THE UNIVERSITY OF SYDNEY

COMP3221 Lab 3

Routing and Communicating

The goal of this lab is to apply a routing algorithm and to connect to a remote machine on the network.

Exercise 1: Router Information Protocol (RIP)

Execute (by hand) the RIP protocol on the communication graph represented in Figure 1 where nodes represent routers and edges represent communication links between routers.

Consider initially that the routing tables are empty (containing only node-local information). Given that no failures occur (neither message losses, nor node failures), write down the final routing table each router obtains after convergence of the protocol.

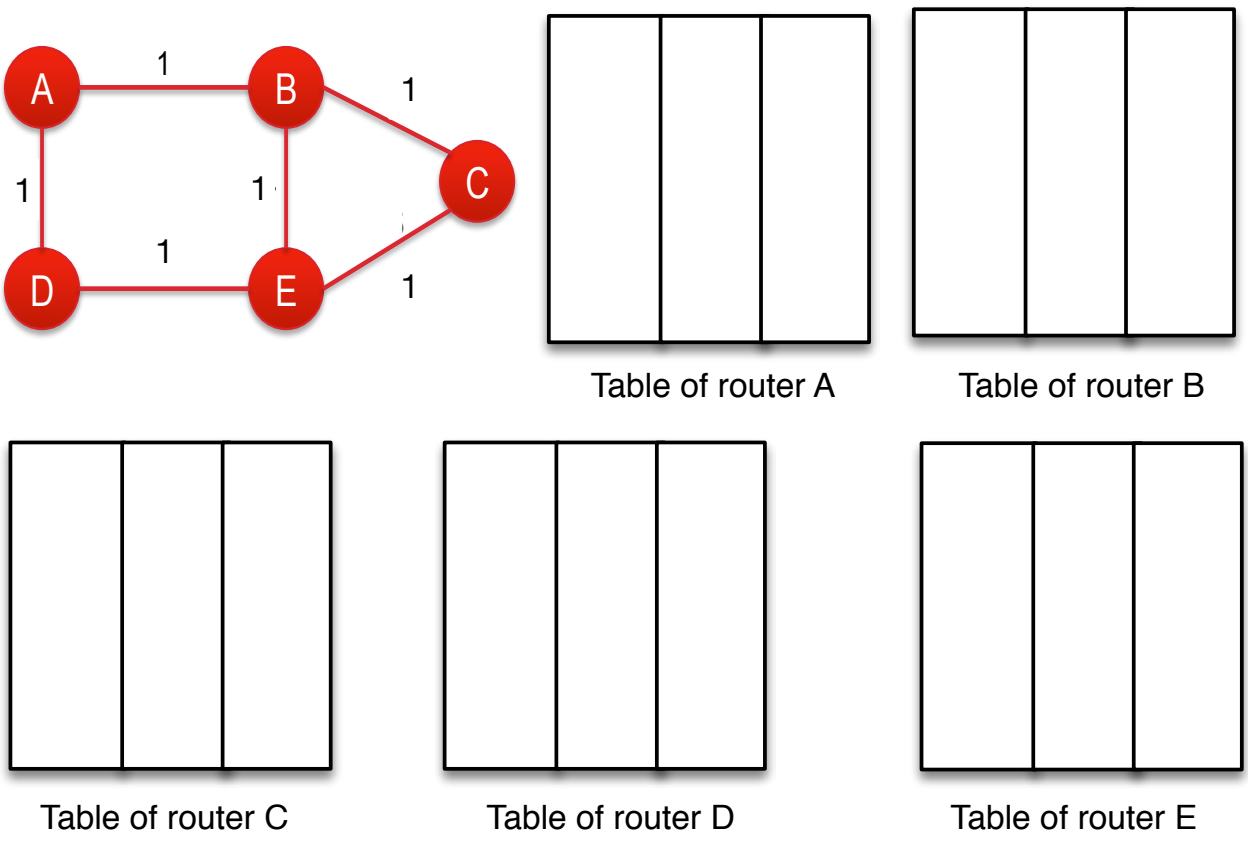

Figure 1: A communication graph with routers A, B, C, D, E.

What happens if the link A-B fails?

Duration: 15 min

Exercise 2: Dijkstra's Shortest Path Algorithm

Complete (by hand) the Table 1 and find the shortest path from node u to each other node in the network following Dijkstra's algorithm. Link costs are given next to each edge. In Table 1, $D(v)$ represents the current cost of path from source to destination v and $p(v)$ represents predecessor node along path from source to v.

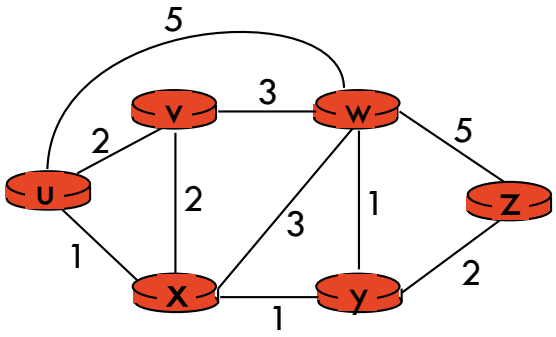

Figure 2: A communication graph with routers u, v, w, x, y, z.

Table 1: Working table for Dijkstra's shortest path algorithm

Step	Working node	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0						
1						
2						
3						
4						
5						
Result						

What is the shortest-path from node u to node z ?

Duration: 10 min

Exercise 3: Path calculation in large scale networks

Now, let's see how to calculate shortest paths programatically in a large scale network. Efficient implementations for many commonly known graph algorithms are readily available. For this task, we use Python **`NetworkX`** graph library[1]. Installation instructions for NetworkX can be found at `https://networkx.github.io/documentation/stable/install.html`.

Create the same network in Figure 2 using the sample code below;

[1] `https://networkx.github.io`

```
import networkx as nx;

G = nx.Graph()
G.add_edge('u','v',weight=2)
G.add_edge('u','w',weight=5)
// complete all edges
```

Now, find (and verify with the result of Exercise 2) the shortest path from node u to node z using the NetworkX shortest path algorithm **`shortest_path`**[2].

What is the total cost of the shortest path ?

What is the shortest path if all edge costs are equal to 1 ?

Duration: 20 min

Exercise 4: Talking to an SMTP server

The second part consists in understanding the interaction between a client and a server. Your machine will play the role of a client communicating with the "Simple Mail Transfer Protocol" (SMTP) server whose hostname is **`mail.usyd.edu.au`**.

The language used in this communication is defined by the SMTP protocol specified in the RFCs. SMTP is associated a specific port, number 25, so that the server expects to hear the client using this specific language that sends a request to this port.

To initiate a connection on port 25 with **`mail.usyd.edu.au`**, use the terminal emulation protocol for TCP/IP through the **`telnet`** command as follow:

```
$ telnet mail.usyd.edu.au 25
```

The server should answer something along these lines if the connection is successful.

```
    Trying 129.78.8.1...
    Connected to mail.usyd.edu.au.
    Escape character is '^]'.
    220 staff.cs.usyd.edu.au. V1.4 ready at Mon, 02 Jul 2012 10:05:29 +1000
```

Try telling him **`HELO hostname`** with any hostname and observe the response from the server. Leave the connection open and continue with the next question.

Duration: 5 min

Exercise 5: Asking for a layered service

Now that the connection is opened, the next step is to ask the SMTP server to send an email. Try to send an email from your email account at the School of IT to yourself, use a different recipient address if you have one.

`MAIL FROM:` is used to indicate the sender of the message. Use angle brackets to encapsulate your

[2] https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.shortest_paths.generic.shortest_path.html

email, and terminate by pressing the "enter" key, as follows:

```
MAIL FROM:<your_login_name@it.usyd.edu.au>
```

If the server accepts the address by returning code 250, then tell him the recipient address.

```
RCPT TO:<your_other_address@if.you.have.one>
```

If the server accepts the address by returning code 250, then you can start giving the data to be sent by typing:

```
DATA
```

Make sure that the server answers you how to indicate that you are done writing your data: "end with a '.' on a line". The last thing to do (next question below) is to write a structured message with the appropriate fields so that the email client of your recipient, once connecting to its receiving mail server (IMAP), will nicely represent this information on the screen of his computer.

Duration: 5 min

Exercise 6: Formating a message

The email information itself is represented in a format specified in another RFC. First, tell the server the format of the content (use the MIME format version 1.0):

```
MIME-Version: 1.0
```

Specify the correct UTC hour with +1000 indicating that Sydney time is 10 hours 00 minutes later.

```
Date: Mon, 15 Aug 2012 15:05:00 +1000
```

Specify your information, the recipient information, the subject and body of the message, and that this is a text message followed by an empty line followed by a dot.

```
From: Your name <your_login_name@it.usyd.edu.au>
To: You <your_other_address@if.you.have.one>
Subject: Testing SMTP

The body of my message...
...spans multiple lines.
Content-Type: text/plain;
.
```

Check whether you successfully received the message by using your web browser. Repeat questions 4 and 5 with the following fields:

```
From: Satoshi Nakamoto <satoshi@nakamoto.com>
To: You <your_login_name@it.usyd.edu.au>
Subject: COMP3221 Lab 3!
```

What do you see? Finally, terminates the connection with:

```
QUIT
```

Duration: 10 min