CS 549: Performance Analysis of Computer Networks
Quick Guide to Virtual Networking

August 19, 2020

Contents

1 Introduction

Linux has rich virtual networking capabilities that are used as the basis for hosting virtual machines (VMs) and containers, and configuring cloud servers. The rest of this Section has a brief description of various components that will enable you to create such a virtual network, and design simple experiments to evaluate its performance. A list of useful commands can be obtained using `ip link help`[1].

1.1 Network Namespace

Linux Containers (LXC) and Docker make extensive use of Linux network namespaces, which allow the user to define and use multiple virtual instances of the resources of a host and kernel. Linux namespaces include Cgroup, IPC, Network, Mount, PID, User and UTS. A network namespace is a copy of the network stack, with its own routing table, firewall rules, and network devices.[3] With network namespaces, we can have different and stand-alone instances of network interfaces and routing tables that operate independent of each other. In container solutions, network namespaces allow individual containers exclusive access to virtual network resources, and each container can be assigned a separate network stack [1],[3].

1.2 Bridge

A Linux bridge is a virtual network switch. It forwards packets between virtual interfaces that are connected to it. It is usually used for forwarding packets on routers, on gateways, or between VMs and network namespaces on a host [1].

1.3 VLAN

A virtual LAN (VLAN) separates broadcast domains by adding tags to network packets. VLANs allow network administrators to group hosts under the same switch or between different switches. We can use a VLAN when we want to separate subnet in VMs, namespaces, or hosts.With VLAN, we can create multiple interfaces on top of a single one and filter packages based on a VLAN tag [1].

1.4 MACVLAN

MACVLAN allows one to create multiple interfaces with different Layer 2 (that is, Ethernet MAC) addresses on top of a single one. Bridge mode is the most commonly used. All endpoints (MACVLAN interfaces) are directly connected to each other with a simple bridge via the physical interface [1].

1.5 Traffic shaping

Traffic shaping is an attempt to control the traffic between interfaces using properties like: delay (it delays each packet) and loss (it drops some packets). Traffic shaping uses traffic classification, policy rules, queue disciplines and quality of service (QoS). A queuing layer exists between the network device and the protocol output. The default queuing discipline is a simple FIFO packet queue. Queuing discipline consists of two key interfaces; one queues packets to be sent, and the other releases packets to the network device for transmission. The queuing discipline decides the order in which packets are sent, relative to the order of the arrival [2].

2 Virtual Networking HowTo

In this section, we will see how to set up different namespaces and MACVLANs in these namespaces. We cover traffic shaping on MACVLANs and running `ssh` in network namespaces.

2.1 Create network namespaces and macvlans

In this section, we will go over the commands required to create and delete namespaces. We will create MACVLANs and add them to namespaces, subsequently assign IP addresses to these MACVLANs.

1. Create network namespace `net1`:
   ```
   ip netns add net1
   ```

2. Delete namespace `net1`:
   ```
   ip netns delete net1
   ```

3. Show all network namespaces:

   ```
   ip netns list
   ```

4. To create Macvlan in network namespace: Creating macvlan in network namespace includes 3 steps:

   ```
   ip link add macvlan1 link wlp2s0 type macvlan mode bridge
   ip link set macvlan1 netns net1
   ip netns exec net1 ifconfig macvlan1 192.0.2.1/24
   ```

   Here, `wlp2s0` is the name of the interface. The name of the interface can be changed to `eth0`, if we are using Ethernet interface.

5. A combination of the above commands can be used to create two macvlans in bridge mode, link their respective network namespaces.

2.2 Ping between namespaces

This section explains how to ping from one namespace to another.

1. Ping between namespaces `net1`:

   ```
   sudo ip netns exec net1 ping -I 192.0.2.1/24 192.0.2.2/24
   ```

   Here, `192.0.2.1/24` & `192.0.2.2/24` are the IP addresses assigned to MACVLANs in namespaces `net1` and `net2` respectively.

2.3 Set delay and loss on MACVLANs

This section explains how to configure a MACVLAN to introduce delay and loss in the packet stream.

1. Set a constant delay of 30ms at macvlan interface `macvlan1`:

   ```
   ip netns exec net1 tc qdisc add dev macvlan1 root netem delay 30ms
   ```

2. Set a variable delay with jitter of +/- 10 on `macvlan1`:

   ```
   ip netns exec net1 tc qdisc add dev macvlan1 root netem delay 30ms
   10ms
   ```

3. Set packet loss of 20% on `macvlan1`

   ```
   ip netns exec net1 tc qdisc add dev macvlan1 root netem loss 20%
   ```

4. Delete rule set on macvlan interface:

   ```
   ip netns exec net2 tc qdisc del dev macvlan2 root
   ```

5. The above commands can also be used to measure the round-trip time between two network namespaces.

2.4 Secure copy between MACVLANs using SSH

This section describes the procedure to securely copy files between namespaces.

1. Add listening to namespace `net1`:

   ```
   ip netns exec net1 /usr/sbin/sshd -o PidFile=/run/sshd-net1.pid
   ```

   The path to sshd can be searched using `locate sshd`.

2. SSH from one namespace `net2` to macvlan1 in namespace `net1`:
   ```
   ip netns exec net2 ssh hostname@192.0.2.1
   ```

3. Copy files using macvlans in different namespaces:
   ```
   ip netns exec net2 scp /path-to-file-to-copy/file.pdf hostname@
   192.0.2.1:destination-path
   ```

2.5 Multiple MACVLANs in a namespace

Here, we explain how to configure several MACVLANs in one namespace, and some of the issues that arise.

1. Setup:
   1st Namespace: 5 MACVLAN interfaces (each MACVLAN interface with different loss and delay)
   2nd Namespace: 4 MACVLAN interfaces (each and every MACVLAN interface with the same loss and delay)

2. Experiment:
   Ping from 1st namespace to 2nd namespace
   Every time using different pairs of MACVLAN interfaces

3. Observation:
   Same loss and delay noted. The loss and delay correspond to MACVLAN interface to which we added the loss and delay first in their respective namespaces.

4. Explanation:
   For a namespace, only one pair of loss and delay parameters can be set. As tc transfers packets through the same physical interface.
   If we delete the first added loss and delay pair, then the next pair of loss and delay is taken up.

2.6 Set Maximum Bandwidth of a Link

Using `tc`, we can configure a maximum transmit rate for an interface.

1. Set bandwidth, burst and latency :
   ```
   ip netns exec net2 tc qdisc add dev macvlan2 root tbf rate 150 mbit
   burst 32 Kbit latency 20 ms
   ```

   The Token Bucket Filter is a classless queueing discipline available for traffic control with the tc(8) command. .

2.7 Setting priority in MACVLAN

Using `tc`, we can set priorities in traffic from MACVLAN.

1. Use the hierarchical token bucket `htb` queuing discipline

2. Create traffic classes using `tc class`

3. Use `tc filter`. It has options like priority (`prio`), protocol, port, classid, which can be configured to set the priorities for traffic classes.

3 References

1. Hangbin Liu, "Introduction to Linux interfaces for virtual networking", https://developers.redhat.com/blog/2018/10/22/introduction-to-linux-interfaces-for-virtual-networking/macvlan. [Accessed: 7 April, 2020]

2. Pardyumna Dash,"Bandwidth Throttling with NetEM Network Emulation", https://opensourceforu.com/2012/06/bandwidth-throttling-netem-network-emulation/. [Accessed: 8 April, 2020]

3. "IP-NETNS(8)", http://man7.org/linux/man-pages/man8/ip-netns.8.html. [Accessed: 8 April, 2020]

4. "Traffic Control", https://wiki.debian.org/TrafficControl. [Accessed: 1 May, 2020]