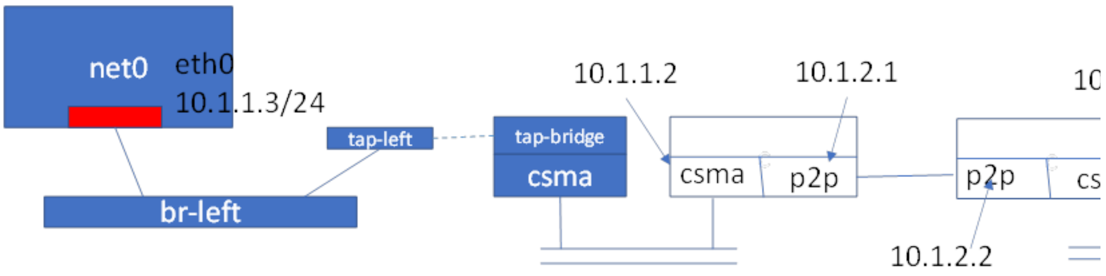
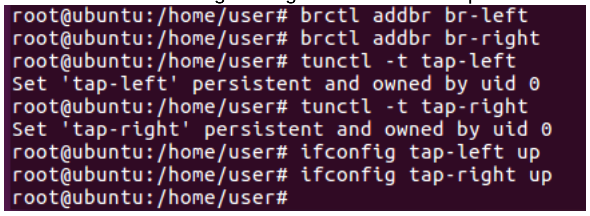
Tap Bridge Model: UseBridge Mode (more complicated scenario) in NS3

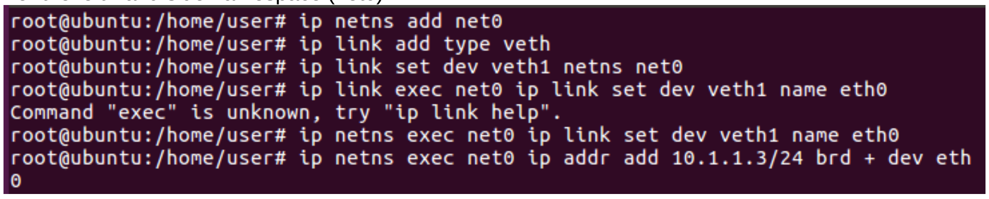
[Topology]

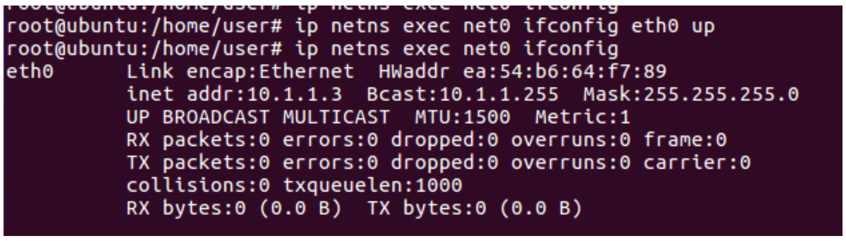


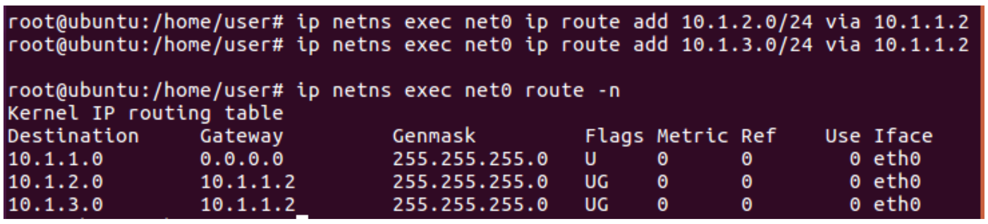
[Steps]
Create br-left and br-right bridges. Also create “tap-left” and “tap-right” taps.

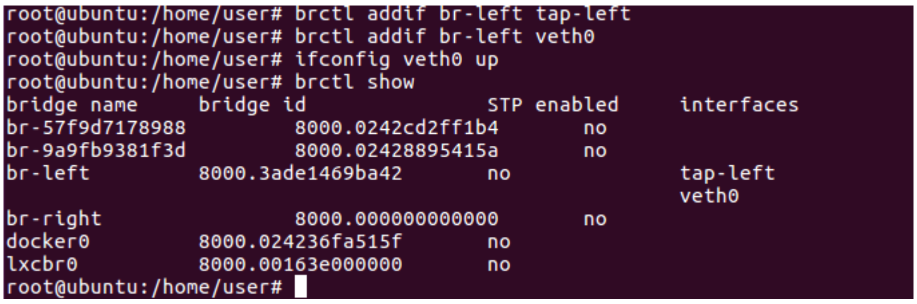

```
root@ubuntu:/home/user# brctl addbr br-left
root@ubuntu:/home/user# brctl addbr br-right
root@ubuntu:/home/user# tunctl -t tap-left
Set 'tap-left' persistent and owned by uid 0
root@ubuntu:/home/user# tunctl -t tap-right
Set 'tap-right' persistent and owned by uid 0
root@ubuntu:/home/user# ifconfig tap-left up
root@ubuntu:/home/user# ifconfig tap-right up
root@ubuntu:/home/user#
```

For the left-hand side namespace (net0)

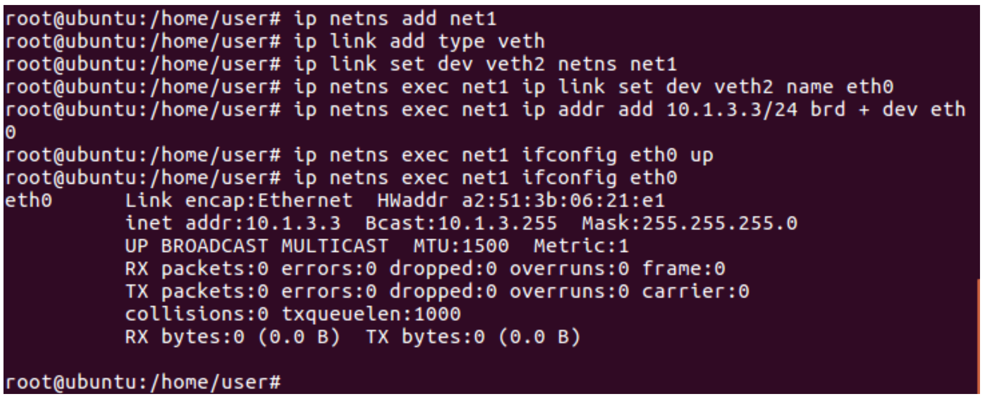

```
root@ubuntu:/home/user# ip netns add net0
root@ubuntu:/home/user# ip link add type veth
root@ubuntu:/home/user# ip link set dev veth1 netns net0
root@ubuntu:/home/user# ip link exec net0 ip link set dev veth1 name eth0
Command "exec" is unknown, try "ip link help".
root@ubuntu:/home/user# ip netns exec net0 ip link set dev veth1 name eth0
root@ubuntu:/home/user# ip netns exec net0 ip addr add 10.1.1.3/24 brd + dev eth
0
```

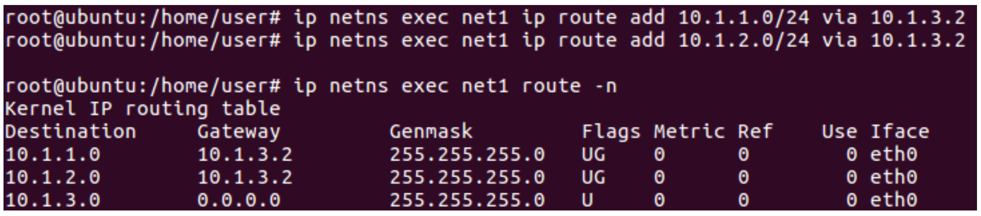

```
root@ubuntu:/home/user# ip netns exec net0 ifconfig
root@ubuntu:/home/user# ip netns exec net0 ifconfig eth0 up
root@ubuntu:/home/user# ip netns exec net0 ifconfig
eth0      Link encap:Ethernet  HWaddr ea:54:b6:64:f7:89
          inet addr:10.1.1.3  Bcast:10.1.1.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

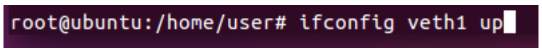

```
root@ubuntu:/home/user# ip netns exec net0 ip route add 10.1.2.0/24 via 10.1.1.2
root@ubuntu:/home/user# ip netns exec net0 ip route add 10.1.3.0/24 via 10.1.1.2

root@ubuntu:/home/user# ip netns exec net0 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
10.1.2.0        10.1.1.2        255.255.255.0   UG    0      0        0 eth0
10.1.3.0        10.1.1.2        255.255.255.0   UG    0      0        0 eth0
```

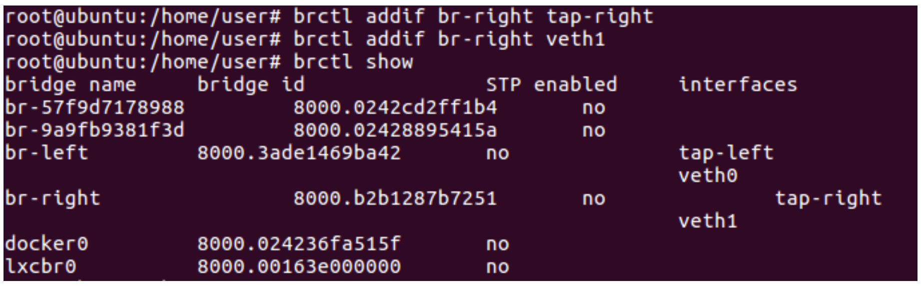

```
root@ubuntu:/home/user# brctl addif br-left tap-left
root@ubuntu:/home/user# brctl addif br-left veth0
root@ubuntu:/home/user# ifconfig veth0 up
root@ubuntu:/home/user# brctl show
bridge name     bridge id               STP enabled     interfaces
br-57f9d7178988         8000.0242cd2ff1b4       no
br-9a9fb9381f3d         8000.02428895415a       no
br-left         8000.3ade1469ba42       no              tap-left
                                                        veth0
br-right                8000.000000000000       no
docker0         8000.024236fa515f       no
lxcbr0          8000.00163e000000       no
root@ubuntu:/home/user# 
```

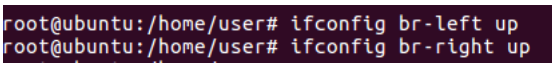
For the right-hand side namespace (net1)


```
root@ubuntu:/home/user# ip netns add net1
root@ubuntu:/home/user# ip link add type veth
root@ubuntu:/home/user# ip link set dev veth2 netns net1
root@ubuntu:/home/user# ip netns exec net1 ip link set dev veth2 name eth0
root@ubuntu:/home/user# ip netns exec net1 ip addr add 10.1.3.3/24 brd + dev eth
0
root@ubuntu:/home/user# ip netns exec net1 ifconfig eth0 up
root@ubuntu:/home/user# ip netns exec net1 ifconfig eth0
eth0      Link encap:Ethernet  HWaddr a2:51:3b:06:21:e1
          inet addr:10.1.3.3  Bcast:10.1.3.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@ubuntu:/home/user#
```


```
root@ubuntu:/home/user# ip netns exec net1 ip route add 10.1.1.0/24 via 10.1.3.2
root@ubuntu:/home/user# ip netns exec net1 ip route add 10.1.2.0/24 via 10.1.3.2

root@ubuntu:/home/user# ip netns exec net1 route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.1.0        10.1.3.2        255.255.255.0   UG    0      0        0 eth0
10.1.2.0        10.1.3.2        255.255.255.0   UG    0      0        0 eth0
10.1.3.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
```


```
root@ubuntu:/home/user# ifconfig veth1 up
```

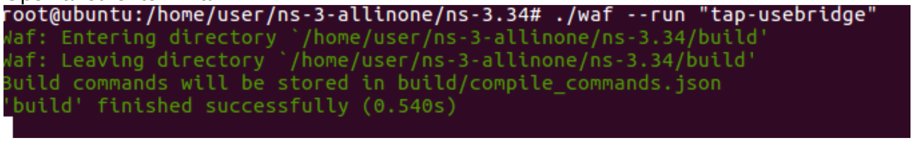

```
root@ubuntu:/home/user# brctl addif br-right tap-right
root@ubuntu:/home/user# brctl addif br-right veth1
root@ubuntu:/home/user# brctl show
bridge name     bridge id               STP enabled     interfaces
br-57f9d7178988         8000.0242cd2ff1b4       no
br-9a9fb9381f3d         8000.02428895415a       no
br-left         8000.3ade1469ba42       no              tap-left
                                                        veth0
br-right                8000.b2b1287b7251       no              tap-right
                                                        veth1
docker0         8000.024236fa515f       no
lxcbr0          8000.00163e000000       no
```


```
root@ubuntu:/home/user# ifconfig br-left up
root@ubuntu:/home/user# ifconfig br-right up
```

tap-usebridge.cc
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

//
// This is an illustration of how one could use virtualization techniques to
// allow running applications on virtual machines talking over simulated
// networks.
//
// The actual steps required to configure the virtual machines can be rather
// involved, so we don't go into that here.  Please have a look at one of
// our HOWTOs on the nsnam wiki for more details about how to get the
// system confgured.  For an example, have a look at "HOWTO Use Linux
// Containers to set up virtual networks" which uses this code as an
// example.
//

#include <iostream>
#include <fstream>
```

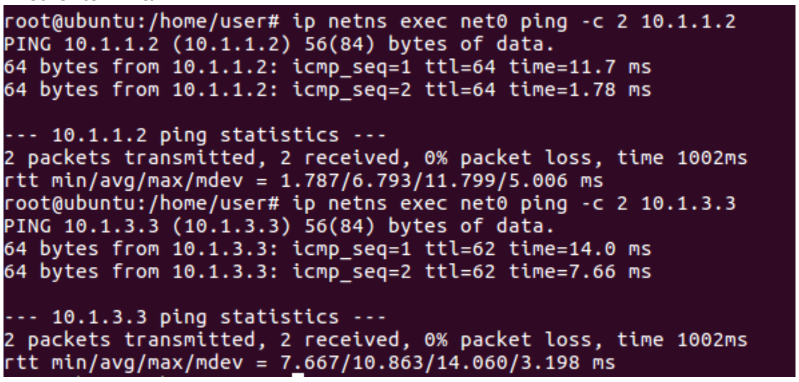
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/tap-bridge-module.h"
#include "ns3/error-model.h"
#include "ns3/point-to-point-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TapCsmaUseBridgeExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);
  cmd.Parse (argc, argv);

  //
  // We are interacting with the outside, real, world.  This means we have to
  // interact in real-time and therefore means we have to use the real-time
  // simulator and take the time to calculate checksums.
  //
  GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealtimeSimulatorImpl"));
  GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

  //
  // Create two ghost nodes.  The first will represent the virtual machine host
  // on the left side of the network; and the second will represent the VM on
  // the right side.
  //
  NodeContainer csma1Nodes;
  csma1Nodes.Create (2);

  NodeContainer p2pNodes;
  p2pNodes.Add (csma1Nodes.Get (1));
  p2pNodes.Create(1);

  NodeContainer csma2Nodes;
  csma2Nodes.Create(1);
  csma2Nodes.Add (p2pNodes.Get (1));

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);

  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

  NetDeviceContainer csma1Devices;
  NetDeviceContainer csma2Devices;
  csma1Devices = csma.Install (csma1Nodes);
  csma2Devices = csma.Install (csma2Nodes);

  Ptr<RateErrorModel> em = CreateObject<RateErrorModel> ();
  em->SetAttribute ("ErrorRate", DoubleValue (0.01));
  em->SetAttribute ("ErrorUnit", StringValue ("ERROR_UNIT_PACKET"));
  p2pDevices.Get (1)->SetAttribute ("ReceiveErrorModel", PointerValue (em));

  InternetStackHelper stack;
  stack.Install (csma1Nodes);
  stack.Install (csma2Nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer csma1Interfaces;
  csma1Interfaces = address.Assign (csma1Devices);

  address.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);
  address.SetBase ("10.1.3.0", "255.255.255.0");
  Ipv4InterfaceContainer csma2Interfaces;
  csma2Interfaces = address.Assign (csma2Devices);

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //TapBridgeHelper tapBridge;
  //tapBridge.SetAttribute ("Mode", StringValue ("ConfigureLocal"));
  //tapBridge.SetAttribute ("DeviceName", StringValue ("thetap"));
  //tapBridge.Install (csma1Nodes.Get (0), csma1Devices.Get (0));

  //
  // Use the TapBridgeHelper to connect to the pre-configured tap devices for
  // the left side.  We go with "UseBridge" mode since the CSMA devices support
  // promiscuous mode and can therefore make it appear that the bridge is
  // extended into ns-3.  The install method essentially bridges the specified
  // tap to the specified CSMA device.
  //
  TapBridgeHelper tapBridge;
  tapBridge.SetAttribute ("Mode", StringValue ("UseBridge"));
  tapBridge.SetAttribute ("DeviceName", StringValue ("tap-left"));
  tapBridge.Install (csma1Nodes.Get (0), csma1Devices.Get (0));

  //
  // Connect the right side tap to the right side CSMA device on the right-side
  // ghost node.
  //
  tapBridge.SetAttribute ("DeviceName", StringValue ("tap-right"));
  tapBridge.Install (csma2Nodes.Get (0), csma2Devices.Get (0));

  //
  // Run the simulation for ten minutes to give the user time to play around
  //
  Simulator::Stop (Seconds (1000.));
  Simulator::Run ();
  Simulator::Destroy ();
}
```

Open another terminal

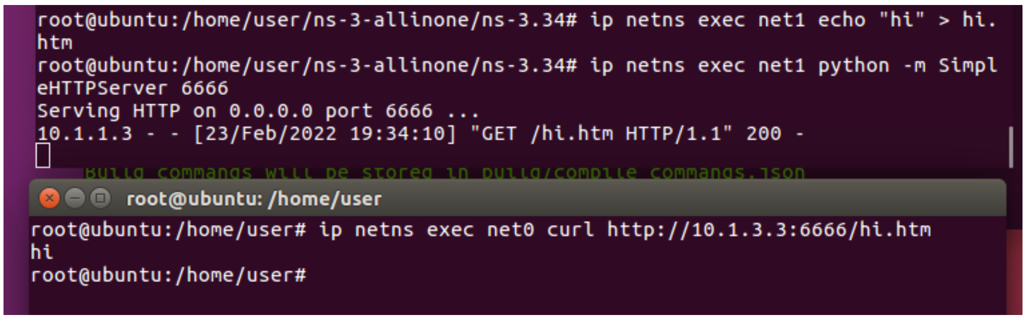


```
root@ubuntu:/home/user/ns-3-allinone/ns-3.34# ./waf --run "tap-usebridge"
Waf: Entering directory `/home/user/ns-3-allinone/ns-3.34/build'
Waf: Leaving directory `/home/user/ns-3-allinone/ns-3.34/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.540s)
```

In other terminal



```
root@ubuntu:/home/user# ip netns exec net0 ping -c 2 10.1.1.2
PING 10.1.1.2 (10.1.1.2) 56(84) bytes of data.
64 bytes from 10.1.1.2: icmp_seq=1 ttl=64 time=11.7 ms
64 bytes from 10.1.1.2: icmp_seq=2 ttl=64 time=1.78 ms

--- 10.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.787/6.793/11.799/5.006 ms
root@ubuntu:/home/user# ip netns exec net0 ping -c 2 10.1.3.3
PING 10.1.3.3 (10.1.3.3) 56(84) bytes of data.
64 bytes from 10.1.3.3: icmp_seq=1 ttl=62 time=14.0 ms
64 bytes from 10.1.3.3: icmp_seq=2 ttl=62 time=7.66 ms

--- 10.1.3.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 7.667/10.863/14.060/3.198 ms
```



```
root@ubuntu:/home/user/ns-3-allinone/ns-3.34# ip netns exec net1 echo "hi" > hi.
htm
root@ubuntu:/home/user/ns-3-allinone/ns-3.34# ip netns exec net1 python -m Simpl
eHTTPServer 6666
Serving HTTP on 0.0.0.0 port 6666 ...
10.1.1.3 - - [23/Feb/2022 19:34:10] "GET /hi.htm HTTP/1.1" 200 -
```

```
root@ubuntu:/home/user# ip netns exec net0 curl http://10.1.3.3:6666/hi.htm
hi
root@ubuntu:/home/user#
```

Back to NS3 Learning Guide
Last Modified: 2022/2/25 done

[Author]
Dr. Chih-Heng Ke
Department of Computer Science and Information Engineering, National Quemoy University, Kinmen, Taiwan
Email: smallko@gmail.com