Programming and Experimenting with Ethernet

Hui Chen
Computer Science
Virginia State University, Virginia 23806

Written on September 14, 2015
Lastly revised on September 15, 2016
Revision: 7a4f92488af8

The objective of this assignment is to gain familiarity with Ethernet including unicast, broadcast, and multicast addresses, Ethernet frame formats, and socket APIs pertinent to programming with Ethernet.

You are required to complete the tasks in this document and ***prepare a report***. In the report, describe *concisely* your activities, observations, and answers.

1 Programming and Experiment Environment

You will use Linux virtual machines for this lab. See the additional document entitled "Setting up Virtual Machines for Network Programming and Experiment" for detail. This additional document is important to understand the network setup of the virtual machines for this exercise.

Unless explicitly stated otherwise, all the operations shown in this document are command line operations.

2 Example Programs

The instructor provides 2 pairs of example programs, the pair of `ethercap` and `etherinj`, and the pair of `ethersend` and `etherrecv`. The programs are hosted on a `git` code repository at Github.com. Upon log in a Linux virtual machine, you can obtain the source code of the programs by cloning the repository using `git` as follows,

```
git clone https://github.com/huichen-cs/ethernet.git
```

To compile the example programs, go to the directory where the cloned repository resides and issue the command `make`, i.e., provided that you cloned the repository under your home directory, do the following,

```
cd ~/ethernet
make
```

To learn how to use the programs, run the programs from the command line without giving any command line arguments and follow the brief instructions that the programs display.

3 Tasks

3.1 Ethernet Frame Capture and Injection

To make the description of the exercise easier, assume that two individuals, Alice and Bob, are communicate with each other on ***two virtual machines***.

Be aware that your Ethernet interface's names and addresses may not be the same as the instructions below. To determine which Ethernet interface and what address to use is a part of the exercise.

1. Find out available Ethernet interfaces and their status on the two computers Bob and Alice are using the command `ip` and determine an Ethernet adaptor on each computer and the adaptors are on the same Ethernet.

2. Assume that the Ethernet interfaces in both Alice and Bob computers are `eth0`. Be aware that `eth0` is *incorrect* for the provided virtual machines. To determine which Ethernet interface to use is a part of the exercise.

3. Check if the adapters are up using the command `ip`. If an adaptor is down, you can turn it on using the `ip` command. For example, to turn on `eth0`, we run

   ```
   sudo ip link set eth0 up
   ```

4. Bob starts capturing fames and saving the captured frames to the file `frame_captured.txt` by,

   ```
   sudo ./ethercap eth0 | tee -a frame_captured.txt
   ```

   Note that you may want to give the captured frame file a more appropriate name from which you can tell which experiment you are doing.

5. Alice injects a frame to the link, for example,

   ```
   sudo ./etherinj -s 10:22:33:44:55:66 -d 60:55:44:33:22:11 -m "Hello, World" eth0
   ```

   where `10:22:33:44:55:66` is the Ethernet address of interface `eth0` on Alice's computer, `60:55:44:33:22:11` is the Ethernet address of interface `eth0` on Bob's computer, and the message that Alice transmits to Bob is "Hello, World".

6. After Alice finishes injecting the frame, Bob may stop frame capturing by press `CTRL-C`.

7. Examine the captured frame file to verify the message is successfully transmitted.

Answer the questions,

1. Locate the frame transmitted from `etherinj` and list the values of all fields of the captured Ethernet frame that carries the "Hello, World" message using Table 1 ,

Table 1: Table for Captured Frame

Captured Frame		
To fill with `<bytes in capture frame in both hexadecimals and characters>`		
Frame Header	Destination Address	To fill with `<bytes in hexadecimals>`
	Source Address	To fill with `<bytes in hexadecimals>`
	Type or Length of Frame	To fill with `<bytes in hexadecimals>`
Body of Frame	To fill with `<bytes in body of frame>`	

2. Can you run `etherinj` and `ethercap` without using `sudo`?

3. Explain from the computer security perspective, why does Linux require you to run such programs using a special permission, such as using `sudo`?

3.2 Unicast, Broadcast, and Multicast

Using the `ethercap` and `etherinj` programs, send and receive a message using `unicast` address, `broadcast` address, and `multicast` address.

Describe the examples and observations in the report.

3.3 Extending Programs for Unicast, Broadcast, and Multicast

You are given two other programs, `ethersend` and `etherrecv`. First, figure out how to run the programs. At present, the `etherecv` program only receives the frames with unicast addresses. Revise the program (or the programs) so that `ethersend` can transmit to `etherrecv` using *unicast*, *broadcast*, and *multicast* addresses.

The instructor suggests that you make a copy of the programs and work on the copy.

Prepare a test procedure and test the program using the test procedure.

3.4 Extending Programs for Duplex Communication

The communication achieved via the two programs `ethersend` and `etherrecv` is one way, i.e., `ethersend` can *only* send message and `etherrecv` can *only* receive message. In other words, these two programs can send and receive messages on two computers on Ethernet, however, only in *simplex* fashion.

Extend the programs so that both of the programs are able to send and receive messages, i.e., in *half-duplex* or *full-duplex* fashion. It is simpler to extend the two programs to communicate in the half-duplex fashion than in the full-duplex fashion.

The instructor suggests that you make a copy of the programs and work on the copy.

Prepare a test procedure and test the program using the test procedure.

4 Submission

Upload the report, along with the extended programs to Blackboard October 5, 2016.