1. A short description of how you implemented your switch [7 points].

1. The switch is implemented using an unbounded queue that offers buffering of packets. The switch uses two threads, one for adding packets to the queue, and one for reading packets from the queue and placing them in the output buffers, i.e. if it can acquire the mutex for that particular output port. This mechanism prevents the thread from overwriting buffers that have valid data that hasn't yet been read. In addition to the mutex associated with the output port, we have added another mutex to lock the value referring to the size of the queue. Depending upon if we dequeue or enqueue a packet to the queue, we have to acquire that particular mutex in order to avoid incrementing or decrementing the value unintentionally.

   In an effort to get lower drop rate, we lock the 4 input ports and try to get packets from all 4, and add to the queue before releasing them. This is to avoid the scenario where the thread that places the object in the input queue gets context switched to before the previous packets it placed in the queue had been all read and added to the queue.

2. The packet drop rate as indicated by our test harness [3 points].

2. The packet drop rate, based on 5 runs using default settings (1000 messages sent) – 0 %.

3. Anything else you want the markers to know.

   Yes – Give us a 100 % on the assignment, please!