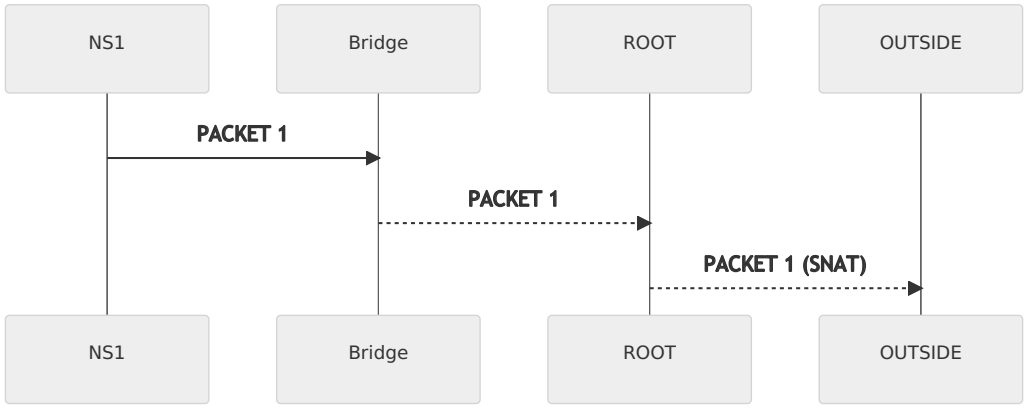
# Network Namespace with Internet and Ingress Traffic

In this doccumentation, we will cover

1. Namespace to Bridge Communication
2. Namespace to Root Namespace Communication
3. Internet Connectivity | ping 8.8.8.8
4. Access to Namespace from Outside of the network

Download PDF

## Egress Packet Flow

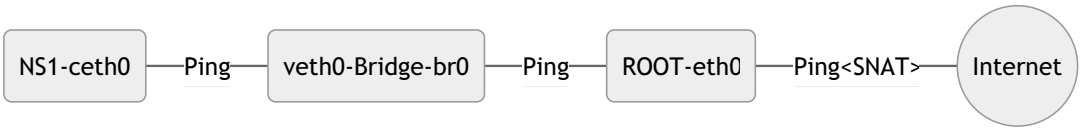



Let's say we are doing ping request from the NS1 to google dns 8.8.8.8

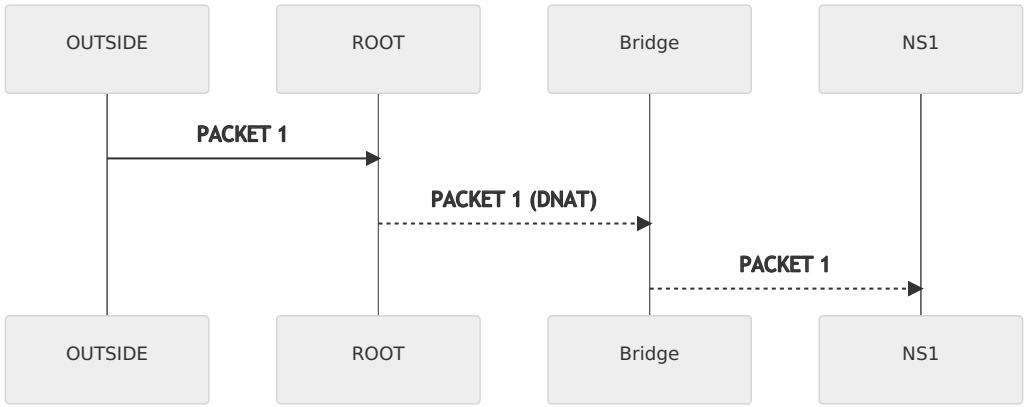
```
ping 8.8.8.8
```

So, how the network packet will travle?

1. The packet will reach from NS1 **ceth0** to the Bridge **br0** via **Default Gateway**
2. Packet will be **forwarded** from Bridge **br0** to RootNS **eth0**
3. Packet will go out from **eth0** to **outside** using **SNAT**

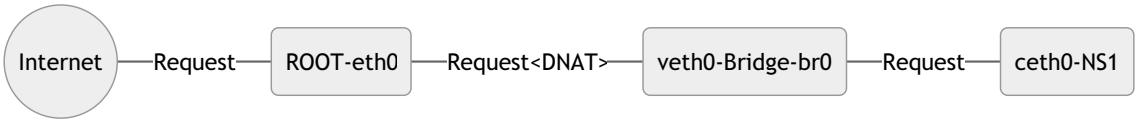



## Ingress Packet Flow




Let's assume we are running a server in NS1 listening on port 5000.
if we try to access this application from outside, how the packet will flow?

1. Packet will reach from **outside** to **eth0**
2. from **eth0** to **br0** using **DNAT**
3. **br0** (Default GW) to NS1 **ceth0**




## Create Network Namespace

```
sudo ip netns add netns1
```

Now varify that your namespaces has been created successfully

```
ip netns list
```

expected output: netns1

## Create a Bridge

A Bridge is a **Layer 2 Switch** which defines a network and can hold multipule namespaces within this network using **veth** (virtual ethernet) cable. For example we are going to define 192.168.0.0/16 network to this bridge. Let's create a bridge first.

```
sudo ip link add br0 type bridge
```

Now verify the bridge is created or not.

```
sudo ip link
```

Expected Output:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group de
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP mode DEFAULT
    link/ether 0a:b8:74:c1:7e:1c brd ff:ff:ff:ff:ff:ff
```

```
4: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default q
    link/ether ba:b7:43:d3:6f:ee brd ff:ff:ff:ff:ff:ff
```

As you can see the br0 bridge interface has been created but it's **DOWN**. Now turn it **UP** and assin an **IP address** to **br0**.

```
sudo ip link set br0 up
sudo ip addr add 192.168.0.1/16 dev br0
```

Now check if br0 can receive packet or not.

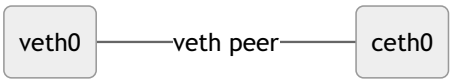
```
ping 192.168.0.1
```

Output:

```
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.042 ms
```

## Configure a veth cable

veth cable




```
sudo ip link add veth0 type veth peer name ceth0
```

**veth0** will be connected to the **Bridge** while the other side of the cable **ceth0** will be connected to the **netns1**. Now verify the interfaces are created or not.

```
sudo ip link
```

Now connect the **ceth0** into the **netns1**

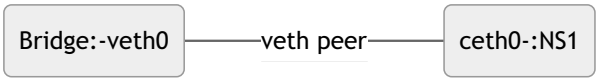
```
sudo ip link set ceth0 netns netns1
```

Now if you check `ip link`, you will see the **ceth0** is **missing**, because it **belongs** to **netns1** now.

After that, Connect the other side of the cable **veth0** into the **Bridge**

```
sudo ip link set veth0 master br0
```

Now let's visualize the outcome




## NS1 to Bridge Communication

Our objective is here to stablish communication between Namespace **netns1** to Bridge **br0**


Enter into **netns1** namespace and check the interfaces.

```
sudo nsenter --net=/var/run/netns/netns1
ip link
```

Output:

```
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
8: ceth0@if9: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group def
    link/ether 3e:32:88:75:f1:43 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

We need to turn UP both the **lo** and **ceth0** and assign an **IP** address to ceth0

```
ip link set lo up
ip link set ceth0 up
ip addr add 192.168.0.2/16 dev ceth0
```

Let's verify with `ip addr`
output:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 10
    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 ::1/128 scope host
       valid_lft forever preferred_lft forever
8: ceth0@if9: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state LOWERLAYER
    link/ether 3e:32:88:75:f1:43 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.0.2/16 scope global ceth0
       valid_lft forever preferred_lft forever
```

Notice! chek the ceth0 carefully, you can see the **LOWERLAYERDOWN**, which means the other side of the cable **veth0** which is still **DOWN**, but we will turn it UP now.

```
exit
```

logging out from Namespace **netns1**

We are in **RootNS** now, turn **UP** the **veth0** interface

```
sudo ip link set veth0 up
```

Now, log into the **netns1** and **ping** to **br0** (192.168.0.1)

```
ping 192.168.0.1
```

Output

```
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.065 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=0.047 ms
```

Communication between Namespace to Bridge is Successful.

```
exit
```

Logging out from the Namespace 1

## NS1 to RootNS Communication

Our objective is here to stablish communication between Namespace *netns1* **ceth0** to *Root NS* **eth0** via *Bridge* **br0**
Lookup eth0 IP address

```
ip addr show eth0
```

Output

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default
    link/ether 0a:b8:74:c1:7e:1c brd ff:ff:ff:ff:ff:ff
    inet 172.31.8.154/20 brd 172.31.15.255 scope global dynamic eth0
       valid_lft 3106sec preferred_lft 3106sec
    inet6 fe80::8b8:74ff:fec1:7e1c/64 scope link
       valid_lft forever preferred_lft forever
```

So **eth0** IP Adress is **172.31.8.154**

Now, log into the Namespace **netns1** and ping to 172.31.8.154

```
sudo nsenter --net=/var/run/netns/netns1
ping 172.31.8.154
```

Output

```
ping: connect: Network is unreachable
```

Let's check the route table

```
route
```

Output:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 ceth0
```

**Root Cause:**

> Opps, The IP doesn't match with any route table entry. Route table match an IP Address using *Longest Prefix Matching* Algorithm and Looks like the Karnel Level **Route Table** has **no information** to send the packet **out** to **eth0** 172.31.8.154/20 network.

**Solution:**

> So need to add a Default Gateway in the route table so that **any not matching** IP address will be forwarded via

> **br0** interface having IP **192.168.0.1**

Ok let's add the default gateway into the Routing Table of NS1

```
ip route add default via 192.168.0.1
route -n
```

Output:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.0.1     0.0.0.0         UG    0      0        0 ceth0
192.168.0.0     0.0.0.0         255.255.0.0     U     0      0        0 ceth0
```

Let's ping to **eth0** again

```
ping 172.31.8.154
```

Output:

```
PING 172.31.8.154 (172.31.8.154) 56(84) bytes of data.
64 bytes from 172.31.8.154: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 172.31.8.154: icmp_seq=2 ttl=64 time=0.045 ms
64 bytes from 172.31.8.154: icmp_seq=3 ttl=64 time=0.066 ms
```

## Ping 8.8.8.8 from NS1

So far we have done NS to Bridge and Bridge to Root NS Communication. Now let's take a walk outside. Let's ping 8.8.8.8 from netns1

```
ping 8.8.8.8
```

Outout:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
```

Now this scenario is little bit tricky, it's not like the Network is unreachable but some how the packet is blocked or stuck in between somewhere. Lucky that Linux has a binary named **tcpdump** to observe the network interfaces to **debug** the **packet flow**.

### Packet Debugging

Now open a new SSH session and observe our gateway **br0** in **RootNS** first.

```
sudo tcpdump -i br0 icmp
```

Output:

```
listening on br0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:29:55.680062 IP ip-192-168-0-2.ap-south-1.compute.internal > dns.google: ICMP echo rec
07:29:56.704054 IP ip-192-168-0-2.ap-south-1.compute.internal > dns.google: ICMP echo rec
07:29:57.728056 IP ip-192-168-0-2.ap-south-1.compute.internal > dns.google: ICMP echo rec
```

Looks like br0 is receiving icmp packets, but how about eth0. Let's check **eth0** interface now.

```
sudo tcpdump -i eth0 icmp
```

Output:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

No packets, There is something wrong. Let's check IPv4 forwarding **/proc/sys/net/ipv4/ip_forward**

```
cat /proc/sys/net/ipv4/ip_forward
```

output: 0

**Root Cause**: IP Forwarding is diabled. Lets **Enable IP Forwarding**

```
sudo nano /proc/sys/net/ipv4/ip_forward
```

change o to 1 and save it, then check it again

```
cat /proc/sys/net/ipv4/ip_forward
```

output: 1

Now observe eth0 again

```
sudo tcpdump -i eth0 icmp
```

Output:

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
07:39:04.544093 IP ip-192-168-0-2.ap-south-1.compute.internal > dns.google: ICMP echo rec
07:39:05.568071 IP ip-192-168-0-2.ap-south-1.compute.internal > dns.google: ICMP echo rec
07:39:06.592091 IP ip-192-168-0-2.ap-south-1.compute.internal > dns.google: ICMP echo rec
```

But still the ping it stuck in **netns1** why is that?
Notice! that the source ip is 192.168.0.2 which is trying to reach out to the google dns 8.8.8.8. 192.168.0.2 is a **private ip**

## Why I can not access internet through Private IP ?

Think of your **IP address** like your **full postal address**. Letters sent to you from anywhere get delivered by the post service reading the address and getting it to the right place.

A **Private IP** address is like having **my bedroom** as the **whole address**. It's Useful inside the house, but of **no use** whatsoever for a letter **trying to reach** you **from another country**. The poor postman does not know where to start, so the letter gets put in the bin. and you get no data.

Simply put, where would the return packets go?
You may be able to send a packet out (*though it'd probably be dropped by a firewall rather early*), but how would anyone get a packet **back to you**? Packets are solely addressed by IP address, and private IPs are, by design, **not unique**. If you do manage to get a packet to a server with a return address of **192.168.0.1**, where would it send its answer?

## Solution

To resolve this problem we somehow need to convert the private ip to public ip. This is called **NAT** (*Network Address Translation*)

We need to add a **NAT** rule in **IP Table** into the **POSTROUTING** chain

```
sudo iptables -t nat -A POSTROUTING -s 192.168.0.0/16 ! -o br0 -j MASQUERADE
```

We have Done **SNAT** ( Source IP NAT ) here
it means we are adding (-A) a IP MASQUERADE Rule into the NAT table (-t) where packets coming from source (-s) 192.168.0.0/16 network via output (-o) interface br0

More about iptables

Now Log into the Namespace **netns1** and try to ping 8.8.8.8 again

```
sudo nsenter --net=/var/run/netns/netns1
ping 8.8.8.8
```

Output:

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=109 time=1.67 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=1.68 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=1.71 ms
```

Eureka!! We have done it. We are able to ping outside

## Reach to your NS1 from outside

### Run a HTTP Server in NS1

Let' log into the **netns1** and fpr example run python3 **http** module listening to **port 5000**

```
sudo nsenter --net=/var/run/netns/netns1
python3 -m http.server --bind 192.168.0.2 5000
```

### Check Accessibility

Open a new SSH Session and let's try to access from RootNS, Let's Check access first.

```
telnet 192.168.0.2 5000
```

output:

```
Trying 192.168.0.2...
Connected to 192.168.0.2.
Escape character is '^]'.
```

Success! Now initiate a curl request

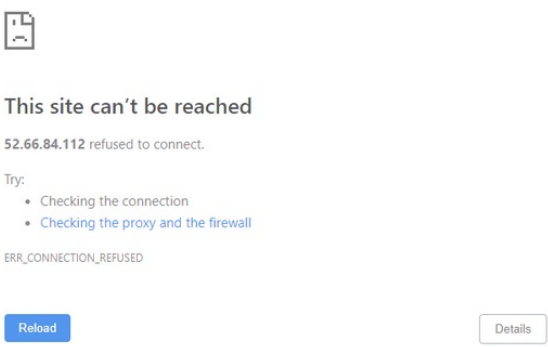
```
curl 192.168.0.2:5000
```

output:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href=".bash_history">.bash_history</a></li>
<li><a href=".bash_logout">.bash_logout</a></li>
<li><a href=".bashrc">.bashrc</a></li>
<li><a href=".cache/">.cache/</a></li>
<li><a href=".profile">.profile</a></li>
<li><a href=".ssh/">.ssh/</a></li>
<li><a href=".sudo_as_admin_successful">.sudo_as_admin_successful</a></li>
</ul>
<hr>
</body>
</html>
```

Awesome we have go the default response. That mean out HTTP server in Up and Running on port 5000.

Let's try to access using `public-ip:5000` from outside. In my case my VM has public ip 52.66.84.112 so I will hit `52.66.84.112:5000` from browser.

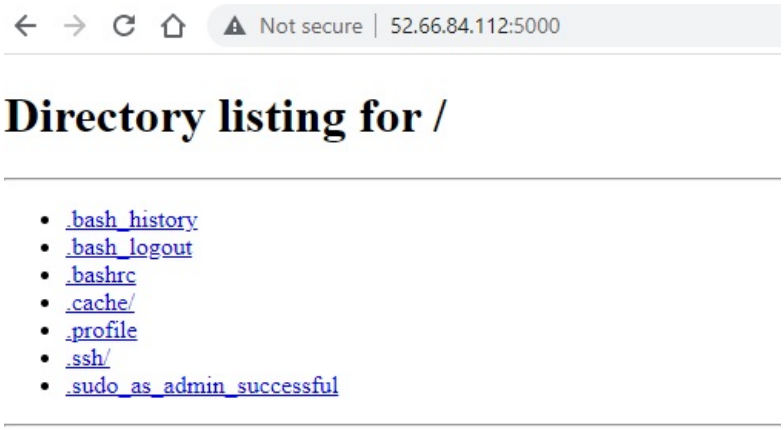



This is expected. Likewise, we need to do **DNAT** (Destination IP NAT)

```
sudo iptables -t nat -A PREROUTING -d 172.31.8.154 -p tcp -m tcp --dport 5000 -j DNAT --t
```

Meaning, any request from exact `matched (-m)` **tcp** `protocol (-p)` with `Destination (-d)` IP 172.31.8.154 (*eth0 ip*) with `destination port (--dport)` 5000 will jump -j into **DNAT** rule to destination `192.168.0.2:5000`

Let's try again from your browser.




Congratulation! We have configured both Ingress and Egress Traffic successfully.

## Reference


1. https://serverfault.com/questions/684439/how-to-connect-custom-network-namespace-to-root-network-namespace
2. https://wiki.archlinux.org/title/Network_bridge
3. https://unix.stackexchange.com/questions/255484/how-can-i-bridge-two-interfaces-with-ip-iproute2
4. https://gist.github.com/dpino/6c0dca1742093346461e11aa8f608a99
5. https://medium.com/@abhishek.amjeet/container-networking-using-namespaces-part1-859d317ca1b8
6. https://serverfault.com/questions/568839/linux-network-namespaces-ping-fails-on-specific-veth
7. https://devconnected.com/how-to-add-route-on-linux/
8. https://www.ibm.com/support/pages/using-tcpdump-verify-icmp-polling
9. https://medium.com/skilluped/what-is-iptables-and-how-to-use-it-781818422e52
10. https://www.quora.com/Why-can-we-not-access-the-Internet-through-a-private-IP