

CSIS0234B Computer and Data Communication (Class B)

Tutorial 6

Understanding Ethernet address translation

In this tutorial, you will see the steps that your computer performs to find the address of another party on the same Ethernet link. A tool called “ethereal” has been installed to let you see clearly the frames sent. It uses the packet capturing library to capture frames, check the protocols used in the various layers of the frame, and display it nicely in a GUI screen. This saves us from a lot of low-level reading of frames. We will use it to understand the complete address translation process involved when a network packet needs to be sent.

Work in pairs and share two computers: our exercise investigate frames between the two.

1. Familiarizing with ethereal

- a. Login as root (only the administrator has the capability to capture all packets, for the obvious reason that otherwise, all the passwords sent on the system can be sniffed by every user). Create a terminal window and run “ethereal”.
- b. In the “Capture” menu, select “Start”. Note that there is a **filter** text box, which allow you to type a capture filter. Make sure that is empty (so that we capture everything), and press “OK”. Wait for 5 seconds, and stop the capturing. There are 3 parts of the resulting display: the first shows a **summary** line for each frame captured, the second shows the frame content in an **overview** format, and the third shows the frame as **plain hexadecimal data**. Try to click on different parts of each pane to see what you get from them.
- c. Note that there are many frames captured. As an effort to reduce the number of frames captured, try to do the same thing again, but **disable promiscuous mode** (which can be specified after you choose “Capture->Start”). Do you notice any reduction in traffic?
- d. Create a capture filter that chooses only frames that are not Ethernet multicast (multicast includes broadcast) and at the same time not destined to the computer you use (by checking the frame does not have the destination being your Ethernet address—your Ethernet address can be found by typing “ipconfig”, and is displayed under the heading “HWaddr”). Can you capture any frame? What you can infer about what is connecting the network—hubs or switches?

2. Ethernet encapsulation

- a. Now clear the filter that you have made, and capture for a while again. What is the size of the smallest frame that you can capture?
- b. The book claims that Ethernet frames are always padded to a minimum of 64 bytes. Is the book wrong? (Answer only after reading again the frame layout of Ethernet!)
- c. In a pure TCP/IP Ethernet network, all packets should have Ethernet frame type 0x800, 0x806 and 0x8035 (IP, ARP and RARP). Can you find any other traffic on the network?
- d. Note that some Ethernet frames has an “unknown” protocol, while some are “raw” (i.e., in place of the “protocol” field, an Ethernet frame length is put). Suggest a way to differentiate between them. (Hint: what is the maximum Ethernet frame size?)

3. Getting hold of IP Addresses

For this part you have to work with your partner. We will call one of them computer 1 and the other computer 2.

- a. In both computers, setup a capturing filter to select only packets that **involves** the Ethernet address of either computer. Start capturing, and try to ping computer 2 from computer 1 using the command `ping -n hw335aNN` where NN is the computer number of computer 2 (using the `-n` flag asks ping not to try to do a reverse DNS lookup, which gives a cleaner display). Stop capturing, and explain why each frame is sent.
- b. Find the DNS response, and compare it with the output of `dig`. Try to understand every byte of the frame. Is there any of them that you cannot map to the result of `dig`?
- c. Try again using an IP address rather than the DNS hostname. Is there any difference in the type of frames sent?
- d. Repeat one of the above steps without the `-n` option, and see what extra frames are sent.

4. Address Resolution Protocol

Now we will focus our attention to the ARP protocol.

- a. You can display the current ARP cache by running the command `arp`. Note that there are quite some entries there already. We want to clean them first.
- b. From the KDE menu, find the item **System -> Network Configuration**, and use it to deactivate the network and activate it again, which should clear the ARP cache. Do this **for both computers**. Repeat the ping exercise, this time restrict the filter so that only ARP frames are selected. Explain every ARP packets that is sent, and try to understand completely the content of each ARP frame. Note in particular **which of these frames are broadcasted**.
- c. Restart the network again for both computers, and use `arp` to make sure that the ARP cache is empty. Start capturing again in computer 1. Stop the network of computer 2, setup its network parameter so that the IP address used is the same as computer 1, and start the network again. Explain the frames sent. (Note: the frames are sent by a program called `arping`, which use is to detect duplicated IP address so that the user know about the clash.)
- d. Start capturing again in computer 1, reset the network configuration of computer 2, and restart the network. Explain every ARP packet sent. (Note: there shouldn't be any RARP/BOOTP/DHCP message transferred, since all computers in the lab has hard-coded IP addresses.)
- e. Look at the arp cache now. Does the Linux implementation tries to enter hardware addresses into ARP cache as they come by? Explain what are the trade-offs.
- f. Start capturing again, and try to ping a non-existent computer `172.16.9.254`. Explain the results. Check the `arp` cache after stopping the ping program (with Control-C).