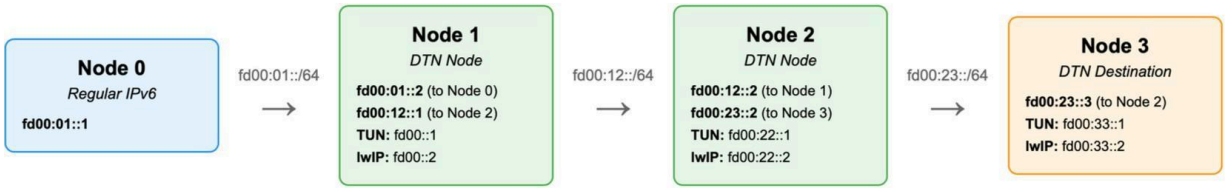
Setup Guide IPv6-DTN Testbed

Step 1: Setting up the virtual machines

The virtual machines have to be set up to any desired scenario. For this setup guide, the following topology is assumed:

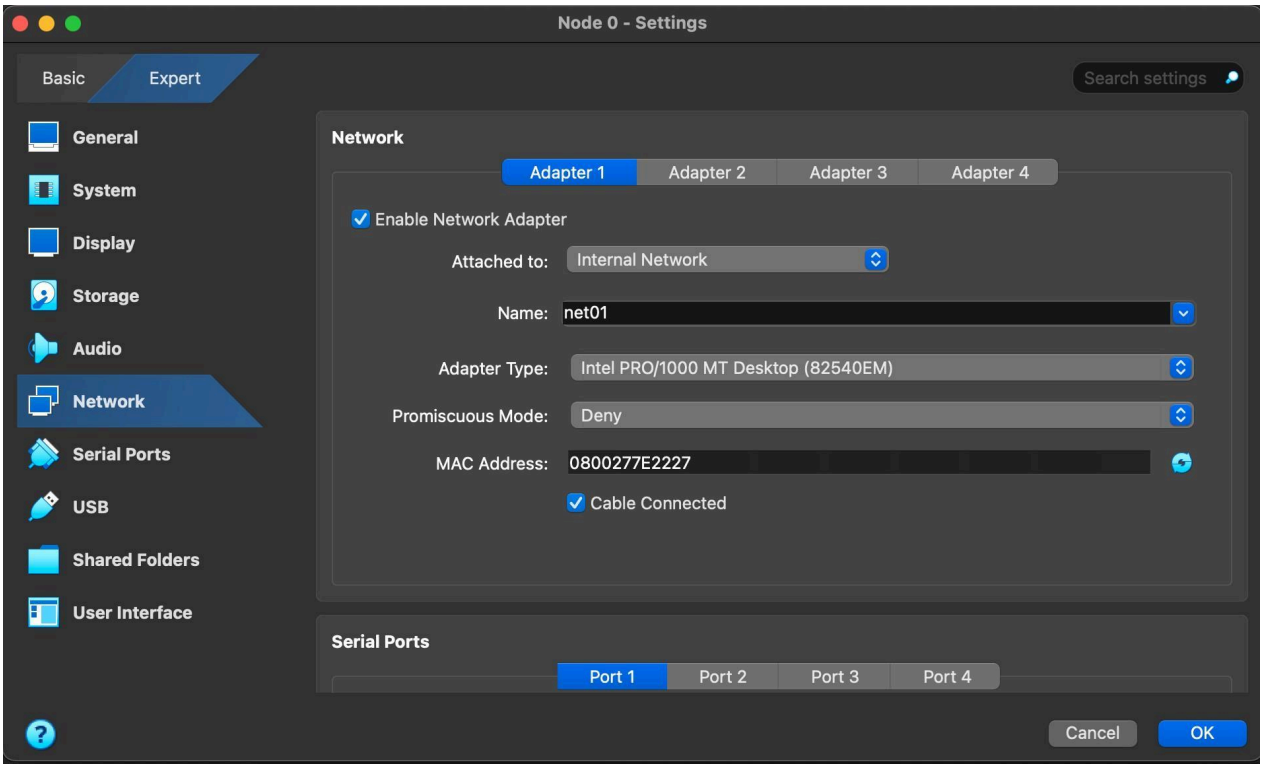


Each node represents one virtual machine. In this case, we use VirutalBox (https://www.virtualbox.org/). The code could also be run on actual devices like raspberries or other different devices using the linux tcp/ip stack.

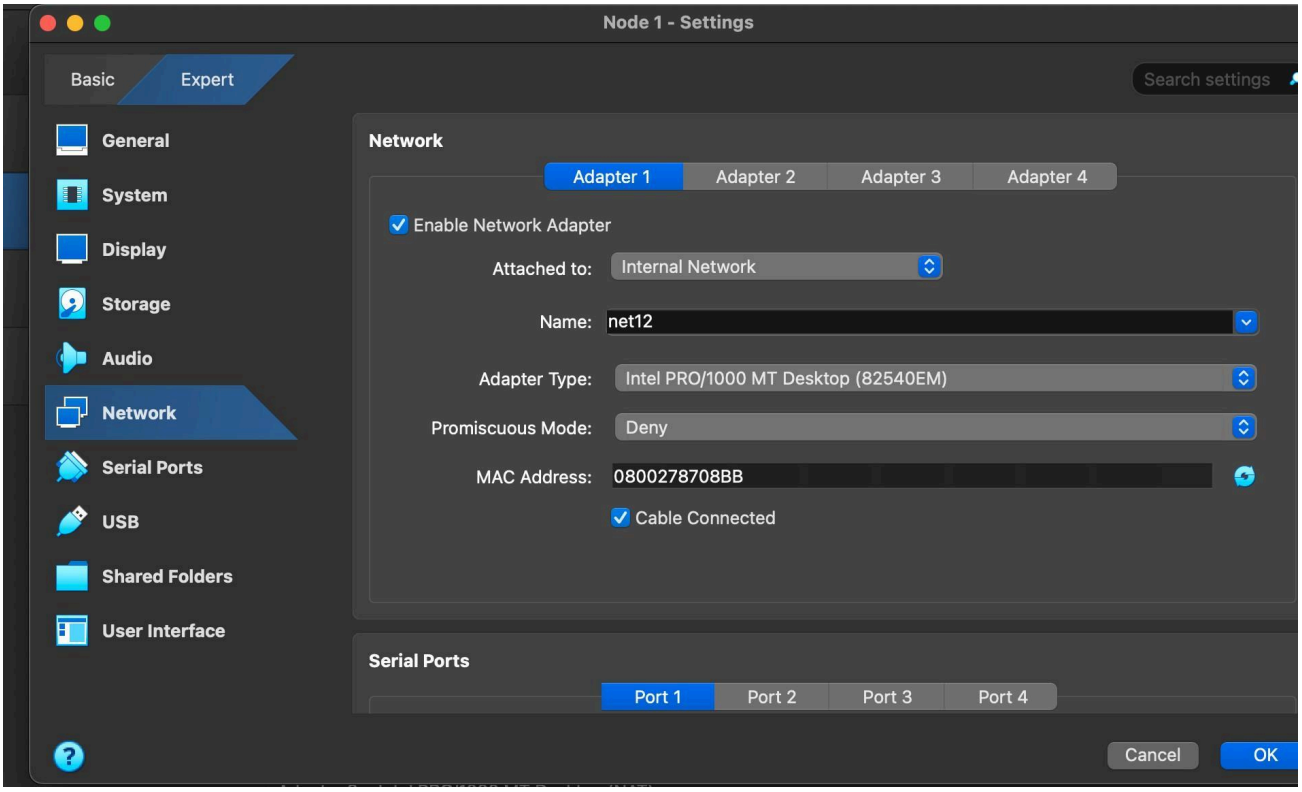
The virtual machines in virtual box have the following network settings to connect to each other via subnets.

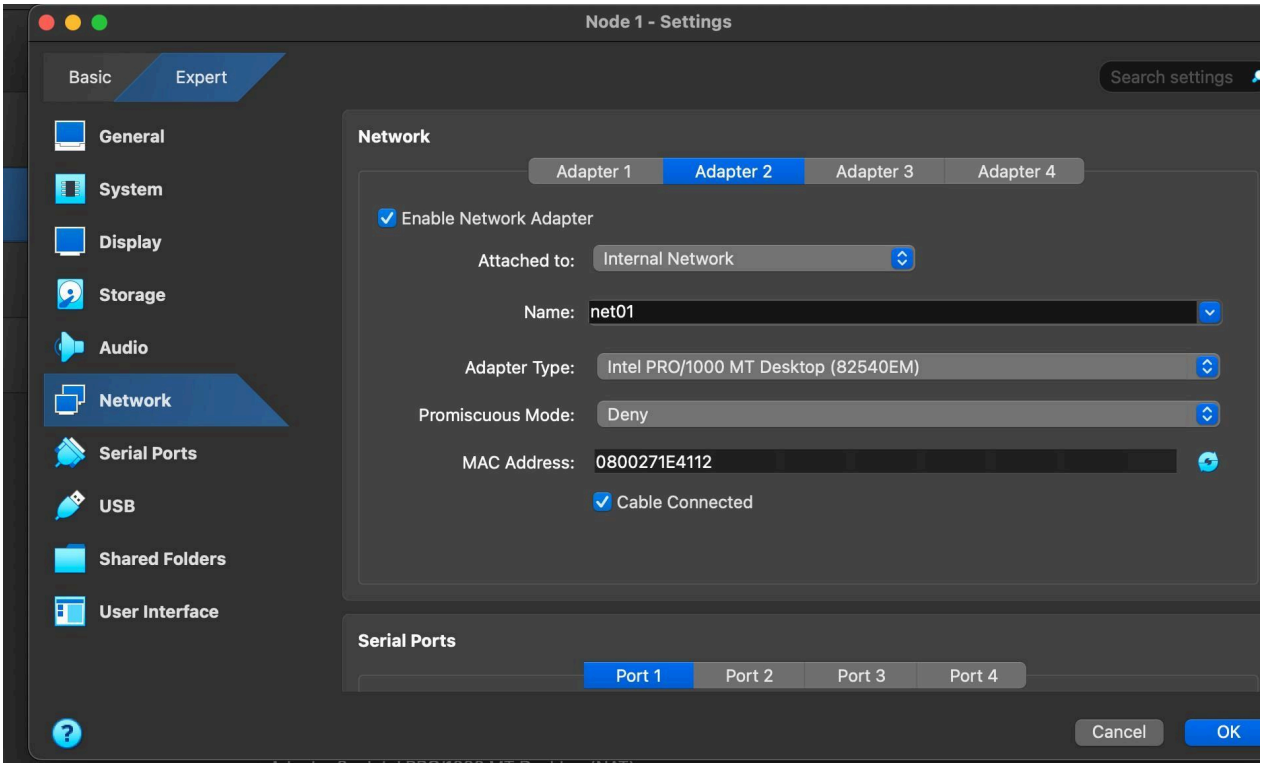
It is important to leave in each node the NAT interface active so that the names of the other interfaces correspond with the ones in the configuration files for step 2.

Node 0:

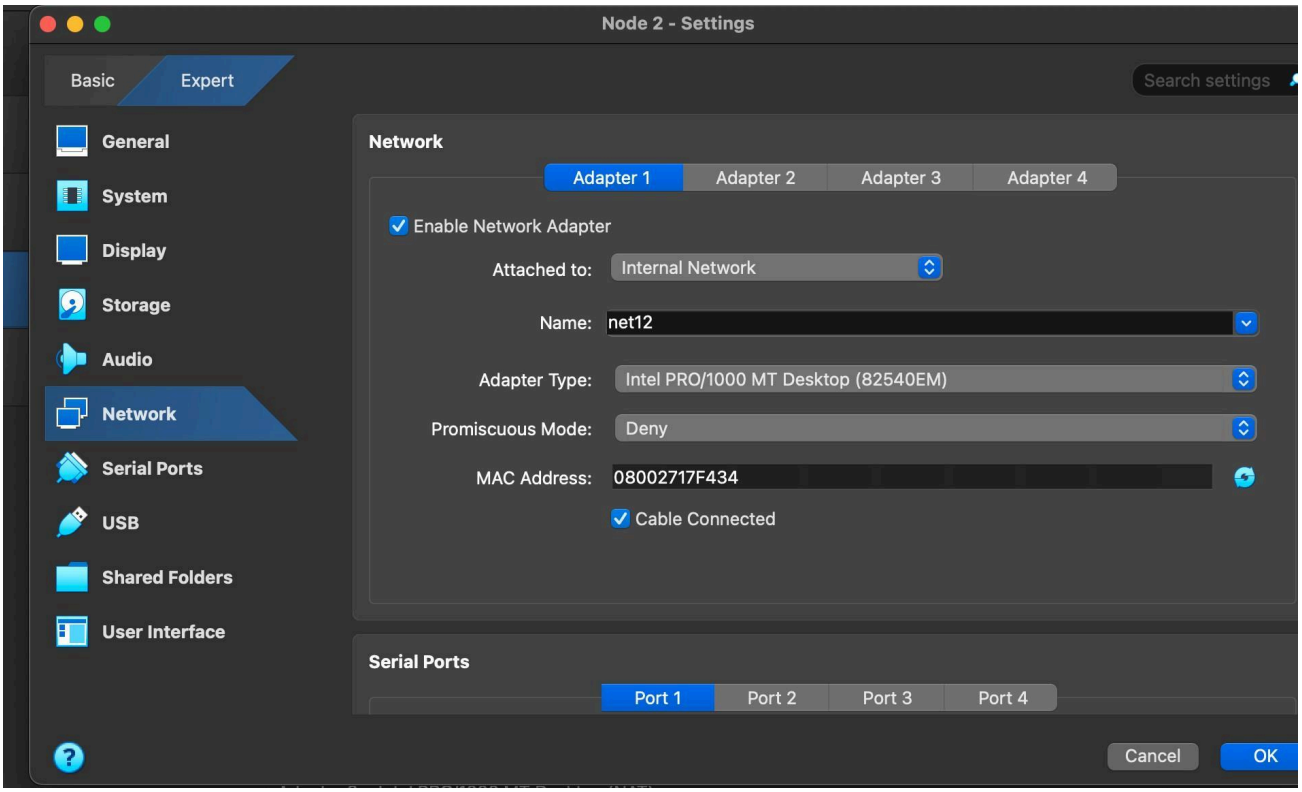


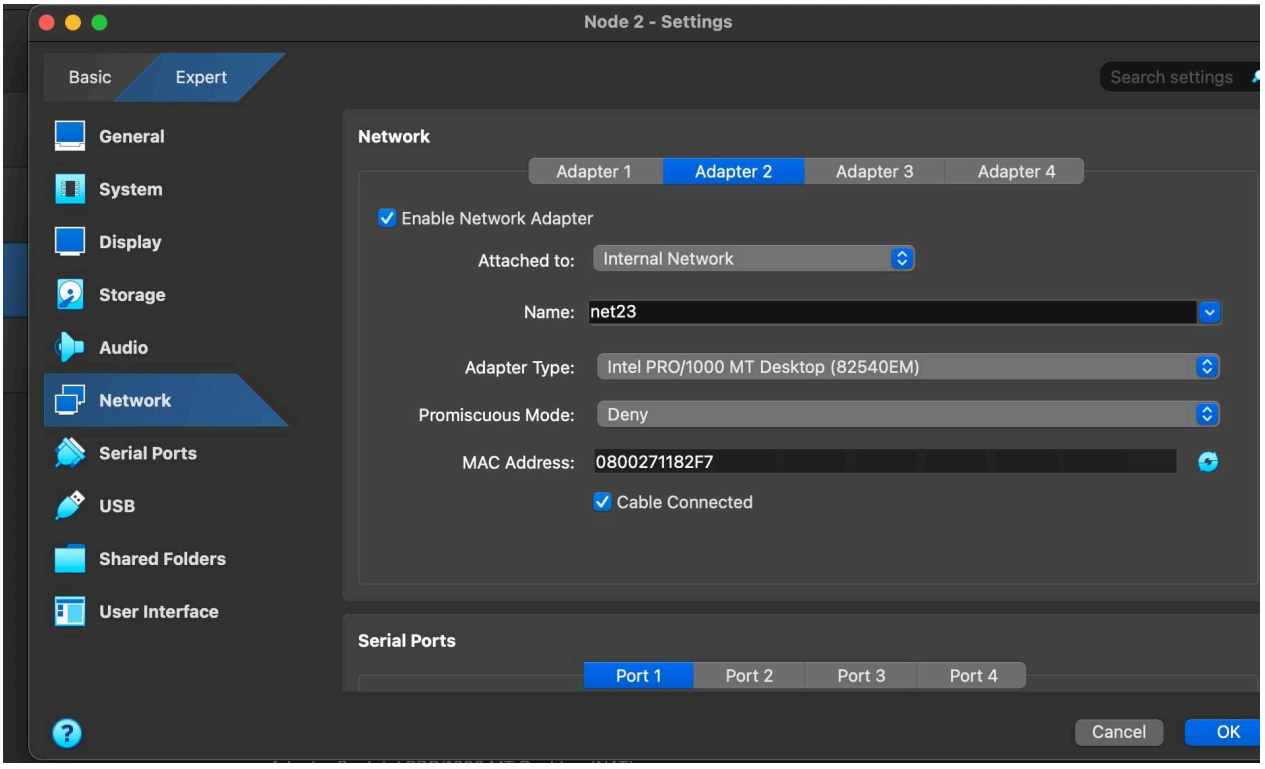
Node 1:


Node 1 - Settings
Basic
Expert
Search settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to:
Internal Network
Name:
net12
Adapter Type:
Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:
Deny
MAC Address:
0800278708BB
Cable Connected
Serial Ports
Port 1
Port 2
Port 3
Port 4
Cancel
OK

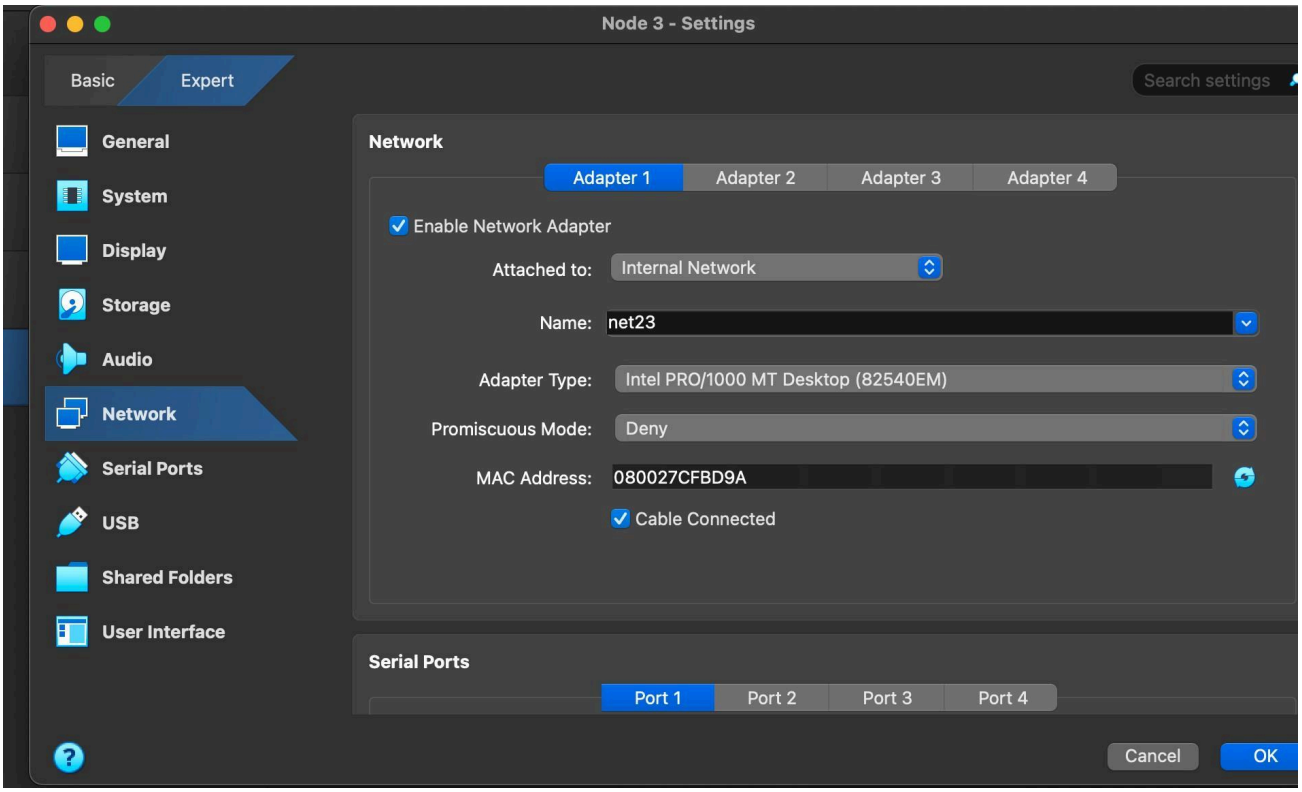

Node 1 - Settings
Basic
Expert
Search settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to:
Internal Network
Name:
net01
Adapter Type:
Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:
Deny
MAC Address:
0800271E4112
Cable Connected
Serial Ports
Port 1
Port 2
Port 3
Port 4
Cancel
OK

Node 2:


Node 2 - Settings
Basic
Expert
Search settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to:
Internal Network
Name:
net12
Adapter Type:
Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:
Deny
MAC Address:
08002717F434
Cable Connected
Serial Ports
Port 1
Port 2
Port 3
Port 4
Cancel
OK


Node 2 - Settings
Basic
Expert
Search settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to:
Internal Network
Name:
net23
Adapter Type:
Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:
Deny
MAC Address:
0800271182F7
Cable Connected
Serial Ports
Port 1
Port 2
Port 3
Port 4
Cancel
OK

Node 3:


Node 3 - Settings
Basic
Expert
Search settings
General
System
Display
Storage
Audio
Network
Serial Ports
USB
Shared Folders
User Interface
Network
Adapter 1
Adapter 2
Adapter 3
Adapter 4
Enable Network Adapter
Attached to:
Internal Network
Name:
net23
Adapter Type:
Intel PRO/1000 MT Desktop (82540EM)
Promiscuous Mode:
Deny
MAC Address:
080027CFBD9A
Cable Connected
Serial Ports
Port 1
Port 2
Port 3
Port 4
Cancel
OK

Step 2: Settings inside the virtual machines

In order to achieve the desired behavior and provide the necessary interfaces to the program, inside the virtual machines the following steps have to be taken:

- Enable IPv6 forwarding
- Create a persistent TUN interface on startup
- Assign the correct addresses to all interfaces with gateways
- Create rules to pass all traffic trough TUN

To achieve this, we have to adjust “/etc/network/interfaces” on all virtual machines and create “/etc/systemd/system/tun0.service” and “/usr/local/sbin/setup-tun0.sh” on all virtual machines.

Node 0:

For /etc/network/interfaces we have to add:

```
auto enp0s8
iface enp0s8
inet6 static
        address fd00:01::1
        netmask 64
        gateway fd00:01::2

post-up sysctl -w net.ipv6.conf.all.forwarding=1
post-up ip -6 route add fd00:12::/64 via fd00:01::2
post-up ip -6 route add fd00::/64 via fd00:01::2
post-up ip -6 route add fd00:22::/64 via fd00:01::2
post-up ip -6 route add fd00:33::/64 via fd00:01::2
```

Node 1:

For /etc/network/interfaces we have to add:

```
auto enp0s8
iface enp0s8 inet6 static
        address fd00:12::1
        netmask 64
        gateway fd00:12::2


post-up ip -6 route add fd00:23::/64 via fd00:12::2
post-up ip -6 route add fd00:33::/64 via fd00:12::2
post-up ip -6 route add fd00:22::/64 via fd00:12::2

auto enp0s9
iface enp0s9 inet6 static
        address fd00:01::2
        netmask 64

post-up sysctl -w net.ipv6.conf.all.forwarding=1
```

We create /etc/systemd/system/tun0.service with:

```
[Unit]
Description=Set up tun0 interface
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/setup-tun0.sh
ExecStop=/usr/bin/ip link delete tun0
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

And we create /usr/local/sbin/setup-tun0.sh with:

```
#!/bin/bash
set -e
ip tuntap add dev tun0 mode tun || true
ip addr add 10.0.0.1/24 dev tun0 || true
ip -6 addr add fd00::1/64 dev tun0 || true
ip link set tun0 up

ip6tables -t mangle -F PREROUTING
ip6tables -t mangle -F OUTPUT
ip6tables -t filter -F INPUT
ip -6 rule del prio 10000 fwmark 1 table 100 2>/dev/null || true
ip -6 rule del prio 10000 2>/dev/null || true

ip6tables -t mangle -A PREROUTING -d fd00:01::2 -j TEE --gateway fd00::2
ip6tables -t mangle -A PREROUTING -d fd00:12::1 -j TEE --gateway fd00::2
ip6tables -t filter -A INPUT -d fd00:01::2 -j DROP
ip6tables -t filter -A INPUT -d fd00:12::1 -j DROP
ip6tables -t mangle -A PREROUTING -i enp0s8 -m addrtype ! --dst-type LOCAL -j MARK
--set-mark 1
ip6tables -t mangle -A PREROUTING -i enp0s9 -m addrtype ! --dst-type LOCAL -j MARK
--set-mark 1
ip6tables -t mangle -A OUTPUT -j MARK --set-mark 1

if ! ip -6 rule list | grep -q "fwmark 1.*lookup 100"; then
    ip -6 rule add fwmark 1 table 100 priority 10000
fi
ip -6 route replace default via fd00::2 dev tun0 table 100
```

Node 2:

/etc/network/interfaces:

```
auto enp0s8
iface enp0s8 inet6 static
        address fd00:12::2
        netmask 64
post-up ip -6 route add fd00::/64 via fd00:12::1
post-up ip -6 route add fd00:01::/64 via fd00:12::1

auto enp0s9
iface enp0s9 inet6 static
        address fd00:23::2
        netmask 64

post-up ip -6 route add fd00:33::/64 via fd00:23::3
post-up sysctl -w net.ipv6.conf.all.forwarding=1
```

/etc/systemd/system/tun0.service:

```
[Unit]
Description=Set up tun0 interface
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/setup-tun0.sh
ExecStop=/usr/bin/ip link delete tun0
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

/usr/local/sbin/setup-tun0.sh:

```
#!/bin/bash
set -e
ip tuntap add dev tun0 mode tun || true
ip addr add 10.0.2.1/24 dev tun0 || true
ip -6 addr add fd00:22::1/64 dev tun0 || true
ip link set tun0 up

ip6tables -t mangle -F PREROUTING
ip6tables -t mangle -F OUTPUT
ip6tables -t filter -F INPUT
ip -6 rule del prio 10000 fwmark 1 table 100 2>/dev/null || true
ip -6 rule del prio 10000 2>/dev/null || true

ip6tables -t mangle -A PREROUTING -d fd00:23::2 -j TEE --gateway fd00::2
```

```
ip6tables -t mangle -A PREROUTING -d fd00:12::2 -j TEE --gateway fd00::2
ip6tables -t filter -A INPUT -d fd00:23::2 -j DROP
ip6tables -t filter -A INPUT -d fd00:12::2 -j DROP
ip6tables -t mangle -A PREROUTING -i enp0s8 -m addrtype ! --dst-type LOCAL -j MARK --set-mark
1
ip6tables -t mangle -A PREROUTING -i enp0s9 -m addrtype ! --dst-type LOCAL -j MARK --set-mark
1
ip6tables -t mangle -A OUTPUT -j MARK --set-mark 1

if ! ip -6 rule list | grep -q "fwmark 1.*lookup 100"; then
    ip -6 rule add fwmark 1 table 100 priority 10000
fi
ip -6 route replace default via fd00:22::2 dev tun0 table 100
```

Node 3:

/etc/network/interfaces:

```
auto enp0s8
iface enp0s8 inet6 static
        address fd00:23::3
        netmask 64
        gateway fd00:23::2

post-up ip -6 route add fd00::/64 via fd00:23::2
post-up ip -6 route add fd00:01::/64 via fd00:23::2
post-up ip -6 route add fd00:12::/64 via fd00:23::2
post-up ip -6 route add fd00:22::/64 via fd00:23::2

post-up sysctl -w net.ipv6.conf.all.forwarding=1
```

/etc/systemd/system/tun0.service:

```
[Unit]
Description=Set up tun0 interface
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/local/sbin/setup-tun0.sh
ExecStop=/usr/bin/ip link delete tun0
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

/usr/local/sbin/setup-tun0.sh:

```
#!/bin/bash
set -e
ip tuntap add dev tun0 mode tun || true
ip addr add 10.0.0.2/24 dev tun0 || true
ip -6 addr add fd00:33::1/64 dev tun0 || true
ip link set tun0 up

ip6tables -t mangle -F PREROUTING
ip6tables -t mangle -F OUTPUT
ip6tables -t filter -F INPUT
ip -6 rule del prio 10000 fwmark 1 table 100 2>/dev/null || true
ip -6 rule del prio 10000 2>/dev/null || true

ip6tables -t mangle -A PREROUTING -d fd00:23::3 -j TEE --gateway fd00::2
ip6tables -t filter -A INPUT -d fd00:23::3 -j DROP
ip6tables -t mangle -A PREROUTING -i enp0s8 -m addrtype ! --dst-type LOCAL -j MARK --set-mark
1
ip6tables -t mangle -A PREROUTING -i enp0s9 -m addrtype ! --dst-type LOCAL -j MARK --set-mark
1
ip6tables -t mangle -A OUTPUT -j MARK --set-mark 1

if ! ip -6 rule list | grep -q "fwmark 1.*lookup 100"; then
    ip -6 rule add fwmark 1 table 100 priority 10000
fi
ip -6 route replace default via fd00:33::2 dev tun0 table 100
```

Each of these configurations should become active once the system has been restarted. Il while checking the interfaces the TUN appears DOWN, execute the file manually with sudo bash -x /usr/local/sbin/setup-tun0.sh.

Step 3: Set values in the DTN application source code

Set all IP addresses to the desired values based on the topology you have. Also, add contacts to the routing table according to your topology and desired scenario. The files that have to be adjusted are:

- **dtn_controller.c** in lines 269 and 279, and again in lines 446 and 456. These are the addresses, on which the lwIP stack and the DTN program are accessible. So speaking from the perspective of the virtual machine the code is being adjusted on, this is “my” address of my user space.
- **dtn_routing.c** Here, CURR_NODE_ADDR needs to be set to one of the ip address interfaces of the node.
- **main.c** HOST_TUN_IPV6_ADDR, HOST_LWIP_IPV6_ADDR, HOST_enp0s9_IPV6_ADDR and HOST_enp0s8_IPV6_ADDR… are set with the corresponding ip addresses.
- **raw_socket.c** chooses the correct physical interface based on subnets. Adjust in the block starting from line 120.

Depending on the number of interfaces a node has, the code needs to be adapted accordingly (The current repository contains the version for a node with two interfaces):

- **dtn_controller.c** In functions dtn_controller_process_incoming and dtn_controller_attempt _forward_stored an additional block of code needs to be added. This block goes from line 269 to line 278.
- **main.c.** As many HOST_enp0sx_IPV6_ADDR constants need to be defined for every interface address. In line 172 a raw socket for each interface is initialized.
- **raw_socket.c** This file needs to be entirely modified in order to fit less or more interfaces. A file raw_socket_one_interface.c was added to the project.
- **lwipopts.h** the variable LWIP_IPV6_NUM_ADDRESSES sets the maximum of ipv6 addresses lwip can have, this number is currently set to 5. Change in case more addresses need to be added.

Step 4: Running the code

Make sure python is downloaded in all VM before compiling and executing. Inside the project folder, run **make clean** and then **make.** Afterwards start the program with
→ **sudo ./lwip_tun**