WIS5059: Homework 3 Project

Instructor: Jaehoon (Paul) Jeong

Date: November 3rd, 2020

Due: November 17th, 2020

**Total Credits: 100 points**

Project: IP Network Emulation

**Note:** If you don't have 'Ubuntu Linux' on your desktop, you can install VM player and Ubuntu in your windows computer as follows:

* **VM Player:** <https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0>
* **Ubuntu Linux:**<http://www.ubuntu.com/download/desktop>
* Recommended to use **Ubuntu Linux 16.04**

1. Overview

In this project, the students can learn how IP hosts can exchange IP datagrams each other through IP network consisting of IP routers. A LAN is typically a broadcast network, where IP hosts are attached to a common shared physical medium and a transmission from any one host is broadcast to and received by all other hosts. However, in the case where the receiver is IP host, the received IP datagram is discarded when the receiver is not the destination of the IP datagram. On the other hand, in the case where the receiver is IP router, it forwards the IP datagram to the appropriate next hop according to its routing information. For this routing information, the routers exchange periodically their routing information with their neighbor routers. Through these routing, a bunch of IP networks can be connected into the Internet. In this project, we adopt Routing Information Protocol (RIP) as routing protocol [1][2]. RIP is a distance vector routing protocol on the basis of Bellman and Ford’s shortest path algorithm [3]. Therefore, a **router** provides network-layer interconnection between two or more LANs and makes it appear to all the hosts on these LANs that there is a single network. This project is about the **IP network emulation** of the functionalities of hosts, hubs and routers using BSD sockets for inter-process communication in place of a physical medium.

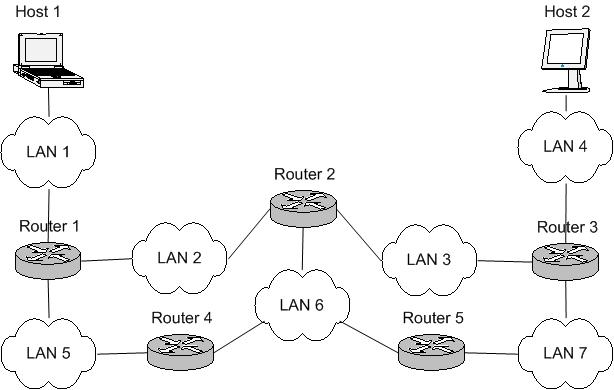


Figure 1. IP Network comprised of multiple LANs

Figure 1 shows a typical IP network consisting of several LANs. We consider a star-shaped bus LAN topology (e.g., as used in 10BASE-T Ethernet LAN) where each host is directly connected to a common central node, called *a* ***repeating hub*** or ***hub*** in short. Each host is connected to the hub. The functionality of a hub is to repeat a frame it receives from one host (via one of its ports) to all the *other* hosts attached to it, i.e., by transmitting the frame through all the ports except for the one from which it receives the frame. This enables every host to hear the transmission from any other host connected to the hub. Note that in our setting there are no collisions.

A host can transmit an IP datagram (or packet) to another host. Before it transmits the packet, it needs to add an IP header. The two most important fields in the header are ***source IP address*** and ***destination IP address***. A host should accept an IP datagram sent by another host which is addressed to it while discarding any other IP datagram which it receives but is not addressed to it.

A router is used to connect two or more LANs to form an extended IP network. It is like any other host except that it is connected to more than one hub and thus be able to receive any IP datagrams transmitted on any LAN it is attached to. The functionality of a router is thus to appropriately forward and filter IP datagrams. It also needs to run RIP routing protocol to exchange routing information with other routers, through which loops in the network should be removed.

You are already provided with the implementation of host and hub. You only need to implement the router by filling in the missing code.

2. Emulation Model

In order to emulate the physical links of a LAN, we will make use of BSD sockets. As you know, the paradigm used in BSD socket programming is a client-server model. In our IP network emulation model, we will have a hub act as a ***server*** and hosts attached to it as its ***clients***, and the physical links between the hub and a host will be emulated by a ***connection-oriented*** socket connection. Thus, in order to allow a host to attach itself to the hub, the hub (like any server) must first create a socket, bind its address and wait for connection requests from clients.

A client has to be aware of the whereabouts of the server in any client-server model (a common approach is to have the server use a well-known port). Here we make ***lan-name*** one of the command line arguments, so that this acts as a common bond between a hub and all its hosts/routers. There can be only one hub for a given ***lan-name*.** Once a hub is ready to listen, it stores its own IP address and the port by creating a link **.<*lan-name>*.info** in the current working directory. This is just one way of sharing information between a server and a client. Given a ***lan-name***, a host, by reading this link, knows the complete address of the hub. Obviously you always have to run a hub first before running a host. You only need to understand but not worry about this since all this is already done for you.

3. Components

Our model consists of three components namely hub, host and router. The program interface and the required functionalities for each of these components are described below.

## 3.1 Hub

A hub takes a command line argument *lan-name*. We have already explained the role of *lan-name*. A hub has to make sure that there is no other hub already existing for the same *lan-name*.

usage : hub <lan-name>

## 3.2 Host

A host takes two arguments, *myname*, *lan-name*. It is assumed that (hardcoded in common.h) the name of a host to its hardware address mapping (name address pairs) is provided in a file named hosts in the current working directory. This allows us to use symbolic names in place of IP addresses while displaying and typing in messages (*packets carry actual IP addresses*). This host's IP address can be figured out by looking up for *myname* in the name to address mapping table. The argument *lan-name* is used as explained earlier.

usage : host <myname> <lan-name>

## 3.3 Router

A router is like any other host, except that it may be connected to more than one hub. It gets attached to all the hubs corresponding to each *lan-name* specified as its command line argument. It performs RIP routing protocol, builds a forwarding table, and either forwards or discards packets based on their destination addresses.

usage : router <myname> <configint> <lan-name> <lan-name> [<lan-name> ...]

The myname is the DNS name of the router. The configint is a parameter used in RIP routing protocol. The configint is the interval between successive generations of RIP messages by each router. RIP message is as follows:

**RIPv1 header:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** |
| [Command](http://www.networksorcery.com/enp/protocol/rip.htm#Command) | | | | | | | | [Version](http://www.networksorcery.com/enp/protocol/rip.htm#Version) | | | | | | | | 0 | | | | | | | | | | | | | | | |
| [RIPv1 entry table](http://www.networksorcery.com/enp/protocol/rip.htm#RIPv1 entry table) ::: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**RIPv1 entry table.** Variable length.  
An array of RIPv1 entries. Each RIPv1 entry contains the following structure:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** |
| [Address family](http://www.networksorcery.com/enp/protocol/rip.htm#Address family) | | | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | | |
| [IPv4 address](http://www.networksorcery.com/enp/protocol/rip.htm#IPv4 address) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Metric](http://www.networksorcery.com/enp/protocol/rip.htm#Metric) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Address family.** 16 bits.

**IPv4 address.** 32 bits.

**Metric.** 32 bits.

Contains a value from 1 to 15 which specifies the current metric for the destination. A value of 16 indicates that the destination is not reachable.

4. Suggested Approach

We suggest that you follow the following steps so that you can gradually complete the project.

## 4.1 Single LAN

Consider a single LAN onto which multiple hosts are attached without any router. You can do this by invoking a single hub and multiple hosts as follows.

% hub lan1

% host mercury lan1

% host peca lan1

Now to send a message, you have to type host name followed by space and then the message. For example to send a message to peca, you can type "peca hello" at mercury. Here is an example transcript.

% host mercury lan1 router1

admin: connected to hub on 'atto.cs.umn.edu' at '37788'

peca hello

peca : how are you

% host peca lan1 router1

admin: connected to hub on 'atto.cs.umn.edu' at '37788'

mercury : hello

mercury how are you

To try this you do not need to do any coding. The hub and host are already implemented and given to you. But for the rest of the following, you need to fill in the missing code in router.c.

## 4.2 Single Router

Now create two LANs and introduce a router between them. Have two hosts, one on each LAN. The invocation would look something like the following.

% hub lan1

% hub lan2

% host mercury lan1

% host peca lan2

% router router1 0 lan1 lan2

This way you can check to see how the router combines two LANs into an IP network and performs IP datagram forwarding.

## 4.3 Two Routers

Now introduce two routers between the same two LANs. Run RIP routing protocol with the same setting. You can do something like this.

% router router1 30 lan1 lan2

% router router2 30 lan1 lan2

In order that a packet is circulated forever between lan1 and lan2, the routers discards the packet which has been fo rwarded with the pair of the source IP address and destination IP address and identification field in IP datagram. The packets would not loop since router2 simply discards all the packets.

Suppose now you kill the router1 (with Ctrl-C), then lan1 and lan2 become disconnected though the router is still alive. This is because router2 can still connect lan1 and lan2.

## 4.4 Multiple Routers

Now introduce fours routers and make a larger IP network with seven LANs like in Figure 1. Host 1 and Host 2 should communicate via a shortest path, (Router 1, Router 2, Router 3). When Router 2 is down, a new path should work for the communication between Host 1 and Host 2, which is (Router 1, Router 4, Router 5, Router 3).

References

[1] C. Hedrick, "RIP: Routing Information Protocol", RFC 1058, June 1988.

[2] D. Comer, “Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture”, 4th Ed., Prentice Hall, 2000.

[3] T. Cormen, “Introduction to Algorithms”, 3rd Ed., MIT Press, 2009.