User Manual for NETMAP based network I/O for application layer VNFs

This manual provides setup and usage of our implementation of packet distribution mechanism based on NETMAP for application layer VNFs. These instructions work for **LINUX based Operating Systems** only. Please read **developer_manual.pdf** for design and implementation details of packet distribution mechanism based on NETMAP.

SETUP AND INSTALLATION INSTRUCTIONS IN HOST

1. *Installation steps for NETMAP/VALE and QEMU in host*

1.1. Installing NETMAP:

Download LINUX source for installed LINUX distribution in host machine

For **UBUNTU** Users **(As normal user)**:

```
$ sudo apt-get source linux-image-$(uname -r)
```

For **Centos 7** Users **(As normal user)**:

```
$ sudo yum install rpm-build redhat-rpm-config asciidoc hmaccalc
 ↪  perl-ExtUtils-Embed pesign xmlto
$ sudo yum install audit-libs-devel binutils-devel elfutils-devel
 ↪  elfutils-libelf-devel
$ sudo yum install ncurses-devel newt-devel numactl-devel pciutils-devel
 ↪  python-devel zlib-devel
# [ Download appropriate kernel-[version].rpm. Use uname -r to know
 ↪  kernel-[version].rpm to be downloaded ]
$ yumdownloader --source kernel
# [ can be used to download kernel rpm ]
$ mkdir -p ~/rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
$ echo '%_topdir %(echo $HOME)/rpmbuild' > ~/.rpmmacros
$ rpm -i kernel-[version].rpm 2>&1 | grep -v exist
$ cd ~/rpmbuild/SPECS
$ rpmbuild -bp --target=$(uname -m) kernel.spec
# [ The kernel source tree will now be found under the
 ↪  ~/rpmbuild/BUILD/kernel*/linux*/ directory.]
```

Download and compile NETMAP in host machine

```
$ git clone https://github.com/luigirizzo/netmap
$ cd netmap/LINUX
# If netmap needs to be compiled along with network drivers
$ ./configure --drivers=[network drivers]  --kernel-sources=[PATH TO LINUX SOURCE]
 ↪  --enable-ptnetmap-host
# If no network drivers needed
$ ./configure --no-drivers --enable-ptnetmap-host
$ make
# config.status file in netmap/LINUX/ contains command lines needed for
 ↪  configuration. To re-configure, config.status can be used.
```

NOTE: We have experimented with netmap interfaces with rings having **2048 slots** (default being **1024 slots**). For enabling **2048 slot ring size**, copy all patches in **patches/2048_slots/** folder to **netmap/sys/dev/netmap/** folder and apply patches followed by **make** in **netmap/LINUX** folder.

1.2. Installing NETMAP patched QEMU:

Install Dependencies

For **UBUNTU** Users **(As normal user)**:

```
$ sudo apt-get install git libglib2.0-dev libfdt-dev libpixman-1-dev zlib1g-dev
$ sudo apt-get install git-email
$ sudo apt-get install libaio-dev libbluetooth-dev libbrlapi-dev libbz2-dev
$ sudo apt-get install libcap-dev libcap-ng-dev libcurl4-gnutls-dev libgtk-3-dev
$ sudo apt-get install libibverbs-dev libjpeg8-dev libncurses5-dev libnuma-dev
$ sudo apt-get install librbd-dev librdmacm-dev
$ sudo apt-get install libsasl2-dev libsdl1.2-dev libseccomp-dev libsnappy-dev
 ↪  libssh2-1-dev
$ sudo apt-get install libvde-dev libvdeplug-dev libvte-2.90-dev libxen-dev
 ↪  liblzo2-dev
$ sudo apt-get install valgrind xfslibs-dev
```

For **Centos 7** Users **(As normal user)**:

```
$ sudo yum install git glib2-devel libfdt-devel pixman-devel zlib-devel
$ sudo yum install libaio-devel libcap-devel libiscsi-devel
```

Download, compile, and install netmap patched QEMU

```
$ git clone https://github.com/vmaffione/qemu
$ cd qemu
$ git checkout ptnet
```

```
$ ./configure --target-list=x86_64-softmmu --python=python2 --enable-kvm
 ↪  --enable-vhost-net --disable-werror --enable-netmap --enable-ptnetmap
 ↪  --extra-cflags=-I[Path to netmap sys directory]
$ make
$ sudo make install
```

Above instructions will install **netmap patched QEMU** as

```
/usr/local/bin/qemu-system-x86_64
```

2. *Installing libvirt*

2.1. Install libvirt and virt-manager:

For **UBUNTU** Users **(As normal user)**:

```
$ sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager
```

For **Centos 7** Users **(As normal user)**:

```
$ sudo yum install @virt* dejavu-lgc-* xorg-x11-xauth tigervnc libguestfs-tools
 ↪  policycoreutils-python bridge-utils
$ sudo yum install qemu-kvm qemu-img virt-manager libvirt libvirt-python
 ↪  libvirt-client virt-install virt-viewer bridge-utils
```

3. *Configuring libvirt*

As **superuser**, do the following to change libvirt configuration:

3.1. Change libvirt configuration to get access to netmap device and ptnetmap device:

For UBUNTU users:

```
        # gedit /etc/apparmor.d/usr.sbin.libvirtd &

                Add line:
                /usr/local/bin/qemu-system-x86_64 rmix,
                below following line:
                /etc/xen/scripts/** rmix,
```

```
# gedit /etc/apparmor.d/abstractions/libvirt-qemu &

        Add line:
        /usr/local/bin/qemu-system-x86_64 rmix,
        below following line:
        # the various binaries


# gedit /etc/libvirt/qemu.conf &

        Set Options:

        security_driver = "none"
        user = "root"
        group = "root"
        clear_emulator_capabilities = 0
        (Do not forget to remove '#' before above lines, if present)

        Add following in 'cgroup_device_acl':
        "/dev/netmap"
```

For Centos 7 users:

```
# gedit /etc/selinux/config &

        Set Option:

        SELINUX=disabled
        (Do not forget to remove '#' before above lines, if present)

# gedit /etc/selinux/semanage.conf &

        Set Options:

        module-store = direct
        expand-check = 0
        usepasswd=False
        bzip-small=true
        bzip-blocksize=5
        ignoredirs=/root
        (Do not forget to remove '#' before above lines, if present)

# gedit /etc/libvirt/qemu.conf &

        Set Options:

        security_driver = "none"
        user = "root"
        group = "root"
```

```
clear_emulator_capabilities = 0
(Do not forget to remove '#' before above lines, if present)

Add following in 'cgroup_device_acl':
"/dev/netmap"
```

3.2. Restart libvirt:

For UBUNTU users:

```
# /etc/init.d/libvirt-bin restart
```

For Centos 7 users:

```
# systemctl restart  libvirtd.service
```

If changes are not reflected, ***reboot the system*** and do the following before creating/modifying/using VMs using libvirt:

```
# cd [Path to netmap/LINUX]
# insmod netmap.ko
```

4. *Creating and modifying VMs*

As **super-user** perform the following steps (Skip the steps already done / not needed):

4.1. Creating a VM (Skip this step if VM is already created):

1. Open **virt-manager**, click on Preferences under **Edit** tab and change **Display** to **VNC**.
2. Now create a new VM using **virt-manager**.

4.2. Cloning a VM (Skip this step if VM is already created):

```
# virt-clone --connect qemu:///system --original OLD_VM_NAME --name NEW_VM_NAME
 ↪  --file /[Desired path for NEW_VM_NAME.qcow2 file]/NEW_VM_NAME.qcow2
```

4.3. Modify a VM to use a PTNETMAP device (NIC, VALE):

NOTE: We have provided sample xml file in **extras/** folder for reference. To use **PTNETMAP** device, change **XML** of **VM** using following steps:

1. Open VM **.xml** file located in **/etc/libvirt/qemu/**.
2. Replace:

   ```
   <domain type='kvm'>
   ```

   with

   ```
   <domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
   ```

3. Set few other XML elements as shown below:

   ```
           (1) <os>
                   <type arch='x86_64' machine='pc-i440fx-2.1'>hvm</type>
           (2) <cpu mode='custom' match='exact'>
                   <model fallback='allow'>Haswell-noTSX</model>
           (3) <emulator>/usr/local/bin/qemu-system-x86_64</emulator>
   ```

4. Add following lines before **</domain>** element:

   ```
   <qemu:commandline>
           <qemu:arg value='-device'/>
           <qemu:arg value='ptnet-pci,mac=[SOME MAC],netdev=someid'/>
           <qemu:arg value='-netdev'/>
           <qemu:arg value='netmap,ifname=[valeX:YY/netmap:[PHYSICAL
            ↪  INTERFACE]],passthrough=on,id=someid'/>
   </qemu:commandline>
   ```

Re-define changed VM:

```
# virsh define [PATH TO CHANGED XML]
```

5. Setup for various VNF designs in host machine

5.1. Setup in host for software based packet distribution:

VALE based packet I/O and distribution

As **super-user** perform the following steps for **each VM** (Skip the steps already done / not needed):

1. Modify VALE switch logic for software RSS based multi-queue packet distribution (skip if already done):
   1.1. In case of netmap rings with **1024 slots**, copy patch **patches/1024_slots/netmap_vale_mq_1024.patch** to **netmap/sys/dev/netmap/** directory and apply the patch.
   1.2. Else, in case of netmap rings with **2048 slots**, first reverse patch **netmap_vale_2048.patch**, if applied previously. Then copy patch **patches/2048_slots/netmap_vale_mq_2048.patch** to **netmap/sys/dev/netmap/** directory and apply the patch.
   1.3. Now, proceed with **make** in **netmap/LINUX** directory to re-make netmap module, and re-insert the module.
2. Create a persistent VALE port using following steps:

```
1) Create a persistent virtual interface using vale-ctl provided by
 ↪  NETMAP:
# cd netmap/LINUX/build-apps/vale-ctl/
# ./vale-ctl -n <interface name> -C <No. of BUFS TX>,<No. of BUFS
 ↪  RX>,<No. of TX RINGS>,<No. of RX RINGS>
2) If POLLING MODE is required (skip this step  if not required):
# ./vale-ctl -p <interface name> -C 0,<HOST CORE NO.>
3) Attach interface to a VALE switch valeX (Replace X with some
 ↪  digit):
# ./vale-ctl -a valeX:<interface name>
4) Assign MAC address to created interface:
# ifconfig <interface name> hw ether <SOME MAC ADDRESS>
```

3. Use persistent VALE port in a VM:

3.1. Open VM **.xml** file located in **/etc/libvirt/qemu/** and set **mac**, **netdev**, **ifname**, and **id** attributes in **<qemu:commandline>** element. Set **ifname=valeX:<some ifname>** and **mac=<SOME MAC ADDRESS>**, where **X <some ifname>** is interface name of **persistent port** created using above steps, and **<SOME MAC ADDRESS>** is MAC address used during port creation.

3.2. Re-define changed VM:

```
# virsh define [PATH TO CHANGED XML]
```

4. Insert netmap module (Remove and re-insert module if module is re-compiled):

```
# cd netmap/LINUX
# insmod netmap.ko
```

5. To turn off polling mode of multi-queue persistent interface:

```
# ./vale-ctl -P <interface name>
```

6. To detach persistent interface from VALE and to remove it:

```
# ./vale-ctl -d valeX:<interface name>
# ./vale-ctl -r <interface name>
```

VALE based packet I/O, with VM based packet distribution

As **super-user** perform the following steps for **each VM** (Skip the steps already done / not needed):

1. Reverse patch applied for VALE based packet distribution. Then, proceed with **make** in **netmap/LINUX** directory to re-make netmap module.

2. Open VM **.xml** file located in **/etc/libvirt/qemu/** and set **mac**, **netdev**, **ifname**, and **id** attributes in **<qemu:commandline>**

element. Set **ifname=valeX:YY**, with **X** and **Y** replaced with **a digit**.

3. Re-define changed VM:

```
# virsh define [PATH TO CHANGED XML]
```

4. Insert netmap module (Remove and re-insert module if module re-compiled):

```
# cd netmap/LINUX
# insmod netmap.ko
```

Physical NIC based packet I/O, with VM based packet distribution

As **super-user** perform the following steps for **each VM** (Skip the steps already done / not needed):

1. Insert netmap module (Remove and re-insert module if module re-compiled):

```
# cd netmap/LINUX
# insmod netmap.ko
```

2. Copy the script **mq_hw.sh** in **scripts/** to folder containing **netmap/** folder.

3. Run script **mq_hw.sh** to enable **1 queue pair** of **NIC** as explained below:

```
# ./mq_hw.sh <No. of rings> <No. of ring buffers> <INTERFACE> <DRIVER
 ↪  NAME> <IPv4 Address> <IPv4 NETMASK>
Example (for 1024 slot rings):
# ./mq_hw.sh 1 1024 ens259f0 ixgbe 169.254.9.3 255.255.0.0
Example (for 2048 slot rings):
# ./mq_hw.sh 1 2048 ens259f0 ixgbe 169.254.9.3 255.255.0.0
```

4. Open VM **.xml** file located in **/etc/libvirt/qemu/** and set **mac**, **netdev**, **ifname**, and **id** attributes in **<qemu:commandline>**

element.
Ex. **ifname=netmap:ens259f0**

5. Re-define changed VM:

```
# virsh define [PATH TO CHANGED XML]
```

5.2. Setup in host for hardware based packet distribution

Modify NETMAP patched network driver seed

Skip these steps if already done:

1. Modify **seed** in files **netmap/LINUX/[driver name]/kcompat.c** and **netmap/LINUX/netmap-tmpdir/[driver name]/kcompat.c** used for **RSS hashing** as follows:

```
static const u8 seed[NETDEV_RSS_KEY_LEN] = {0x05, 0x05, 0x05, 0x05,
                    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05,
                    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05,
                    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05,
                    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05,
                    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05,
                    0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05, 0x05};
```

2. Re-compile module:

```
$ cd netmap/LINUX
$ make
```

Insert network driver and modify XML

As **super-user**, do the following:

1. Insert netmap module (Remove and re-insert module if module re-compiled):

```
# cd netmap/LINUX
# insmod netmap.ko
```

2. Copy the script **mq_hw.sh** in **scripts/** to folder containing **netmap/** folder.

3. Run script **mq_hw.sh** to enable desired number of queue pairs of NIC with desired number of buffers as explained below:

```
# ./mq_hw.sh <No. of rings> <No. of ring buffers> <INTERFACE> <DRIVER
 ↪  NAME> <IPv4 Address> <IPv4 NETMASK>
Example (for 1024 slot rings):
# ./mq_hw.sh 4 1024 ens259f0 ixgbe 169.254.9.3 255.255.0.0
Example (for 2048 slot rings):
# ./mq_hw.sh 4 2048 ens259f0 ixgbe 169.254.9.3 255.255.0.0
```

4. Open VM **.xml** file located in **/etc/libvirt/qemu/** and **mac**, **netdev**, **ifname**, and **id** attributes in **<qemu:commandline>** element.
   Ex. **ifname=netmap:ens259f0**

5. Re-define changed VM:

```
# virsh define [PATH TO CHANGED XML]
```

SETUP AND INSTALLATION INSTRUCTIONS IN GUEST VM

6. Download and compile NETMAP, and insert NETMAP module (Skip the steps already performed / not needed)

```
$ git clone https://github.com/luigirizzo/netmap
$ cd netmap/LINUX
$ ./configure --no-drivers --enable-ptnetmap-guest
$ make
# If virtio-net header not required in packets
$ sudo insmod netmap.ko ptnet_vnet_hdr=0
# Else
$ sudo insmod netmap.ko
# config.status file in netmap/LINUX/ contains command line needed for
 ↪  configuration. To re-configure, config.status can be used.
```

NOTE: We have experimented with netmap interfaces with rings having **2048 slots** (default being **1024 slots**). For enabling **2048 slot ring size**, copy all patches in **patches/2048_slots/** folder to **netmap/sys/dev/netmap/** folder and apply patches followed by **make** in **netmap/LINUX** folder.

7. Download moodycamel ConcurrentQueue

```
$ git clone https://github.com/cameron314/concurrentqueue
```

8. Install Google Sparse Hash

```
$ sudo apt-get update
$ sudo apt-get install sparsehash
```

9. Download and install libcuckoo

```
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:george-edison55/cmake-3.x
$ sudo apt-get update
$ sudo apt-get install cmake
$ sudo apt-get install cmake-curses-gui
$ git clone https://github.com/efficient/libcuckoo
$ cd libcuckoo
$ cmake CMakeLists.txt -DBUILD_EXAMPLES=1 -DBUILD_TESTS=1 -DBUILD_STRESS_TESTS=1
 ↪  -DBUILD_UNIT_TESTS=1 -DBUILD_UNIVERSAL_BENCHMARK=1
$ make all
$ sudo make install
```

10. Configurable parameters for NETMAP based packet distribution

Packet distribution logic has been implemented in files **netmap_api.h** and **netmap_api.cpp**.

10.1. Configurable parameters in *netmap_api.h*

- ***SEND_PIPES_IFNAME***: Interface used to open egress distribution netmap pipes.
- ***RECV_PIPES_IFNAME***: Interface used to open ingress distribution netmap pipes.
- ***BUSY_WAIT***: If enabled **ioctl()** is used, otherwise **poll()** is used. Generally disabled.

- ***MAX_BATCH_SEND***: To adjust send batch of packets (Varied between 1 and 5, varied on basis of number of messages per connection).
- ***MAX_BATCH_RECV***: To adjust receive batch of packets (Varied between 1 and 128, varied on basis of number of messages per connection).
- ***DATA_LEN***: Message length copied to send buffer or received from receive buffer by application.
- ***NO_OF_SAFE_ITR***: Packet send/receive iterations after which throughput calculation is to be started.
- ***OPT_FOR_SINGLE_CORE***: Option to be enabled if single-queue-single-core / multi-queue enabled.
- ***IFNAME***: vNIC interface name preceded by **netmap:**.
- ***N_MINUS_1_CONFIG***: **0** if **OPT_FOR_SINGLE_CORE** is used or for two core kernel bypass case, **1** otherwise.
- ***N_MINUS_2_CONFIG***: **0** if **OPT_FOR_SINGLE_CORE** is used or for one core kernel bypass case, **1** otherwise.

10.2. Static ARP Entries in netmap_api.cpp

If required, static ARP entries can be made in function **source_hwaddr()**. Before the end of function, sample ARP entries have been provided for **single-queue** as well as **multi-queue** operations.

10.3. Configurations for sample TCP/UDP application provided

Our sample epoll-based client-server application can be used to test various packet distribution setups.

Configurations for TCP/UDP

1. Insert netmap module in VM to enable ptnetmap enabled vNIC.

2. Steps **3, 4, 5 / 6** are to be performed in both **client and server VMs**.

3. In **netmap_api.h** set ***MAX_BATCH_SEND*** and ***MAX_BATCH_RECV*** appropriately. We have tested our sample TCP client / server application with **1 MB** TCP buffer size with ***MAX_BATCH_SEND*** set to **4 / 5** based on performance and ***MAX_BATCH_RECV*** set to **128**. Similarly, we have tested our sample UDP client / server application with **8 MB** UDP buffer size with ***MAX_BATCH_SEND*** set to **8** and ***MAX_BATCH_RECV*** to **1024**.

4. Set **IFNAME** to **vNIC** interface to be used for packet I/O. Set **DATA_LEN** to appropriate value (in bytes for payload size), if our sample epoll-based TCP client-server application is to be used for testing.

5. In **netmap_api.h** enable option ***OPT_FOR_SINGLE_CORE*** for **VALE / NIC** based packet distribution. Disable it for **VM** based packet distribution.

6. In **netmap_api.h**, for one core kernel bypass in case of **VM** based packet distribution, set ***N_MINUS_1_CONFIG*** to **1** and ***N_MINUS_2_CONFIG*** to **0**. Skip this step for **VALE / NIC** based packet distribution.

7. For two core kernel bypass in case of **VM** based packet distribution, set ***N_MINUS_1_CONFIG*** to **0** and ***N_MINUS_2_CONFIG*** to **1**. Skip this step for **VALE / NIC** based packet distribution.

8. Please make sure that path to **netmap/sys** in **Makefile** is correct before make.

9. To test our stack using our sample TCP/UDP client/server application, perform steps mentioned below, else skip them.

10. Set **MAX_THREADS** in TCP/UDP client/server programs appropriately as mentioned below:

```
n   : if multi-queue VALE port / NIC is used
n-1 : if 1-core kernel bypass is used
n-2 : if 2-core kernel bypass is used
where, n = number of cores in the VM
```

11. In case of **physical NIC** based packet distribution, enable option **MQ_NIC** in TCP client program, else disable it. If enabled, array **mq_ports** needs to be populated manually based on number of cores in the VM, by identifying a port corresponding to each core that can be used at given cores based on TCP/UDP 4-tuples.

12. Set parameters **CLIENT_IP**, **SERVER_IP**, **CLIENT_START_PORT**, **SERVER_PORT** in client program. Set **SERVER_IP**, **SERVER_PORT** in case of UDP server program.

13. In **config.sh**, set **num_cores** to number of application usable cores. Set **port** to **vNIC interface name** and set TCP/UDP send and receive buffers appropriately.

14. Make sure that path to **netmap/sys** folder is correct in **compile.sh** before compiling the application.