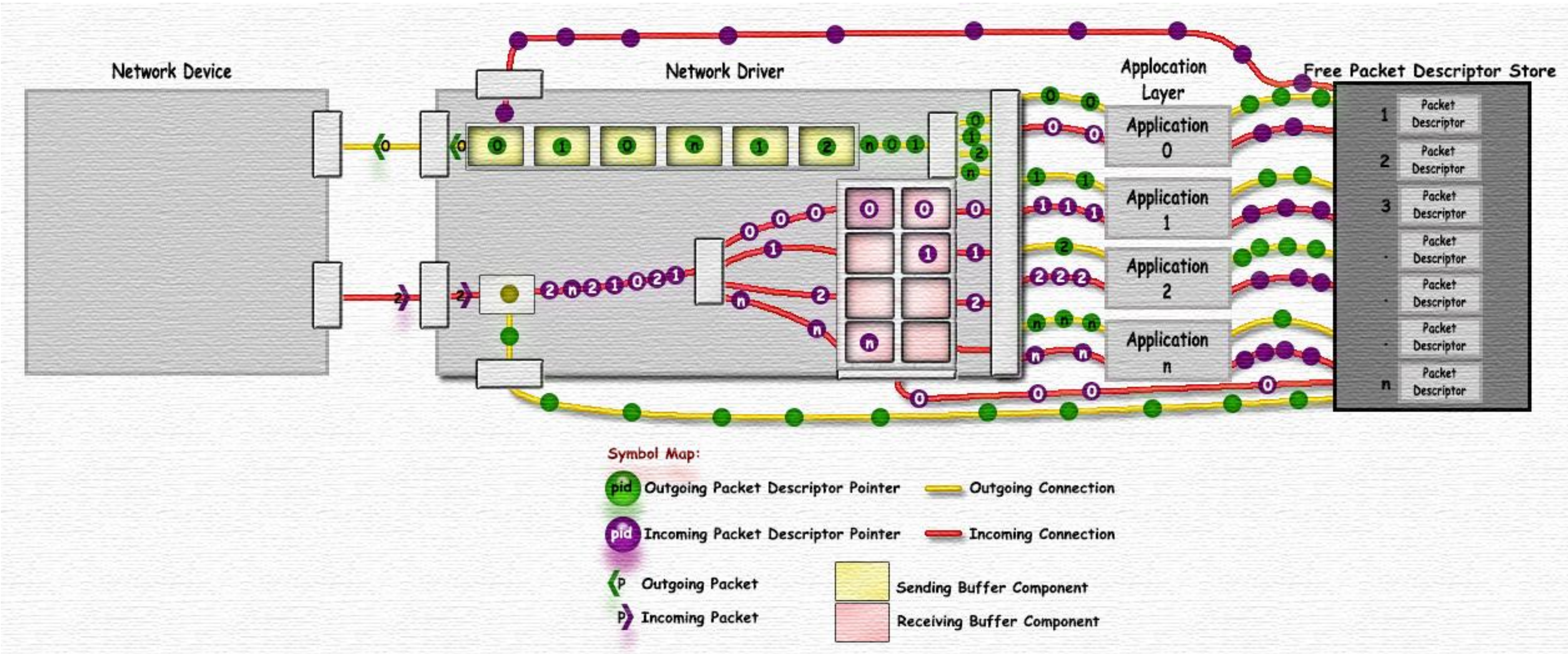
Report 2 – CIS 415
By: *Matt Almenshad*

Project 2 Report

<u>Overview</u>



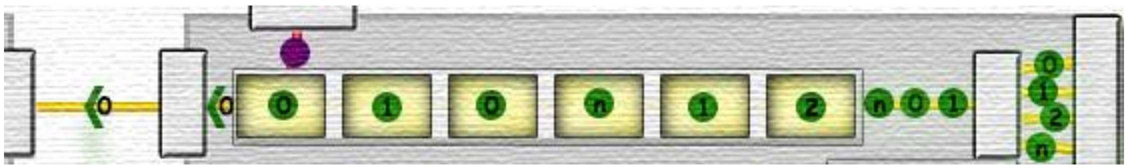
The network driver interfaces the network device and the applications through moving pointers to packet descriptors.
In the following report I will break down all the components and their role in my design as well as comment on the code associated with them.

Report 2 – CIS 415
By: *Matt Almenshad*

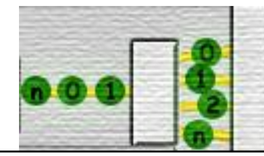
The overall driver

The driver is consistent of two functionalities. The receiving component that is a thread managing the transmission of the data sent by the network device to the applications. The transmitting component that is a thread managing the transmission of the data from the applications to the network device. In order to achieve the synchronization, I have used the provided bounded buffers and their functionality.

Sending data



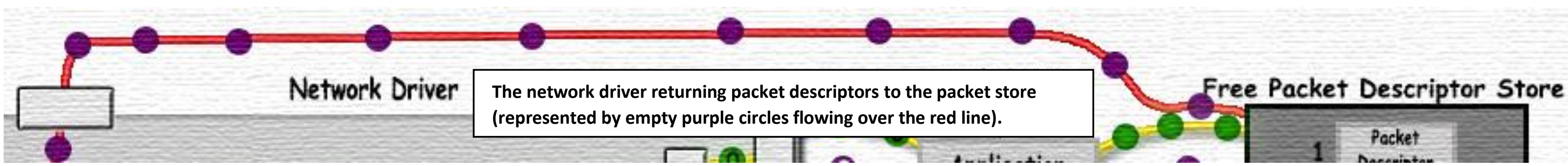
First, the driver takes in multiple packet descriptors from the applications. Then the driver queues the data up into a bounded buffer of size 10.
If an application requests a blocking send, it will execute a blocking write on the buffer. The call will make the application block until it has written its data to the buffer. A non-blocking call, on the other hand, will return immediately on returning an integer value indicating a value of 1 to represent success, or a value of 0 to represent failure.



Packets descriptors are blocked waiting to be queued

Full sending buffer

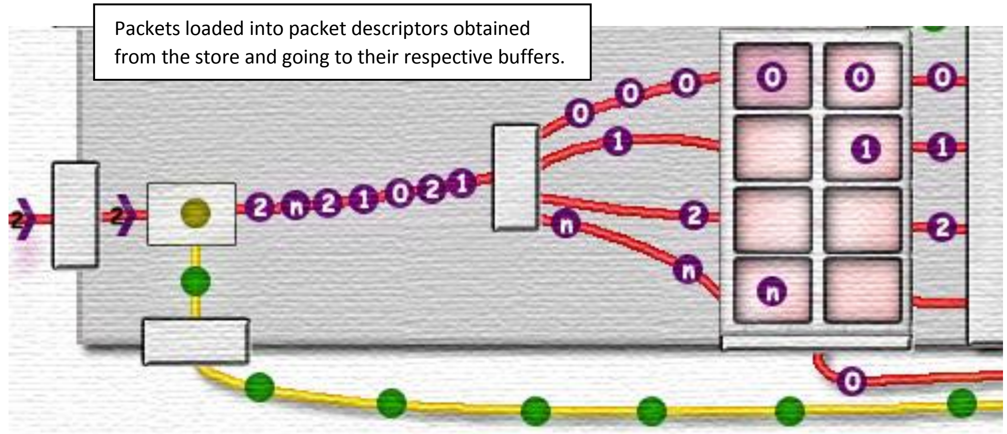
The buffer component queues the packet descriptors in to be for sending in the order that there are received. The thread then takes the packet descriptors out of the queues, sending the packets to the network device. If the driver failed to send the packet, it will reattempt to send the packet. Upon failing three times, the driver will give up and return the packet to the free packet store. The driver will then attempt to send next packet from the sending buffer to the network device.



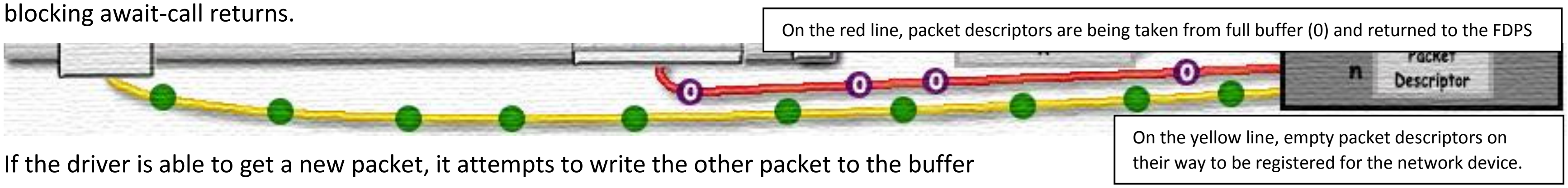
The network driver returning packet descriptors to the packet store (represented by empty purple circles flowing over the red line).

Report 2 – CIS 415
By: *Matt Almenshad*

Receiving data



The network device will send packets to the driver. The driver will always insure that there is a packet descriptor awaiting the next packet. The method I implemented always starts with an initialized packet descriptor waiting. The process of initializing registering the first packet descriptor occurs at my driver’s initialization function. I take advantage of the fact that the device will not be sending me data until my initialization function is completed. This also insures that I have at least one packet descriptor at the store ready to be taking through a blocking get call. After my initialization function is completed, my driver attempts to get a new packet at the start of every iteration once the blocking await-call returns.



If the driver is able to get a new packet, it attempts to write the other packet to the buffer associated with the application that is calling it.

If the application fails to get a new packet descriptor, it uses the current packet descriptor to store the next packet sent by the network driver rather than writing it into a buffer. If it succeeds in obtaining a new packet descriptor but fails to write the data into the application buffer then that would mean that the application buffer is full. The driver then discards the extra packets. The applications can only have two uncollected packages at most. When an application calls a get function, it is effectivally reading its corrosponding reieced buffer. If the application fails to get its packets fast enough, extra packets coming for the same application are discarded.