Advanced Computer Networks

Assignment 2 – Virtual Circuit Switching
Assignment given on: 18th Sep 2020
Due Date: 2nd Oct 2020, on Moodle
(1% Penalty per 24-hour period)
Team: Work as a team of two

Assignment 2 – Virtual Circuit Switching Application

The objective of this assignment is to understand link-disjoint routing algorithms and Virtual Circuit Switching concepts.

1. Inputs

The command line will specify the following parameters:

```
$ ./routing -top topologyfile -conn connectionsfile -rt routingtablefile -ft forwardingfile -path pathsfile -flag hop|dist|relb|degree -p 0|1
```

- The topology file that contains on the first line: NodeCout(N), Edgecount (E) in the network. Nodes are numbered 0 through N -1

  On each subsequent line, there are five numbers, where the first two integers represent the two endpoints of a bi-directional link; the third integer denotes the link's delay (in ms), the fourth integer denotes the link's capacity (in Mbps); the fifth value denotes the link's average reliability (in range 0 to 1).

- 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 5 numbers, where the first two integers represent the source and destination node of a *unidirectional connection*, and the remaining 3 integers denote the connection's stated capacity (in Mbps), i.e. a connection request *i* specifies the requested bandwidth using three integers: (b^i_{min}; b^i_{ave}; b^i_{max}), which respectively specify the minimum, average and maximum bandwidth needed. Note that there may be several connections between the same source-destination pair.

2. Processing

The program will first determine a pair of shortest-sum-cost link-disjoint paths for all the node pairs, using either hop, distance(delay), reliability or degree metric. You have to use **Suurballe algorithm** to find the two disjoint paths for all the nodes and to find the shortest path between two nodes, make use of **Dijkstra's algorithm**. For reference, you can go through: *Suurballe JW, Disjoint paths in a network, Networks, vol. 4, pp. 125145, 1974.*

The command line parameter will specify the metric choice. If reliability metric is selected, then the most reliable path(s) will be determined. If degree metric is selected, then the cost of a link from node ***s*** to node ***t***, the cost will be the degree of node ***t***.

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

Optimistic Approach: This is used if -p command-line argument has value 0. Let C_l denote the total capacity of a link ***l***. Let

$$b^i_{equiv} = \min[b^i_{max}, b^i_{ave} + 0.25 * (b^i_{max} - b^i_{min})].$$

A connection is admitted if the following condition is met, along each link of the path selected for connection i:

$$b^i_{equiv} <= (C_l - \sum^n_{j=1} b^j_{equiv})$$

where ***n*** denotes the number of existing connections sharing a given link (along the path selected for the given connection) and ***j*** denotes the index of a connection assigned on this link.

Next, for each source-destination pair, select the two link-disjoint paths. If adequate bandwidth is not available along the first shortest-cost path, then the network attempts to set up the connection on the second shortest-cost paths. If this also fails, then the connection is NOT admitted, i.e. it fails to meet the admission control test.

Once an available path is identified, the connection will be set up along the path. This primarily involves setting up link-unique VC IDs (Virtual Circuit ID) for a given connection request along all links of the path, and updating the corresponding **forwarding tables** at each intermediate node along the path.

Pessimistic Approach: This is used if *-p* command-line argument has value 1. Let C_l denote the total capacity of a link ***l***. A connection is admitted if the following condition is met, along each link of the path selected for connection ***i***:

$$b^i_{max} <= (C_l - \sum^n_{j=1} b^j_{max})$$

where ***j*** denotes the index of a connection assigned on a given link along the path selected for the given connection. Repeat the above for the pessimistic approach.

3. Outputs

The ***routingtablefile*** will contain the routing table information for all the nodes. For each network done, the corresponding routing table (i.e. the two link-disjoint paths from the given node to all other nodes) displayed with the following fields:

Destination Node	Path(from Source to Destination)	Path Delay	Path Cost

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 following fields:

Node ID of Incoming Port	VC ID	Node ID of Outgoing Port	VC ID

The ***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:

Conn. ID	Source	Destination	Path	VC ID List	Path Cost

The above list will be sorted based on connection ID.

4. What to Submit

A single zip file containing:

- Source files, compiled executable
- Example Input files used and Corresponding Output Files generated
- Makefile
- README File
- Technical Report: For the 14-node NSFNET and 24-node ARPANET topologies, compare the blocking probability for your own set of link capacities/traffic requests (100, 200 and 300 requests).
- The TAs will evaluate on a different set of input files.

5. Grading (Evaluation Criteria)

- Routing Setup: 20%
- Connections Setup: 50 %
- Demo: 20%
- Report: 10 %

Submission Details:

1. Please read the questions carefully and complete it.

2. Make a directory with name <Your_Roll_Number> and copy your all program (source code) and output file to that folder.

3. You can implement using either C/C++/Python or Java.

4. Please copy your output and paste it to related program at the end of source code (please comment it)/ if necessary you can take screen shot name it with its question number and put it in a same folder.

5. Test well before submission. Follow some coding style uniformly. Provide proper comments in your code.

6. Submit only through moodle and well in advance. Any hiccups in the moodle/Internet at the last minute is never acceptable as an excuse for late submission. Submissions through email

will be ignored.

7. Please zip your folder and submit to moodle within 2-10-2020 (23:55 PM)