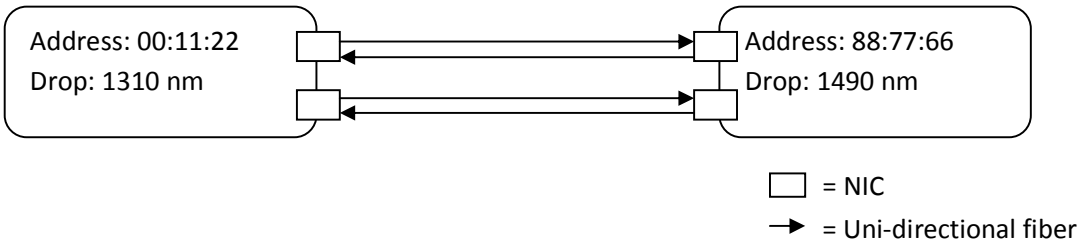
18-756 – Project 1 - SONET

Overview

You have been given some Java code that represents an Optical network. Unfortunately your TA's were so busy reading paper reviews they didn't have time to write all of the code themselves; your job is to complete the Optical network. The SONET network in the code given to you (and described in this document) does not work exactly like the real SONET system; as this would take you around 6 months to implement.

The first thing to do is look at Example.java. This file creates a basic Optical network, consisting of two SONET routers.



Our implementation of a SONET router uses a FDM system to decide where data is sent. Many SONET routers really use TDM and WDM to send data, however our SONET network does not constantly send data, so we are using FDM to make our implementation simpler. Each router has one (or more) drop frequencies. If a SONET router receives a light wave that matches one of its drop frequencies, it removes it from the ring and does 'something' with it. If a SONET router receives a wavelength that is not on one of its drop frequencies it continues to forward it around the ring. Other SONET routers in the network must know the wavelength that the destination router is dropping on to be able to send data to it. These wavelengths are configured manually by the SONET administrator.

Another piece of luck in our SONET network is that exactly one SPE fits in one SONET frame, and our SONET routers don't send SONET frames unless there is data to be sent. In real life, SPE's can be split across multiple frames, and SPE's could contain many other multiplexed signals. However, we are using an STS-1 type signal where the SPE is synchronized with the frame.

When answering the questions, please provide a small description and explanation of your code on the answer sheet. Also, make clear in your source code where you have changed things and comment your source code so that it is clear what you are doing and why. We want to give you points but can only do so if you make it clear you have answered the question.

To complete each question in the coursework you must have completed the question before it, otherwise your code will not function correctly. Make sure your submitted code runs, no matter what questions you reached in your attempt. If the code doesn't run you get a zero. Also, if you do not

complete the entire coursework, please indicate clearly where you answered up to. The later parts of the coursework may 'break' the earlier parts of the coursework as they will fix problems that are in the system. So make sure you have the answers for the first questions before you complete later ones.

Grading

Implementation 80 points: *How well did you implement you answers? Did your code work correctly? Was it implemented efficient? Did you understand and implement what was being asked?*

Report 20 Points: *Was your report clear and concise? Did you report help the TA's understand your work? Did you answer questions satisfactory? Did you explain where and why you added/altered code?*

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). Call the folder 'andrewid_18756_project1' (use **your** AndrewID, not the word 'andrewid')

2) A project report with all the answers and code explanation, named as andrewid_project1_report.doc/pdf, to be placed in the project folder (the folder containing the src and bin folders)

Zip the working folder into a ZIP file only. Name the file as andrewid_18756_project1.zip (again using **your** Andrewid in replace of 'andrewid').

Upload the zip file in the blackboard under Assignments > Coursework > Project 1

Points will be deducted for every step not followed.

Getting started

The following instructions detail the installation of Eclipse in a Windows environment. If you already have an installed version of JRE, please make sure it is updated to the latest version by NOT skipping step 1.

1. Go to http://www.oracle.com/technetwork/java/javase/downloads/index.html to download the latest version of Java Standard Edition (Java SE) JRE. At this time of writing, it should be "Java SE 8". Click on "Download JDK", and choose your platform. Note the architecture of the platform that you choose (either 32-bit or 64-bit), since you will need to download the same architectural version of Eclipse.

2. Install the JDK.

3. Go to http://www.eclipse.org/downloads/ to download the latest version of "Eclipse IDE for Java Developers". Download the version which architecture matches the JRE architecture which you downloaded above.

4. Extract the downloaded Eclipse software, which should be downloaded as a `.zip` file, to your directory of choice. Eclipse is then essentially installed in that directory.

5. Add the "`<Java installation directory>\bin`" to your system PATH environment variable. The directory should look like "`C:\Program Files\Java\jre8\bin`". For more information on how to do this, visit http://www.java.com/en/download/help/path.xml

6. Download and extract the project 1 files onto your local computer. Find the directory "`andrewid_18756_project1`", and note its full system path.

7. Navigate to your Eclipse directory, and execute "`eclipse.exe`". If any errors about Java JRE are reported, then you may have installed Eclipse which has a different architecture from your JRE installation, or you may have not configured your system PATH properly.

8. If Eclipse executes successfully, you should be shown a fancy "Welcome" screen. (You may also be asked to select a default workspace.) This is not what we need, so locate the "Welcome" tab in the top left hand corner of the application and click on the "X" button to close the "Welcome" screen.

9. Go to "File" > "New" > "Java Project".

10. Under project name, enter "`andrewid_18756_project1`", without quotes.

11. Uncheck the option "Use default location", and instead manually set the "Location" field to the full system path of the "`andrewid_18756_project1`" directory in step 6, or browse to locate the path to that directory.

12. Leave everything else in their default settings, and click on "Finish".

13. Under Package Explorer, expand the "`andrewid_18756_project1`" root project.

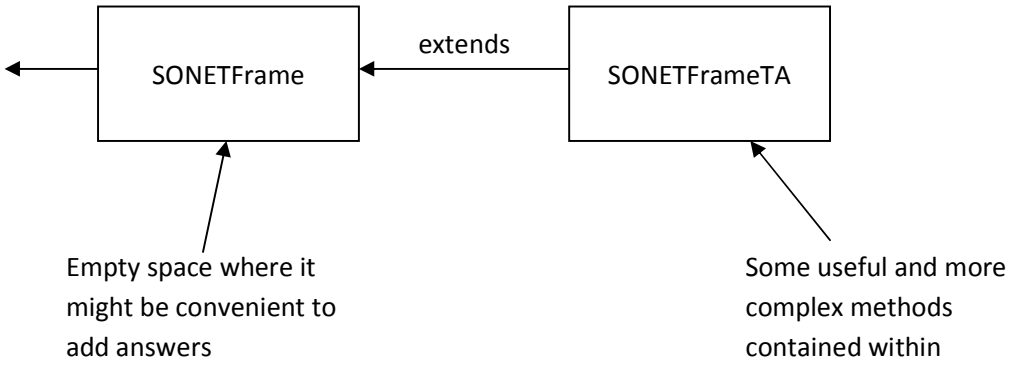
14. Navigate to "src" > "(default package)".

15. Right-click on "ExampleTA.java", and move down to "Run As", and click on "1 Java Application".

16. You should see the text "Setting up two routers" in the console window.

Code setup

The code is setup to help you answer questions, although at first it may seem confusing. There is a TA and an extended type to each object. You are free to edit any of the files you want; however, you may find that the TA objects already have methods that are of use to you and that the extended object has a blank area where you could put your own code without having to edit the larger and more complex TA files.



Last point: **DO NOT HARD CODE**
Our test code will not be the same as your answer code, so hard coding router addresses to go to certain NICs will not work. Use the functions to setup the information during the creation of the network.

Questions

1.a) [**15pt**] The extract below (from ExampleTA.java) tries to send a new SPE to 88:77:66 from 00:11:22

```
router1.source(new SONETFrame(new SPE(0)), 1490);
```

Unfortunately the `receiveFrame` method doesn't contain any code. Fill in the method with the following specifications:

- If a frame is on the routers drop frequency it takes it off of the line (it might be for that router, or has to be forwarded to somewhere off of the ring)
- If a frame is on the routers drop frequency, and is also the routers destination frequency, the frame is forwarded to the `sink(frame, wavelength)` method. In real life the `sink` method would be sending the data to the layer above)
- If a frame is not on the routers drop frequency the frame is forwarded on all interfaces, except the interface the frame was received on (`sendRingFrame()`)

1.b) [**3pt**] Why does the frame arrive at 88:77:66 twice? Why do we want this to happen?

1.c) [**3pt**] What happens if the following code is run (in ExampleTA.java)? Why does this happen? (Make sure you followed the specification in 1.a. You can edit ExampleTA.java for this question, although any of your code in there will be deleted at grading time)

```
router1.source(new SONETFrame(new SPE(0)), 1310);
```

1.d) [**7pt**] You may notice that when you send a frame from 00:11:22 to 88:77:66 the delay is zero.

```
(SONETRouter) 88:77:66 has received a frame at the sink on wavelength
1490 frame delay: 0 SPE delay: 0
```

Edit the OtoOLink (and the frame and SPE objects) to add delay of 5 every time a SPE travels over a link.

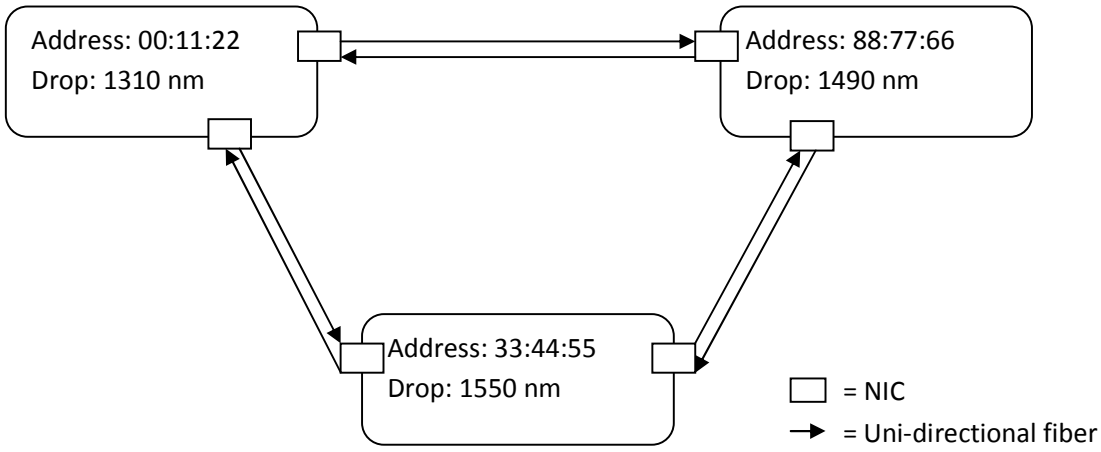
NOTE: SONET frames and SPE's do not have a field for the delay in real life, it is a property of the speed of light. However, since we do not have a 10km optical fiber to send our frames down, we will add a fake delay value to the data we send.

1.e) [**6pt**] You may notice that the second packet has a delay of 10 even though it has only traveled over one link. In real life you don't send the same photon down two different links, you send two photons that are the same. In addition to this, SONET routers strip the frame headers and send add their own when they forward a frame. Change the `sendRingFrame` method to send newly created frames containing a copy (clone) of the SPE object, rather than sending the same frame and SPE object down both links.

Both frames should now have a frame delay and SPE delay of 5.

1.f) [**6pt**] What happens if you send data on a frequency that neither router 1 or router 2 is dropping on? Fix this problem so that the data of unknown frequencies are never sent. Edit your code that sends frames to ensure we have the frequency of the frame being sent in our destinationFrequencies object.

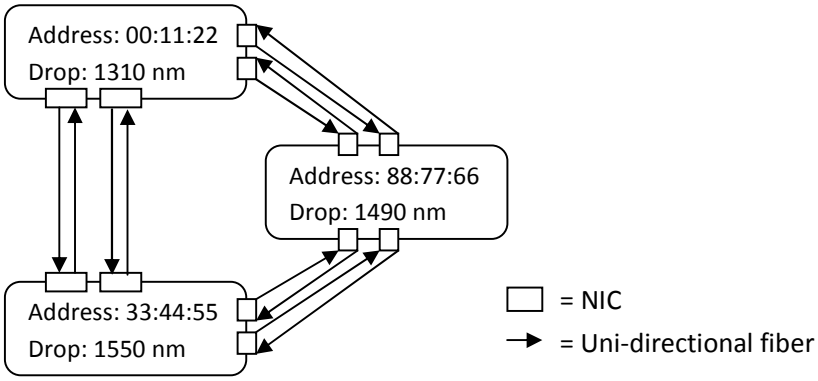
2.a) [**2pt**] Now that we have two routers working, we want to create a SONET ring. Create the topology below in q2a.java. You can copy most of the code from Example.java. Make sure you add the new destination and frequency in each routers routing table.



2.b) [**2pt**] Now, send a new SPE from 00:11:22 to 88:77:66. Why (if you have implemented the code correctly so far) do you get the following output? Explain the delay times and why the packet is received twice.

```
(SONETRouter) 88:77:66 has received a frame at the sink on wavelength 1490
      frame delay: 5 SPE delay: 10
(SONETRouter) 88:77:66 has received a frame at the sink on wavelength 1490
      frame delay: 5 SPE delay: 5
```

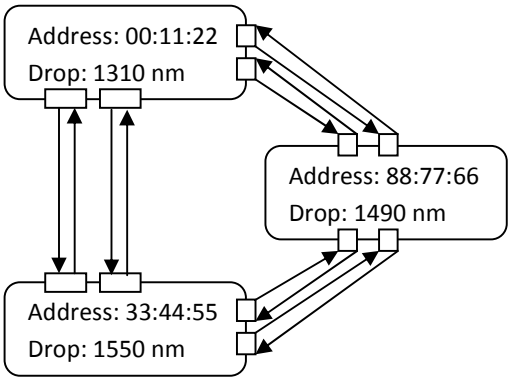
2.c) [**6pt**] We now want to improve our implementation to use BLSR. The first step is to add the extra NICs and links that the routers will need. Change your network to have an additional NIC and link between each of the SONET routers (answer in q2c.java).



Your current implementation will not work as BLSR as you have implemented a UPSR system.

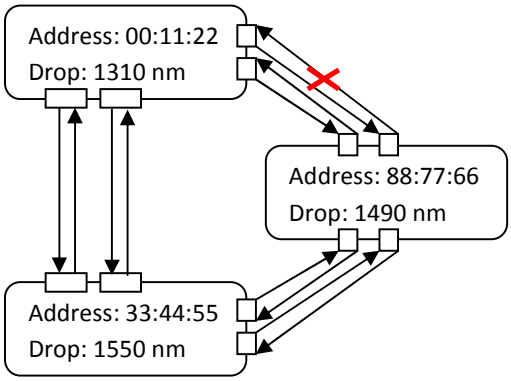
2.d) [**30 pts**] Implement BLSR instead of UPSR in your SONET router. For simplicity, we will assume that the largest size of the ring is 3 SONET routers (this makes things a lot simpler). Your BLSR implementation should send both the working and protection traffic via the shortest path if possible. If not, the working traffic should take the shortest path and the protection traffic the longer path. Finally, if both links are down that provide the shortest path, both the working and protection traffic should take the alternative path.

For example,



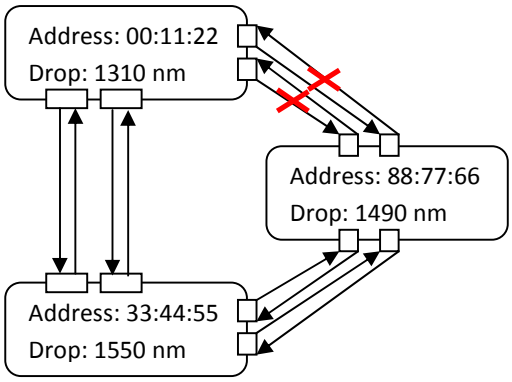
From 00:11:22 to 88:77:66

```
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 5
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 5
```



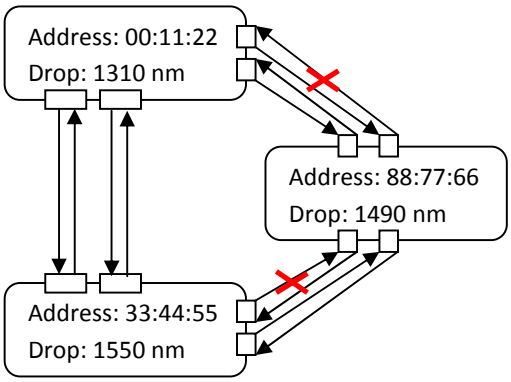
From 00:11:22 to 88:77:66

```
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 5
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 10
```



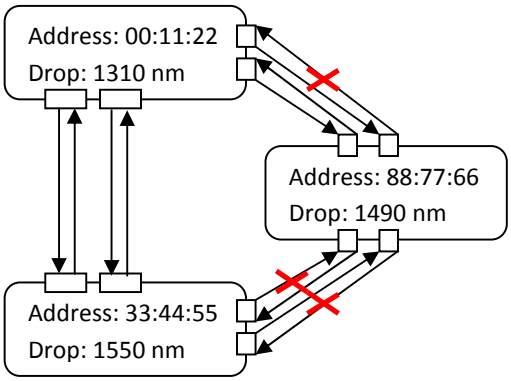
From 00:11:22 to 88:77:66

```
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 10
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 10
```



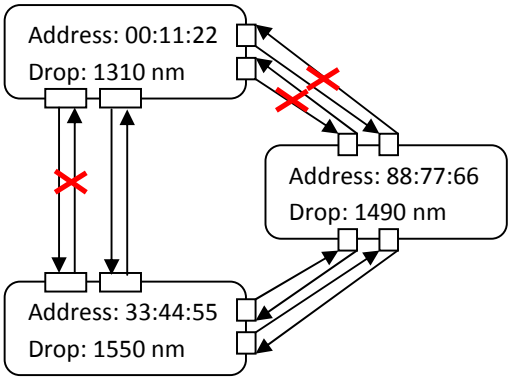
From 00:11:22 to 88:77:66

```
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 5
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 10
```



From 00:11:22 to 88:77:66

```
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 5
```

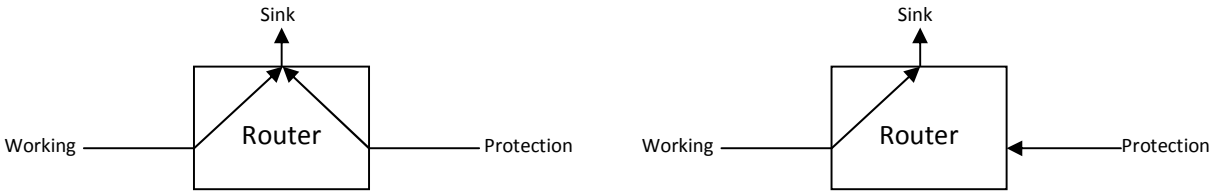


From 00:11:22 to 88:77:66

```
Trace (OpticalNIC):88:77:66 has received a frame at the on 1490 frame delay:
      5 SPE delay: 10
```

You can test how you BLSR is performing by using the `OtoOLink.cutLink` method. Additionally, your network should return to using the shortest path once the link becomes uncut.

There are also some additional factors you should consider. For example, the sink should only receive one copy of the SPE, where as currently you implementation acts like the first picture.



You may want to edit the receiveFrame function to use a preferred NIC for a signal. However, be careful if that link goes down as you need to know to use the protection NIC to listen for the frames.

Summary

1) Take the shortest path if available
2) Working path takes preference over protection path if only one shortest path is available
3) Traffic will start taking the shortest path again if a link that was unavailable becomes available again
4) The router should receive each SPE twice if there are two **full** distinct paths available (see the last example picture, only one SPE arrives)
5) The sink should only receive one SPE and it should be the working path, i.e. the shortest path SPE

You are free to implement the BLSR however you wish as long as it conforms to the instructions above (and is not hardcoded to only work with the exact 3 routers you have created). Including some signaling to indicate when paths should be changed might be the easiest way to complete this question. As we know the network topology (as we create the network) we should know what NICs it is best to send the traffic down (in the configuration of the network – not hardcoded in the router object). Additionally, since this is not really SONET and we don't really have Frames constantly streaming even when there is no data to be sent, you can create a Frame and send it if you need to send data from one router to another (in real life there is always frames being sent so this isn't a problem, you can always send you signaling data – though obviously not down a link that has been cut).