Name: Abhinav Kuruvadi
Andrew ID: akuruvad

18-756 Coursework 1 Answer sheet

Please keep any code change descriptions and explanations (in this document) to under 10 lines – the code comments and structure should allow the TA to easily see what you have done. Also, remember to comment in your source code where you have made changes. Example:
On answer this sheet: "
Ex1)
Explanation: I changed the SONETRouter.receiveFrame to start a fire when it receives a frame. I did this using if(receive==true) and the .setIsOnFire(true)
Code:

```
//Added question 1.h) onFire code below
public SONETRouter.receiveFrame(){
// check that we did receive a frame
If(receive==true){
        // as we have received a frame set the datacenter on fire, and update the isOnFire variable
        this.startFire();
        this.setIsOnFire(true);

        // Do the next thing
        ….
}
}"
```

ANSWERS

1.a)

Explanation: I have implemented the three requirements specified in the Question in three different conditions. I have used the arrays `this.dropFrequency` and `this.destinationFrequencies` to verify if the wavelength is right.

Code:

```
public void receiveFrame(SONETFrame frame, int wavelength, OpticalNICTA nic){

		if ( this.dropFrequency.contains(wavelength) &&
this.destinationFrequencies.get(this.address) == wavelength) {
			System.out.println("Frame reached its destination - " + this.address);
			this.sink(frame, wavelength);
		} else if ( this.dropFrequency.contains(wavelength) &&
!(this.destinationFrequencies.get(this.address) == wavelength) ) {
			System.out.println("Taking the frame off as it exists in the diestination
frequencies.");
		} else {
			System.out.println("Forwarding the frame to all interfaces except the
interface the frame was received on- " + this.address);

			this.sendRingFrame(frame, wavelength, nic);
		}
	}
```

1.b)

Explanation:

The frame(generated by router 00::11::22) arrives twice since the routers, other than the destination router (here router 00:11:22), in the ring also pass the frame back to the ring on their output channels, in this case there is just one more router in the network with both the output channels facing towards the destination router. We want this to happen as the second frame received will serve as a backup from the recovery channel in which data flows in the opposite direction.

1.c)

Explanation:

The frame is received at 00:11:22 only once. This is happening because the frame is created by 00:11:22 and the wavelength of the frequency generated corresponds to the origin of the frame and hence it is consumed by the router 00:11::22 and not forwarded to other channels.

1.d)

Explanation: I have added implemented the addDelay function in SONETFrame and SPE classes. Also, in OtoOLink, I have added code to add the delay of 5 to the frame and the frame's SPE.

Code:

In public class SONETFrame :-

```
public void addDelay(int delay){
		this.delay += delay;
	}
```

In public class SPE :-

```
public void addDelay(int delay){
		this.delay += delay;
	}
```

In Public class OtoOLink :-

```
public void sendData(SONETFrame frame, int wavelength, OpticalNICTA source){


		//Adding the delay to the frames before sending them across to the destination.
		int delay = 5;

		frame.addDelay(delay);
		frame.getSPE().addDelay(delay);

		if(dest==null)
			System.err.println("Error (OtoOLink): You tried to send data down a line that doesn't
go anywhere");
		else if(this.source != source)
			System.err.println("Error (OtoOLink: You tried to send data down a line you are not
connected to");
		else if(this.linkCut)
			System.err.println("Error (OtoOLink: You tried to send data down a line that is
cut");
		else{
			this.dest.receiveData(frame, wavelength);
		}
	}
```

1.e)
Explanation :
I have added the delay in the clone() method of the SPE class to incorporate the delay. Also, in sendRingFrame() method of SONETRouter class, I have cloned the frame before forwarding it to the ring.

Code:

```
public SPE clone(){
		SPE clone = new SPE(this.getVTPointer());
		clone.addDelay(this.delay);
		return clone;
	}
```

```
public void sendRingFrame(SONETFrame frame, int wavelength, OpticalNICTA nic){

		// Check if the router knows a destination for this particular frequency.
		// And Loop through the interfaces sending the frame on interfaces that are on the ring
		// except the one it was received on. Basically what UPSR does

		if ( this.destinationFrequencies.containsValue(wavelength)) {
			for(OpticalNICTA NIC:NICs)
				if(NIC.getIsOnRing() && !NIC.equals(nic)) {
					SONETFrame newFrame = new SONETFrame(frame.getSPE().clone());
					NIC.sendFrame(newFrame, wavelength);
				}
		} else {
			System.out.println("This destination of this frequency is unknown");
		}
	}
```

1.f)

Explanation:

I am verifying in `sendRingFrame()` method if the destination wavelength is present in destinationFrequencies before sending the frame across to the ring.

Code:

```
public void sendRingFrame(SONETFrame frame, int wavelength, OpticalNICTA nic){

		// Check if the router knows a destination for this particular frequency.
		// And Loop through the interfaces sending the frame on interfaces that are on the ring
		// except the one it was received on. Basically what UPSR does

		if ( this.destinationFrequencies.containsValue(wavelength)) {
			for(OpticalNICTA NIC:NICs)
				if(NIC.getIsOnRing() && !NIC.equals(nic)) {
					SONETFrame newFrame = new SONETFrame(frame.getSPE().clone());
					NIC.sendFrame(newFrame, wavelength);
				}
		} else {
			System.out.println("This destination of this frequency is unknown");
		}
	}
```

2.a)

Explanation: In the ring() method, I have set up 3 SONET routers as given in the question. The comments in the code have explained the process.

Code:

```
public void ring(){
		/*
		 * Setup the SONET ring
		 */
		System.out.println("Setting up 3 routers");
		// Create two SONET routers
		SONETRouter router1 = new SONETRouter("00:11:22");
		SONETRouter router2 = new SONETRouter("88:77:66");
		SONETRouter router3 = new SONETRouter("33:44:55");

		// tell routers a wavelength to add/drop on (in this case their own frequencies)
		router1.addDropWavelength(1310);
```

```
                  router2.addDropWavelength(1490);
                  router3.addDropWavelength(1550);

                  // tell router 1 the wavelength each router is add/dropping on
                  router1.addDestinationFrequency("00:11:22", 1310);
                  router1.addDestinationFrequency("88:77:66", 1490);
                  router1.addDestinationFrequency("33:44:55", 1550);

                  // tell router 2 the wavelength each router is add/dropping on
                  router2.addDestinationFrequency("00:11:22", 1310);
                  router2.addDestinationFrequency("88:77:66", 1490);
                  router2.addDestinationFrequency("33:44:55", 1550);

                  // tell router 3 the wavelength each router is add/dropping on
                  router3.addDestinationFrequency("00:11:22", 1310);
                  router3.addDestinationFrequency("88:77:66", 1490);
                  router3.addDestinationFrequency("33:44:55", 1550);

                  // Create an interface for each router
                  OpticalNICTA nicRouter11 = new OpticalNICTA(router1);
                  nicRouter11.setID(11);
                  OpticalNICTA nicRouter12 = new OpticalNICTA(router1);
                  nicRouter12.setID(12);

                  OpticalNICTA nicRouter21 = new OpticalNICTA(router2);
                  nicRouter21.setID(21);
                  OpticalNICTA nicRouter22 = new OpticalNICTA(router2);
                  nicRouter22.setID(22);

                  OpticalNICTA nicRouter31 = new OpticalNICTA(router3);
                  nicRouter21.setID(31);
                  OpticalNICTA nicRouter32 = new OpticalNICTA(router3);
                  nicRouter22.setID(32);

                  // Create two-uni directional links between the routers

                  //Links between Routers 1 and 2
                  OtoOLink OneToTwo1 = new OtoOLink(nicRouter11, nicRouter21);
                  OtoOLink TwoToOne1 = new OtoOLink(nicRouter21, nicRouter11);

                  //Links between Routers 1 and 3
                  OtoOLink OneToThree1 = new OtoOLink(nicRouter12, nicRouter31);
                  OtoOLink ThreeToOne2 = new OtoOLink(nicRouter31, nicRouter12);

                  //Links between Routers 2 and 3
                  OtoOLink TwoToThree2 = new OtoOLink(nicRouter22, nicRouter32);
                  OtoOLink ThreeToTwo2 = new OtoOLink(nicRouter32, nicRouter22);

                  /*
                   * Sent a frame on the network
                   */
//                router1.source(new SONETFrame(new SPE(0)), 7777); //Unsupported frequency
//                router1.source(new SONETFrame(new SPE(0)), 1490);
                  router1.source(new SONETFrame(new SPE(0)), 1550);
```

2.b)

Explanation:

At the end of the method, I have added the line where router one is generating a frame aimed at Router2.

Code:

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

2.c)

Explanation:

I have set up the SONET ring network with three routers, where each router has four NICs and

Code:

```
public void twoRings(){
                /*
                 * Setup the SONET ring
                 */
                System.out.println("Setting up 3 routers");
                // Create three SONET routers
                SONETRouter router1 = new SONETRouter("00:11:22");
                SONETRouter router2 = new SONETRouter("88:77:66");
                SONETRouter router3 = new SONETRouter("33:44:55");

                // tell routers a wavelength to add/drop on (in this case their own frequencies)
                router1.addDropWavelength(1310);
                router2.addDropWavelength(1490);
                router3.addDropWavelength(1550);

                // tell router 1 the wavelength each router is add/dropping on
                router1.addDestinationFrequency("00:11:22", 1310);
                router1.addDestinationFrequency("88:77:66", 1490);
                router1.addDestinationFrequency("33:44:55", 1550);

                // tell router 2 the wavelength each router is add/dropping on
                router2.addDestinationFrequency("00:11:22", 1310);
                router2.addDestinationFrequency("88:77:66", 1490);
                router2.addDestinationFrequency("33:44:55", 1550);

                // tell router 3 the wavelength each router is add/dropping on
                router3.addDestinationFrequency("00:11:22", 1310);
                router3.addDestinationFrequency("88:77:66", 1490);
                router3.addDestinationFrequency("33:44:55", 1550);

                // Create an interface for each router
                OpticalNICTA nicRouter11 = new OpticalNICTA(router1);
                nicRouter11.setID(11);
                OpticalNICTA nicRouter12 = new OpticalNICTA(router1);
                nicRouter12.setID(12);
                OpticalNICTA nicRouter13 = new OpticalNICTA(router1);
                nicRouter13.setID(13);
                OpticalNICTA nicRouter14 = new OpticalNICTA(router1);
                nicRouter14.setID(14);

                OpticalNICTA nicRouter21 = new OpticalNICTA(router2);
                nicRouter21.setID(21);
                OpticalNICTA nicRouter22 = new OpticalNICTA(router2);
                nicRouter22.setID(22);
                OpticalNICTA nicRouter23 = new OpticalNICTA(router2);
                nicRouter23.setID(23);
                OpticalNICTA nicRouter24 = new OpticalNICTA(router2);
                nicRouter24.setID(24);

                OpticalNICTA nicRouter31 = new OpticalNICTA(router3);
                nicRouter31.setID(31);
                OpticalNICTA nicRouter32 = new OpticalNICTA(router3);
                nicRouter32.setID(32);
                OpticalNICTA nicRouter33 = new OpticalNICTA(router3);
                nicRouter31.setID(33);
                OpticalNICTA nicRouter34 = new OpticalNICTA(router3);
                nicRouter34.setID(34);

                // Create two-uni directional links between the routers

                //Links between Routers 1 and 2
```

```
          OtoOLink OneToTwo1 = new OtoOLink(nicRouter11, nicRouter21);
          OtoOLink TwoToOne1 = new OtoOLink(nicRouter21, nicRouter11);
          OtoOLink OneToTwo2 = new OtoOLink(nicRouter12, nicRouter22);
          OtoOLink TwoToOne2 = new OtoOLink(nicRouter22, nicRouter12);

          //Links between Routers 1 and 3
          OtoOLink OneToThree1 = new OtoOLink(nicRouter13, nicRouter31);
          OtoOLink ThreeToOne1 = new OtoOLink(nicRouter31, nicRouter13);
          OtoOLink OneToThree2 = new OtoOLink(nicRouter14, nicRouter32);
          OtoOLink ThreeToOne2 = new OtoOLink(nicRouter32, nicRouter14);

          //Links between Routers 2 and 3
          OtoOLink TwoToThree1 = new OtoOLink(nicRouter23, nicRouter33);
          OtoOLink ThreeToTwo1 = new OtoOLink(nicRouter33, nicRouter23);
          OtoOLink TwoToThree2 = new OtoOLink(nicRouter24, nicRouter34);
          OtoOLink ThreeToTwo2 = new OtoOLink(nicRouter34, nicRouter24);

          /*
          ArrayList<Integer> Routes12 = new ArrayList<Integer>();
 ArrayList<Integer> Routes13 = new ArrayList<Integer>();
 ArrayList<Integer> Routes21 = new ArrayList<Integer>();
 ArrayList<Integer> Routes23 = new ArrayList<Integer>();
 ArrayList<Integer> Routes31 = new ArrayList<Integer>();
 ArrayList<Integer> Routes32 = new ArrayList<Integer>();

 Routes12.add(0, 11);
 Routes12.add(1, 12);
 Routes12.add(2, 13);
 Routes12.add(3, 14);
 router1.addDestinationHopCount(1490, Routes12);


 Routes13.add(0, 14);
 Routes13.add(1, 13);
 Routes13.add(2, 12);
 Routes13.add(3, 11);
 router2.addDestinationHopCount(1550, Routes13);


 Routes21.add(0, 21);
 Routes21.add(1, 22);
 Routes21.add(2, 23);
 Routes21.add(3, 24);
 router2.addDestinationHopCount(1310,Routes21);


 Routes23.add(0, 24);
 Routes23.add(1, 23);
 Routes23.add(2, 22);
 Routes23.add(3, 21);
 router2.addDestinationHopCount(1550, Routes23);


 Routes31.add(0, 31);
 Routes31.add(1, 32);
 Routes31.add(2, 33);
 Routes31.add(3, 34);
 router3.addDestinationHopCount(1310,Routes31);


 Routes32.add(0, 34);
 Routes32.add(1, 33);
 Routes32.add(2, 32);
 Routes32.add(3, 31);
 router3.addDestinationHopCount(1490, Routes32);
          */

/*
 router1.computeNewRoutes();
```

```
        router2.computeNewRoutes();
        router3.computeNewRoutes();
        */

                 /*
                  * Sent a frame on the network
                  */
//               router1.source(new SONETFrame(new SPE(0)), 7777); //Unsupported frequency
//               router1.source(new SONETFrame(new SPE(0)), 1490);
                 router1.source(new SONETFrame(new SPE(0)), 1550);


         }
```

2.d)

Explanation:

I was not able to complete this code for a general case.

Until the evening of the deadline, I had this code working but I had assumed the maximum no of routers for this problem is going to be three. I got confused by the statement "For simplicity, we will assume that the largest size of the ring is 3 SONET routers (this makes things a lot simpler)." in the question.

When I tried to make it work for a general case, I broke my code and I am not able to compile properly and the whole code is a bit messy right now.

PS: This question is not solved and do not compile q3c.java in my zip file.

Code:

Grade:	/ 40	/40	/20	/100