18-756 – Project 3 – Label Switching

Overview

You have been given some Java code that represents some ATM routers on a network. You job is to complete the code to allow communication over the ATM network. Our communication scheme is not exactly like ATM as a full implementation would take months, however, we do have a basic label distribution and label switching scheme (In fact, our scheme is more like RSVP). You may implement the code however you wish, however, ensure that you output conforms to the grading section. You have been provided with an example network with which you can test your code but the grading scheme will use a more complex network; so, ensure sure your code works in more complex networks, not just with the example network. Many of the methods needed to complete the project are already implemented, but you may use your own methods if you decide they are required.

Lecture notes 4, slides 50-54 will be useful. As will lecture notes 7, slides 32 and 61.

Grading

The output results from your code will be worth 100% of your project grade. We will test your code against a test vector and check that your outputs match the correct values. The test vectors will be more complex than the example code you have been given, so you are encouraged to try your own scenarios to ensure your code is robust.

The grading criteria will contain the following test:

1. [10%] Can the code support a network topology different from that given in the example file?
2. [20%] Does the code allow VCs to be established in a network topology different from that given in the example file?
3. [20%] Does the code allow cells to be transmitted through the network from any source to any destination (given that a VC has been established between those two points)?
4. [10%] Does the code allow a connection to be torn down (i.e. cells can no longer be sent down a VC that used to exist)?
5. [5%] Does the code correctly implement RED at an ingress router?
6. [5%] Does the code correctly implement RED at an intermediate router?
7. [5%] Does the code correctly implement Tail drop at an ingress router?
8. [5%] Does the code correctly implement Tail drop at an intermediate router?
9. [5%] Does the code correctly implement PPD at an ingress router?
10. [5%] Does the code correctly implement PPD at an intermediate router?
11. [5%] Does the code correctly implement EPD at an ingress router?
12. [5%] Does the code correctly implement EPD at an intermediate router?

The skeleton code that you have been provided with includes methods to output to the console events that occur within your router. You **must** use these methods so that we can see that your implementation does in fact do what you claim.
On November 4th (a day after the deadline of the project submission), the test vector used for the project grading will be posted on Blackboard. This will provide you with another learning opportunity to find errors and mistakes you may have made in either your algorithm or codes.

As stated in the course policy, any late submission will get a zero grade without any exceptions. Please make sure you submit the coursework in advance in order to avoid any unfortunate events such as power or network outage.

Deliverables

1) All the Java source files only (src folder in eclipse). Delete the .class files from the working folder. (In eclipse this is the bin folder which at the time of submitting needs to be empty)

Zip the working folder into a ZIP file only. Name the file as *andrewid*_18756_project3.zip
Upload the zip file in the blackboard under Assignments > Coursework > Project 3

Hints

You are free to implement the code however you choose. You may find it convenient to place lots of your router logic in the ATMRouter.recieveCell method. If-else statements and the cell.getData().startsWith(“”) commands may also be useful. Remember to call the console output methods whenever you send and receive commands so that you will receive full points. If you invest an hour or two understanding the flow of the code before you attempt to answer the questions, the actual implementation is very straightforward.

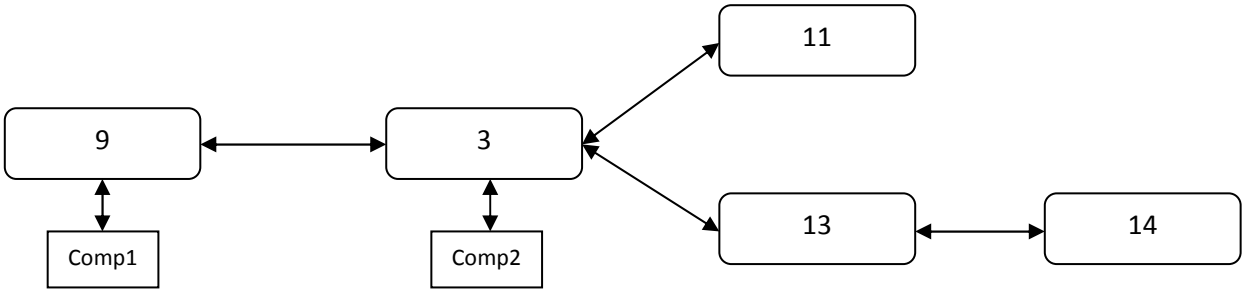
Questions

1. [50 pts] Currently the code does not allow for cells to be sent across the network; each ATM router can only send cells to routers to which it is directly connected. To allow cells to be sent across the network you must implement the code that will forward VC setup requests across the network. To do this, each ATM routers will need to keep a mapping of VC's and also be able to forward the setup request to the next router. See lecture notes 7, slide 61 for details. In our implementation we only need to setup one connection at a time, other setup requests are told to wait and keep retrying until the router is free to service their request.

The messages that you network needs to support are:

Command	**Explanation**
setup *dest_address*	Tells a router that a downstream router wants to setup a connection
call proceeding	To be sent to the downstream router if the router that received a **setup** request is going to forward the request. A router may send a **setup *dest_address*** command as soon as it has sent its **connect *vc_number*** command for the previous setup request (if present and if you so choose to not wait another tock), it does not have to wait until it has received the **connect ack** command
wait *dest_address*	To be sent to the downstream router if the router that received a **setup** request is already trying to setup a connection. Upon receiving a **wait dest_address** command, the receiver sends back a **setup *dest_address*** command to the sender of the **wait** command
connect *vc_number*	Sent in reply to a downstream router once the VC number for the router-to-router traffic has been decided
connect ack	Sent in reply to the **connect *vc_number*** to inform the upstream router that the connection is setup correctly
end *vc_number*	Tells a router that a downstream router has decided they no longer require the use of a certain VC
end ack	Sent in reply to an **end *vc_number*** to confirm that the VC has been torn down

In our current implementation computers can only connect to one VC, and they send cells to a destination router. In real life a computer could be connected to multiple VC, and could communicate with multiple computers or routers. Once the connection is established, the computer will remember what VC to send its cells on in order for them to reach their destination. At the destination (end) router, remove the cells from the network and call the .cellDeadEnd method

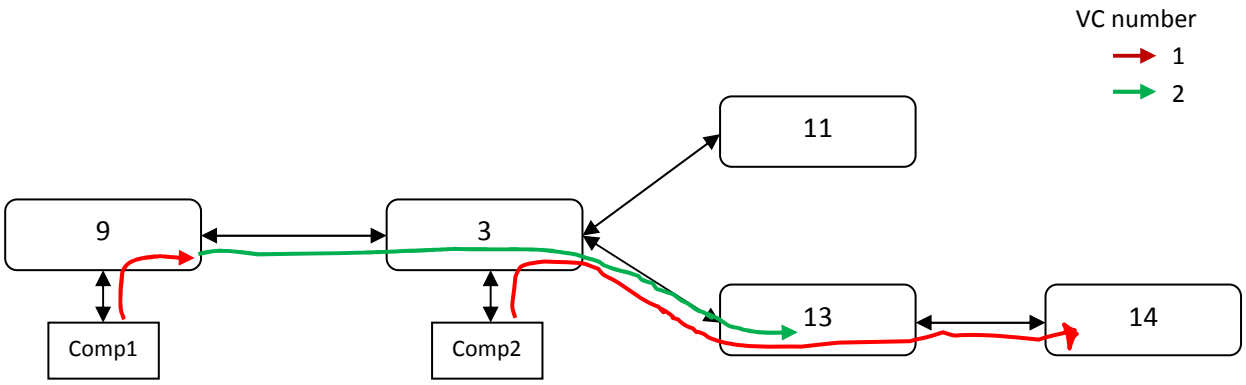


In the above network, if we setup a connection from Comp1 to 13, and Comp2 to 14 we should have output similar to that given below. Your commands may not follow the exact timing, but they will be very similar. Also, your trace IDs will be different – they are only there to help you see the flow of cells.

```
** TIME = 0 **
SND SETUP: Computer 1 sent a connect setup 62496
SND SETUP: Computer 2 sent a connect setup 83256
** TIME = 1 **
REC SETUP: Router 9 received a setup message 62496
SND CALLPRO: Router 9 sent a call proceeding message 40587
SND SETUP: Router 9 sent a setup 62496
REC SETUP: Router 3 received a setup message 83256
SND CALLPRO: Router 3 sent a call proceeding message 74698
SND SETUP: Router 3 sent a setup 83256
** TIME = 2 **
SND WAIT: Router 3 sent a wait message 74699
REC SETUP: Router 13 received a setup message 83256
SND CALLPRO: Router 13 sent a call proceeding message 64919
SND SETUP: Router 13 sent a setup 83256
REC CALLPRO: Computer 1 received a call proceeding message 40587
REC CALLPRO: Computer 2 received a call proceeding message 74698
** TIME = 3 **
REC WAIT: Router 9 received a wait message 74699
SND SETUP: Router 9 sent a setup 40588
REC CALLPRO: Router 3 received a call proceeding message 64919
REC SETUP: Router 14 received a setup message 83256
SND CALLPRO: Router 14 sent a call proceeding message 70064
Trace (ATMRouter): First free VC = 1
SND CONN: Router 14 sent a connect message 70065
** TIME = 4 **
SND WAIT: Router 3 sent a wait message 74700
REC CALLPRO: Router 13 received a call proceeding message 70064
REC CONN: Router 13 received a connect message 70065
SND CONN: Router 13 sent a connect message 64920
SND CALLACK: Router 13 sent a connect ack message 70065
** TIME = 5 **
REC WAIT: Router 9 received a wait message 74700
SND SETUP: Router 9 sent a setup 40589
REC CONN: Router 3 received a connect message 64920
SND CONN: Router 3 sent a connect message 74701
SND CALLACK: Router 3 sent a connect ack message 64920
REC CALLACK: Router 14 received a connect ack message 64921
```

```
** TIME = 6 **
REC SETUP: Router 3 received a setup message 40589
SND CALLPRO: Router 3 sent a call proceeding message 74703
SND SETUP: Router 3 sent a setup 40589
REC CALLACK: Router 13 received a connect ack message 74702
REC CONN: Computer 2 received a connect message 74701
The connection is setup on VC 1
SND CALLACK: Computer 2 sent a connect ack message 83257
** TIME = 7 **
REC CALLPRO: Router 9 received a call proceeding message 74703
REC CALLACK: Router 3 received a connect ack message 83257
REC SETUP: Router 13 received a setup message 40589
SND CALLPRO: Router 13 sent a call proceeding message 64922
Trace (ATMRouter): First free VC = 2
SND CONN: Router 13 sent a connect message 64923
** TIME = 8 **
REC CALLPRO: Router 3 received a call proceeding message 64922
REC CONN: Router 3 received a connect message 64923
SND CONN: Router 3 sent a connect message 74704
SND CALLACK: Router 3 sent a connect ack message 64923
** TIME = 9 **
REC CONN: Router 9 received a connect message 74704
SND CONN: Router 9 sent a connect message 40590
SND CALLACK: Router 9 sent a connect ack message 74704
REC CALLACK: Router 13 received a connect ack message 74705
** TIME = 10 **
REC CALLACK: Router 3 received a connect ack message 40591
REC CONN: Computer 1 received a connect message 40590
The connection is setup on VC 1
SND CALLACK: Computer 1 sent a connect ack message 62497
** TIME = 11 **
REC CALLACK: Router 9 received a connect ack message 62497
** TIME = 12 **
```

Router 9 receives wait messages as Router 3 is already setting up a connection from Comp2. You can sue the trace numbers to see which router received which cell. You may also notice that the VC numbers chosen are 1 and 2. When the first VC is setup ‘1’ can be used over the entire path, as no router has any VC’s setup. When the setup request from Comp1 finally gets through, VC 2 is used in the routers that already have VC 1 in use, but Router 9 can still choose VC 1 as it is not in use on that router; leaving VC 2 free and ready for use. Router 9 can then change the VC of the outgoing cells to VC 2 for the rest of the network.



2. [10 pts] Now we have created VC's across the network, we also want to be able to tear them down. Implement the **end** and **end ack** commands to allow VC's to be freed up for reuse. When a computer sends a cell on a VC that is not setup the Router should call the cellNoVC method.

3. [10 pts] Currently any cell sent to a router is placed in the output buffer, i.e. the output buffer is infinitely big. Change the .runTailDrop() method in ATMNIC to only allow a maximum number of cells, as defined by maximumBufferCells.

4. [10 pts] Change the .runRED method to randomly drop cells (with an increasing probability as the maximum approaches) once a threshold of startDropAt is reached. This is not a full implementation of the RED algorithm – it is a simplified version. Run the algorithm on the single output buffer and drop cells with an increasing probability as required (raising to a probability of 1 if the buffers maximum size is reached)

5. [10 pts] Change the .runPPD method to drop all cells belonging to the same IP packet if any of the cells belonging to that packet are dropped. Use a method similar to your RED method to decide if a cell is dropped, and drop all subsequent cells for that IP packet if any part of the packet is dropped. You can tell if cells belong to the same IP packet as the cells for the same packet will contain data until a cell with a new IP header arrives.

6. [10 pts] Change the .runEPD method to drop all cells from a packet if any part of that packet will be discarded. You can tell how many cells will arrive for a given packet by its size. Use a method similar to your RED method to decide if any cells belonging to the packet will be dropped.