DeepDive:
Local Docker Networking

Martin Leischner
April 7, 2022

Docker local networking structure

- **Problem statement: The docker local networking structure is very complex**
  - **Every docker container running on the local system is a communicating micro service.**
  - **A lot of interfaces.**
  - **Local virtual networks build by bridged subnets.**
  - **Internal switching, routing and gateway routing.**
- **Building blocks of the linux/ubuntu local networking infrastructure:**
  - **Interfaces**
    → `ip addr show` / `ip a`
  - **Bridges**
    → `brctl show`
  - **Subnets**
    → via interfaces
  - **Routing tables**
    - **`ip route show table main` / `ip route show` / `ip r` :**
      Content of routing table main manageable by an administrator (even used by install). Useful in most cases.
    - **`ip route show table local` / `ip r s t local` :**
      routing table of local addresses managed by the kernel

Our network analysis methodology

- Building the docker infrastructure step-by-step:
  1. <u>basis</u>: Ubuntu server 20.04. with one standard interface (and with ssh)
  2. <u>add</u>: docker server/client (no container)
  3. <u>add</u>: running one simple container providing a webserver at port 80
  4. initialize docker swarm

- Analyze every building step by (only IPv4):
  - Interfaces
  - Bridges and subnets
  - Routing table
  - Connections and listening ports :

    `netstat -an` use grep additionally if necessary

    `-a` all active unix sockets, `-t` tcp sockets, `-u` udp sockets

    `-n` show ports as numbers (instead of resolving dns)

    `-l` only ports bound to listen

    `-p` show program name / PID

Step 1: Ubuntu server 20.04 with only one standard interface

- **Interfaces**

```
➢ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
                                                    group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ...
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
qlen 1000
    link/ether 00:0c:29:5e:fc:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.101/24 brd 192.168.178.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 ...
```

- **Bridges and subnets: none**

```
➢ brctl show
➢
```

Step 1: Ubuntu server 20.04 with only one standard interface

Routing table (ip r)

```
➢ ip r
default via 192.168.178.1 dev ens33 proto static
192.168.178.0/24 dev ens33 proto kernel scope link src 192.168.178.101
```

- Listening ports (via sudo netstat –tulpn)

```
➢ netstat -tulp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address       Foreign Address      State       PID/Program name
tcp          0      0 localhost:domain  0.0.0.0:*            LISTEN      930/systemd-resolve
tcp          0      0 0.0.0.0:ssh       0.0.0.0:*            LISTEN      984/sshd: /usr/sbin
tcp6         0      0 [::]:ssh          [::]:*               LISTEN      984/sshd: /usr/sbin
udp          0      0 localhost:domain  0.0.0.0:*                        930/systemd-resolve
```

Step 2: Ubuntu server 20.04. with Docker and nothing else

- **Interfaces**

```
➢ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ...
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
qlen 1000
    link/ether 00:0c:29:5e:fc:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.101/24 brd 192.168.178.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 ...
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default
    link/ether 02:42:75:43:9b:39 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

- **Questions on `docker0` Interface:**
  - **How to interpret interface `docker0` in this context?**
  - **Why is it down?**

Step 2: Ubuntu server 20.04. with Docker and nothing else

- Bridges and subnets

```
➢ brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.024275439b39       no
```

- Questions on bridge `docker0`:
  - Why are there no interfaces?
  - Which subnet belongs to bridge `docker0` ?
  - Can you give a coherent explanation of the relationship between `docker0`-Bridge and `docker0`-Interface?

- Routing table (`ip r`):

```
➢ ip r
default via 192.168.178.1 dev ens33 proto static
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.178.0/24 dev ens33 proto kernel scope link src 192.168.178.101
```

Step 2: Ubuntu server 20.04. with Docker and nothing else

- Listening ports (via `netstat –tulpn`)

```
➢ sudo netstat -tulp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address         Foreign Address        State       PID/Program name
tcp        0      0 localhost:41851       0.0.0.0:*              LISTEN      983/containerd
tcp        0      0 localhost:domain      0.0.0.0:*              LISTEN      934/systemd-resolve
tcp        0      0 0.0.0.0:ssh           0.0.0.0:*              LISTEN      1006/sshd: /usr/sbi
tcp6       0      0 [::]:ssh              [::]:*                 LISTEN      1006/sshd: /usr/sbi
udp        0      0 localhost:domain      0.0.0.0:*                          934/systemd-resolve
```

- Connections (via `netstat –tupn`)

```
➢ sudo netstat -tupn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0     64 192.168.178.101:22      192.168.178.50:60668    ESTABLISHED 1442/sshd: mleisc2m
```

- Connections (via `netstat –tup`)

```
sudo netstat -tup
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0     64 lokserver:ssh           pc-home2.lau50c.h:60668 ESTABLISHED 1442/sshd: mleisc2m
```

Step 3: Ubuntu server 20.04. running one simple container

- **Interfaces**

```
➢ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ...
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
qlen 1000
    link/ether 00:0c:29:5e:fc:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.101/24 brd 192.168.178.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 ...
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:de:61:87:86 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 ...
9: vethc519f84@if8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0
state UP group default
    link/ether 52:e9:85:cc:99:ce brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 ...
```

- **Questions: Where is interface `9`? What does `vethc519f84@if8` mean?**

Step 3: Ubuntu server 20.04. running one simple container

- **Bridges and subnets**

```
➢ brctl show
bridge name     bridge id               STP enabled     interfaces
docker0         8000.0242de618786       no              vethc519f84
```

- **Routing table (ip r)**

```
default via 192.168.178.1 dev ens33 proto static
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1 linkdown
192.168.178.0/24 dev ens33  proto kernel  scope link  src 192.168.178.101
```

Zugriff auch mit IPv6 möglich!

- **Listening ports (via netstat –tulpn)**

```
➢ sudo netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address       Foreign Address     State       PID/Program name
tcp          0      0 127.0.0.1:46469   0.0.0.0:*           LISTEN      976/containerd
tcp          0      0 0.0.0.0:8080      0.0.0.0:*           LISTEN      2526/docker-proxy
tcp          0      0 127.0.0.53:53     0.0.0.0:*           LISTEN      926/systemd-resolve
tcp          0      0 0.0.0.0:22        0.0.0.0:*           LISTEN      992/sshd: /usr/sbin
tcp6         0      0 :::8080           :::*                LISTEN      2532/docker-proxy
udp          0      0 127.0.0.53:53     0.0.0.0:*                       926/systemd-resolve
```

Step 3: Ubuntu server 20.04. running one simple container

Go inside the container and look around!

- **Interfaces**

```
➢ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

- **Bridges and subnets: none**
- **Routing table (`ip r`)**

```
➢ ip r
default via 172.17.0.1 dev eth0
172.17.0.0/16 dev eth0 scope link  src 172.17.0.2
```

Step 3: Ubuntu server 20.04. running one simple container

Go inside the container and look around!

- **Listening ports (via `netstat -tulpn`)**

```
➢ netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 :::80                   :::*                    LISTEN      1/node
```

Started with PID 1

- **Running processes within the container**

```
➢ ps -e
PID   USER     TIME  COMMAND
    1 root      0:00 node miniwhoami.js
   27 root      0:00 sh
   51 root      0:00 ps -e
```

Step 4: Ubuntu server 20.04. + docker swarm init

```
docker swarm init --advertise-addr 192.168.178.101
```

- **Interfaces:**

```
➢ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:5e:fc:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.101/24 brd 192.168.178.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 ...
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:ae:70:e1:dc brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
8: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:f3:29:c3:d1 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
    inet6 ...
10: vethd944f19@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default
    link/ether ca:4c:2d:16:ea:79 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 ...
```

Step 4: Ubuntu server 20.04. + docker swarm init

- **Bridges and subnets**

```
➢ brctl show
bridge name             bridge id               STP enabled     interfaces
docker0                 8000.0242ae70e1dc       no
docker_gwbridge         8000.0242f329c3d1       no              vethd944f19
```

- **Routing table**

```
➢ ip r
default via 192.168.178.1 dev ens33 proto static
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
172.18.0.0/16 dev docker_gwbridge proto kernel scope link src 172.18.0.1
192.168.178.0/24 dev ens33 proto kernel scope link src 192.168.178.101
```

Step 4: Ubuntu server 20.04. + docker swarm init

- **Listening ports (via `sudo netstat –tulpn`)**

```
➢ sudo netstat -tulpn
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      2557/docker-proxy
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      929/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      998/sshd: /usr/sbin
tcp        0      0 127.0.0.1:36387         0.0.0.0:*               LISTEN      980/containerd
tcp6       0      0 :::8080                 :::*                    LISTEN      2564/docker-proxy
tcp6       0      0 :::22                   :::*                    LISTEN      998/sshd: /usr/sbin
tcp6       0      0 :::2377                 :::*                    LISTEN      1185/dockerd
tcp6       0      0 :::7946                 :::*                    LISTEN      1185/dockerd
udp        0      0 0.0.0.0:4789            0.0.0.0:*                           -
udp        0      0 127.0.0.53:53           0.0.0.0:*                           929/systemd-resolve
udp6       0      0 :::7946                 :::*                                1185/dockerd
```

Port 2377: For swarm managers, not for docker clients (→ TLS).

Port 4789: UDP for the container overlay network.

Port 7946: TCP/UDP for container network discovery.

Dangerous (but very practical): Enable TCP port 2375 for external connection to Docker API via http

.