In this project, you will modify the wunet router to implement a WDRR packet scheduler for each output link, allowing each vnet to have a separate queue on each link. You will then examine how the scheduler behaves in a couple different situations and comment on your observations. You will find a new source tarfile (srcTarFile3) on the web site, which has everything you will need for the project. The link is in the left margin of the main page for the course (by the link for the system project description). This includes all the source files that you will need. While only some files have changed, the full set (including makefiles) is provided for your convenience (and mine). More details below.

1. To implement the scheduler, you will need to extend the qMgr class. You will find a partial version of the new qMgr in srcTarFile3. You should not need to make any further changes to the header file, but you will need to implement the enq and deq methods. The provided files include comments sketching out what the new version must do. Please read all the comments carefully and study the original version of qMgr, so that you understand how it works. If there is something that you don’t understand, be sure to ask.

A few other things must also be modified. You will find a new version of wunet.h a slightly modified version of vnetTbl and a new version of wuRouter. The new wunet.h includes a macro called truPktLeng() that returns the number of bytes that are sent on a link for a wunet packet of a given length. You should use truPktLeng() in your revised version of qMgr. In particular, your WDRR scheduler should “charge” a queue for the number of bytes actually sent on the link, rather than the number of bytes in the wunet packet.

Once you have made your modifications to qMgr and compiled your code, you should verify its operation in ONL. SrcTarFile3 provides an experimental setup that you can use for this purpose. More details under part 2.

Turn in a copy of the source code for the revised version of qMgr. If you change any other source code files, provide copies of these as well.

2. In srcTarFile3 you will find an ONL configuration file (wunet2.onl) which you should open in the RLI when running your ONL session for this part. You will also find a directory called qtest1 that includes router configuration files and packet generation files. There is also a script qtScript1 that you can run to send traffic through the configured network. Make qtest1 a subdirectory of wunetRuns (or, alternatively, modify qtScript1, if you prefer).

You should study all the files and make sure you understand the traffic that is being sent through the routers and how the routers process that traffic. Using the real-time displays, make a table of the measured traffic rates flowing through the network. Your table should
include the packet per second rate AND the bit rate for the total traffic from R1 to R2, traffic from n1p1, n1p2 and n1p3 to R1, and the traffic from R2 to n2p1, n2p2 and n2p3. For each value, explain why that specific value makes sense, based on the traffic and the way the routers are configured (explanations should be quantitative, not just descriptive). For the bandwidth values, please show the data in Mb/s.

NOTE. The NPRs report bandwidth based on the number of bytes in each IP packet. That is, they do not account for the Ethernet overhead. This will cause some discrepancy between the values you measure and what you might otherwise expect. Your explanations of the traffic data should account for this discrepancy (quantitatively).

Turn in a screenshot showing the four monitoring windows when you run the test. Make sure that your screenshot is large enough so that all the labels are legible. You should also turn in a copy of your table of data and your explanations for each of the values.

3. In srcTarFile3 you will find a directory called qtest2 that includes router configuration files and packet generation files for use in this part. There is also a script qtScript2 that you can run to send traffic through the configured network.

Study the bandwidth charts produced by the experiment and compare it to your expectations. How does the actual performance differ from the ideal behavior one would like to see? What is it about the WDRR scheduler that accounts for this difference? Turn in a screenshot showing all the monitoring windows along with your comments.