Programming Assignment 2
Due 12/11/2004 at 5:00pm

1 What this assignment is about?
In this assignment, you are required to implement a subset of routing software that runs on most routers. Specifically, you will write a simplified version of "Routing Information Protocol" (RIP). The RIP algorithm is explained in section 4 and problem definition is given is section 5. Please read those sections carefully.

For this programming assignment, we are providing you with a network simulator that is written in C++ programming language. The implementation of our simplified version of RIP is also required in C++. The programming assignments will be graded on a Solaris system; your code must compile and run correctly on suraj on which all the registered students will get an account. You should use the Makefile provided with this assignment to compile and link your code. You can do the development on other systems as well but it is your responsibility to make sure that your code correctly runs on suraj. Details for downloading the appropriate files and programming environment help are given on the class web site.

2 Routing Protocols
Internet is composed of units each of which represent a collection of IP addresses under common technical administration. Such a unit is called an Autonomous System (AS). Every AS is assigned a 16-bit number by InterNIC; this number uniquely identifies an AS.

The routing protocols used within an AS are called Interior Gateway Protocols (IGPs) and routing protocols used between two and more AS are called Exterior Gateway Protocols (EGPs) as shown in Figure 1.

Figure 1: Depicting routing protocols in the Internet. Figure reproduced from [1].
Some well-known examples of IGPs are Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Intermediate System to Intermediate System (IS-IS). A well known EGP is Border Gateway Protocol (BGP).

Routing protocols are divided into two categories: **distance vector protocols** and **link state protocols**. A distance vector routing protocol uses distance (usually, hop count) to compute the shortest routes. Each router knows its distance to all immediate neighbors and advertises this information to all of them, but the routers have no information regarding the overall state of the network. A link state protocol, on the other hand, is based on the status and speeds of the physical links. Each router has a complete map of the network and route changes are immediately propagated to every router in the network.

RIP is a distance vector protocol whereas OSPF is a link state protocol. Distance vector protocols take longer to converge as compared to the link state protocols. On the other hand, link state protocols are complex, difficult to configure, and resource intensive.

### 3 Introduction

RIP is based on the Bellman-Ford algorithm. RIP sends its routing table in a RIP Packet to its immediate neighbors. The routing table is a collection of RIP entries. Each RIP Entry consists of:

- A destination IP address,
- Cost to reach that destination through a shortest path, and
- Next hop router’s IP address on that shortest path.

Upon receipt of a RIP Packet, the routers update their routing tables accordingly and then send that routing table to their own immediate neighbors. In this way, routes are calculated and routing updates reach all the routers in the network.

### 4 RIP Algorithm

1. When the router is first booted up, the routing table consists of a single RIP Entry. That RIP Entry has its destination router as itself and cost equal to zero. For example, a router with IP address 1 will have the following RIP Entry:

<table>
<thead>
<tr>
<th>Destination IP</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

2. The router broadcasts its RIP Entries to all of its neighbors either at predetermined intervals or triggered by some event.

3. RIP Entries are extracted from each RIP Packet received from the neighbors. For each RIP Entry, if the destination IP contained in that RIP Entry is not
found in the routing table of the receiving router, a new RIP Entry is created after adding cost of the link on which that RIP Packet is received (this cost is usually 1). Thus, when a RIP Packet containing the RIP Entry of step 1 is received by a neighboring router with IP address 2, the routing table of the receiving router will become:

<table>
<thead>
<tr>
<th>Destination IP</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>1</td>
<td>0+1 = 1</td>
<td>1</td>
</tr>
</tbody>
</table>

For creating a RIP Entry in the routing table, a router does the following:

a) The Next Hop is set as that immediate neighbor from whom the RIP Packet (containing that RIP Entry) was received.
b) The destination IP is the same as the destination IP in the RIP Entry.
c) Cost is set as the sum of Cost contained in the Entry and the cost of link on which the RIP Packet (containing that RIP Entry) was received.

Consider the example shown in Figure 2. Three routers, running the RIP protocol, are connected as indicated. Assume that Router-3 is the first to send its RIP Entries (routing table) to its neighbor, Router-2, in a RIP Packet. The new routing table for Router-2 will then look as in step 1 in Table 1. At that time, in the same step, the routing table of Router-1 has the only RIP Entry which is present when a router boots up. After some time, Router-2 sends its RIP Entries to its neighbors, Router-1 and Router-3. Router-1 will update its cost and next hop as shown in step 2 in Table 1. Notice that the next hop is different from the destination IP.

**Figure 2:** Example with 3 routers.

**Table 1:** Routing table updates for Example in Figure 2.
4. After a few iterations, it is likely that a router will receive a RIP Packet with a RIP Entry whose destination IP matches that of an already existing RIP Entry in the routing table. In this case, the router will calculate the “new cost” for the destination IP (sum of cost in the received RIP Entry and the link cost) and then perform the following checks:
   
i. If the RIP Entry is received from the router which IS LISTED as the next hop router in the existing RIP Entry in the routing table, simply replace the cost in the existing RIP Entry by minimum of “new cost” and 16. That is, if the new cost is 16 or higher, simply replace the cost in the existing RIP Entry in the routing table with 16; otherwise replace the cost in the existing RIP Entry with the “new cost”. Note that new cost may be higher or lower than the existing cost.
   
ii. If the RIP Entry is received from a router which is NOT listed as the next hop router in the existing RIP Entry in the routing table, then compare the “new cost” with the existing RIP Entry cost:
      
   a) If “new cost” is greater than or equal to the existing cost, leave the existing RIP Entry as is (no change in the routing table).
   
   b) If “new cost” is smaller than the existing cost, update the existing RIP Entry. Use “new cost” as the cost and sending neighbor router as the next hop.

5. Routers continuously send their RIP Entries to the neighbors after a fixed period of time dictated by the network simulator.

4.1 What if a link breaks?

If a link breaks, the routers which are located at the ends of the broken link will discover this failure immediately. These routers will update their RIP entries which may list the neighbor across the failed link as the next hop by changing the cost to infinity (i.e., 16 which is considered infinity in RIP). For example, consider a router with IP address 7 having following RIP Entries in the routing table:

<table>
<thead>
<tr>
<th>Destination IP</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Suppose that the link to router with IP address 5 is broken. Then the new routing table will be as follows:

<table>
<thead>
<tr>
<th>Destination IP</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>(change to) 16</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>(change to) 16</td>
<td>5</td>
</tr>
</tbody>
</table>

Once the new routing table gets advertised, all routers will update their routing table accordingly and find new alternative routes.

### 4.2 Count to infinity

During RIP operation, routers keep only the best route (i.e., a route with minimum cost) while adding routes to their routing table. As explained in the previous section, routers increase (by updating!) the cost of a route only if it is announced by the same router which is listed as the next hop for that route.

Consider the previous example of three routers and assume that the link between Router-2 and Router-3 breaks as shown in Figure 3. Further assume that Router-1 and Router-2 had the correct RIP Entries for reaching Router-3 before the link breaks.

![Figure 3: Demonstrating counting to infinity in RIP.](image)

After detecting the broken link, Router-2 will update its routing table to infinity (16). If Router-1 gets a chance to advertise its RIP Entries to Router-2 before Router-2 can advertise its entries to Router-1, this will lead to a situation where both routers count to infinity (16), as shown in Figure 3.
4.3 Split Horizon (SH)

Split horizon solves the problem of count to infinity by not allowing routers to advertise routes in the direction from which those routes are learned. That is, before advertising RIP Entries to a neighboring router, each router will remove those RIP Entries from the RIP Packet (a RIP Packet is what gets advertised!) whose next hop is that neighboring router itself. Therefore, if routers implement Split Horizon in example of Figure 3, then Router-1 will not advertise its RIP Entry regarding Router-3 to Router-2 because the next hop for that route is Router-2 itself.

4.4 Split Horizon with Poisonous Reverse (SHPR)

In split horizon with poisonous reverse, routers advertise a cost of infinity (16) to neighbors for routes which are received from those neighbors. To elaborate this, consider example of Figure 3 in which Router-1 can reach Router-3 using Router-2, i.e., Router-2 is the next hop for that route. Thus, if SHPR is implemented, when Router-1 advertises its routing table to Router-2, the RIP Entry about Router-3 in that advertisement has a cost of infinity (16). SHPR converges faster than simple SH.

4.5 Triggered Updates

Triggered updates allow a RIP router to announce changes in cost values almost immediately rather than waiting for the next advertisement opportunity which comes after fixed periods.

4.6 RIP Entry’s Timer

Every RIP Entry inside the routing table has a timer attached with it. The RIP Entry is invalidated on expiration of its associated timer. You do not need to worry about timers in this programming assignment.

5 Problem Definition

You are required to implement the algorithm running on each router; the algorithm is assumed to perform the following:

- Send appropriate RIP Packets as dictated by the version: simple RIP, RIP with simple split horizon, and RIP with split horizon with poisonous reverse; the version is chosen by the user.
- Correctly function whether triggered update is switched on or off.
- Update the router RIP Entries based on the RIP Entries received according to RIP Protocol, described in section 4. This piece of code should be written in recieveRIPPacket(...) function.
5.1 Functions

You are provided with a header file Router.h that contains the definition for each of the functions to be implemented. You will be writing code in Router.cpp for the following three functions:

1. void sendRIPPackets(...);
2. void receiveRIPPacket(...);
3. RIPPacket splitHorizonRIPPacket(...);

You will be using many predefined functions while writing your code.

6 The Class Diagram

Below is the class diagram of the simulator; most of the code is already written for you. The Class Diagram uses UML (unified modeling language).
6.1 Description of Classes

Here is a brief overview of classes used in this implementation. For more details about a particular class and its methods, you should look at the comments in the header files. You are not required to read all the header files for successful completion of this assignment.

**RIP Class:** The RIP Class is the entry point for the RIP simulator and contains the `main(…)` method. It uses a TopologyReader, a ConfigReader, and a number of Router objects. The `main(…)` method takes the filename of a configuration file as an input parameter. The RIP class populates ConfigReader and TopologyReader objects by reading data from respective text files. The RIP class also executes a loop to continuously call each Router’s `sendRIPPackets(…)` function in a round-robin fashion and updates the network topology for any changes in the links of a Router.

**RouterBase Class:** This class simulates a real-world router. Most of the code needed to build a router is included in this class and has been written for you. You will write the rest of the code needed to complete the router functionality in the Router class which is derived from RouterBase class. RouterBase class contains many “Port” and “RIPEntry” objects which are inherited by the Router class.

**Router Class:** This class provides part of the router functionality that is written by you. The methods you need to implement for this class are `receiveRIPPackets(…), sendRIPPackets(…)` and `splitHorizonRIPPacket(…)`.

**Port Class:** Port class represents a single port on a router. Every Port object has a unique IP assigned to it. This class contains the Link class.

**Link Class:** Link class is to connect/link two Port objects of two different Routers.

**RIPEntry Class:** A routing table of a RIP router is composed of a number of RIP Entries. RIPEntry class exactly represents such an entry. Every RIPEntry includes a destination address, the cost, and the next hop for that destination address.

**RIPPacket Class:** RIPPacket class is used by a Router class to send all of its RIP Entries to its immediate neighbors.

**ConfigReader Class:** This class reads the configuration for the simulator from a text file. The name of the text file is provided as a command line argument by the user. ConfigReader class is extended from FileReader and uses the utility functions from its super class (i.e. FileReader) to read the configuration file efficiently.

**TopologyReader Class:** This class reads the network topology from a text file and populates the Router objects. TopologyReader class also uses FileReader utility functions to read data from the file efficiently.
7 The Network Simulator

You are provided with a network simulator to which your router code interfaces. The simulator uses two input text files: one specifies the network topology and the other specifies configuration (options) to use.

7.1 Simulator Description

The network simulator performs the following functions:

1. It allows you to define the network topology in term of routers, port, and links that interconnect them. One can create more then one topologies at the same time using the topology file.

2. It is responsible for calling sendRIPPacket(...) for each Router in a round-robin fashion. Note that the function sendRIPPacket(...) is responsible to send RIP Entries to its neighbors.

3. It allows you to set different options using a configuration file. For example, setting trigger update to “On” and version of split horizon to be used.

4. It allows you to change the network links (but not the routers) while the program is running. By disconnecting links or making new links, one can demonstrate count to infinity problem and split horizon solution.

5. It prints useful information so that the user knows what is going on.

To run the simulator from command line, you need to type rip <path/config.txt>, where config.txt is the name of the configuration file. Composition of the configuration file is given in Section 7.3.

7.2 Network Topology File

The network topology is described in a text file (e.g., topology.txt) whose filename is specified in the configuration file. You can place comments anywhere in the topology file. Comments start with hash (#) and the simulator ignores lines that start with a hash. You can also place “Enters” and “spaces” in the topology file to make it more readable. Simulator ignores spaces within a command. A single command, however, should span only a single line. For example,

```
<First command>

<Second command>
```

In above example, empty lines between <first command> and <second command> will be ignored. Simulator will also ignore the extra spaces in the <first command>. Each command is used to define routers and their ports. You can define any number
of routers with any number of ports. The memory for routers is dynamically allocated by the network simulator at runtime after reading the topology file.

**Defining Routers with Example:**

```plaintext
Routers
# router IP, port 1 IP, port 2 IP, ... (this is a comment)
1, 11, 12, 13,
2, 21, 22,
3, 31,
```

Above example defines three different routers; keyword “Routers” is necessary to intimate the simulator that the commands that follow will define routers. No two routers can share an IP address. Similarly, port IP addresses are also globally unique.

**Defining Links with Example:**

```plaintext
Links
# A link connects ports of two different routers
# router A IP, router A port, router B IP, router B port, link Cost
# Cost is optional, default value is taken as 1
1, 11, 2, 21,
2, 22, 3, 31, 5,
```

Above example defines two links. First link connects port with IP address 11 (existing on router with IP address 1) to port with IP address 21 (existing on router with IP address 2) with default cost 1. Second link connects port with IP address 22 (existing on router with IP address 2) to port with IP address 31 (existing on router with IP address 3) with cost 5. Once again, the keyword “Links” is necessary. You are not allowed to create a link to an undefined port. This implies that the Routers keyword should precede the Links keyword in the topology file.

For the above example with Routers and Links, the topology is shown in Figure 4.

**Figure 4:** Pictorial representation of a topology file.
7.3 Configuration File

The configuration file, passed as a command line argument to the network simulator, contains the possible options to use with RIP simulator. Following is a sample configuration file with comments.

```
# split_Version can be 0, 1, 2 ...
# 0 means no split horizon, 1 means SH and 2 means SHPR
# Anything other then 0, 1, 2 defaults to 0
split_version = 0

# debug can be “on” or “off”
# If debug is on then it will print lots of extra debugging information
download = on

# steps_to_run
# It can be regarded as number of iterations or time for the simulation
# In each iteration, sendRIPPacket(...) is called for every Router.
steps_to_run = 20

# trigger_update can be either “on” or “off”. Other values default to off
trigger_updates = off

# complete name with path of the topology file to be used
topology_file = Z:\topology.txt
```

8 Deliverables and Instructions

You are required to submit your source files and a README file that contains:

First Name:
Middle Name:
Last Name:
Student ID:

If you wish, you can also make **BRIEF** comments about your program in the README file. Your comments, if any should not exceed a few sentences.

8.1 Submission instructions:

1. Make sure you remove any debugging `printf` statements in your code. Any such statements left therein may result in significant grade reduction due to the automatic nature of grading procedure.
2. Make sure your code runs with many different topology files. Check your result for count to infinity, split horizon (different versions), triggered updates on/off, etc.

3. Your code should be well commented and should be efficient enough.

4. zip the files to be submitted and drop them in the following folder:
   ```plaintext
   \badar\common\cs573\pa2
   ```

   Late days will be automatically deducted using the time of submission. DO NOT use any extra days except for your late days, if any are remaining. LATE SUBMISSIONS WILL INCUR VERY SIGNIFICANT PENALTIES.

9 References