libVNF: User Manual

May 16, 2018

1 Setup

The VNF will run on VMs hosted on commodity servers in case of kernel/netmap setup and on host in case of DPDK version. For the mTCP stack the netmap/DPDK setup have to be done on the server as well as the VMs.

Netmap Setup:
Netmap uses vale as the software switch. Follow steps (1-4) in :
https://github.com/networkedsystemsIITB/Modified_mTCP/blob/master/mTCP_over_Netmap/docs/netmap_docs/user_manual.pdf
In netmap setup, mTCP requires multi-queue in vale.

- Copy the netmap_vale.c in dependencies folder to netmap/sys/dev/netmap/ and compile netmap again. We have compiled netmap with igb driver.
- Create interface for each VM with no.of of queues=no of cores of VM.

```
./vale-ctl -n viX -C 2048,2048,Y,Y
//Y is the number of cores of the VM, X will be a number (1,2,3..)
./vale-ctl -a vale0:viX
ifconfig viX hw ether 00:aa:bb:cc:dd:0Z //MAC address of the VM
change ifname in XML to vale0:viX
```

- Install mTCP in VM as given in the mTCP github page
  https://github.com/eunyoung14/mtcp

DPDK Setup:
We could not get mTCP to work on VMs as we were unable to expose multi-queue to the VM. Follow the steps for the DPDK +mTCP setup on host machine:
https://github.com/eunyoung14/mtcp

2 Prerequisites for library

Install the following dependency

```
sudo apt-get install libboost-all-dev
```

3 Configuration

Before compiling do the following changes in the files:
mTCP setup:

- Put the folder in apps/ folder of mtcp
- change the interface name in server.conf according to your interface
- change the number of cores in server.conf
- in lib.h change the MAX_THREADS=number of cores of the VM (netmap)/ host (DPDK)
- To run for lower cores in DPDK setup on host turn of the remaining cores
- echo 1 > /proc/sys/net/ipv4/tcp_tw_reuse on VMs running the kernel stack

kernel setup:

- in lib.h change the MAX_THREADS=number of cores of the VM (netmap)/ host (DPDK)
- echo 1 > /proc/sys/net/ipv4/tcp_tw_reuse on VMs running the kernel stack
- ulimit -n 65535

4 Compilation

If running the sample A-B-C code change the IP addresses in clientA.cpp, b.cpp and c.cpp.

- compile clientA.cpp (on VM A) and c.cpp (on VM C) with g++ and -lpthreads
- make clean and make the code on B.
- run b as sudo user (sudo ./b) in mTCP and ./b in kernel setup

In general to compile any code using the libVNF API compile with lib.o, libpacket.o and utils.o files provided in the library_files folder

5 Distributed Setup

Load Balancer setup:
A VNF can run in multiple VMs in a distributed fashion. To, split traffic between these replica VNF, we provide a load balancer. The load balancer can be the VNF built over our library or a kernel module. Both the version of the load balancer are provided in the Load_Balancer subfolder of example folder.
For using the load balancer the network stack of VNF needs to be modified. For mTCP stack, replace the mtcp folder with the one provided in dependencies/mtcp_when_using_lb and set the lb IP in server.conf. For kernel stack do the following steps on the backend server:

- Install the module given in backend_kernel_module. Replace the source IP to your load balancer IP.
- echo 1 > /proc/sys/net/ipv4/ip_forward

To install library version of load balancer; inside the **Data Store Setup:**

- Download the KeyValueStore folder.
- Use step in *https://github.com/networkedsystemsIITB/NFV_LTE_EPC/blob/master/NFV_LTE_EPC-2.0/KeyValueStore/Docs/UserManual.pdf* for the client and server installation on the data store VM. Use server IP as 127.0.0.1.
- Copy paste the dependencies/data_store_wrapper folder into client folder
- Update the IP in redis_ep_server.cpp.
- Make and execute the code (rd_ep.out).
- Verify that the DS ports in lib.h are the same as the data store ports in rd_ep_server.cpp.