Project I: Routing over the Internet

Introduction

Internet routing: We are living in the Internet world, and the Internet Protocol (IP) is one of the most successful protocols, in the human history. In this project, we simulate one of two primary functions in IP protocol: routing.

Routing is the way to find a path to the destination in the Internet world. On the Internet, there are more than 3 billion hosts. Routing helps us to identify where to send data from a location to any destination in the world.

The Internet allows us to collaborate with multiple different networks. The Internet routers are located at the edge of a network, delivering the data to a nearby network. IP network is a packet-switching network. Packet by packet, an Internet router 'forwards' packets from a network to another network.

To forward packets, an Internet router needs to know the next network toward billions of different destination hosts. Each router maintains a huge table, called routing table in order to manage routes to different networks. A routing table consists of multiple route records. A route entry remembers the next router to forward packet towards the destination.

To make concrete paths, routers communicate each other, exchanging the route information. The routers use 'routing protocols' for exchanging the routing information. This project simulates routing protocol over the Internet host. You need to implement a routing protocol that creates routing tables among the computers, and a forwarding engine that delivers packet data to the nearby router.

Project I: Routing over the Internet

Project description

In this project, you have to implement a routing protocol at the application layer. You can use multiple computers or multiple processes, each of which serves as an Internet router. The router processes are connected through virtual links. Nearby routers (that are connected through virtual links) communicate with the routing protocol. The routing protocol iteratively updates the routing information until it converges.

With the router processes and virtual links, an application can send the information over the virtual links. The application sends packets over the virtual links. The user data is divided into packets, fixed size sending unit over the virtual link. Once the router receives a packet from a link, the router looks up the routing table and finds the next router to forward. If the destination is found, the router process sends the packet to the next router over the virtual link.

You have to implement some of routing functions. The primary goal is to implement

a. Routing protocol among the routing processes.

Basic requirements:

1. Submission guideline. Use github. Leave your activity logs in the github by committing the codes regularly. Show your active participation to the group by pushing your own commits. Let me know the github repository in the submission.
2. Please format your documents (reports). If you have unique design considerations, please specify it. Please attach the working screen shot.
3. Routing protocol should work with five or more nodes (routing daemons).
   A. Routing daemon runs on some (IP:port).
   B. Initial links between the routers are given in text form.
      i. Example format) route1

//dstip	dstport	nextip	nextport	metric
192.168.0.1	8888	192.168.0.2	8888	1
Destination router		next hop router		distance

4. A routing daemon process should maintain a routing table that remembers the next router in the path to the destination.
   A. You can define your own simple route entry format, but you can refer RIP or the followings:
      i. <destination, nexthop router, the number of hops to destination>
5. Routers periodically exchange the routing tables, updating its own table.
   A. If the shorter path is identified, the route should be updated.
   B. If a path to the unknown destination is known, the routing table should also be updated along with the new path.
   C. After some finite exchanges, the routing table should be stablized.

6. Note that some part requires multi-threading or I/O multiplexing because while you're reading from socket, you cannot sending any data on the single thread.
   A. To avoid this, there are generally three ways to handle multiple concurrent sessions.
      i. Multi-process using fork: make a child process that handles a specific session
      ii. I/O multiplexing using select/poll: wait on multiple files at the same time, and do the job which is ready. OS tells you which files are ready.
      iii. Multi-threading using pthread_create: do the job with multiple threads, which makes easy to share data.
   B. Each routing protocol session exchanges route information, which is maintained by the routing daemon. If the multiple threads/processes are working concurrently, then the information should be carefully shared.
      i. You need to synchronize the shared data or
      ii. You need to cross the process boundary using IPC facility (msgq, sharemem, etc.) when multiple processes are involved.
7. Demonstration that can best show your code work.
   A. Partial screen-shot is okay, but
   B. In-class demo should be the best!

Extra options: There are many open options to implement; and here are some samples you can consider.

1. You can use sophisticated data structure than simple array, or linked list. The routing table of an Internet router has a large number of entries. Among the entries, you have to find the specific one that matches your destination router, and compares the metric values with the incoming route update message. Hash table or enhanced tree can be effectively used here. Of course, you can implement data structure or use library.
2. Link can be broken down. The routers check the link status, and if the link is broken, redirects the paths on the broken link. You can add programming interface that simulates link down.
3. Routers have forwarding engine. Every routers have routing queue (or packet queue), and sends data (not the routing protocol data, but user packets) according to the routing table. You can implement multiple traffic over the routers, and simulates how the traffic flows.
4. Different routing protocols. Routing protocols are generally categorized into two algorithms: Distance vector and Linkstate. You can implement two different algorithms, comparing the pros and cons of the different algorithms.
5. Policy-driven routing. Some Internet routers use policy-based routing for specific purposes. For example, specific routersA on the national border are avoided or all the traffic from routerB must also go through along with the specific routerC. Some high-cost, high-speed routers are avoided for fulfilling

lowest price charge. You can add some more rules/policies in your routers/routing protocols.

No programming language limits, and you can freely add your own assumptions.

Please submit your work by the last minute of April, including the reports/documents. 2 or 3 persons group work. Please make your group as early as possible.

Last, but not least, no plagarism is allowed. You have little references on the Internet, I am somewhat sure for this.

For your reference, look for Unix Network Programming, Linux system programming.

Happy hacking!

Seehwan

Appendix for your refs

referenced syscalls/lib functions

socket, pthread, sigaction, timer, open, close, send, recv, bind, accept, listen, connect, memset, htons/l, ntohs/l, memset, malloc/free