KHALIFA UNIVERSITY

Spring 2014
Computer Networks
CMPE323

Laboratory Experiment No. 2: Introduction to Ethernet Networks

Aims and Objectives:

- Introduce students to layer 2 aspects of Ethernet,
- the structure of Ethernet frames,
- the functionality of broadcast domains,
- foresee the scalability issues of broadcast domains,
- and basic protocol analysing techniques.

Materials Required:

- Ethernet switches,
- PCs with Ethernet adapters,
- and Straight-through/crossover/rollover cables.

Change Log:

- 23-2-2014: original document – mkhonji.
- 24-2-2014: multi-switch broadcast domain question moved to VLANs lab, fixed various typos – Dr. Amjad.

1 Introduction

Ethernet (specified in the standard IEEE 802.3) is a set of protocols that facilitate communication among devices over Local Area Networks (LANs). Such protocols span the OSI layers 1 and 2. For example, the standard IEEE 802.3 specification includes (not limited to):

- layer 1 specifications (e.g. cable specifications, line coding method[1]),
- and layer 2 specifications (e.g. MAC frames, LLC).

However, this lab will only focus on the most-widely used layer 2 aspects of Ethernet. ~~Layer 1 details will be omitted as they are closer to electrical engineering and communications than the subjects of CMPE323 labs.~~

1.1 High-level overview of Ethernet networks

For simplicity, this lab is focused on common layer 2 aspects of Ethernet II only, which is the most wide-spread Ethernet type.

- When nodes (e.g. PCs) transmit data over the network, the data must be encapsulated in a packet that is formatted according to Figure 1. **Note:** the hardware (e.g. Ethernet NICs) handle the fields `PREAMBLE`, `SFD`, `FRAME CHECK SEQUENCE` transparently and does not pass them to the upper software layers (the Operating System (OS) cannot see those fields). The same applies when sending messages, there is no need to ~~adds~~ add the named fields earlier in any system call as the hardware layer will add them too. In other words, we need to only send the fields that are marked as *frame* (as the hardware will add the surrounding *packet* fields automatically, which we cannot disable with normal NICs that we have in our lab).
- This packet encapsulation facilitates a number things, most importantly:
  - The ability to specify which target node (e.g. PC) should receive the data. This is possible since each Ethernet NIC in a PC has a unique address called the *MAC Address* and that intermediate Ethernet switches forward such Ethernet packets based on these addresses,
  - the ability to specify the protocol[2] by which the payload data is formatted as. This allows the receiver node to know how to interpret the Ethernet frame data payload,
  - A checksum value to allow the receiver to determine whether a received MAC frame has any errors[3].

[1] Basically, line coding methods (e.g. Manchester encoding) can describe how to signal bits of 0s and 1s to a remote end by transitioning voltage over a medium such as the copper wires in Cat5e cables that are available in the lab.

[2] The *Length/Type* field can be either used for the frame's length (in bytes) by which an LLC header is required, or the payload's protocol type as in Ethernet II (which this lab is about).

[3] Examples of errors could be the change of a bit of (say) 1 into a 0 due to electromagnetic interference that could be caused by electrical devices around the copper medium (as in our Cat5e cables).

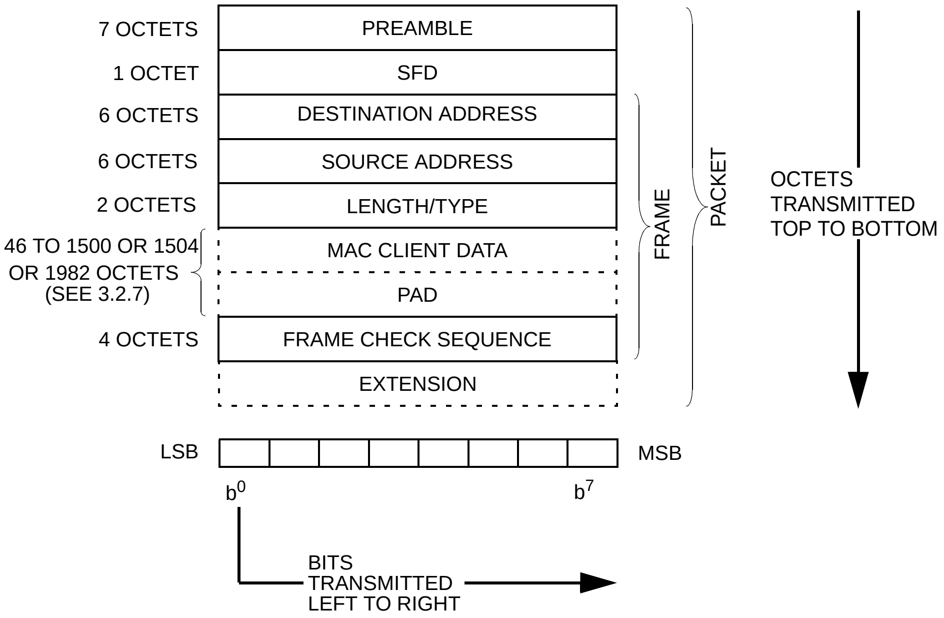

Figure 1: Ethernet Media Access Control (MAC) packet. Note: an octet is 8 bits — Source: IEEE Std. 802.3-2012.

- When an Ethernet switch receives a MAC packet, it generally performs the following tasks:
  - Populates its local MAC address table:
    1. For every MAC packet that enters the switch, the switch reads the source MAC address and adds an entry in its MAC address table that maps the source MAC address to the port identifier by which the corresponding MAC packet entered the switch.
  - Forwards the MAC packet:
    1. For every MAC packet that enters the switch, the switch reads the destination MAC address from the MAC packet.
    2. If the destination MAC address is `FF:FF:FF:FF:FF:FF` (in hex), the switch will forward the received MAC packet to all of its ports except the one from which it received the packet (i.e. broadcasting).
    3. For other MAC addresses, the switch looks up its MAC address table[4]. If the Ethernet switch finds an entry in the MAC address table, it will forward the MAC packet only to the Ethernet port associated with that entry. Else, it will forward the MAC packet to all of its ports (except the one it received the MAC packet from).

[4] The MAC address table basically contains entries that map MAC addresses to local Ethernet switch ports that lead to the node with the given MAC address.

2 Lab Preparation

- Connect PCs to an Ethernet switch as depicted in Figure 2.

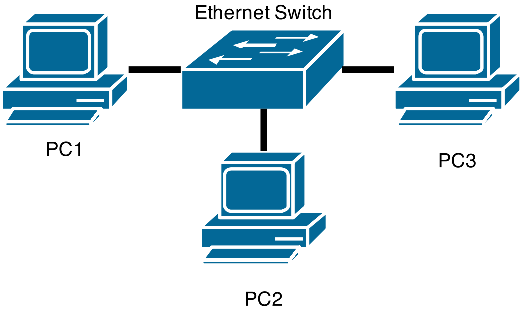

Figure 2: Experiment's physical topology.

- Erase any configuration from the switch:

  1. Connect the switch's console port to your PC's serial port using the provided roll-over cable (ask the lab engineer for help).
  2. If you are using Windows, use HyperTerm or Putty to connect to the serial port of your PC. If you are using a Linux distribution, execute the following command `screen /dev/ttySx` Where `x` is the ID of the serial port that you have connected the console port to.
  3. Once connected to the switch via the serial port (using `screen` or whatever Windows alternative), execute the following commands:
     (a) `enable` // to activate privileged mode.
     (b) `erase startup-config` // to delete previous configurations.
     (c) `reload` // to reboot the switch. Submit `N` when asked to save the configuration

- Run the protocol analyzer application *Wireshark* on all of the three PCs. You can do this by either following the GUI's menus or by simply executing the command `wireshark` in a terminal.

3 Lab Experiments

3.1 Send your 1^{st} manually-crafted MAC packet from PC1 ⟶ PC3

[33 points]

Note: when done, show your work to the the lab engineer for grading purposes.

1. View[5] the MAC address table of the switch, and note down the *dynamic* MAC address entries in the table.

[5] You can view the MAC address table of a Cisco switch by executing the command `show mac-address table` in the privileged mode.

2. From PC1, send[6] the following message:
    - Source MAC address (6 bytes in hex): `11:11:11:11:11:11`.
    - Destination MAC address (6 bytes in hex): `33:33:33:33:33:33`.
    - EtherType[7]: `0x05FF`
    - Message payload: “Hello” (in hex: `0x48, 0x65, 0x6c, 0x6c, 0x6f`).
3. Once you send the message above, answer the following:
    - Did the message appear on PC3’s Wireshark instance?
    - Did the message appear on PC2’s Wireshark instance?
    - View the MAC address table of the switch again, do you see any new added addresses?
    - What is the newly added MAC address entry in the MAC address table?
    - What interface is the new MAC addressed mapped to?

Note: for the purpose of this task, you can choose any 2 byte value for EthType. However, we suggest `0x05FF` as it is not assigned to any protocol so that Wireshark won’t attempt to parse the payload once you capture it (for visual clarity).

3.2 Send your 1st manually-crafted MAC packet from PC3 ⟶ PC1

[33 points]

Note: When done, show your work to the the lab engineer for grading purposes.

1. From PC3, send the follows back to PC1’s MAC address:
    - Source MAC address (6 bytes in hex): `33:33:33:33:33:33`.
    - Destination MAC address (6 bytes in hex): `11:11:11:11:11:11`
    - EtherType: `0x05FF`
    - Message payload: “Hello” (in hex: `0x48, 0x65, 0x6c, 0x6c, 0x6f`).
2. Once you send the message above, answer the following:
    - Did the message appear on PC1’s Wireshark instance?
    - Did the message appear on PC2’s Wireshark instance?
    - View the MAC address table of the switch again, do you see any new added addresses?
    - What is the newly added MAC address entry in the MAC address table?
    - What interface is the new MAC addressed mapped to?

Note: the same MAC addresses that PC1 used earlier. Recall the added entry in the MAC address table?

[6] You can send such low-level messages using `scapy`’s `sendp` command as follows:
```
sendp('\x11\x11\x11\x11\x11\x11\x33\x33\x33\x33\x33\x33\x05\xFF\x48\x65\x6C\x6C\x6F')
```

[7] Named as *Type/Length* in Figure 1.

3.3 Logical conclusions

[34 points]

Note: When done, show your work to the the lab engineer for grading purposes.

Based on the experiments that you have performed earlier, answer the following questions:

- How do Ethernet networks identify ~~its~~ their nodes (or PCs)? In other words, how can the Ethernet network know which node you wish to send your MAC packet to?
- If an Ethernet switch receives a MAC packet from its interface X, and the destination MAC address that does not exist in its MAC address table, then:
  - How would the switch update the MAC address table?
  - How would the switch forward the packet?
- If an Ethernet switch receives a MAC packet with a destination MAC address that does indeed exist in its MAC address table such that it is mapped to some interface X, how would the switch forward the packet?
  - How would the switch forward the packet?

  ~~Suppose that you have 1,000,000 nodes (or PCs) connected to a set of connected switches forming a single broadcast domain, also suppose that all of the nodes have unique MAC addresses and are communicating with each other. How many MAC addresses do you expect to exist in each Ethernet switch?~~

3.4 ~~Bonus~~ Extra questions (not graded)

Based on the behaviour that you saw in earlier experiments, below are additional interesting questions that you are encouraged to think about (and answer):

Note: when done, show your answers to the lab engineer for feedback. If the lab time is not enough, take your time and submit it in a later time. Although not graded, it can strengthen your understanding of the subject.

- What do you think are the possible outcomes if we connect ~~the~~ 1,000,000 actively inter-communicating nodes to a set of daisy-chained switches were one of the switches has a maximum capacity of storing 500 MAC addresses only?
- How would the switch behave if its MAC address table is full?
- Repeat the experiments 3.1 and 3.2 except for replacing the MAC address of PC1 to `FF:FF:FF:FF:FF:FF`. What is the difference? Do you notice anything special with this address?