**EECE.7290 Special Topics on SDN and Data Plane Programming**

**Lab5**

**Introduction:**

In Lab2-RX we have learned how to use single RX queue to receive the packets. But, single RX queue would not be sufficient in high-volume network. Also, single RX queue would not take the advantage of multicore architecture. In this Lab5, we are going to learn how to use RSS (Receive Side Scaling) to increase the scalability of the DPDK application.

**Tasks:**

1. Read the example source code of l3fwd and ip-pipeline to find out how the RSS is used to implement multiple RX queues. You can use *grep -ri “rss”* to quickly find out where RSS is being used.

2. The main.c code of Lab2-RX can only run with a single RX queue, you need to modify the main.c by using what you have learned from the example source code, such that main.c can support multiple RX queues.

**Note:**

|  |
| --- |
| After you finish Lab2 and Lab3, you should have basic understanding of the DPDK, which means you can easily answer these questions:   * **Which API is used to setup EAL?** * **Which API is used to configure the device (hardware port), and specify the number of RX queues and TX queues etc?** * **Which API is used to setup RX queue, and in order to setup RX queue, which API is used to setup memory pool?** * **Which API is used to setup TX queue?** * **Which API is used to enable the promiscuous mode?** * **Which API is used to start the link?** * **Which API is used to launch the user-defined application for each enabled core?** * **Which API is used to receive the packets from the device?** * **Which API is used to send the packets out to the device?**   These questions can help you understand the Big-Picture of any DPDK application.    The final goal of Lab5 is to enable and use multiple RX queues to receive the packets. You need to use "-c" EAL option to enable multiple cores in your program, and then you need to control each lcore to work on its own RX queue.    First, you need to read the l3fwd example source code and try to answer the above questions again, so in this way, you can easily find out the Big-Picture of the l3fwd example. Once you figure out the Big-Picture, please then focus on the codes used to setup of the RX queues, since this is the main task in lab5. Please find out the differences on how to setup the RX queues between the RX code(Lab2) and l3fwd example, and apply the changes accordingly to the RX code (Lab2), so in this way your modified code can support multiple RX queues.    Second, the developer of the l3fwd example has not yet fully finished the RSS part, so you probably can only see the received packets in only one lcore. Then your next task is going to read and learn the init.c of the example ip-pipeline. You are going to mainly learn how to correctly use "rte\_eth\_dev\_rss\_reta\_update()" to modify the redirection table.  In general, you need to follow the steps in the below to enable multiple RX queues.  1. Change the configuration for the PORT  struct rte\_eth\_conf conf;  conf.rxmode={  .mq\_mode = ETH\_MQ\_RX\_RSS,    .rx\_adv\_conf = {  .rss\_conf = {  .rss\_key = NULL,  .rss\_key\_len = 40,  .rss\_hf = ETH\_RSS\_IPV4,  },  },  }  2. Change the number of RX queues  rte\_eth\_dev\_configure(pmd\_id, NUM\_RX\_QUEUES, 0, &conf);  3. Setup the mempool for each RX queue and configure the RX queue  for\_each\_RX\_queue {  struct rte\_mempool \* mp;      mp = rte\_pktmbuf\_pool\_create( name, pool\_size, cache\_size, priv\_size, data\_size, socket\_id); //name for each pool should be distinct      rte\_eth\_rx\_queue\_setup(pmd\_id, queue\_id, rx\_desc\_size, socket\_id, &rx\_queue\_conf, mp);  }  4. Change the redirection table by the API: rte\_eth\_dev\_rss\_reta\_update()  5. In the subroutine for each core, rte\_eth\_rx\_burst() needs to use different queue\_id, since we want each core to work on a different RX queue. |

**Submissions:**

A single report which includes the following

1. Summarize the key steps which are used to enable the multiple RX queues with RSS.

2. Your modified main.c.