CS60203: Design Optimization of Computing Systems *November 17, 2024*

Network Namespaces and Linux Ethernet Bridge

Instructor: Sandip Chakraborty *Scribe: Chinmay Bhusari*

1 Network Namespaces

1.1 Overview of Network Namespaces

1.1.1 Concept of Namespaces

Namespaces in Linux are a fundamental aspect of containerization, allowing processes to have isolated views of certain system resources. When a process is placed in a new namespace, it has its own instance of that resource.

1.1.2 Purpose of Network Namespaces

Network namespaces isolate the network environment for processes. This means that each namespace can have its own:

- Network interfaces
- IP addresses
- Routing tables
- Firewall rules
- `/proc/net` and `/sys/class/net` entries

This isolation enables multiple applications or containers to run on the same host without interfering with each other's network configurations.

1.2 How Network Namespaces Work

1.2.1 Isolation Mechanism

When a new network namespace is created, the Linux kernel provides a new instance of the network stack for that namespace. This includes:

- **Network Interfaces**: The namespace starts with only the loopback interface (`lo`) unless additional interfaces are assigned.
- **IP Addressing**: Separate IP address configurations can be set within each namespace.
- **Routing Tables**: Each namespace maintains its own routing table, determining how packets are forwarded.
- **Netfilter Rules**: Firewall and NAT rules using `iptables` are specific to each namespace.

1.2.2 Communication Between Namespaces

By default, network namespaces are completely isolated. To enable communication:

- **Virtual Ethernet Devices**: Use `veth` pairs to create a link between namespaces.
- **Bridging**: Connect multiple namespaces to a common bridge to simulate a LAN.
- **Routing**: Set up routing rules and use tools like `ip route` to forward packets between namespaces.

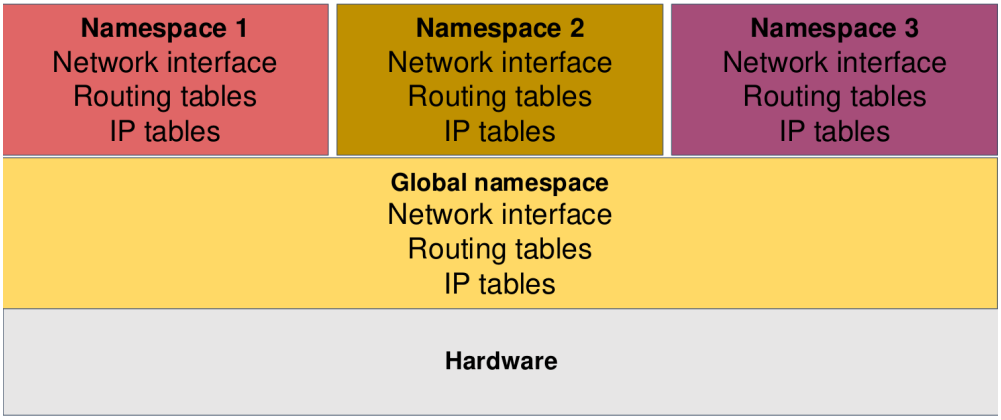

Figure 1: Isolation of Network Resources in Namespaces

1.3 Managing Network Namespaces

1.3.1 Creating a Network Namespace

Use the `ip` command to create network namespaces:

```
sudo ip netns add ns1
sudo ip netns add ns2
sudo ip netns add ns3
```

This creates three network namespaces named `ns1`, `ns2`, and `ns3`.

1.3.2 Listing Network Namespaces

To list existing network namespaces:

```
ip netns list
```

1.3.3 Deleting a Network Namespace

To delete a network namespace:

```
sudo ip netns delete ns1
```

1.3.4 Executing Commands in a Network Namespace

Use `ip netns exec` to run commands inside a namespace:

```
sudo ip netns exec ns1 bash
```

This opens a shell inside `ns1` where you can run commands as if they were executed in that namespace.

2 Connecting Namespaces with Virtual Ethernet Pairs

2.1 Virtual Ethernet Pairs (`veth`)

2.1.1 What are Virtual Ethernet Pairs?

A **virtual Ethernet pair** acts like a virtual cable connecting two network interfaces. It consists of two interconnected interfaces; packets sent on one appear on the other. They are essential for:

- Connecting network namespaces to each other.
- Connecting namespaces to the host network.
- Building complex network topologies within a single host.

2.1.2 Properties of `veth` Pairs

- **Bidirectional**: Traffic can flow in both directions.
- **Appears as Ethernet Interfaces**: Behave like regular Ethernet interfaces.
- **No Physical Medium**: Completely virtual, existing only in the kernel.

2.2 Creating and Assigning `veth` Pairs

2.2.1 Creating `veth` Pairs Between Namespaces

To connect multiple namespaces, you'll create `veth` pairs as follows:

1. **Connect `ns1` and `ns2`:**

```
# Create veth pair
sudo ip link add veth1 type veth peer name veth2

# Assign to namespaces
sudo ip link set veth1 netns ns1
sudo ip link set veth2 netns ns2
```

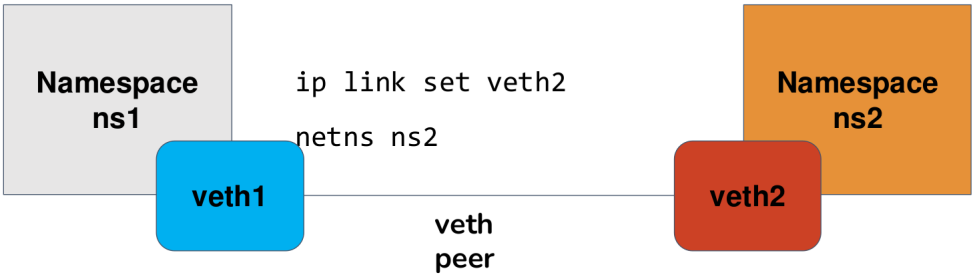

Figure 2: Veth pair Connecting 2 namespaces

2. **Connect `ns2` and `ns3`:**

```
# Create veth pair
sudo ip link add veth3 type veth peer name veth4

# Assign to namespaces
```

```
sudo ip link set veth3 netns ns2
sudo ip link set veth4 netns ns3
```

2.2.2 Configuring Interfaces Inside Namespaces

After assigning the `veth` interfaces to the respective namespaces, configure them:

1. **Configure** `ns1`:

```
sudo ip netns exec ns1 ip addr add 192.168.1.1/24 dev veth1
sudo ip netns exec ns1 ip link set veth1 up
sudo ip netns exec ns1 ip link set lo up
```

2. **Configure** `ns2`:

```
sudo ip netns exec ns2 ip addr add 192.168.1.2/24 dev veth2
sudo ip netns exec ns2 ip addr add 192.168.2.1/24 dev veth3
sudo ip netns exec ns2 ip link set veth2 up
sudo ip netns exec ns2 ip link set veth3 up
sudo ip netns exec ns2 ip link set lo up
```

3. **Configure** `ns3`:

```
sudo ip netns exec ns3 ip addr add 192.168.2.2/24 dev veth4
sudo ip netns exec ns3 ip link set veth4 up
sudo ip netns exec ns3 ip link set lo up
```

2.3 Linux Ethernet Bridge

2.3.1 Why Use a Linux Ethernet Bridge?

When connecting more than two namespaces, directly connecting them with multiple `veth` pairs becomes complex and unmanageable. A **Linux Ethernet bridge** simplifies the topology by acting as a virtual Layer 2 switch, allowing multiple interfaces to communicate seamlessly.

2.3.2 Creating a Linux Ethernet Bridge

Create a bridge interface in the host namespace:

```
sudo ip link add name br0 type bridge
sudo ip link set br0 up
```

2.4 Connecting Multiple Namespaces via Bridge

To connect ns1, ns2, and ns3 via a bridge, follow these steps:

2.4.1 Connecting `ns1` to `ns2`

1. **Create `veth` Pair**:

   ```
   sudo ip link add veth1 type veth peer name veth2
   ```

2. **Assign to Namespaces**:

   ```
   sudo ip link set veth1 netns ns1
   sudo ip link set veth2 netns ns2
   ```

3. **Configure Interfaces**:

   Inside `ns1`:

   ```
   sudo ip netns exec ns1 ip addr add 192.168.1.1/24 dev veth1
   sudo ip netns exec ns1 ip link set veth1 up
   sudo ip netns exec ns1 ip link set lo up
   ```

   Inside `ns2`:

   ```
   sudo ip netns exec ns2 ip addr add 192.168.1.2/24 dev veth2
   sudo ip netns exec ns2 ip link set veth2 up
   sudo ip netns exec ns2 ip link set lo up
   ```

2.4.2 Connecting `ns2` to `ns3`

4. **Create `veth` Pair**:

```
sudo ip link add veth3 type veth peer name veth4
```

5. **Assign to Namespaces**:

```
sudo ip link set veth3 netns ns2
sudo ip link set veth4 netns ns3
```

6. **Configure Interfaces**:

   Inside `ns2`:

```
sudo ip netns exec ns2 ip addr add 192.168.2.1/24 dev veth3
sudo ip netns exec ns2 ip link set veth3 up
sudo ip netns exec ns2 ip link set lo up
```

   Inside `ns3`:

```
sudo ip netns exec ns3 ip addr add 192.168.2.2/24 dev veth4
sudo ip netns exec ns3 ip link set veth4 up
sudo ip netns exec ns3 ip link set lo up
```

2.4.3 Creating and Connecting the Bridge

To bridge veth2 (from ns2 to ns1) and veth3 (from ns2 to ns3):

6. **Add `veth2` and `veth3` to Bridge**:

```
# Bring up veth2 and veth3 in the host namespace
sudo ip link set veth2 up
sudo ip link set veth3 up

# Add to bridge
sudo ip link set veth2 master br0
sudo ip link set veth3 master br0
```

7. **Ensure Bridge is Up**:

```
sudo ip link set br0 up
```

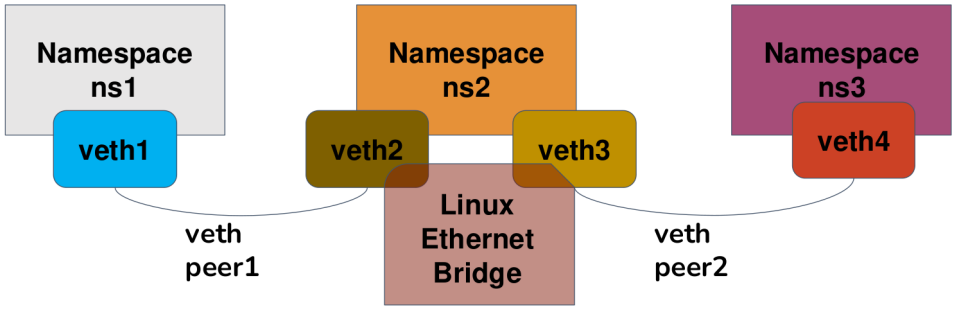

Figure 3: Linux Ethernet Bridge Connecting Multiple Namespaces

2.5 Testing Connectivity Between Namespaces via Bridge

To test connectivity:

```
# From ns1 to ns3
sudo ip netns exec ns1 ping 192.168.2.2

# From ns3 to ns1
sudo ip netns exec ns3 ping 192.168.1.1
```