Simulate Link State Routing Protocol with Java Socket Programming

Goal

In this project, you are supposed to develop a pure user-space program which simulates the major functionalities of a routing device running Link State Routing protocol.

To simulate the real-world network environment, you have to start multiple instances of the program, each of which connecting with (some of) others via socket. Each program instance represents a router or host in the simulated network space; Correspondingly, the links connecting the routers/hosts and the IP addresses identifying the routers/hosts are simulated by the in-memory data structures.

By defining the format of the messages transmitting between the program instances, the parser and the handlers of these messages, you simulate the routing protocol with the user-space processes.

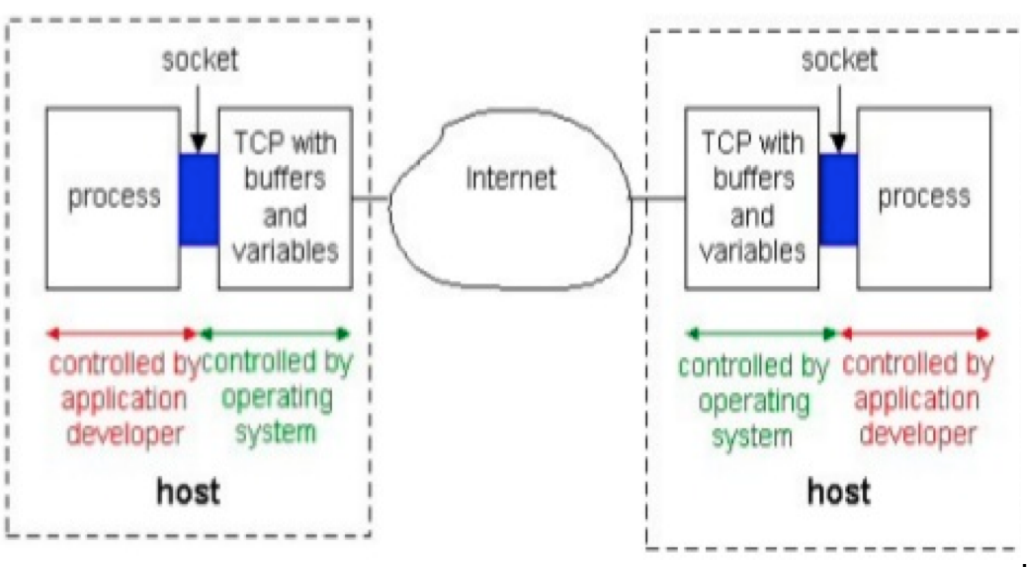
Prerequest

Before you do this PA, please ensure that you understand the basic concept of routing, especially Link State Routing, which is taught in class.

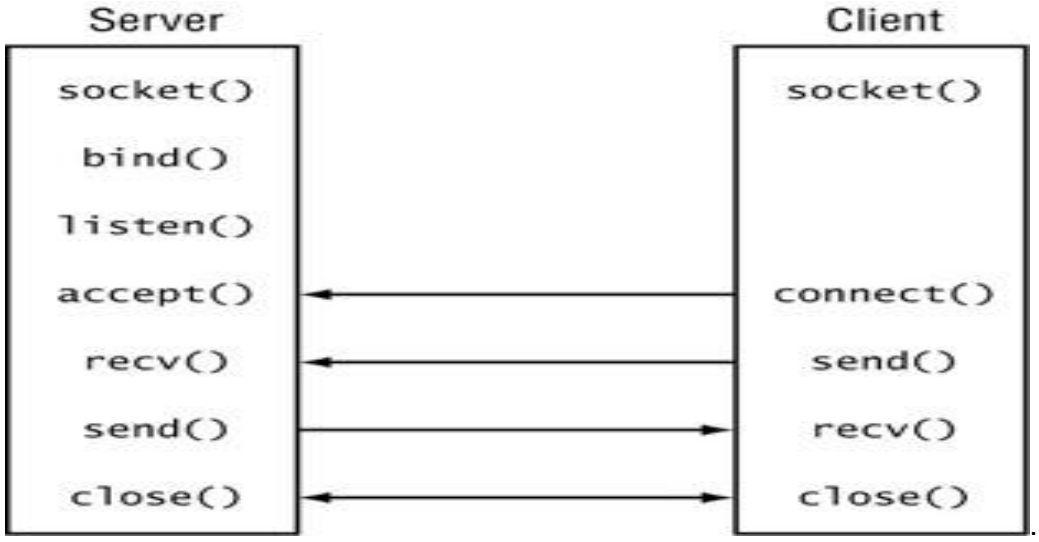
Recommended Reading: Link State Routing

Socket Programming 101

Socket is the interface between the application layer and tranmission layer



The existence of Socket interface greatly reduce the complexity of the developing network-based applications. When the processes communicate via socket, they usually fall into two categories of roles, `server` and `client`. The usage of socket in these two roles are different.



In `server` side, the program create a socket instance by calling `socket()`. With this socket instance, you can `bind()` it to a specific IP address and port, call `listen()` for waiting to active connections, `accept()` to accept the connection. After you call `accept()`, you can then call `recv()` and `send()` to transmit data between the client and itself. After you finish all tasks, you can call `close()` to shutdown the socket.

In `client` side, the story seems a bit simpler, after you call `socket()` to create a socket instance, you only need to call `connect()` with the specified IP address and port number to request service from the server side. After the connection is established, the following process is very similar to server side, i.e. transmit data with `recv()` and `send()` and shutdown with `close()`.

This is the general process of socket-based communication. To understand it better, you are suggested to read the article in here. The article is described in C programming language, which exposes many details of network data transmission but helpful to understand the concept.

Java Socket Programming

Different programming languages offers their own abstraction over socket programming. You are requestd to finish this project in Java. Java provides higher level abstraction for socket. In server side, You only need to call `ServerSocketserverSocket = new ServerSocket(port);` to create socket, bind, listen in one shot. In client side, `Socket client = new Socket(serverName, port);` creates socket instance and connect to the remote server.

The data tranmission between the server and client is through stream. For example, the following code snippet writes data to remote server and wait for the feedback.

```
OutputStream outToServer = client.getOutputStream();
DataOutputStream out =
               new DataOutputStream(outToServer);

out.writeUTF("Hello from "
             + client.getLocalSocketAddress());
InputStream inFromServer = client.getInputStream();
DataInputStream in =
                new DataInputStream(inFromServer);
System.out.println("Server says " + in.readUTF());
```

Here we say `wait for the feedback`, because `getInputStream().read()` is a blocking method. According to the Java API, "This method blocks until input data is available, the end of the stream is detected, or an exception

is thrown." (Blocking Read)).

This is not a good thing for high-throughput scenarios, e.g. routers. We have to develop some way to handle concurrent socket requests.

Recommended Reading: Java Socket Programming

Handle Concurrent Socket Requests

In this project, you are supposed to develop a multi-threaded server to handle concurrent socket requests from the client. Recommended Reading: Multi-threaded Java Server

Simulation of Link State Routing

The basic idea of link state routing is that every router maintains its own description of the connectivity of the complete network. As a result, each router can calculates the next best hop for all possible destinations in the network. The key point in Link State Routing Protocol is to synchronize the network description in all nodes (In this project, we only consider a single LAN deployment).

In the following paragraphs, we will describe your tasks in this project in details.

Router Design

The first task in this project is to design the `Router` class which represents a router instance in the simulated network.

Each router is identified by a simulated "IP address", it is simply a String variable. In this project, the routers is assumed to have 4 ports, which means that the router class contains a 4-length, `Link` typed array. When the router is connected with the other, you shall fill the array with a Link object. If the ports are all occupied, the connection shall not be established.

The most important component in router side is the `Link State Database`. Each router maintains its own `Link State Database` which is essentially a map from router IP address and the neighborhood description which is originated by the corresponding router. The shortest path algorithm is ran over this database.

Besides the components introduced above, you have to develop a terminal for the router. When you start the router program, the terminal interface (command line based) should popup, and it allows the user to input following commands:

- detect [IP Address]: output the routing path from this router to the destination router which is identified by [IP Address].
- attach [IP Address]: establish a connection to the remote router which is identified by [IP Address]
- disconnect [IP Address]: disconnect the connection between this router and the remote router which is identified by [IP Address]
- start: start this router and initialize the database synchronization process
- quit: disconnect all connections to others and quit the program

The above commands all trigger the Link State Database update and synchronization, which we will introduce in the next section.

Link State Database Synchronization and Update

Shortest Path Finding

Evaluation