Networked Systems Coursework 3: L4 Load Balancer

Viyan Raj

1 DPDK Load Balancer

1.1 Implementing the Load Balancer

My DPDK load balancer implementation is in the net.c file within this folder. I used the external 'rte_tcp.h' functions to extract tcp headers from the packet. My hash function simply added the three varying values (source ip, destination ip, destination port), then applied modulo 2. This generated either 0 or 1, as there are two possible namespaces to forward to (h2 or h3). If a packet arrives from h2 or h3 it is simply forwarded to 10.0.0.1.

When extracting certain information from headers such as ports, I used the 'rte_be_to_cpu_16' function to convert the 16-bit value from big endian to CPU order.

Finally, I used the provided get_mac_for_ip() function to get the ethernet mac address to forward into, and passed this to the eth_out() function.

1.2 Testing the Load Balancer

I set up two netcat servers in the two namespaces (running 'h2 nc -l 8080' and 'h3 nc -l 8080' in different terminals). Then I ran 'nc 10.0.0.10 8080' from terminal to connect to one of these servers. When I entered text from this terminal, it was echoed in the h2 server. After running it a few more times, the text was echoed in the h3 server instead. This is because, when server 10.0.0.10 tries to transmit TCP packets, they go via the load balancer middlebox. This performs the hash function on the packet and uses it to determine whether to forward to h2 or to h3. This allows the traffic to be shared between both.

When trying to take screenshots during testing, VirtualBox repeatedly crashed, and attempting to ssh into the VM didn't work either. Given more time, these screenshots of both the server responses and tcpdump packet capture would have been included.