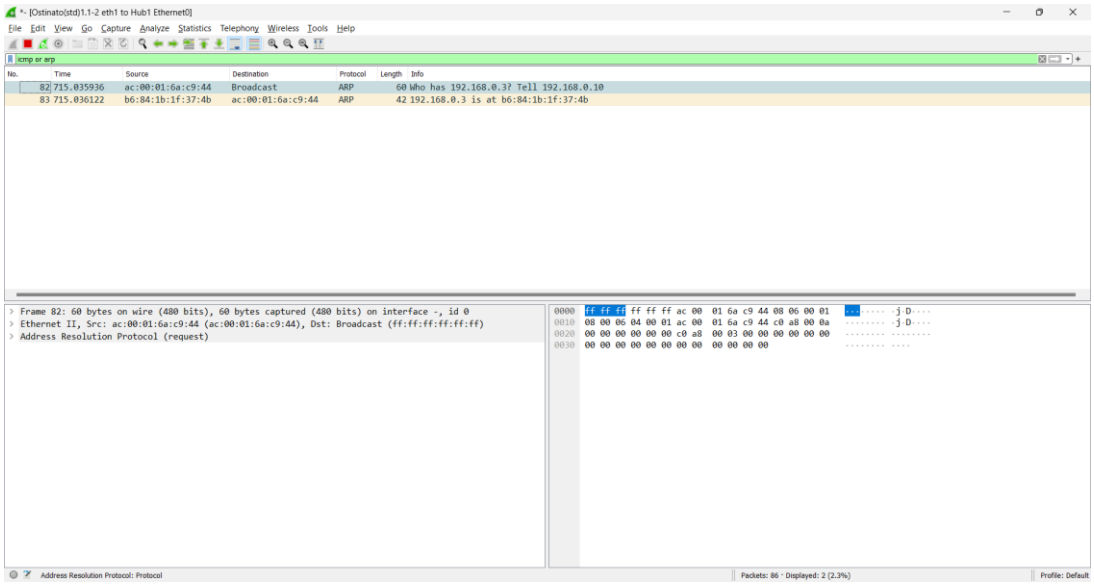

Fig. 33: Note that the unicast is sniffed too, although it does not pertain to alpine-1

8. Now, run the packet stream and inspect Wireshark's capture.

QUESTION 5

For each packet, state source and destination IPv4 address. Compare these latter two addresses with the IPv4 address bound to alpine-1 eth0, and justify the outcome of your comparison.

5 marks

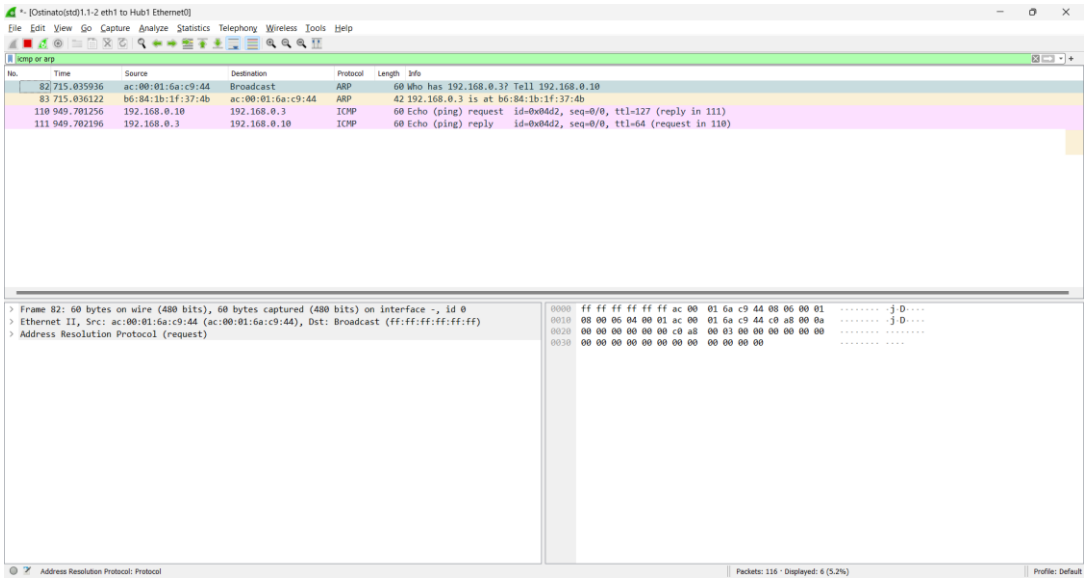

Fig. 34: Wireshark's capture, post-execution of packet stream

2.2.3 Procedure: phase 2

In phase 2, the experiment with dynamic ARP will be repeated, but the scope now changes to the devices connected to the switch. That is: the same procedure will be carried out but packet exchange will be observed on a switched link. A detailed sequence is not given here; the sequence used during phase 1's work should be repeated and adapted where necessary.

Topology deployment and IP address configuration

The sender is alpine-2 and the receiver is alpine-4; see Fig. 35, which focuses upon topology in scope.

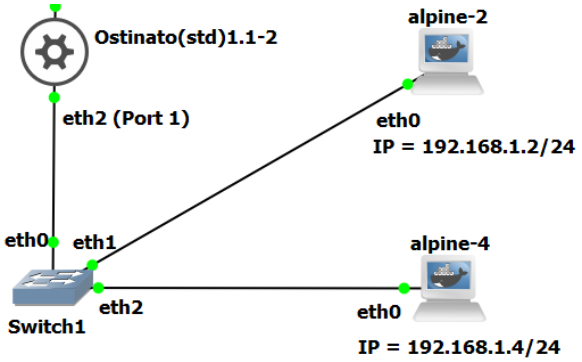

Fig. 35: Subset of the topology that is in scope in phase 2

In order to better observe the difference between a hub's (phase 1) and a switch's behaviour (phase 2), you should capture (concurrently) on both links: Switch1 ↔ alpine-2 AND Switch1 ↔ alpine-4.

2.2.4 Verification of results obtained in phase 2

After ***implementing*** the ICMP Echo request stream on the link between Switch1 and alpine-4, the output should be similar to that shown in Fig. 36.

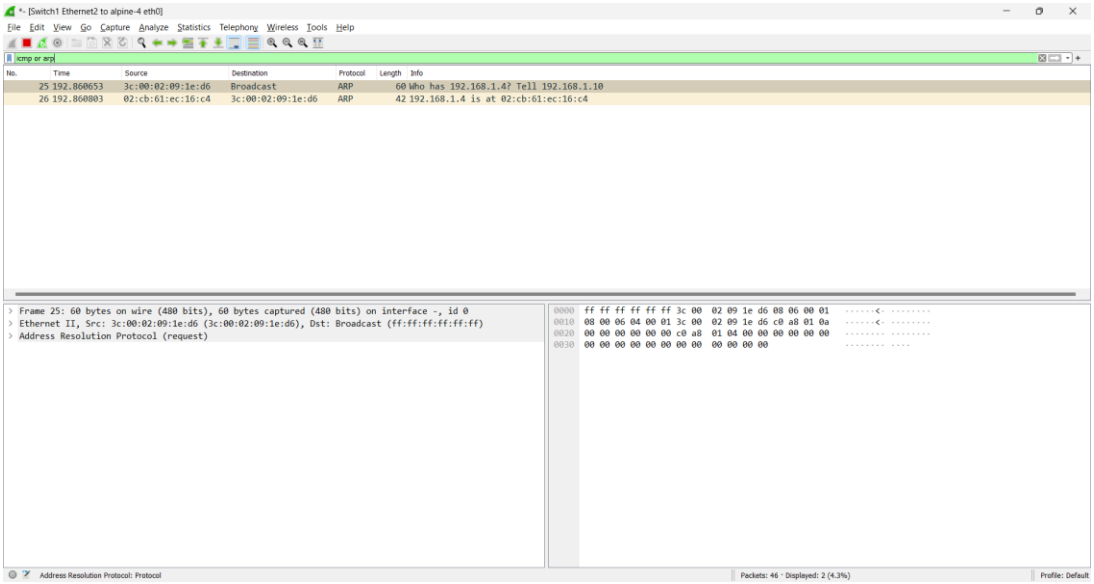

Fig. 36: Packets captured on Switch1 ↔ alpine-4 after implementing the packet stream

If you were to concurrently capture packets on the link between the switch and alpine-2, you should obtain output similar to that shown in Fig. 37. Note that the broadcast is captured, but the unicast is not. This is because the switch has now learnt that Ostinato is not on the switch port linked to alpine-2.

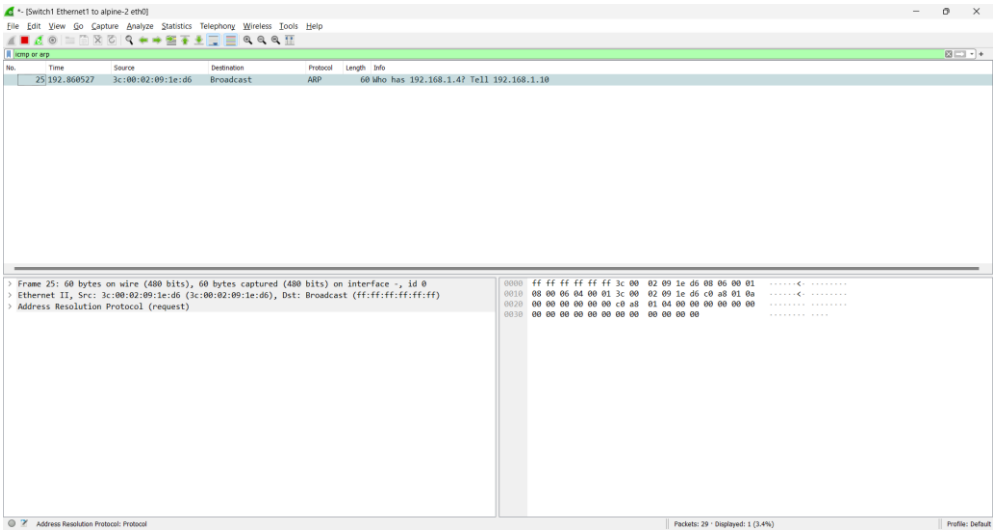

Fig. 37: Packets captured on Switch1 ↔ alpine-2 after implementing the packet stream

QUESTION 6

Compare the packet(s) you capture on Switch1 ↔ alpine-2. State at least one difference between your output and that shown in Fig. 37, and explain it.

5 marks

After running the ICMP echo request stream, the output should be similar to that shown in Fig. 38. Note that, here, alpine-4 autonomously inserts an ARP request after the ICMP echo request – reply pair. This was not controlled by the experiment, but by alpine-4's operating system.

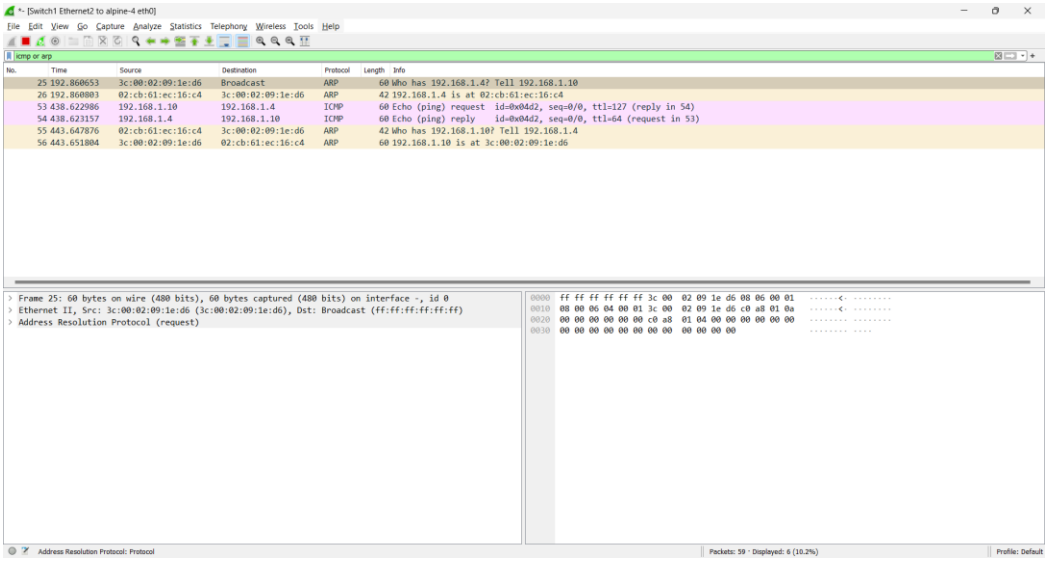

Fig. 38: Packets captured on Switch1 ↔ alpine-4 after running the packet stream

QUESTION 7

7.1 Present a screenshot that shows the packets you have captured on Switch1 ↔ alpine-4.

7.2 For each packet, explain how it fits into the sequence that is generated as a result of running the packet stream you have configured on Ostinato.

7.3 Select the ICMP packet generated by Ostinato. Use a tabular structure, exemplified by Fig. 39, to name **all** the fields, for all the layers, in the ICMP packet. For each field name, write a prefix to indicate which layer the field pertains to, e.g., L1 for layer 1, L2 for layer 2, etc.

10 marks

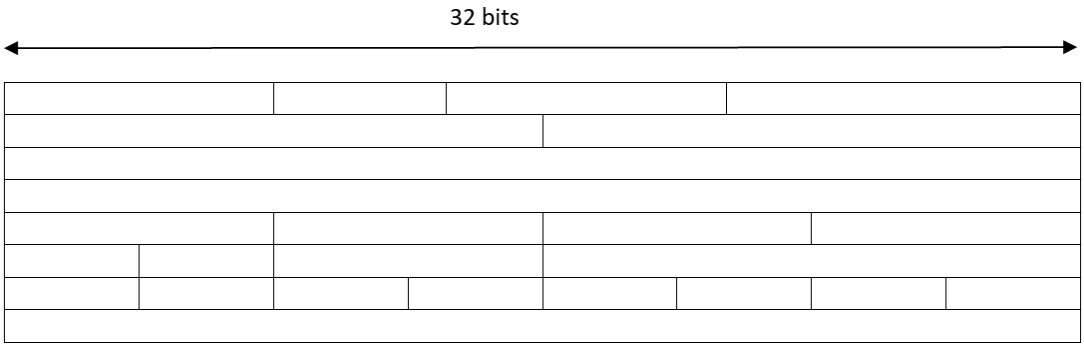

Fig. 39: Tabular presentation of the fields of the packet

If you were to concurrently capture packets on the link between the switch and alpine-2, the output should be similar to that shown in Fig. 40. The captured output is unchanged from the previously captured output on the same link.

QUESTION 8

Explain why the packets captured on Switch1 ↔ alpine-2 do not change between the point in time just before running the packet stream on Ostinato, and the point in time just after running it.

5 marks

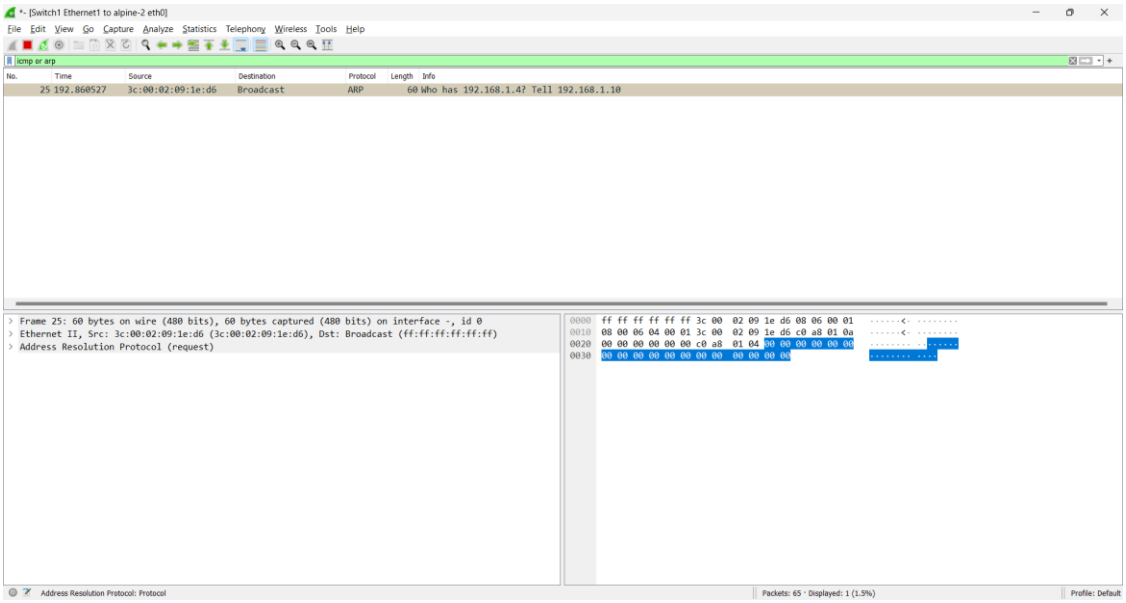

Fig. 40: Packets captured on Switch1 ↔ alpine-2 after running the packet stream

2.2.4 Procedure: phase 3

In this phase, the ***reliability*** of (layer 4) transport protocols (Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)) is investigated. In the proceedings, the ***dynamics of transmission*** are investigated.

2.2.4.1 Phase 3, part 1

In this phase, a TCP connection will be:

1. established,
2. used to transfer data and
3. torn down.

The sender is alpine-2 and the receiver is alpine-4 (see Fig. 41). The behaviour of TCP's automated congestion control is observed.

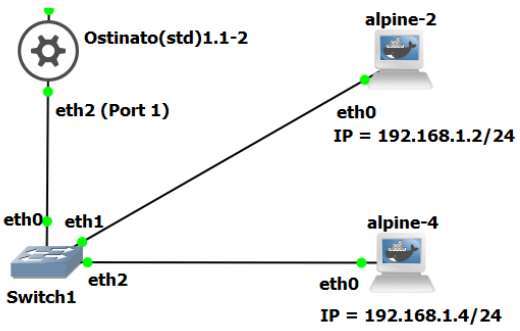

Fig. 41: Subset of the topology that is in scope in phase 3

A large file will be transferred between sender and receiver, using TCP. Therefore, the following items are necessary:

(a) a large file: this can be created using a combination of basic Linux commands;
(b) a tool to send and receive the large file: the Linux utility **netcat** can be used to establish a TCP connection, transfer data and tear it down.

While the data transfer is taking place, data will be captured and its behaviour will be monitored.

1. **Large file creation, on alpine-2**
   a. On the sender (alpine-2), type the following at the command line to create a text file, named large_file, that is exactly 100MiB large.
      `yes This is plain text | head -c 104857600 > large_file`
   b. Verify that the large file has been created using the command:
      `ls -al ./large_file`
   c. alpine-2 should respond as follows, confirming that the file has indeed been created:
      `-rw-r--r--    1 root     root      10485760 Oct 21 15:47 ./large_file`

2. **Set up a listener, on alpine-4**
   At alpine-4, type the following at the command line to prepare netcat to listen on TCP port 4444 and redirect what it receives to a file named `received_file`.
   `nc -l -p 4444 > received_file`
   Note that the command prompt (the # symbol) will not return, as netcat is occupying the console.

3. **Capture sequence (lossless)**
   a. Prepare alpine-2 for transmission (step b, below), then proceed to steps c and d, and thereafter immediately switch to step e.
   b. **(Do not hit enter after typing the following command)** On alpine-2, type the following at the command line to transfer `large_file` to alpine-4:
      `nc 192.168.1.4 4444 < large_file`
   c. On the GNS3 canvas, start capture, using Wireshark on the link between alpine-2 and switch-2.
   d. Apply the display filter tcp.port==4444
   e. Quickly, switch back to alpine-2's console and hit <Enter>.
   f. Switch over to Wireshark's UI and look for the segment wherein alpine-2 signals, through the FIN flag that it has finished sending data (Fig. 42).

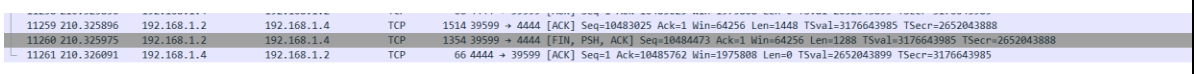

Fig. 42: The TCP segment carrying a set FIN flag is the one that signals the sender's closure

   g. Stop capture (from within GNS3) as soon as the signal is detected (note that this is what is known as a half-close – only alpine-2 closes the connection).
   h. Select Statistics menu → TCP Stream Graphs → Throughput (Fig. 43)

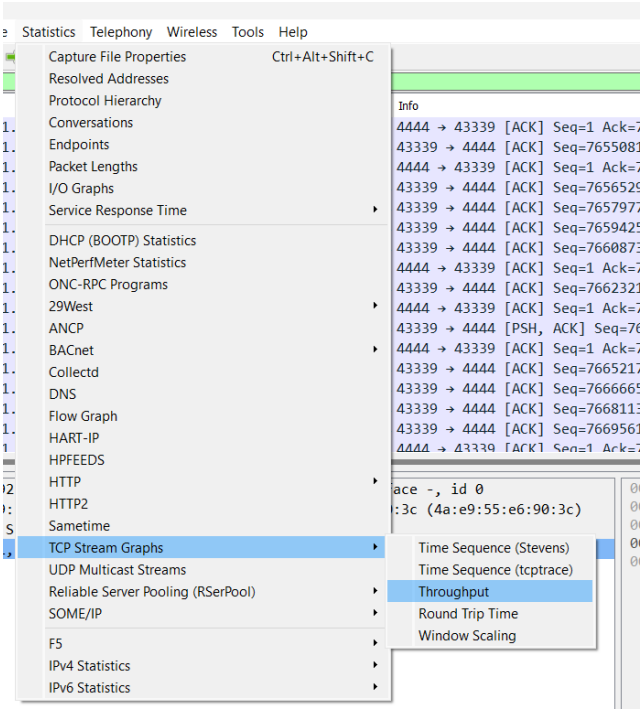

Fig. 43: Menu sequence leading to graphing of TCP throughput

i. In the graph's window, ensure that you are viewing the 1-second moving average throughput only, and viewing it in the direction from alpine-2 to alpine-4.

j. At alpine-4, inspect the file named received_file, using the command:
`ls -al ./received_file`.
Inspect the size of the file.

k. In the throughput graph's window, set the duration of the moving average (MA) window to 1 second and clear the "segment length" check box (see Fig. 44).

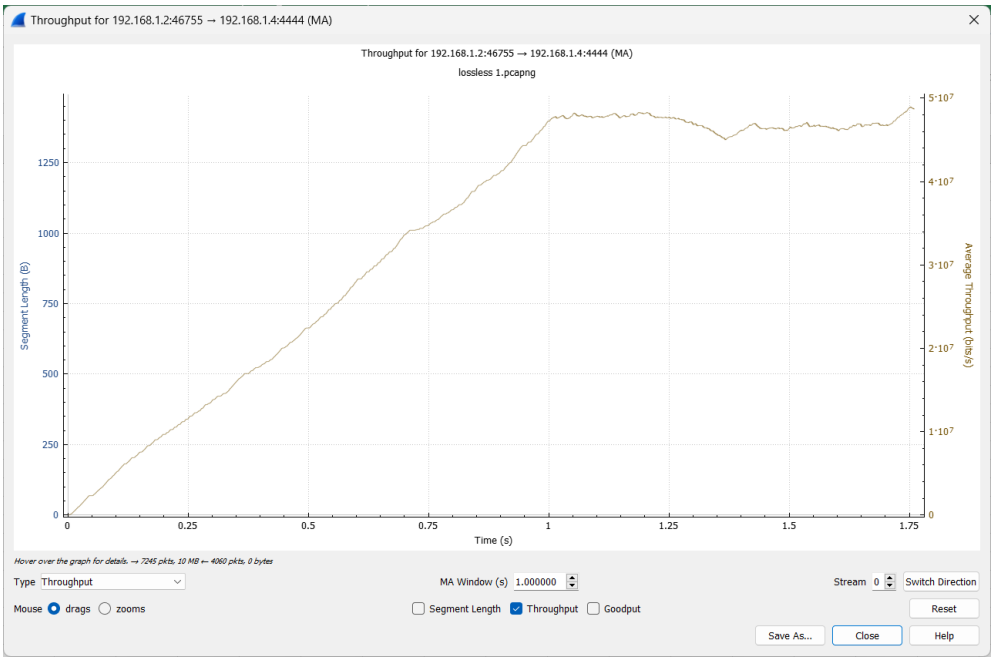

Fig. 44: TCP stream throughput, with MA = 1s

4. **Capture sequence: 1% packet loss, 3% packet loss, 5% packet loss**
In this sub-phase of phase 3 of the experiment, your objective is to compare:
   - the profile of the 1-second moving average of the throughput, obtained with lossless paths, with
   - the profile of the 1-second moving average of the throughput, obtained with lossy (1%, 3% and 5%) paths.

   a. Bring up the context menu on the link between alpine-2 and switch-1 and select "Packet filters" from the context menu.

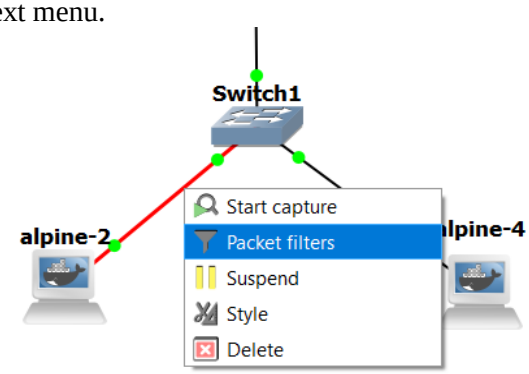

Fig. 45: First step in the configuration of a packet filter

   b. Navigate to the "packet loss" tab, set a loss probability of 1% and click OK (Fig. 46)

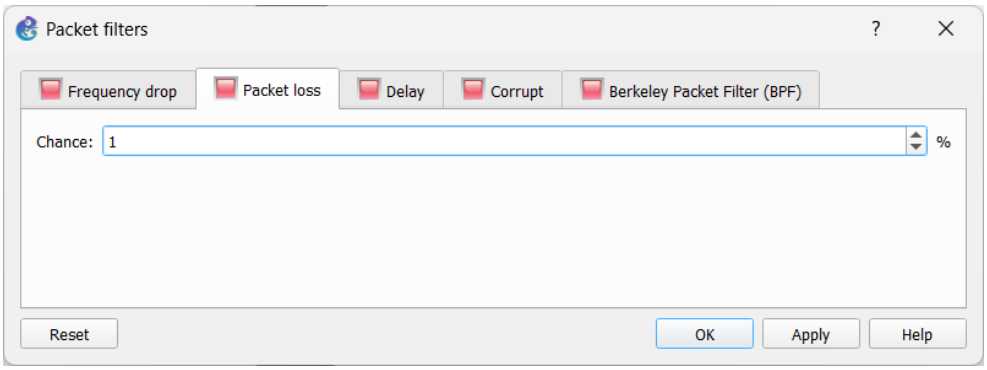

Fig. 46: Second step in the configuration of a packet filter

   c. The link will now show an icon, representing the imposed filter (Fig. 47).

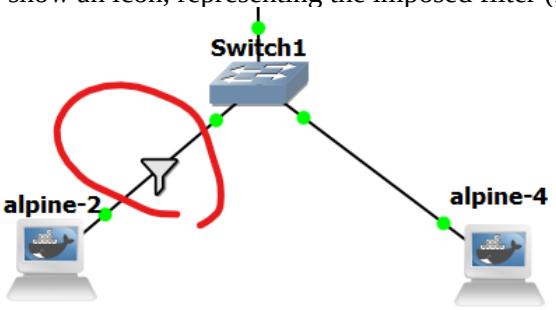

Fig. 47: The funnel icon represents the filter configured on the link

d. Stop netcat (nc) at the receiver, and restart its listener. Repeat the capture sequence procedure. This will result in a fresh file capture, which you can then inspect for file size.
e. Compare the 1-s moving average throughput with that obtained for the lossless path.

5. **Repeat, with 3% and 5% packet loss probability.**

QUESTION 9

This concerns the dynamics of transmission using TCP. For each of the four cases (lossless, and packet loss at the rate of 1%, 3% and 5% respectively):

9.1 Plot a 1-s MA of the throughput, and

9.2 calculate the average throughput

5 marks

QUESTION 10

Consider the packet sequence pertaining to the lossless case.

10.1 List the flags (within square brackets) pertaining to the first three packets.

10.2 What part of connection establishment do the first three packets pertain to?

10.3 List the sequence (Seq=) and acknowledgement (Ack=) numbers pertaining to the first three packets.

10.4 Identify the maximum segment size which the two parties in the connection state.

10.5 Identify the range of packets involved in the data transfer phase.

10.6 Identify the packet(s) involved in the connection teardown phase.

5 marks

Show screenshots that allow a reader to validate your answers.

QUESTION 11

Consider the packet sequence pertaining to the 5% packet loss case.

List `received_file` on alpine-4 and compare its size with that of `large_file`.

Show screenshots that allow a reader to validate your answers.

5 marks

2.2.4.2 Phase 3, part 2

UDP is connectionless; this means that it does not implement the function of "end-to-end sequence control on individual connections" (see "Networking Foundations" slides). The capture sequence will be repeated, using UDP as the transport protocol. As with the TCP capture sequence, it is necessary to prepare alpine-2 for transmission (step 1, below), then proceed to steps 2 and 3 and thereafter immediately switch to step 4.

1. **Capture sequence, UDP, lossless**
    a. On alpine-4, set up a UDP listener (the server), by typing the following command:
    b. `nc -l -u -p 4444 > received_file`
    c. Do not hit enter after typing the following command: on alpine-2, type the following at the command line to transfer large_file to alpine-4 over UDP:
    d. `nc -u 192.168.1.4 4444 < large_file`
    e. On the GNS3 canvas, start capture, using Wireshark on the link between alpine-2 and switch-2.
    f. Apply the display filter udp.port==4444
    g. Switch back to alpine-2's console and hit <Enter>.

h. Switch over to Wireshark's UI. This time, there is no FIN flag to indicate end of transmission, because UDP is connectionless. Your only means of detecting the end of transmission is an end to changes in the packet list pane.
i. Stop capture (from within GNS3).
j. Move to the top packet in the packet-list pane, highlight it and press CTRL-T. This latter CTRL key combination sets the time reference (t = 0) to the time of capture of the first packet (Fig. 48).

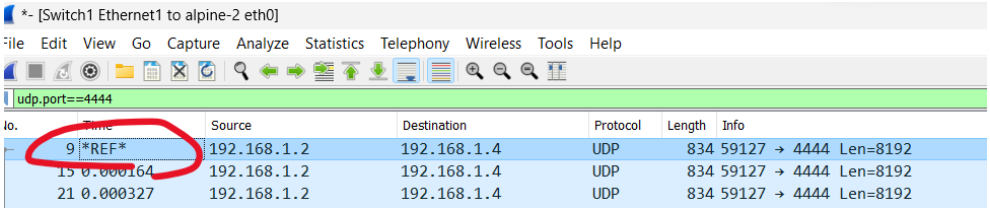

Fig. 48: Setting the time reference to t=0 facilitates inspection and graphing

k. Throughput can now be viewed using Statistics -> I/O Graphs (Fig. 49).

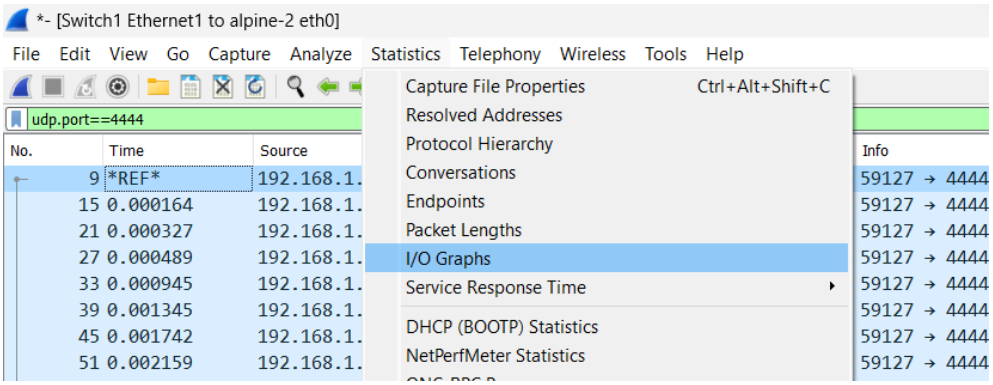

Fig. 49: Graphing data rate for UDP transport

l. In the graph window, set the time interval for capture to 10 ms (Fig. 50). This interval is the time over which Wireshark will aggregate packets. In the example shown in Fig. 50, in the first 10 ms, 200 kbits were captured. In the successive 10 ms, 230 kbits were captured, and in the third interval, 180 kbits were captured.

QUESTION 12

Consider the packet sequence pertaining to the lossless case.

12.1 List `received_file` on alpine-4 and compare its size with that of `large_file`.

12.2 Go to File->Export Packet Dissections, export the captured packets as CSV and inspect the CSV file in Excel. How does the number of octets sent (as you determine from the CSV file) compare with the size of `received_file`?

12.3 Inspect the first few packets in the UDP window and identify the length of the UDP datagram ("Len="). How does this compare with the Ethernet frame's MTU of 1500? Explain any difference you observe.

15 marks

Show screenshots that allow a reader to validate your answers.

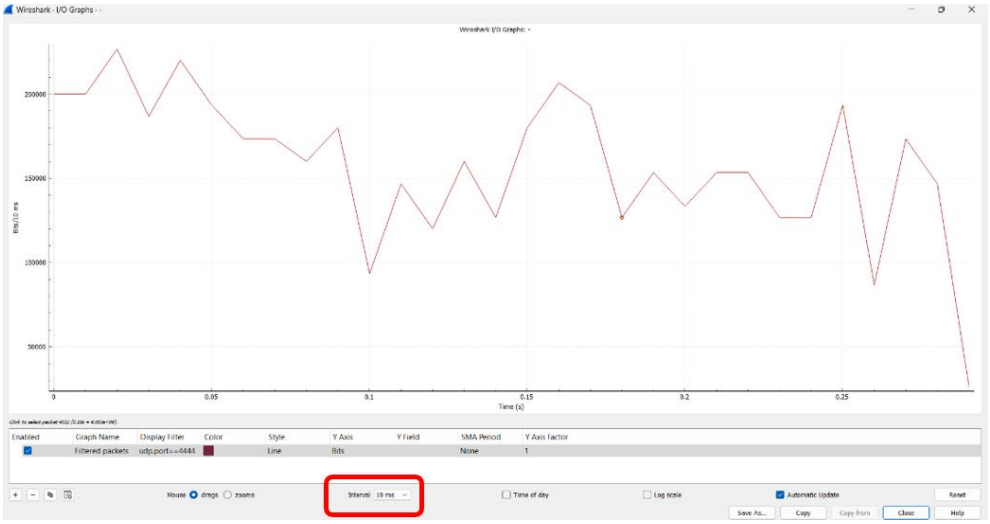

Fig. 50: Set the capture aggregation interval to 10 ms

QUESTION 13

Explain the difference between your observations in Q11 and Q12.1.

10 marks