CS6040 – Router Architectures and Algorithms
Lab 1: Circuit Switching

July-Nov. 2021 Semester, Prof. Krishna Sivalingam
Due date: Aug. 25, 11PM, On IITM Moodle (`courses.iitm.ac.in`)
Updated with example table entries: Aug. 21, 2021

Assigned: Aug. 13, 2021

The purpose of this lab experiment is to understand Circuit Switching concepts, through a basic implementation. In this system, the system can provide higher bitrate support (in multiples of 64 Kbps).

1 Inputs

The command line will specify the following parameters:

```
% ./routing -top topologyfile -conn connectionsfile -rt routingtablefile \
 -ft forwardingfile -path pathsfile
```

- The topology file that contains on the first line: NodeCount (N), Edgecount (E) in the network.

  Nodes are numbered 0 through $N-1$.

  On each subsequent line, there are four numbers, where the first two integers represent the two endpoints of a bi-directional link; the third integer denotes the link's propagation delay (in *ms*), the fourth integer denotes the link's capacity; this is to be intrepreted as a T1 or a T3 link; thus, only two values are valid (1 and 3). Note that a T1 link has 24 slots and a T3 link has 672 slots.

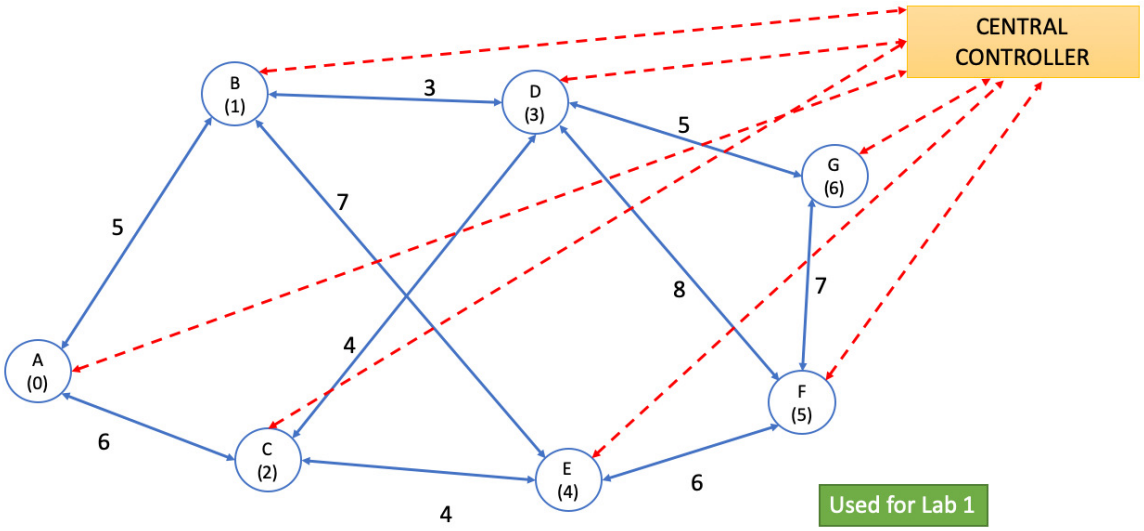

Figure 1: Example Network Topology.

An example file, corresponding to the graph in Week2 Slides (Slide No. 13) and shown in Fig. 1, will look as follows:

```
7 10
0 1 5 3
0 2 6 3
1 3 3 1
1 4 7 3
2 3 4 3
2 4 4 1
3 5 8 3
3 6 5 1
4 5 6 3
5 6 7 1
```

Note that nodes $A, B, C, D, \ldots, G$ in the graph are respectively denoted by $0, 1, 2, 3, \ldots, 6$ above.

- The connections file that contains on the first line: Number of Connection Requests (R).

  Connections are implicitly numbered 0 through $R-1$.

  On each subsequent line, there are 3 numbers:

  - First two integers represent the source and destination node of a *unidirectional connection*,
  - The third integer denotes the connection's requested bitrate, in slots per TDM frame. Note that there may be several connections between the same source-destination pair.

  An example input file corresponding to the above topology will look as follows:

```
8
0 2 24
0 3 20
0 6 16
0 5 4
1 5 120
2 1 24
3 4 76
4 0 12
```

2 Processing

Routing: The program will first determine the shortest cost paths for all node-pairs, using the path propagation delay metric.

If you use any publicly available algorithm implementation, you must specify the same in your README file.

Connections: The program will then process the specified set of connection requests.

For each connection request, the program will attempt to find if sufficient resources are available along the shortest path. Note that the number of slots (S_r) requested by each connection request r is greater than 1. A connection request can be accepted only if there are S_r **contiguous** slots available along each link of the path. Assume that slots are allotted from the lowest available slot number.

If it is available, the call will be set up and the forwarding tables will be updated at the intermediate switches; otherwise, the program will print an error message that specifies: the connection request details; the nodes along the shortest path; the capacity requested for this connection; and the available capacity along the links of the shortest path.

3 Output Files

Routing: The *routingtablefile* will contain the routing table information for all the nodes. For each network node, the corresponding routing table displayed with the following fields (sample entries are given for the topology shown in Fig. 1:

Routing Table for Node 0		
Destination Node	Path (from Source to Dest)	End-to-end Path Delay
0	{0}	0
1	{0, 1}	5
2	{0, 2}	6
3	{0, 1, 3}	8
4	{0, 2, 4}	10
5	{0, 1, 3, 5}	16
6	{0, 1, 3, 6}	13

Routing Table for Node 1		
Destination Node	Path (from Source to Dest)	End-to-end Path Delay
0	{1, 0}	5
1	{1}	0
2	{1, 3, 2}	7
3	{1, 3}	3
4	{1, 4}	7
5	{1, 3, 5}	11
6	{1, 3, 6}	8

. . . and so on.

Forwarding: The *forwardingfile* will contain the forwarding table information for all the nodes. For each network node, the corresponding forwarding table for all established connections will be displayed with the fields, as shown in the example below, for the set of connections given earlier.

Forwarding Table for Node 0			
Node ID of Incoming Port	Starting Slot Number	Node ID of Outgoing Port	Starting Slot Number
-	-	2	0
-	-	1	0
-	-	1	20
2	24	-	-

Forwarding Table for Node 1			
Node ID of Incoming Port	Starting Slot Number	Node ID of Outgoing Port	Starting Slot Number
0	0	3	0
0	20	3	20

Note that instead of interface/port identifiers, we are using the identifiers of the incoming port and the outgoing port.

Paths: The output *pathsfile* will first contain one line that has two integers: the total number of requested connections and the total number of admitted connections.

Then, for each connection that is admitted into the network, the output format is as follows:

Connection ID	Source ID	Destination ID	Slots	Path	Starting Slot Number List	PathCost
0	0	2	24	{0, 2}	{0, -}	6
1	0	3	20	{0, 1, 3}	{0, 0, -}	8
3	0	5	4	{0, 1, 3, 5}	{20, 20, 0, -}	16
7	4	0	12	{4, 2, 0}	{0, 24, -}	10

The above list will be sorted based on the connection ID.

For example, consider the request number 0 between node 0 and 2 with bitrate of 24 slots. The shortest path between the two nodes is the direct link $\{0,2\}$. Assuming that slots are allotted from the lowest available slot number, the output line will look like:

```
0   0    2   24  {0, 2}  {0, -}    6
```

Next, consider the request number 1 between node 0 and 3 with bitrate of 20 slots. The shortest path between the two nodes is: $\{0,1,3\}$. Since the first 20 slots are available on both the links, they are allotted. The output line will look like:

```
1    0    3   20  {0, 1, 3}  {0, 0, -}    8
```

Next, consider the request number 2 between node 0 and 6 with bitrate of 16 slots. The shortest path between the two nodes is: $\{0,1,3,6\}$. There are sufficient free slots on link $(0,1)$ which is a T3 link and has 672 slots. However, the link $(1,3)$ is a T1 link with only 24 slots of which 20 are currently alloted to request 1. Therefore, this request cannot be granted and is dropped.

Next, consider the request number 3 between node 0 and 5 with bitrate of 4 slots. The shortest path between the two nodes is: $\{0,1,3,5\}$. There are sufficient free slots on link $(0,1)$ which is a T3 link and has 672 slots. The link $(1,3)$ is a T1 link, of which 20 are currently alloted to request 1 and there are 4 slots still available. Therefore, this request can be granted. The output line will look like:

```
3    0    5   4  {0, 1, 3, 5}  {20, 20, 0, -}    16
```

4 What to Submit on IITM Moodle

A single tar.gz file containing:

- Source files, compiled executable
- Example Input files used and Corresponding Output Files generated
- Makefile
- README File that explains how to compile and run the program; whether your programs works correctly or whether there are any known bugs/errors in your program.

5 Grading

- Routing: 25 points
- Forwarding and Connections Setup: 60 points
- Viva Voce: 15 points

6 Policies

- This is an INDIVIDUAL assignment. Please refer to first day handout (on Moodle) regarding penalties for any form of academic dishonesty, plagiarism, etc. There should be no downloaded code.
- Software for checking plagiarism the code will be used.
- Please start on the assignment right away – you can start working on implementing the input file parsing, routing algorithm implementation, etc.
- The program can be written in C/C++/Java/Python.
- Example input files will be provided for checking. However, for grading purposes, other input files may be used.