LABORATORY WORK NO. 8

ETHERNET, ARP and NDP

1. Objectives

The objectives of this practical activity consist of understanding the structure of the Ethernet frame and the techniques used for discovering other devices within an Ethernet based network. Additionally, the simulation mode of the Cisco Packet Tracer tool is explored.

2. Theoretical considerations

This practical activity is concerned with Layer 2 operations performed on switches. Layer 2 refers to the Data Link Layer of the ISO/OSI model, which corresponds to the Network Access Layer in the TCP/IP model.

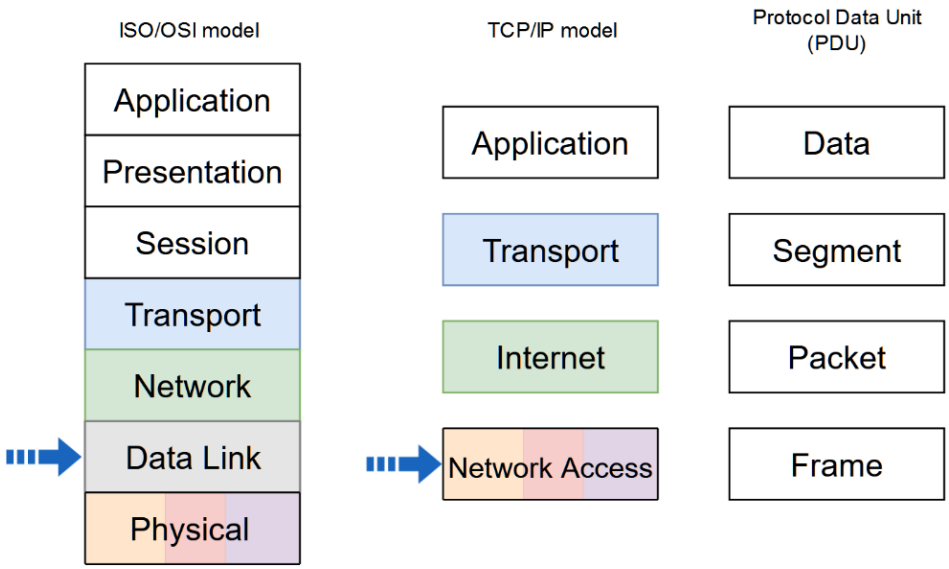

Figure 9.1 *Network stack models and PDU naming in each level. The arrows indicate the addressed layers in the current activity*

In order for a switch to forward a packet on a specific port, it maintains a switching table which contains a correspondence between a destination MAC address and the switch's port number. The MAC addresses used for communication are found in the Ethernet (Layer 2) frame header and a switch does not decapsulate the frame any further when manipulating the packet contents (Figure 9.2). This practical activity continues with providing more details regarding Layer 2 operations.

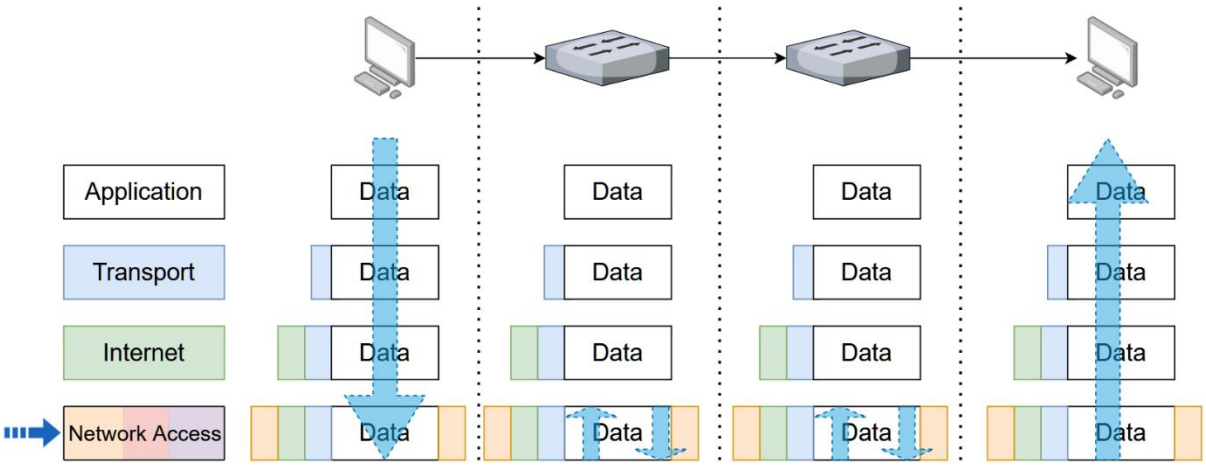

Figure 9.2 Switching operation, showing frame *serialization/deserialization in the Network Access layer*

2.1 Ethernet, Ethernet II and IEEE 802.3

Ethernet, Ethernet II and IEEE 802.3 are often used interchangeably as terms. Even though the terms tend to refer to very similar standards, they are slightly different, both historically and technically. In 1981 a consortium formed by Digital Equipment Corporation, Intel and Xerox (abbreviated as DIX) developed the Ethernet standard (also referred to as DIX 1.0 or Ethernet I). It was replaced with DIX 2.0, much more commonly known as Ethernet II in 1982. In 1983, IEEE introduced the 802.3 standard in an attempt to standardize the protocol beyond the DIX consortium. Nowadays, Ethernet II is generally the more popular approach, for reasons that will be described shortly.

There are two main differences between Ethernet II and IEEE 802.3. The first one is that Ethernet II uses a Type (also referred to as EtherType) field, which specifies the protocol encapsulated within the payload, whilst 802.3 uses that field for specifying payload length. The second difference is that, in order to run 802.3 within a TCP/IP stack, some additional information needs to be used (based on the SNAP and 802.2 format – beyond the scope of this practical activity) and, as such, taken from the Data field. This totals to 8 bytes which 802.3 uses from the Data field, reducing this field down to a range of 38 to 1492 bytes. Historically though, the length field has been considered not necessary and networks operate just as fine without it – this is the reason why Ethernet II is the more commonly used standard. All modern operating systems however work with both 802.3 and Ethernet II.

Note that when most engineers refer to Ethernet, they are generally referring to Ethernet II or, more rarely, to the IEEE 802.3 standard. This practical activity will use the term Ethernet and Ethernet II interchangeably, since Ethernet I is no longer used.

2.2 Ethernet II Frame Structure

Figure 9.3 presents the Ethernet II/IEEE 802.3 frame structure and the number of bytes allocated for each field. The following section describes the meaning of each field within an Ethernet frame.

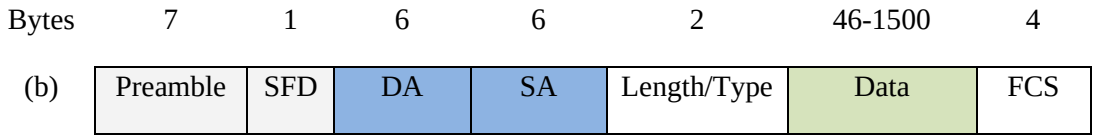

Figure 9.3 Ethernet II / IEEE 802.3 Frame Structure

Since Ethernet defines protocols for both the Physical and the Data Link Layer parts of a networking stack, some fields are handled by the Physical layer (Preamble and SFD), whilst some by the Data Link Layer (other fields).

The Preamble field is a 56-bit series of alternating '0' and '1' bits. They are used so that the devices involved in communication can synchronize their respective clocks and thus adjust the sampling rate accordingly for correct reception of the frame. The concept of using a preamble does not impose a fixed length but is rather adjusted on an individual protocol basis, even if Ethernet uses a fixed length of 56 bits. Using more bits allows more time for the communicating devices to synchronize but increases the overhead of communications, whilst reducing the preamble length has opposite effects.

The Start Frame Delimiter (SFD) field is a byte used to break the bit pattern in the Preamble and mark the start of the rest of the Ethernet frame. Specifically, it is "10101011" or 0xD5 (again, this is specifically in Ethernet; other protocols might use different SFD values). Please consider the fact that the bits are transmitted left to right and interpreted in LSB order.

The Destination Media Access Control (MAC) Address (DA) and Source MAC Address (SA) are identifiers which are uniquely assigned to the Network Interface Controller (NIC) of each device. Note that a device can have multiple NICs and, therefore, multiple corresponding MAC addresses. The role of the DA and SA will be discussed in more detail in the following sections.

The Type field is used to indicate the type of message encapsulated in the frame. Table 9.1 indicates some specific Type values.

Table 9.1 EtherType Examples

Hex Value	Protocol Type
0x0000-0x05DC	Length field for IEEE 802.3
0x0600	Xerox
0x0800	IPv4
0x0801	X.75

0x0806	ARP
0x86DD	IPv6

Note that due to the minimum requirement of 46 bytes used for data transmission if the length is any less than that value the Data Link Layer adds padding bytes to the Data field. The 46 byte value is based on the CSMA/CD mechanism (presented during the lecture) and beyond the scope of this activity. Alternatively, this field is considered to represent Length for 802.3, when its value is less than 0x05DC.

The Data field corresponds to the payload which is encapsulated within the frame. This is typically higher level protocol data.

The Frame Check Sequence (FCS) field is used for verifying integrity of the message. It is a four-byte Cyclic Redundancy Check (CRC). It is a numerical value which is computed based on all data within the frame, with the exception of the actual FCS (and, obviously, the Preamble and SFD). On reception this value is recalculated and compared with the original FCS. If the two values are different then the frame contains errors and is discarded.

2.3 Address Resolution Protocol

The Address Resolution Protocol (ARP) is a very important protocol in networking. As seen during the lectures and previous activities, addressing is handled separately by OSI (or TCP/IP) stack layers. The DLL handles MAC addressing (even though they are sometimes referred to as physical addresses and are dependent on the NIC, the Physical layer does not generally handle MAC addresses), the network layer handles IP addresses and the Transport layer handles port numbers. The Transport layer is not addressed in this practical activity. In a typical networking scenario, when a device intends to send a message to a destination, it already knows the destination IP address from a DNS server. However, in order to correctly assemble a frame, the DLL needs to know the MAC address, which is not handled by DNS servers and manual handling is extremely impractical. As such, ARP provides a simple mechanism to figure out the MAC address for a known IP address, a process known as *Address Resolution*.

Each device contains an internal data structure, known as an ARP cache, which stores the mappings between IP addresses and MAC addresses on a network. ARP is used to populate this cache. Figure 9.4 illustrates the contents of the cache through running the Windows *arp -a* console command.

```
C:\Users\admin>arp -a

Interface: 192.168.0.103 --- 0x12
  Internet Address      Physical Address      Type
  192.168.0.1           c4-6e-1f-37-70-61     dynamic
  192.168.0.101         9c-2e-a1-ed-55-ab     dynamic
  192.168.0.255         ff-ff-ff-ff-ff-ff     static
```

Figure 9.4 ARP Cache

In order to do so two ARP frames are generally needed: an *ARP Request* and an *ARP Reply* frame. Let us consider two devices on a network: device A intends to transmit a message to device B. The ARP algorithm is as follows:

1. Device A checks its ARP cache. If there is an entry with B's IP address it will jump to step 5

2. Device A broadcasts an ARP request containing the target IP. All devices receive this broadcast since A doesn't yet know the MAC address of B

3. If device B is on the network it will reply with an ARP response containing its own MAC address. All other devices will silently (i.e. without sending a message announcing this) discard the request

4. Device A will update its ARP cache

5. Device A will send the intended message as a unicast to B

ARP caches contain two types of entries: static and dynamic. Static entries are introduced by the user and are kept permanently in the cache, unless specifically removed. Dynamic entries are introduced by ARP and are periodically deleted. Each ARP entry can be deleted after periods of seconds, up to several hours, depending on the network, the device type, OS features and individual configurations. Deleting dynamic entries is an automatic process (but which can be initiated manually) and is useful because some ARP entries might not be needed anymore. Some examples of this situation are:

- A device changes its IP address (especially if using DHCP)
- A device is removed from the network so the entry might not be needed anymore
- A device has its NIC changed and, implicitly, its corresponding MAC address in the network

The reason for periodically removing entries in the cache is to remove any unnecessary or unused entries (especially since the cache is of fixed size, which might lead to certain strategies of cyber-attacks). Only ARP messages update ARP caches, so a device will update an entry either when receiving an ARP request, when receiving an ARP reply as part of the resolution process or when receiving an ARP broadcast (this final scenario only updates ARP entries, it does not add new ones – also used for avoiding overflowing the cache).

Static entries should generally be used only when a device is intended to remain in the network for a long time (e.g. a router). With the exception of some forms of cyber-attacks, ARP does not generate much overhead.

Other ARP use cases are: using Proxy ARP, which implies one device responding to an ARP request on behalf of another device and using a gratuitous ARP, which implies sending an ARP

broadcast so that other hosts can update their respective entries. These use cases are beyond the scope of this activity.

2.4 Neighbor Discovery Protocol

Neighbor Discovery Protocol (NDP or simply ND) is a protocol used with IPv6 which has multiple roles. It defines five ICMPv6 packet types, some of which have already been presented. These are: Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA) and Redirect packets.

NDP fulfills several roles, of which the current activity work only briefly presents MAC and IPv6 address resolution.

In IPv6 NS and NA messages are used to replace ARP, and are, to a certain extent, equivalent to the ARP Request and ARP Reply messages. Much like the ARP cache, IPv6 enabled devices use an IPv6 Neighbor Table or IPv6 Neighbor Discovery Cache. There are, however, certain optimizations with NDP.

One significant optimization is brought on by the use of multicast addresses: instead of sending a broadcast ARP request, NDP implies sending an NS to the target device's multicast address, which reduces network overhead.

Another optimization is brought on by using five states which describe an IPV6 ND Cache entry:

1. Incomplete (NS has been sent and NA not yet received)

2. Reachable (NS has been sent and NA received or ND entry successfully used by upper layer protocol)

3. Stale (Timeout interval elapsed)

4. Delay (Timeout interval elapsed but recent packet sent to target, state moving to Probe after sending an NS)

5. Probe (NS has been sent from delay, waiting for NA)

IPv6 ND builds upon ARP but has multiple functions and is much more complex than simply resolving MAC addresses to IP addresses.

3. Practical activity

In the following activity you will use Wireshark in order to analyze the ARP protocol. Using the local ARP cache on the device requires administrator privileges. The current instructions are for Windows based systems. On Unix based systems, the **sudo** command might be required to manipulate the local ARP cache. The activity has two parts:

1. Working on the local device
    a. Clearing the local ARP cache
    b. Examining an ARP request
    c. Examining an ARP reply

2. Working in Packet Tracer
    a. ARP Simulation Mode
    b. NDP Simulation Mode

3.1 Working on the local device

Step 1: First you will need to open a command line or PowerShell with Administrator privileges. To do this, right click the appropriate program and select Run as Administrator. Enter the password when prompted. Use the **ipconfig /all** command and write down your IPv4 address, your appropriate MAC address and the default gateway's IPv4 address.

Step 2: Open a Wireshark capture on the appropriate interface. In order to clear the ARP cache you need to use the **arp -d** command. To view the ARP cache the *arp -a* command is used. Since the ARP cache is continuously updated, in order to make sure that it is cleared you can combine the two instructions using the & character, as follows: **arp -d** & **arp -a** (optional, if arp -d does not work, use one of the following commands: **arp -ad** or **netsh interface ip delete arpcache**). To make sure that the MAC address of the default gateway is reintroduced in the cache ping the default gateway's IPv4 address. Stop the Wireshark capture. You can use the **arp -a** command again to check that the cache now contains the default gateway's corresponding entry.

Step 3: Use the *arp* filter in Wireshark in order to view only ARP frames. Select the first broadcast message. This is an ARP request message. Notice that the type is Ethernet II (unless specifically using a different protocol). Expand the appropriate Ethernet II tab in Wireshark. Check that the message originated from your device either using the source MAC address or the source IP address (there might be other request messages on the network). If the first request isn't yours, go ahead and navigate until you find your own.

Step 4: Now that you've identified the correct ARP request message go ahead and analyze it (Figure 9.5). Notice that the DA field is *FF:FF:FF:FF:FF:FF*. This is a broadcast address. Observe that the type field is *0x0806* which correctly indicates an ARP frame. Note that the addresses in your own case will be different.

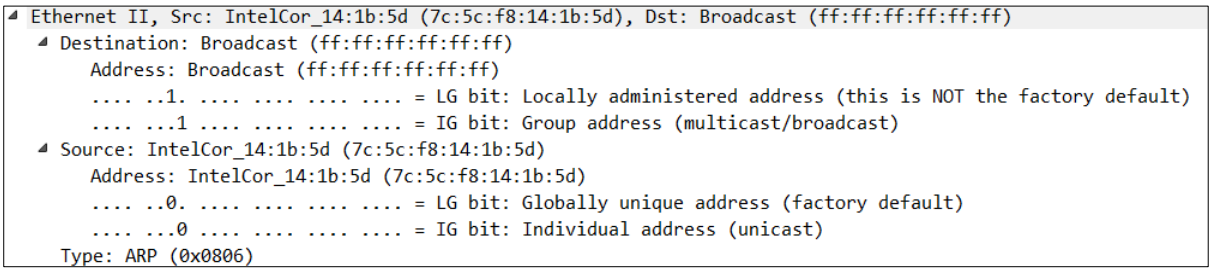

Figure 9.5 Wireshark capture of a detailed ARP Request frame

Step 5: Let us investigate the actual ARP contents. Expand the appropriate selection, as seen in Figure 9.6.

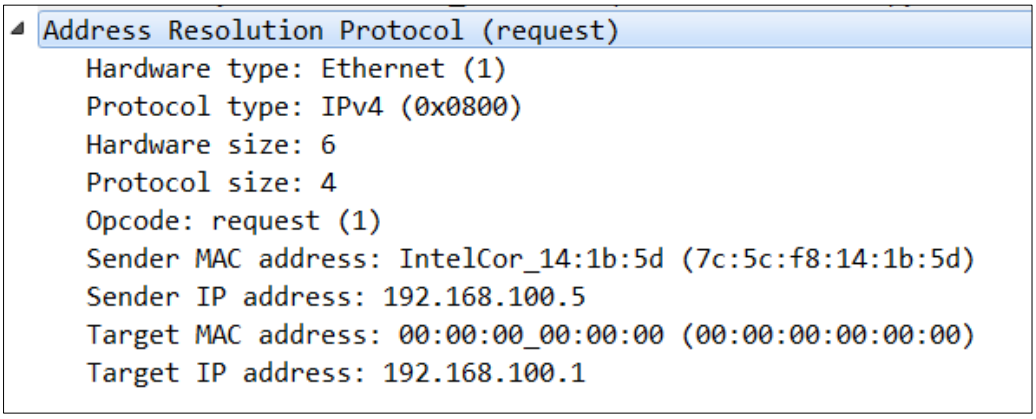

Figure 9.6 Wireshark capture of an ARP Request frame

Let us analyze each field and understand their respective purpose and meaning. Hardware type and Protocol type refer to what types of addresses are being mapped to one another. In this case a MAC address is mapped to a known IPv4 address (remember that this is the purpose of ARP). The following two fields refer to the size of each address: a MAC address is 6 bytes long whilst an IPv4 address is 4 bytes long. The Opcode is, in the case of ARP, one of two options: "1" represents a request and "2" represents a response. The Sender MAC and IP addresses are obviously your own (including the sender MAC in the request ensures that the reply can be sent as unicast to the requester). What is noteworthy though is that the protocol includes the Target MAC and Target IP addresses. A **very important** distinction is the use of *Target* instead of *Destination*. Even though the destination is a broadcast, as previously seen, the target represents the device whose MAC address is being resolved. Hence the distinction between destination and target. The IP address is clearly the default gateway and because the target MAC address has yet to be resolved this field is left unpopulated.

Step 6: Let us have a look at the corresponding ARP reply (Figure 9.7). First you need to find the correct reply – it should be from the default gateway to your device. Depending on how long the capture was running for there might be multiple requests and replies – this is due to the ARP cache refresh rate.

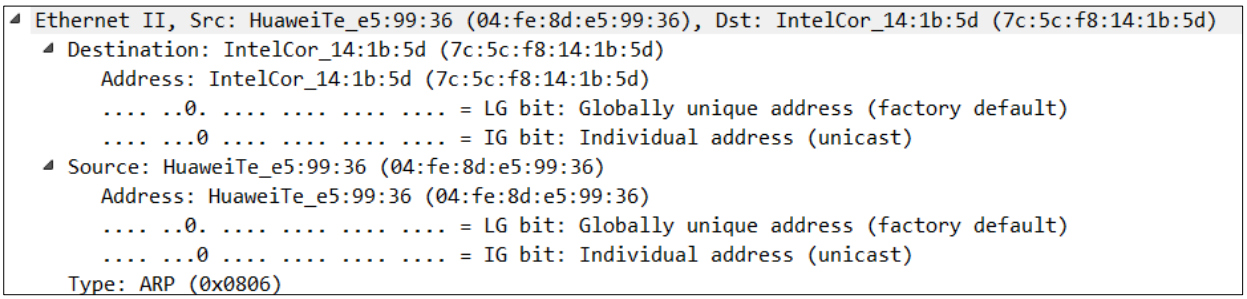

Figure 9.7 Wireshark capture of an ARP Reply frame

It can be seen that this is a unicast message from a MAC on the network (check that it's the default gateway based on what you previously noted).

Step 7: Let us investigate the contents of the actual protocol (Figure 9.8).

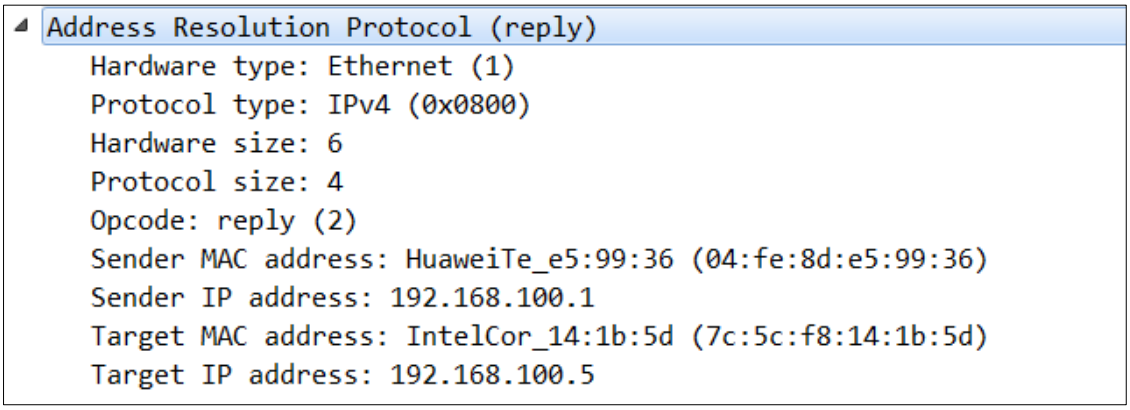

Figure 9.8 Wireshark capture of an ARP Reply frame

Notice the Opcode is changed and that the Sender MAC address is now visible (during the request this corresponded to the unknown Target MAC address). In conclusion, our own device receives this reply from the default gateway and can thus populate its ARP cache with the appropriate MAC address. Communications can continue now without exchanging any more ARP messages, except if the entry is deleted after a timeout.

3.2 Working in Packet Tracer

a. ARP Simulation Mode

This part of the practical activity uses the Simulation mode of the Cisco Packet Tracer tool to verify how network packets travel inside a network. This will also clarify why the first echo

request of a **ping** command can sometimes be an unsuccessful timeout (as probably noticed in previous activities).

Step 1: Launch Packet Tracer, create a network topology containing only switches and endpoint devices and navigate to the Simulation mode, as indicated by the arrow in Figure 9.9.

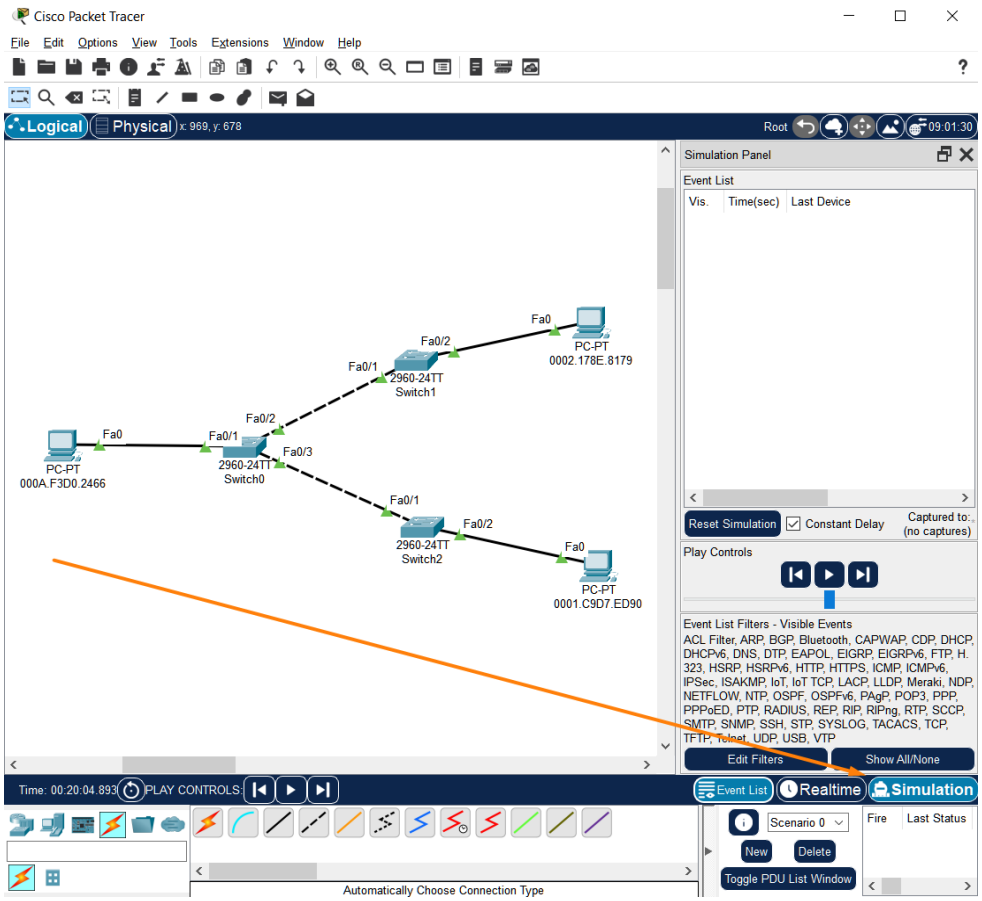

Figure 9.9 PacketTracer Simulation Mode

Step 2: In the Simulation window, click on the Show All/None button to clear all filters and then click on the Edit Filters button and select only ARP and ICMP (Figure 9.10).

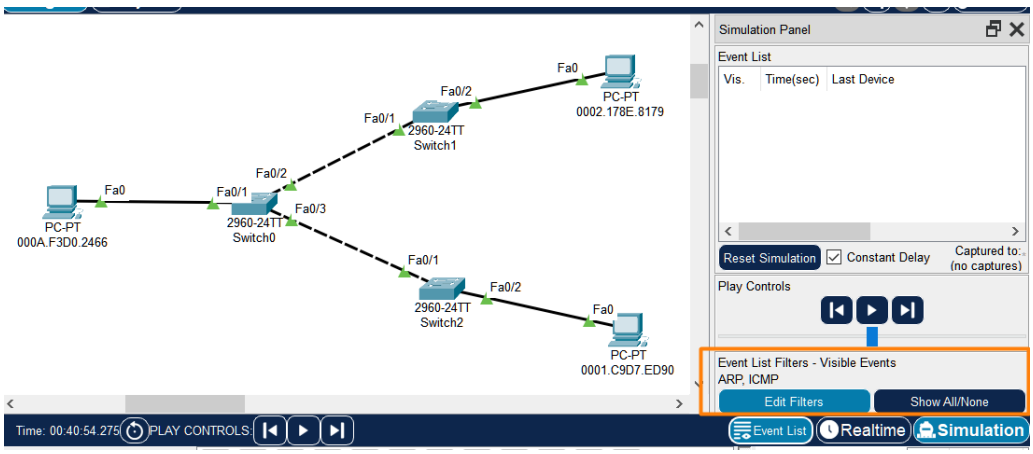

Figure 9.10 PacketTracer Simulation Filters

Step 2: Rename the PC names with their own MAC addresses; they can be found in the PC menu -> Config tab -> FastEthernet0 interface .

Step 3: Assign each PC an IP address from the same network (e.g. 10.0.0.1, 10.0.0.2 and 10.0.0.3, all of them /8 – feel free to use a different network/subnet).

Step 4: Open the command prompt on one of the PCs and verify that its arp cache is empty. If it is not empty run the **arp -d** command to clear it.

Step 5: Ping another PC's IP address and inspect the simulation. At this point the ARP cache of the PC is empty so it cannot populate the entire packet (specifically it cannot populate the Ethernet frame DA field), therefore it launches an ARP broadcast request in the network – recall that this is the first stage of the ARP protocol. The step by step traffic analysis (Figure 9.11, Figure 9.12 and Figure 9.13) shows the path that the request takes; note that only the target device replies to the broadcasted request and all other devices silently drop the packet.

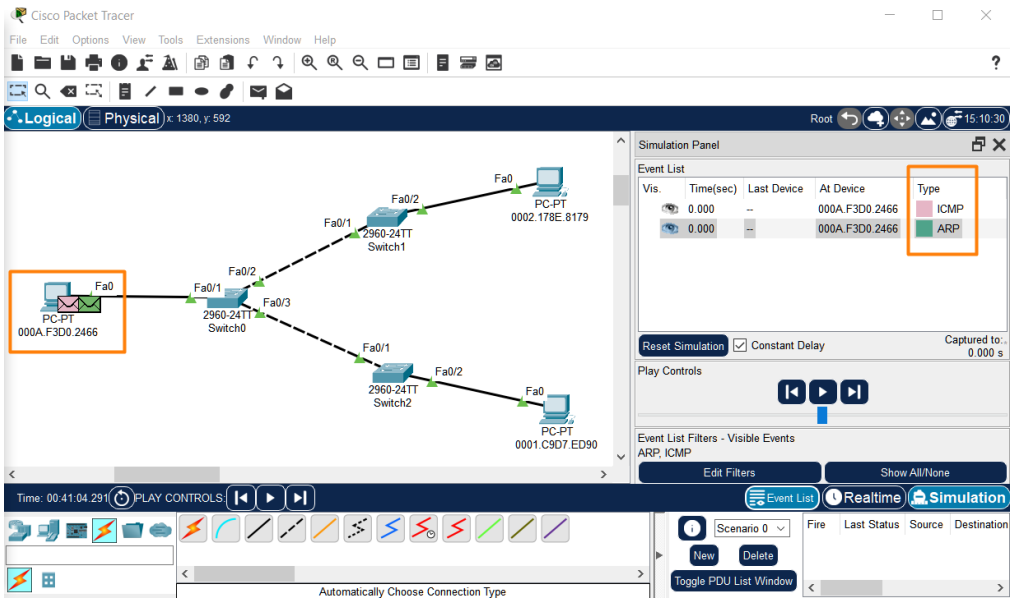

Figure 9.11 ARP request

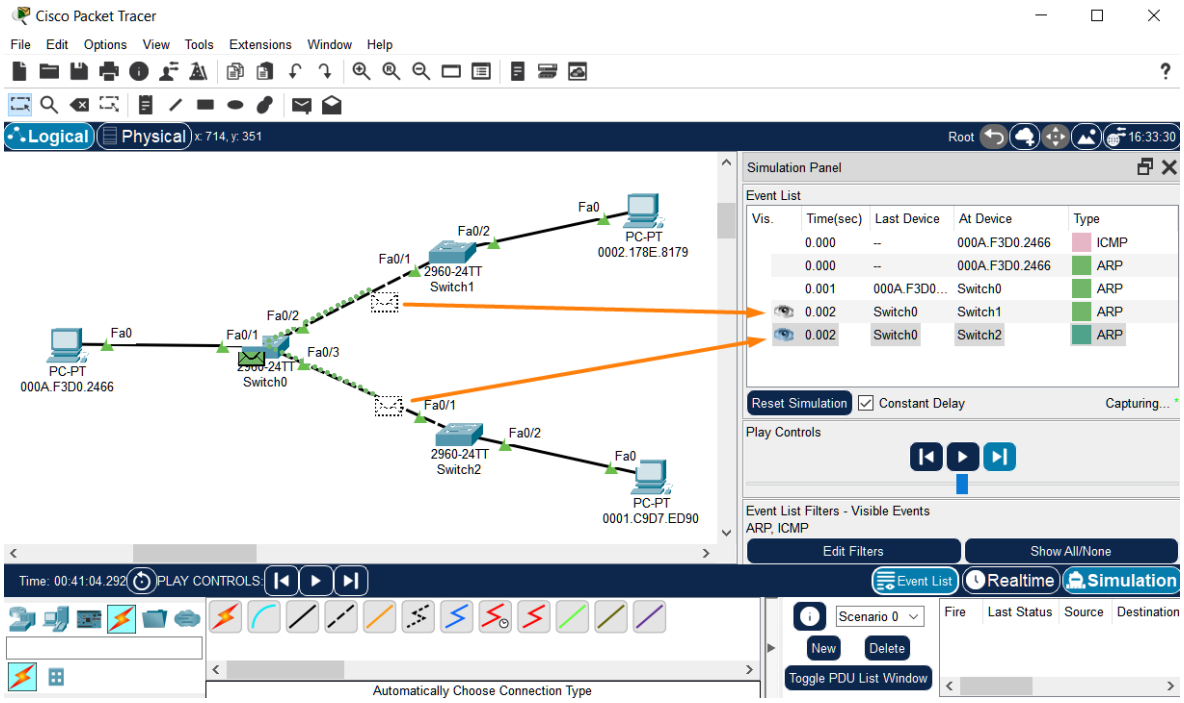

Figure 9.12 ARP broadcast

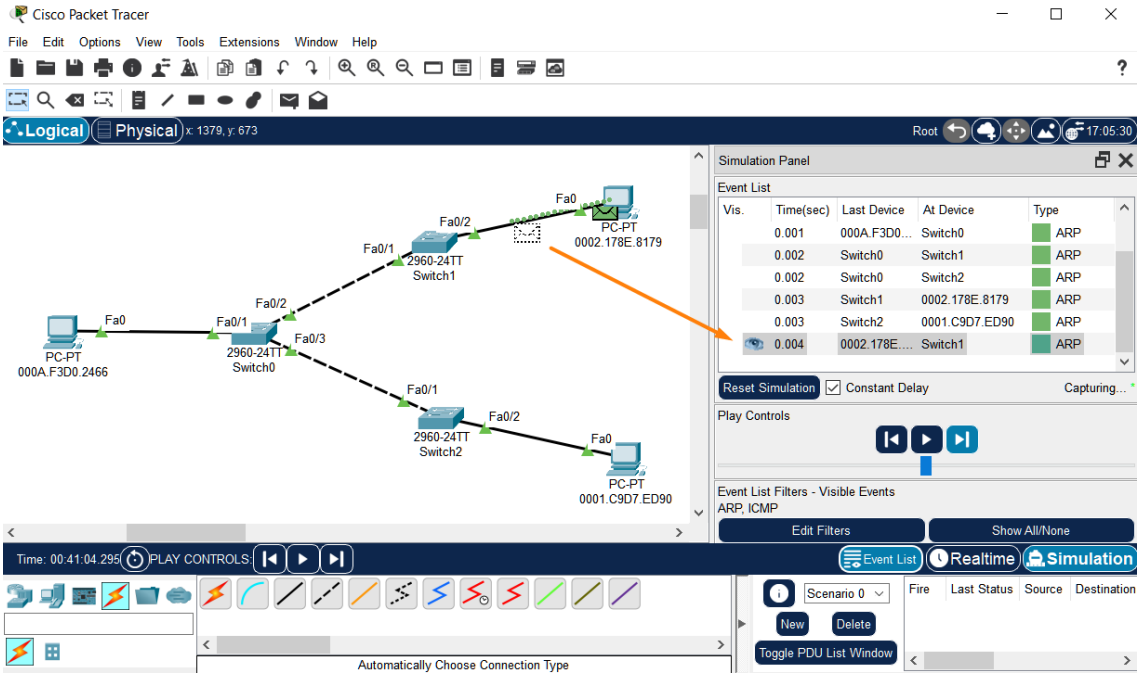

Figure 9.13 ARP reply

Step 6: The packet content can be explored by double clicking the packet in the Event List (Figure 9.14). The same information which was discovered in the first part of the practical activity can be seen here (information at all 7 layers of the OSI model, the same as in Wireshark).

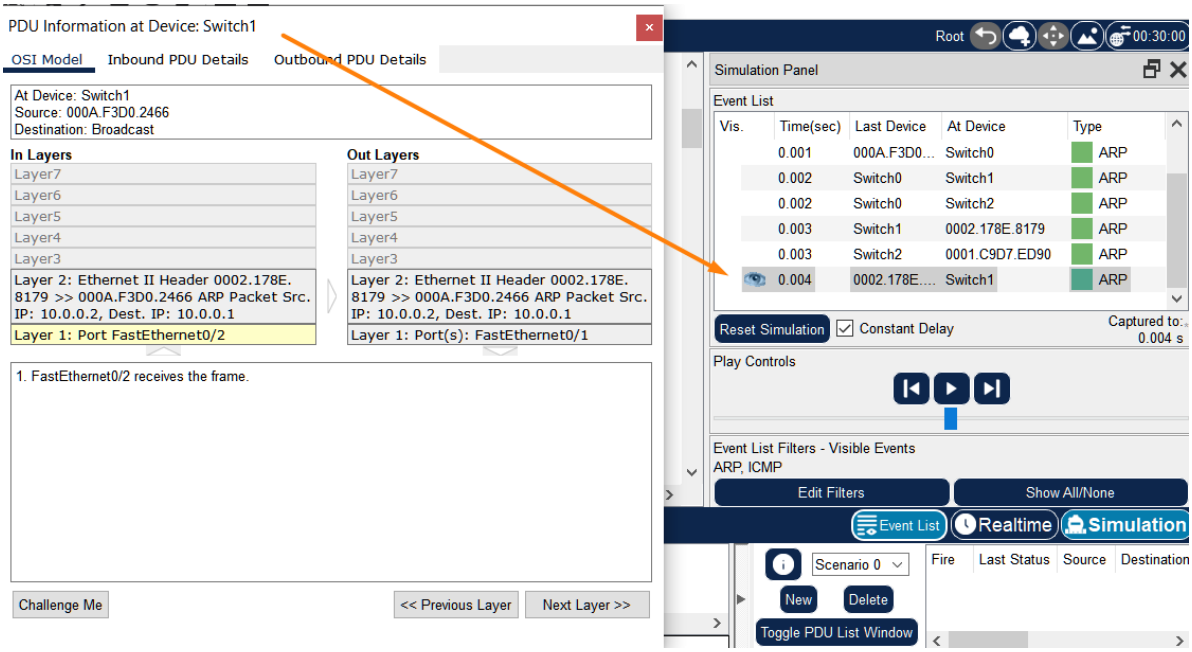

Figure 9.14 Inspection of frame content in PacketTracer

Step 7: Resume the packet flow in the network and inspect the console of the PC running the **ping** command. If the ARP reply takes too long to return then the first ICMP echo reply message might not reach the PC in time, resulting in a request timeout (recall that the **ping** utility uses ICMP echo requests and replies). This explains why you probably noticed in previous Packet Tracer activities that, especially on a newly formed network, some messages time out (Figure 9.15).

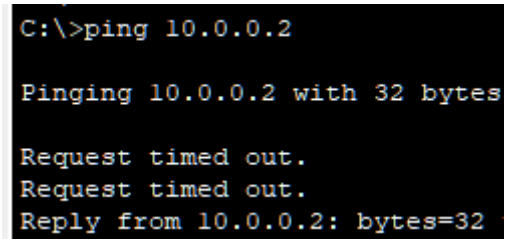

Figure 9.15 Ping timeout exemplified in PacketTracer

Step 8: Use the following commands on the switch components to inspect their MAC address tables and how they get populated when the first ARP requests/replies travel through the network.

Switch>enable
Switch#show mac address-table
Switch#clear mac address-table

b. NDP Simulation Mode

Using the IPv6 .pkt file which was created in a previous activity for the static routing functionality, apply the NDP packet filter in the Edit Filters window and inspect the traffic according to the description in the activity text. Find the RS, RA, NS and ND packets using the Simulation mode.