Network and protocols

Lecture 13.5:
Operating Systems and Networks
Behzad Bordbar

Routing example

<table>
<thead>
<tr>
<th>Routing from A</th>
<th>To</th>
<th>Link</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>local</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routing from B</th>
<th>To</th>
<th>Link</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>local</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

RIP routing algorithm

Variables: Ti local table, Tr table received.
Send: Each t seconds or when Ti changes, send Ti on each non-faulty outgoing link.
Receive: Whenever a routing table Tr is received on link n:

for all rows Rr in Tr {  
  if (Rr.link != n) {
    Rr.cost = Rr.cost + 1;
    Rr.link = n;
  }  
  if (Rr.destination is not in Ti) add Rr to Ti;
  else for all rows Rl in Ti {
    if (Rr.destination = Rl.destination and (Rr.cost < Rl.cost or Rl.link = n)) Rl = Rr;
    // Rr.cost < Rl.cost : remote node has better route
    // Rl.link = n : remote node is more authoritative
  }
}

Exercise

Suppose that the router A receive a table from link 2 for Routing from E. What would be new

<table>
<thead>
<tr>
<th>Table from E</th>
<th>To</th>
<th>Link</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>local</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

First octet give the class

<table>
<thead>
<tr>
<th>Class</th>
<th>1st octet range</th>
<th>1st octet highbits</th>
<th>No Networks</th>
<th>No hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1-126</td>
<td>0</td>
<td>126 (2^7 - 2)</td>
<td>16,777,214 (2^24 - 2)</td>
</tr>
<tr>
<td>B</td>
<td>128-191</td>
<td>10</td>
<td>16,382 (2^14 - 2)</td>
<td>65,534 (2^16 - 2)</td>
</tr>
<tr>
<td>C</td>
<td>192-223</td>
<td>110</td>
<td>2,097,150 (2^21 - 2)</td>
<td>254 (2^8 - 2)</td>
</tr>
<tr>
<td>D</td>
<td>224-239</td>
<td>1110</td>
<td>multicast</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>240-254</td>
<td>1111</td>
<td>Experiment and research</td>
<td></td>
</tr>
</tbody>
</table>

- first octet can NOT be 127 (kept for trouble shooting and testing your local system )
- local host: 127.0.0.1
- why 2 is subtracted?

Why private IP address?

- originally for testing and training.
- But we have lots of them? count them:
  - Class A (private) above is 16M addresses
  - Class B (private): 1M
  - Class C (private): 65K
- Some companies assign these reserved addresses for their internal use. On the firewall they use Network Address Translation (NAT) to extend the range of addresses used in IPv4
- How does NAT work?
**Subnet**

- Process of dividing a large network to smaller interacting networks to increase efficiency and manageability.
- IP addresses are hierarchical by nature (Network ID and host ID, but only at two layers). Allows create multilayers.

**Other reasons for using subnets**

- Security: protect parts differently
- Organisational and division of jobs: different department
- Political: we want to be independent.

**How to use mask subnet address?**

- Look up table? **NO**
- Do you know Bitwise “and” operation? Two important case:
  - If mask 255 corresponding part of the IP address repeated
  - If mask 0 corresponding part of the IP address is set to zero
- Suppose that 181.92.56.5 is IP address of a computer. The network has mask 255.255.200.0. What is the subnet address?
  - IP 181 92 56 5
  - Mask 255 255 200 0
  - Subnet 181 92 8 0
- Why 8?
  - 56 = (0011 1000)
  - 200 = (1100 1000)
  - 0000 1000 which is 8

**Exercise**

- Consider a C class address 193.171.120.0. How can you divide the addresses to four subnets using a mask?
- Notice if you use mask is 255.255.255.0, you can play with 8 bits 0000 0000 your subnet addresses.
- Choose the first two bits for subnet 1100 0000 (or the first three bits 1110 0000).
- To do so I can use the subnet mask of 255.255.255.192.
- Why 192?
  - Cause 1100 0000 (base 2) = 128 + 64 (base 10) = 192 (base 10)
- How many subnet I will have this case?
  - Answer: 10... 01... 11... 00... so the 254 addresses are divided into four subnets and within each I can have private address. Put a router in between them.
- Use a subnet calculator www.subnet-calculator.com/

**Back to future**

- A couple of slides about programming aspect of TCP and UDP so that you have more time for your 2nd assignment.
- We will study theory of TCP and UDP next week.

---

**Communication via message passing**

- Port 6789

See [https://docs.oracle.com/javase/tutorial/networking/sockets/](https://docs.oracle.com/javase/tutorial/networking/sockets/) 
Do this tutorial?

**In the next example...**

- TCP Client
  - Makes connection, sends a request and receives a reply
- TCP Server
  - Makes a connection for each client and then echoes the client’s request
TCP client example

```java
public class TCPClient {
    public static void main(String args[]) {
        // arguments supply message and hostname of destination
        Socket s = null;
        try{
            int serverPort = 7896;
            s = new Socket(args[1], serverPort);
            DataInputStream in = new DataInputStream(s.getInputStream());
            DataOutputStream out = new DataOutputStream(s.getOutputStream());
            out.writeUTF(args[0]);
            // UTF is a string encoding,
            String data = in.readUTF();
            System.out.println("Received: "+ data);
            s.close(); // why do we have to close the socket?
        } catch (UnknownHostException e) {System.out.println("Sock:"+e.getMessage());}
        catch (EOFException e) {System.out.println("EOF:"+e.getMessage());}
        catch (IOException e) {System.out.println("IO:"+e.getMessage());}
    }
    finally {if (s!=null) try {s.close();} catch (IOException e)…..}
}
```

TCP server example

```java
public class TCPServer {
    public static void main(String args[]) {
        try{
            int serverPort = 7896;
            ServerSocket listenSocket = new ServerSocket(serverPort);
            while(true) {
                Socket clientSocket = listenSocket.accept();
                Connection c = new Connection(clientSocket);
                //see next slide for Connection
            }
        } catch(IOException e) {System.out.println("Listen :"+e.getMessage());}
    }
}
```

TCP server example ctd

```java
class Connection extends Thread {
    //why thread?
    DataInputStream in;
    DataOutputStream out;
    Socket clientSocket;
    public Connection(Socket aClientSocket) {
        try {
            clientSocket = aClientSocket;
            in = new DataInputStream(clientSocket.getInputStream());
            out = new DataOutputStream(clientSocket.getOutputStream());
            this.start();
        } catch(IOException e) {System.out.println("Connection:"+e.getMessage());}
    }
    public void run(){
        try {
            // an echo server
            String data = in.readUTF();
            out.writeUTF(data);
        } catch(EOFException e) {System.out.println("EOF:"+e.getMessage());}
        catch(IOException e) {System.out.println("IO:"+e.getMessage());}
    }
    finally {try {clientSocket.close();} catch (IOException e)…..}
}
```