Daily Report File for the Semester Project

Furkan Karakaş

February 20, 2020

9 February 2020

I did the first installation. I also forked the tendermint repository to my GitHub repository `https://github.com/FurkanKarakas/tendermint`. We can run a single node tendermint at the moment but to run local testnets, we can utilize *docker*. To start a 4 node testnet, run:

```
make localnet-start
```

After executing multiple nodes, a new folder `build` is created. In this folder, each node has a log file, which is called `tendermint.log`. These log files contain the commit and execute steps of the program, written with their corresponding timestamps.

The nodes bind their RPC servers to ports 26657, 26660, 26662, and 26664 on the host. This file creates a 4-node network using the localnode image. The nodes of the network expose their P2P and RPC endpoints to the host machine on ports 26656-26657, 26659-26660, 26661-26662, and 26663-26664 respectively.

To stop and restart the environment, one can use the following commands:

```
make build-linux
make localnet-stop
make localnet-start
```

20 February 2020

We need to simulate network delays and packet losses in the Docker containers. There is a very good article here about it: `https://alexei-led.github.io/post/pumba_docker_netem/`. Linux allows us to manipulate the traffic flow using a tool called `tc`, which is available in `iproute2`. Another tool called `netem` is an extension of it. It allows us to emulate network failures such as delay, packet loss, packer reorder, duplication, corruption, and bandwidth rate.

For controlling traffic flow in the Docker containers, we will be using a software called *pumba*, which utilizes Linux traffic control and network emulation softwares *tc* and *netem*, respectively. A good documentation of it could be found at the website `https://github.com/alexei-led/pumba`.

Figure 1 illustrates the basic topology of a Docker host. Basically, `veth` interfaces are used in order to create a communication between each container and the docker itself. As I type the `ifconfig` command in the command line, I am able to see the aforementioned virtual interfaces, each corresponding to $node_i$ where $i \in \{0, 1, 2, 3\}$:

```
veth282987a: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::dc1b:3cff:fe06:7484  prefixlen 64  scopeid 0x20<link>
```
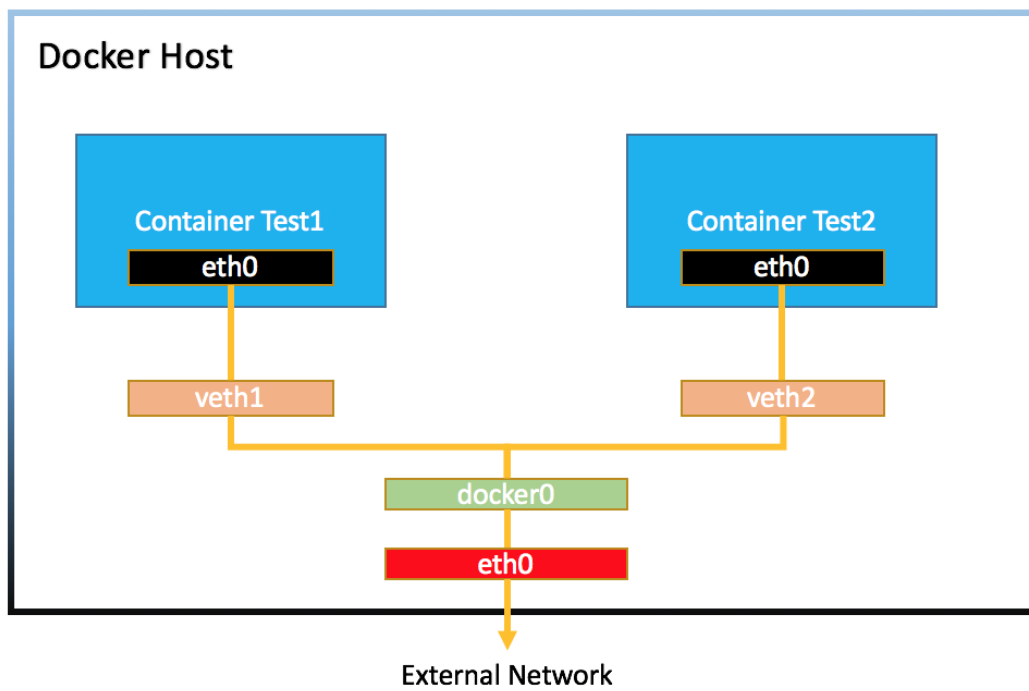
Figure 1: The basic network topology inside a Docker host

```
        ether de:1b:3c:06:74:84   txqueuelen 0  (Ethernet)
        RX packets 94167   bytes 57527418 (57.5 MB)
        RX errors 0   dropped 0   overruns 0   frame 0
        TX packets 94874   bytes 58402540 (58.4 MB)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0

veth454c67a: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>   mtu 1500
        inet6 fe80::24fd:25ff:fee5:45f   prefixlen 64   scopeid 0x20<link>
        ether 26:fd:25:e5:04:5f   txqueuelen 0  (Ethernet)
        RX packets 95109   bytes 58670384 (58.6 MB)
        RX errors 0   dropped 0   overruns 0   frame 0
        TX packets 95470   bytes 58291946 (58.2 MB)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0

vethba77776: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>   mtu 1500
        inet6 fe80::b0e9:54ff:fe9d:52cb   prefixlen 64   scopeid 0x20<link>
        ether b2:e9:54:9d:52:cb   txqueuelen 0  (Ethernet)
        RX packets 95284   bytes 58311392 (58.3 MB)
        RX errors 0   dropped 0   overruns 0   frame 0
        TX packets 94967   bytes 58303854 (58.3 MB)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0

vethe97798a: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>   mtu 1500
        inet6 fe80::4c55:84ff:fe93:4da8   prefixlen 64   scopeid 0x20<link>
        ether 4e:55:84:93:4d:a8   txqueuelen 0  (Ethernet)
        RX packets 94980   bytes 58487742 (58.4 MB)
        RX errors 0   dropped 0   overruns 0   frame 0
        TX packets 95391   bytes 58199768 (58.1 MB)
        TX errors 0   dropped 0 overruns 0   carrier 0   collisions 0
```

Following is a simple command to delay the traffic at `node0` 20 seconds with a duration of 30 seconds:

```
sudo pumba netem --duration 30s --interface eth0 --tc-image \
'gaiadocker/iproute2' delay --time 20000 node0
```