COMP 3331/9331: Computer Networks & Applications

Programming Assignment: Routing Performance Analysis

Due Date: Friday 31 Oct 2014, 11:59 pm (Week 13)

Change Log

a. Version 1.0 released on September 16th, 2014

b. Version 2.0 released on October 20th, 2014

The due date was updated.
Some typos have been fixed.

C. Version 3.0 released on October 27th, 2014

The missing part of the tabulated summary has been fixed.

***NOTE**: Please make sure that you read notes on plagiarism before commencing work on this assignment. Despite our warnings, every session we detect several cases of plagiarism. Please note that copying from past assignments (or outsourcing on web sites) gets reported to us via various tools that we use and falls under plagiarism.*

Assessment: 60 marks (towards the Practical component)

Goal: The purpose of this assignment is to gain insights into the performance of different network-layer routing algorithms. Your task is to develop a program that will evaluate the performance of 3 different routing protocols over a *virtual circuit network* (i.e. a network with a connection-based network layer, unlike the Internet) as well as a *virtual packet network* (like the connection-less network layer of the Internet).

Learning Objectives: On completing this assignment you will gain sufficient expertise in the following skills:

- Understanding and developing routing protocols
- Performance analysis of some routing protocols over virtual circuit and virtual packet networks

Specification:

You will implement the following program:

RoutingPerformance

- The *RoutingPerformance* program will accept the following command line arguments:
  - NETWORK_SCHEME: this argument specifies the network type that will be evaluated. You program should implement two different network types, called here as virtual circuit network and virtual packet network. Accordingly, the argument will

take one of the following values: CIRCUIT and PACKET corresponding to those network types, respectively.
  - ROUTING_SCHEME: this specifies the routing scheme that will be evaluated. Your program should implement 3 routing protocols, which are essentially variants of Dijkstra's algorithm, as explained later in the specification. This argument will take on one of the following values: SHP, SDP and LLP corresponding to the 3 routing protocols: Shortest Hop Path (SHP), Shortest Delay Path (SDP) and Least Loaded Path (LLP), respectively.
  - TOPOLOGY_FILE: this file contains the network topology specification.
  - WORKLOAD_FILE: this file contains the *virtual connection* requests in the network.
  - PACKET_RATE: this positive integer value shows the number of *packets* per second which will be sent in each virtual connection.

- The network topology that will be used for the evaluation will be specified in the TOPOLOGY_FILE (the third argument). We will be using both a virtual circuit network and a virtual packet network to evaluate the performance of the routing protocols. **Note that a routing protocol is required in both networks to determine the path between the source and destination of a virtual circuit during the connection.** The main difference between the virtual circuit network and virtual packet network is how end-to-end path is determined. The former is similar to the circuit switching networks and the path is determined only at the connection establishment phase and such a path will be used for all packet transmission through the connection. In other words, one virtual connection in the virtual circuit network follows the same *virtual circuit*[1] for transmitting all packets. However, a virtual connection in the virtual packet network uses the routing protocol to determine the path for each packet independently. Thus, such virtual connection needs to invoke the routing protocol *N* times, where *N* is the number of packets transmitted through the connection. In other words, one virtual connection in the virtual packet network has *N* virtual circuits for transmitting *N* packets.
- A simple example of a network topology specification for a network with 4 routers (routers will be referred to as nodes in the rest of the specification), labelled A, B, C and D is as follows:

```
A B 10 19
A C 15 20
B C 20 20
B D 30 70
C D 8 20
```

This example network topology has 5 links connecting the 4 nodes as shown in the figure below (Figure 1 on the next page). Each line in this file defines a point-to-point link. For example, the first line specifies a link from node A to node B, with a one-way propagation delay of 10 milliseconds and a capacity that can accommodate up to 19 simultaneous virtual circuits at any given time. All links are assumed to be bi-directional, with identical propagation delay in both directions (hence, the ordering of the names of the two endpoints of the point-to-point link does not matter, i.e. the first line in the example above could have been replaced with "B A 10 19"). Further, each connection in either virtual circuit network or virtual packet network is also assumed to be bi-

[1] Here we assume that a virtual circuit is a path suggested by a routing protocol for sending data between a particular source and destination.

directional and consumes unit resource (i.e. a resource of 1). There will be at most one direct link between any two nodes in the graph. You may assume that the topology will form a connected graph i.e. there will be no isolated nodes. The first task for your program is to read in the topology file and construct a suitable internal representation of the network topology, using an appropriate data structure. You may want to consult standard data structures textbooks for standard representations of undirected graphs, which would be an appropriate way to model the network. You may assume that all node names are single upper-case alphabetic characters (i.e., a maximum of 26 nodes from A to Z), all propagation delays *d* are positive integers (*0 < d < 200*) and expressed in milliseconds, and all link capacities *C* are positive integers (*0 < C < 100*) and indicate the number of virtual connections that can be supported by a link. A more complex network topology, which you may use for testing your code is available on the assignment webpage (*topology.txt*).

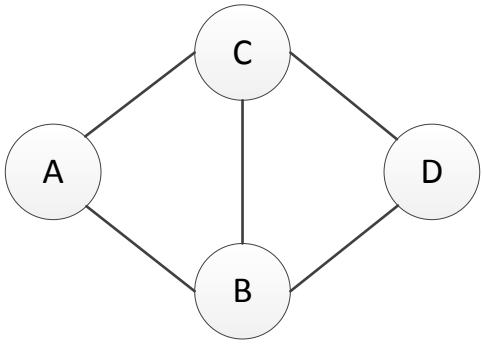

Figure 1: Topology for the above example

- The network is initially empty, i.e., there are no virtual circuits established. The virtual circuit requests that arrive in the network will be specified in the virtual connection requests in WORKLOAD_FILE (fourth argument), which is ordered by time. The next step for your program is to read in the arriving virtual connection request workload (from the file), one request at a time, in timestamp order, and attempt to establish the virtual circuits in the network according to the routing algorithm in use (routing algorithms are explained later). The virtual connection workload for the network is specified in a simple four-column format as follows:

  0.123456 A D 12.527453
  7.249811 B C 48.129653
  8.975344 B D 6.124743
  10.915432 A C 106.724339
  15.817634 B C 37.634569

  Each line of this file describes one virtual connection request. The first column specifies the time (in seconds) at which the connection is established. You may assume that time starts at 0 seconds and that time values will be represented up to 6 decimal digits (i.e. microseconds). The second column specifies the originator (source node) for the request, and the third column specifies the recipient (destination node) for the request. The final column specifies the time duration for which this virtual connection remains active for sending the packets. As an example, the first line in the above file contains a connection request that originated at time 0.123456 seconds for a virtual circuit to be established from node A to node D. This request will be active for duration of 12.527453 seconds. A more comprehensive virtual connection workload, which you may use for testing your

program, is available on the assignment webpage in the *workload.txt* file. (Note that, this workload is consistent with the topology specified in *topology.txt*)

- We assume that all virtual connections have the same value for packet rate (fifth argument), which determines the number of packets per second in a connection. For example, if the packet rate value is 2 packets per second, the first virtual connection in the above example will have 26 packets (rounded value of 2 * 12.527453), the first packet starts at exactly the connection establishment time 0.123456 and finishes at 0.123456+0.5, the second one starts at time 0.123456+0.5 and finishes at 0.123456+1, …, the last packet starts at time 0.123456+12.5 and finishes at 0.123456+12.527453. Clearly, for the case of virtual circuit network, you will create only one virtual circuit using the routing protocol for sending all packets. In other words, you don't need to consider the packet rate as each virtual connection will be assigned to only one virtual circuit with the same specification presented for the connection (including above four features). However, in the case of the virtual packet network, you need to extract the starting time and duration of each packet according to the packet rate value as you must invoke the routing protocol to find an appropriate path for each packet. For example in the first virtual connection of the above list and when the packet rate value is 2, we have 26 packets and accordingly we have 26 virtual circuits. The first virtual circuit starts at time 0.123456 and finishes at 0.123456+0.5, the second one starts at time 0.123456+0.5 and finishes at 0.123456+1,…, the last virtual circuit starts at time 0.123456+12.5 and finishes at 0.123456+12.527453.

- For each virtual circuit request in the above workload, your program must use the specified routing algorithm to determine if the circuit can be established. To be more specific, your program must select the "best" route depending on the routing protocol in use (routing protocols are explained in the next bullet point) from the source to the destination of the circuit and then determine if there is sufficient capacity along each link of this end-to-end path to accommodate the circuit. Recall that each virtual circuit uses unit capacity on each link. You may assume that the routing decision for each request can be made in zero processing time. A virtual circuit that is successfully established must be counted as such, and the network resources associated with that circuit marked as "busy" for the duration of the circuit. For this purpose, assume that each circuit consumes exactly one unit (i.e., one "circuit") of link capacity on each link. Of course, recall that virtual circuits are bidirectional. When a circuit terminates, the resources (i.e. unit capacity) along the individual links of the end-to-end path are released, and become available for subsequent circuits that are established in the network. A circuit request that is not routed successfully is said to be "blocked". A blocked request means that there is no physical path currently available from the specified source to the specified destination (according to the routing protocol in use). Such requests (packets) must be immediately discarded from further consideration by your program. Your program must count and report the number of blocked requests (packets). A list of performance metrics that your program must measure and report is specified later in the specification.

- Your program should implement the following three routing protocols to determine the least cost path between the source and destination. The first argument in the argument list (as explained earlier) will determine which protocol should be used for a particular run of your program. The algorithms are essentially variants of Dijkstra's algorithm (i.e. link-state routing) with each scheme using a different "cost" metric.

(1) Shortest Hop Path (SHP): This algorithm tries to find the shortest path currently available from the source to the destination, where the length of a path refers to the number of hops (i.e. links) traversed. In essence this is Dijkstra's algorithm with the cost of each link set to 1. Note that, this algorithm ignores the delay and load associated with each link.

(2) Shortest Delay Path (SDP): This algorithm tries to find the shortest path currently available from the source to the destination, where the length of the path refers to the cumulative propagation delay for traversing the chosen links in the path. In other words, this is Dijktra's algorithm with the cost of each link set to the propagation delay. Recall that the network topology file specifies the delay along each link in the network. Note that, this algorithm ignores the number of hops and the load associated with each link.

(3) Least Loaded Path (LLP): This algorithm tries to find the least loaded path currently available from the source to the destination, where the load of a path is defined to be the maximum load on any link in the path. The load on a link is defined as the ratio of its current number of active virtual circuits to the capacity, C, of that link for carrying virtual circuits. Note that, this algorithm ignores the number of hops and the delay associated with each link. There are two main differences between LLP and the other two algorithms (SHP and SDP). Firstly, the path cost in LLP is not an additive function, as is the case with the other two algorithms (in SHP and SDP the cost of the path is simply the sum of the cost along each individual link that constitutes the path). Secondly, link costs (i.e. the link load) change with time, whereas in both SHP and SDP the link costs are static over the entire lifetime.

- For all the above algorithms, whenever ties occur, they should be broken arbitrarily. In other words, if a particular routing algorithm determines that there are two or more paths between the source and destination with exactly similar costs then the algorithm should choose one of these paths randomly.

- Your program MUST NOT look for alternate paths between the source and destination of a virtual circuit request in case the path selected by the routing protocol results in a blocked request. Simply count this as a blocked request (packet) and move on to the next one.

- Your program will need to keep track of time. You may assume that time starts at 0. Recall that the workload file specifies the time at which each virtual connection request originates in the network and the duration of each virtual circuit (provided it is established). Clearly, in the case of virtual packet network, you must extract the starting time and duration for each packet and then make a virtual circuit request per each packet. Note that when the connection request is divided into the packets in the virtual packet network, your program must be careful about the order of the starting time of the packets of different connection requests. As discussed earlier the routing decision for each circuit request can be processed in zero time. Remember to free up the resources dedicated for a circuit once the duration of the circuit elapses.

- Your program should maintain the following statistics:
  - The total number of virtual circuit requests.
  - The total number of packets.
  - The number (and percentage) of successfully routed packets.

  - The number (and percentage) of blocked packets.
  - The average number of hops (i.e. links) consumed per successfully routed circuit.
  - The average source-to-destination cumulative propagation delay per successfully routed circuit.

  Note that in the case of virtual circuit network, your program make one circuit for all packets of a requested connection, thus the program should cascade the results of the virtual circuit over all packets in the corresponding connection. For example, in this case, if a connection has been blocked, it means that all packets of the connection are blocked.

- The following illustrates an example initiation of your program:
  java RoutingPerformance CIRCUIT SHP topology.txt workload.txt 2 (for JAVA)
  RoutingPerformance CIRCUIT SHP topology.txt workload.txt 2 (for C)

- Once your program has read through the entire workload file and finished processing all virtual circuit requests, it should output all of the above statistics to the terminal and then exit. A sample output from one run of the program is as follows:

```
total number of virtual circuit requests: 200

total number of packets: 4589

number of successfully routed packets: 3654

percentage of successfully routed packets: 79.63

number of blocked packets: 935

percentage of blocked packets: 20.37

average number of hops per circuit: 5.42

average cumulative propagation delay per circuit: 120.54
```

  Your output format **MUST** be exactly same as the above sample output. You need only to change the performance metrics values according to your program results. Also, the maximum number of decimal places for real performance values must be set to 2. Moreover, you **MUST** write the above output format in the standard output.

  We will wait for reasonable amount of time (3 - 4 minutes) for your output to appear (with the provided *workload.txt* file). If you observe that your program is taking longer than 4 minutes to execute the workload file then please indicate this in your report so that we know that your program is not in an infinite loop.

Additional Notes

- You may choose to work either individually or in a group of two. If you form a group, please send an email with the names/student-ids of your group members to the class account (cs3331@cse.unsw.edu.au) by Friday, **26th October 2014**, 11:59 pm. We neither are able to help you choose a group member nor would be accepting more than 2 people in a group. However, you are encouraged to use message board. Marking criteria will remain the same. Unless advised otherwise, contribution from each member will be considered equal. If there is any problem with group members not cooperating, this must

be reported to the LIC immediately as dealing with group problems post submission will be hard to manage. Please keep a log of your contributions/meetings.

- Non-programming assignment is allowed as an exception to non-CSE and non-EE&T students who do not have experience with programming (e.g: Mechatronics). Note that non-CSE students are encouraged to attempt the programming assignment. Check details about this on the alternate assignment webpage. You must send an email to class account indicating that you are opting for non-programming assignment by Friday, **26th September 2014**, 11:59 pm. If we DO NOT receive e-mail from such students, we will assume that they have opted for the programming assignment. Students cannot change their decision past the deadline.

- While you are encouraged to adopt good programming practices, your coding style is **NOT** subject to marking.

- The programs will be tested on CSE Linux machines. So please make sure that your program runs correctly on these machines. This is especially important if you plan to develop and test the programs on your personal computers (which may possibly use a different OS or version). We are unable to accommodate the request to have you demonstrate it on your laptop/desktop machines. Please plan ahead for any porting issues.

- **Tips on how to get started:**

    - Focus first on making sure that your program can read in and model the network topology. The 4-node example discussed earlier is a good one to start with.
    - Next focus on getting ONE routing algorithm working. The Shortest Hop Path (SHP) is the easiest one, since it entails implementing Dijkstra's algorithm using a link cost of 1 for each hop traversed. Spend a lot of time testing your Dijkstra's algorithm to make sure that it is working properly. This will definitely pay off in the long run.
    - The Shortest Delay Path (SDP) is a simple variant of SHP and will involve a simple modification (i.e. changing the link cost) to your implementation of SHP. The Least Loaded Path (LLP) will require reasonable modifications but is still relatively easy to implement.
    - Test your routing algorithm(s) with a SMALL network topology and a SMALL virtual circuit workload file. Links with very low capacity (e.g. 1) are useful in initial testing. Make sure that the routing is working, and that your program is able to compute the required statistics.
    - Extend your program for the virtual packet network case and repeat all tests for all three routing algorithms.
    - Once you are fairly confident that your program is working correctly for a small topology, and then move on to the provided sample topology. It may be useful to initially test out the workload file incrementally (e.g.: first 10 requests, first 50 requests, first 100 requests and so on).

- **Language and Platform**: You are free to use either C or C++ or JAVA or Python to implement this assignment (you can choose only one of these languages for the entire assignment, not a combination of them). Your assignment will be tested on the **Linux** Platform. Make sure you develop your code under Linux (check the version on CSE machines).

- **Submission:** We will inform you about the details of submission 1 week before the deadline (check the notice board closer to submission date). You really need only one file: **RoutingPerformance.c** (or **RoutingPerformance.java**). If you are using C and if you are going to use any other files besides these two such as header files or other .c files then you will have to submit a **Makefile** along with your code. This is because we need to know how to resolve the dependencies among all the files that you have provided. If you are using Java and have multiple files, a makefile will not be necessary if your program doesn't need any CLASSPATH settings (javac *.java usually work here). In addition you should submit a report, **report.pdf** (no more than **3 pages**) including the following 3 items:
    - An explanation on the data structure(s) that you have used for the internal representation of the network topology.
    - A tabulated summary of the comparison of the performance metrics for the 3 routing protocols over the virtual circuit network. This comparison MUST be carried out using the provided topology and workload files (topology.txt and workload.txt) and **the packet rate must be set by 1**. Your table must contain 8 columns (one each for the 8 performance metrics) and 3 rows (one each for the 3 routing protocols for the virtual circuit network).
    - An analysis of the results, i.e., where possible you MUST provide reasons for the performance results that you observe. In particular comment on the reason behind the differences in the following metrics for the 3 protocols: percentage of blocked requests, average number of hops and the average propagation delay.
    - Performance evaluation of the virtual packet network with respect to the packet rate and different routing protocols. To this end, you need to collect the values of three performance metrics percentage of successfully routed packets, average number of hops per circuit, average cumulative propagation delay per circuit by varying the packet rates in the range of 1 - 5 and changing the routing protocols over the topology.txt and workload.txt files. Then, you will create three plots corresponding to each performance metric. In each plot, the x-axis is the different values of packet rates and the y-axis is the performance values. Also, in each plot you draw three line charts corresponding to each routing protocol. Finally, try to describe the plots and the reason of the results by one paragraph for each plot.

- **Late Submission Penalty**: Late penalty will be applied as follows:
    - 1 day after deadline: 10% reduction
    - 2 days after deadline: 20% reduction
    - 3 days after deadline: 30% reduction
    - 4 days after deadline: 40% reduction
    - 5 or more days late: NOT accepted

  NOTE: The above penalty is applied to your final total. For example, if you submit your assignment 1 day late and your score on the assignment is 60, then your final mark will be 60 – 6 (10% penalty) = 54.

- **Plagiarism**: You are to write all of the code for this assignment **yourself**. All source codes are subject to strict checks for plagiarism, via highly sophisticated plagiarism detection software. These checks may include comparison with available code from Internet sites and ***assignments from previous semesters***. In addition, each submission will be checked against all other submissions of the current semester. Please note that we take this matter quite seriously. The LIC will decide on appropriate penalty for detected cases of plagiarism. The most likely penalty would be to reduce the assignment

mark to **ZERO** and reported to school plagiarism register. We are aware that a lot of learning takes place in student conversations, and don't wish to discourage those. However, it is important, for both those helping others and those being helped, not to provide/accept any programming language code in writing, as this is apt to be used exactly as is, and lead to plagiarism penalties for both the supplier and the copier of the codes. Write something on a piece of paper, by all means, but tear it up/take it away when the discussion is over.

- **Forum Use:** You are free to discuss (and are in fact strongly encouraged to do so) issues relevant to the assignment on the course forum. However, refrain from posting large code-fragments on the forum. Students will be heavily penalised for doing so.

Sequence of Operation for Marking

The following shows the sequence of events that will be involved in marking your assignment:

1) We will first test your program with the network topology and workload files that are provided on the assignment webpage (*topology.txt* and *workload.txt*). All 3 routing protocols and two networks will be tested.

2) We will repeat the tests for a different topology and associated workload file. Note that, the format of the topology and workload files will be consistent with those provided on the webpage.

3) Finally, your report will be marked.

Marking Guidelines:

The marking will consist of the following sequence of tests:

Test 1: Correct Compilation

Correct compilation of all files: **2 mark**

Test 2: Shortest Hop Path (SHP) Test

We will first evaluate the SHP routing protocol for both networks. We will first verify the performance for the provided topology and workload files and then test with a different topology and associated workload: **12 marks**

Test 3: Shortest Delay Path (SDP) Test

We will next evaluate the SDP routing protocol for both networks. The same topology and workload files as in the SHP test will be used for this evaluation: **14 marks**

Test 4: Least Loaded Path (LLP) Test

Next we will evaluate the LLP routing protocol for both networks. The same topology and workload files as in the SHP test will be used for this evaluation: **18 marks**

Test 5: Report

Your report is worth **14 marks**. Your report will be evaluated as follows:

- Description of the data structure used to create the internal representation of the network topology: **2 marks**
- Tabulated comparison of the performance metrics of the 3 routing protocols for the provided topology and workload files: **2 marks**
- Explanation of the performance results observed: **2 marks**
- Plots for performance evaluation of the virtual packet network: **4 marks**
- Explanation of the plots: **4 marks**

IMPORTANT NOTE: We will not read through your code to evaluate your coding style, etc. For assignments that fail to execute **all** of the above tests, we will be unable to award you a substantial mark.

Consultation:

Please note that Mr. Mohsen Rezvani is in-charge of this assignment and all queries should be directed to him via cs3331@cse.unsw.edu.au with subject heading *Assignment Query*. Additionally, Mr. Rezvani will also provide weekly Consultation. The consultation timing and venue would be 4:00PM -5:00PM on Friday (from week 9 to week 13) and 4:00PM - 5:00PM on Tuesday (week 11 to week 13) in Room 508, Building K-17. The consultation sessions will be started from Friday, 26th September 2014.