CPSC 441: Computer Networks
Assignment 4
due April 7, 2017

Important Notes

- This an individual assignment. You should submit your own work.
- The assignment is due 11:59pm on April 7, 2017 (if you wish to submit the written part as hard copy, you may do so by making use of the physical dropbox in Maths Science building, 1st floor. However, deadline in this case will be 4pm on April 7, 2017)
- Start the assignment early and avoid procrastination.
- Assignment has got two parts, programming and written part. The programming part has to be implemented in Java.

Objective (Programming Part - 85 Marks)

The objective of the programming part is to implement Dijkstra's link state routing algorithm. By completing this part, you will learn how routers broadcast link state message among other nodes in the network and find the least cost distance to any router in the network.

Note: The terms "router" and "node" are used interchangeably

Definitions

- Node label and id: This is used to represent a router in the network. Each node in the network is given a unique label and id. The node label is represented using alphabets in English language (upper case), starting from A. The node id is represented by integers greater than or equal to 0. There is a one-to-one mapping between node id and label. For example, A is mapped to 0, B is mapped to 1, C is mapped to 2 and so on.

- Link state: Each router broadcast its link state vector to other routers in the network as a link state message. A link state vector of a router provides information about a router's neighbors and corresponding link costs. Link cost to the same router is set as 0. Link cost to those routers which are not neighbors is set as infinity. Hence, link state vector also provides information about non-neighbors of a node. Furthers, link costs to each node are arranged in increasing order of node id. You may denote infinity as 999. For example, link state vector of node A in the sample network topology (Figure 1) is [0 1 999 999 999]. Similarly, for B it is [1 0 2 2 1], for C it is [999 2 0 999 4], for D it is [999 2 999 0 5] and for E it is [999 1 4 5 0]

- Previous node: This is the node that lies just before the destination node in the least cost path from a source node to a destination node.

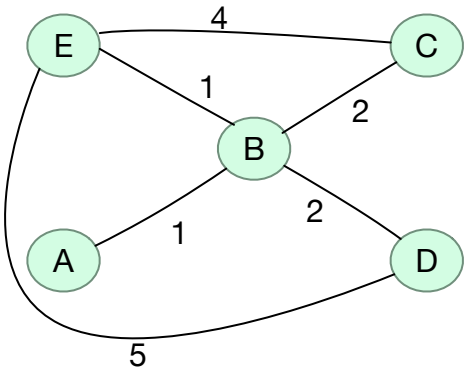

Figure 1: Sample Network Topology

Overview

You need to implement a program called *Router* which emulate the behavior of a router, in particular (1) broadcast link state message to other routers in the network and (2) compute the least cost distance to any router in the network. You need to run multiple instances of the *Router* program, each emulating exactly one router in the network. Hence, if there are 5 routers in the network, 5 instances of the *Router* program need to be run. **The routers exchange link state messages over UDP transport protocol**. To keep it simple, we will ONLY consider networks with nodes less than or equal to 10.

A node, apart from its details, only knows the total number of nodes in the network, its neighboring nodes and corresponding link costs. Hence, a node can ONLY send its link state vector as link state message to its neighboring nodes. The neighboring nodes need to propagate this message further in the network so that all nodes in the network receives this message. You can come up with different approaches to implement *broadcast* in such a scenario. ANY approach is acceptable and additional overhead due to the approach selected can be ignored for this assignment. You may use your creativity to come up with a suitable approach. Few options are:

- Approach-1: Each node will send its link state message to its neighboring nodes. Every neighboring node that receives link state message will record and then forward it to its neighboring nodes which again repeat this step. This results in link state messages "living" in the network forever. Further, each message received at a node creates 1 or more new messages.

- Approach-2: Similar to approach-1 except that each message also contains a "count" along with link state vector and other information. Every router, before forwarding

the message, will also decrement the "count" by one. When count becomes zero, a router can stop forwarding the message. This helps to remove every link state message created from the network at some point in time. Since number of nodes is restricted to 10 or less, you can easily come with a suitable initial value for "count".

- Approach-3: Each router will send its link state message to its neighboring nodes. Every neighboring node that receives link state message will record and then forward it to its neighboring nodes (which again repeats this step) that haven't forwarded the same message. Each message may require an ID along with the link state vector and other information. This approach will ensure that messages do not "live" forever in the network and also avoid unnecessary message forwards.

You may come with any other approach as well.

Protocol Details and Implementation Logic

Your router program should implement the following *three operations in parallel*:

- Send link state message: Each router should send its link state vector to its neighbors every 1000 ms. A message containing link state vector and other details is called a link state message.
- Receive link state message: Each router should listen for incoming link state message, record the same for its use and then forward it to its neighboring nodes depending on the broadcast algorithm.
- Compute route information: Each router should compute the least cost distance to any node in the network using Dijkstra's algorithm and print the route information. This needs to be done every 10000 ms. If a router hasn't received link state vectors from all nodes, it should defer from running Dijkstra's algorithm and printing the route information. The exact format of the route information is discussed later in the assignment description.

Please note that different routers start executing at different times. For example, router 0 **starts at time** t**, router** 1 **at** $t + 10$**, router** 2 **at** $t + 13$ **etc.**

Assumptions

- The underlying graph is connected, i.e, there at least one path from any node to any other node in the network.
- Network topology does not change during a single run of the router program.

Program skeleton, Source Codes, Run Time Arguments and Output Format

The skeleton code, `Router.java`, is provided to you. Please read comments in the skeleton code. You are allowed to edit the signature of the given constructor and other methods provided in the source file. You may also introduce additional classes, methods and constructors.

Your *Router* program **should output routing information in the following format**:
`<RouterID> <Distance> <Prev RouterID>`

For example, route information at node C for the given network topology is shown below:

RouterID	Distance	Prev RouterID
0	3	1
1	2	2
2	0	2
3	4	1
4	3	1

Your implementation should include appropriate exception handling code to deal with various exceptions that could be thrown in the program. For simplicity, run all instances of the router program in a single machine.

Your router program **should be run as**:

`java Router <routerid> <routerport> <configfile>`

`<routerid>`: is the router id

`<routerport>`: is the port number used by the router to send and receive link state messages over UDP and

`<configfile>`: is the configuration containing information about total number of routers, neighboring routers and corresponding link costs.

The format of the configuarion is as follows:

`<Total number of nodes> (first line)`
`<neighbor node label> <space> <neighbor node id> <space> <link cost> <space> <neighbor node port number> (a line per neighbor)`

The configuration file for node C is shown below as an example (please see uploaded configuration files of all nodes on D2L, for the given sample network):

5
B 1 2 5001
E 4 4 5004

Please note that configuration file for each node is different and WOULD NOT contain entry for nodes which are not neighbors. You need to keep this in mind when you design and implement your broadcast algorithm.

Note: For easiness and to save your time, message implementation to exchange link

state vector among routers is provided, `LinkState.java`. *Read Javadoc documentation of the class on how to use it. You are allowed to edit* `LinkState.java` *depending on your broadcast algorithm. You may also discard this message implementation and develop your own.*

Restrictions

- You are not allowed to change the format of the configuration file.
- You are not allowed to change the number, type and order of run time arguments of the *Router* program.

Objective (Written Part - 15 Marks)

The objective of the written part is to get familiarized with *Wireshark*, a protocol analyzer, and do a simple packet trace analysis.

Overview

All questions are based on the packet trace (`cpsc441-assignment-4-trace`) provided as part of the assignment. The trace contains DHCP message exchanges between a laptop (client) and a DHCP server, as part of obtaining IP address for the laptop. Marks for each question is given in square brackets.
Note: To filter DHCP messages, use the keyword "bootp" in Wireshark.

Questions

1. The broadcast IP address used by the client to send DHCP discover/request messages? [1]
2. What is the destination MAC address for the DHCP discover/request messages? [1]
3. What is the destination IP address of DHCP offer/ack messages? [1]
4. What is the destination MAC address for the DHCP offer/ack messages? [1]
5. What is the IP address of DHCP server? [1]
6. What is the MAC address of client and DHCP server? [2]
7. What is the IP address offered to the client by the the DHCP server? [1]
8. What is the subnet address to which client is connected? What is the broadcast address for this subnet? [2]

9. List any three information apart from offered IP address that is provided by DHCP server to the client? [3]

10. How do you think a client can identify its DHCP offer/ack messages from the DHCP server (or vice versa, DHCP server can separate DHCP discover/request messages from multiple clients) if two or more clients try to get connected to the same subnet and obtain IP address simultaneously? [2]