Knit workshop.

We present two experiments.

Experiment 1

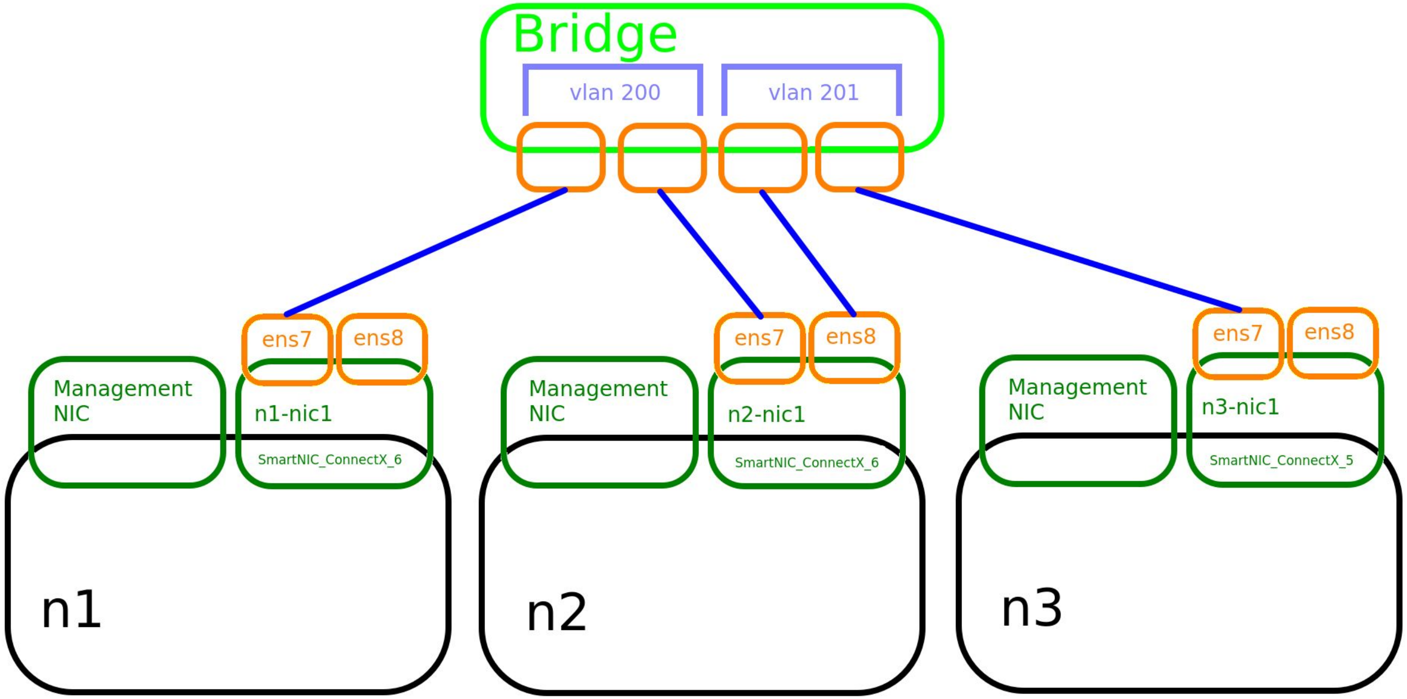
Motivation

We want to connect three nodes to a bridge.

The bridge will use vlan tags to isolate two paths between the nodes.

One node will have a path to each of the other two nodes. And it will act as a router between them.

We reserve the following topology



The topology consists of 3 nodes, each having an extra nic.

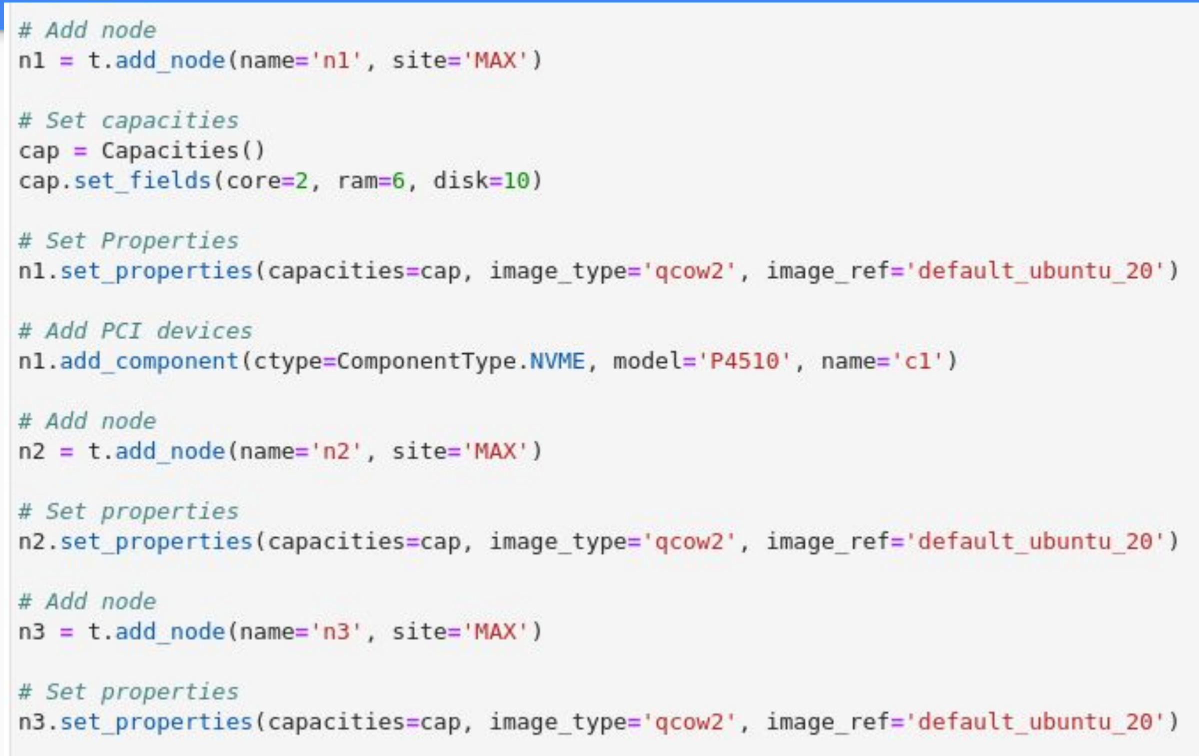
The three NICs are connected to a bridge.

The path between the n1 and n2 NICs is isolated with a vlan tag (200), and the path between the n2 and n3 NICs is isolated with another vlan tag (201).

n2 will act as a router between n1 and n3. n1 and n3 are going to send traffic through n2.

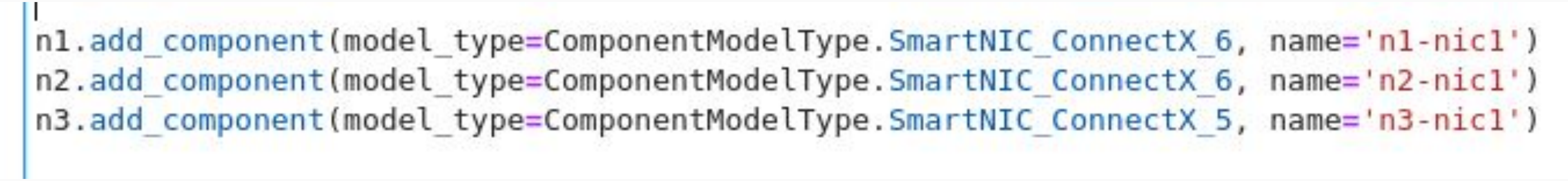
Code Walkthrough

First we reserve the nodes



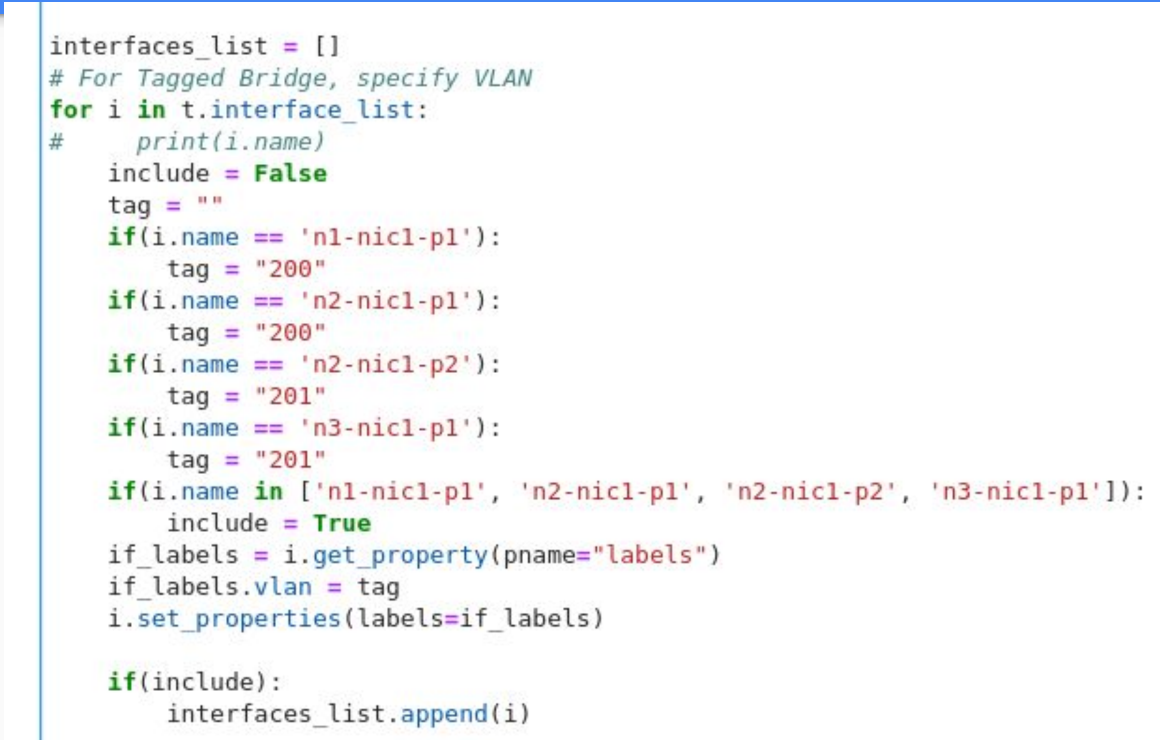
```
# Add node
n1 = t.add_node(name='n1', site='MAX')

# Set capacities
cap = Capacities()
cap.set_fields(core=2, ram=6, disk=10)

# Set Properties
n1.set_properties(capacities=cap, image_type='qcow2', image_ref='default_ubuntu_20')

# Add PCI devices
n1.add_component(ctype=ComponentType.NVME, model='P4510', name='c1')

# Add node
n2 = t.add_node(name='n2', site='MAX')

# Set properties
n2.set_properties(capacities=cap, image_type='qcow2', image_ref='default_ubuntu_20')

# Add node
n3 = t.add_node(name='n3', site='MAX')

# Set properties
n3.set_properties(capacities=cap, image_type='qcow2', image_ref='default_ubuntu_20')
```

Then we add the NICs to them.



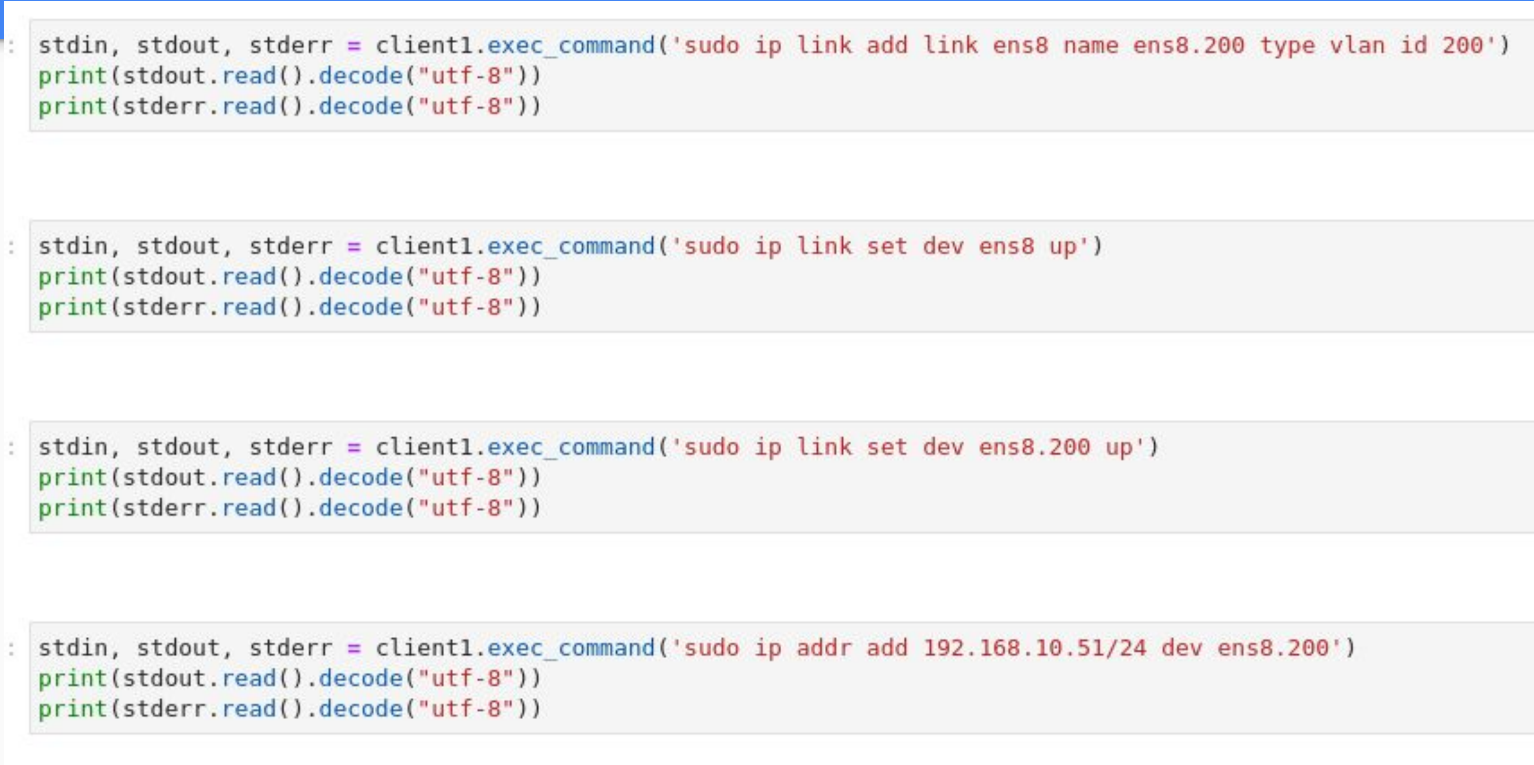
```
n1.add_component(model_type=ComponentModelType.SmartNIC_ConnectX_6, name='n1-nic1')
n2.add_component(model_type=ComponentModelType.SmartNIC_ConnectX_6, name='n2-nic1')
n3.add_component(model_type=ComponentModelType.SmartNIC_ConnectX_5, name='n3-nic1')
```

And we specify the vlan tags.



```
interfaces_list = []
# For Tagged Bridge, specify VLAN
for i in t.interface_list:
#     print(i.name)
    include = False
    tag = ""
    if(i.name == 'n1-nic1-p1'):
        tag = "200"
    if(i.name == 'n2-nic1-p1'):
        tag = "200"
    if(i.name == 'n2-nic1-p2'):
        tag = "201"
    if(i.name == 'n3-nic1-p1'):
        tag = "201"
    if(i.name in ['n1-nic1-p1', 'n2-nic1-p1', 'n2-nic1-p2', 'n3-nic1-p1']):
        include = True
    if_labels = i.get_property(pname="labels")
    if_labels.vlan = tag
    i.set_properties(labels=if_labels)

    if(include):
        interfaces_list.append(i)
```

We create a virtual interface, activate the interfaces, and give the NICs IPs.



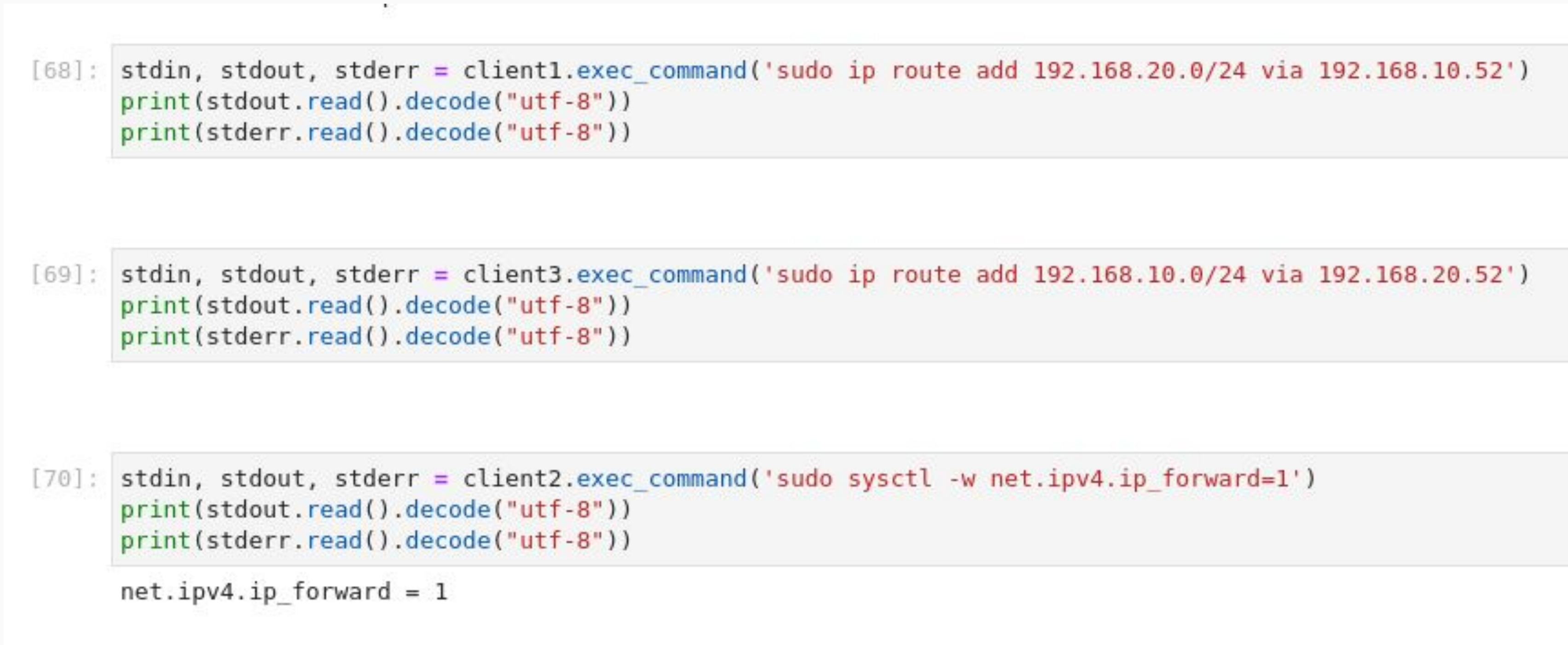
```
stdin, stdout, stderr = client1.exec_command('sudo ip link add link ens8 name ens8.200 type vlan id 200')
print(stdout.read().decode("utf-8"))
print(stderr.read().decode("utf-8"))
```

```
stdin, stdout, stderr = client1.exec_command('sudo ip link set dev ens8 up')
print(stdout.read().decode("utf-8"))
print(stderr.read().decode("utf-8"))
```

```
stdin, stdout, stderr = client1.exec_command('sudo ip link set dev ens8.200 up')
print(stdout.read().decode("utf-8"))
print(stderr.read().decode("utf-8"))
```

```
stdin, stdout, stderr = client1.exec_command('sudo ip addr add 192.168.10.51/24 dev ens8.200')
print(stdout.read().decode("utf-8"))
print(stderr.read().decode("utf-8"))
```

We setup routes, and enable packet forwarding.

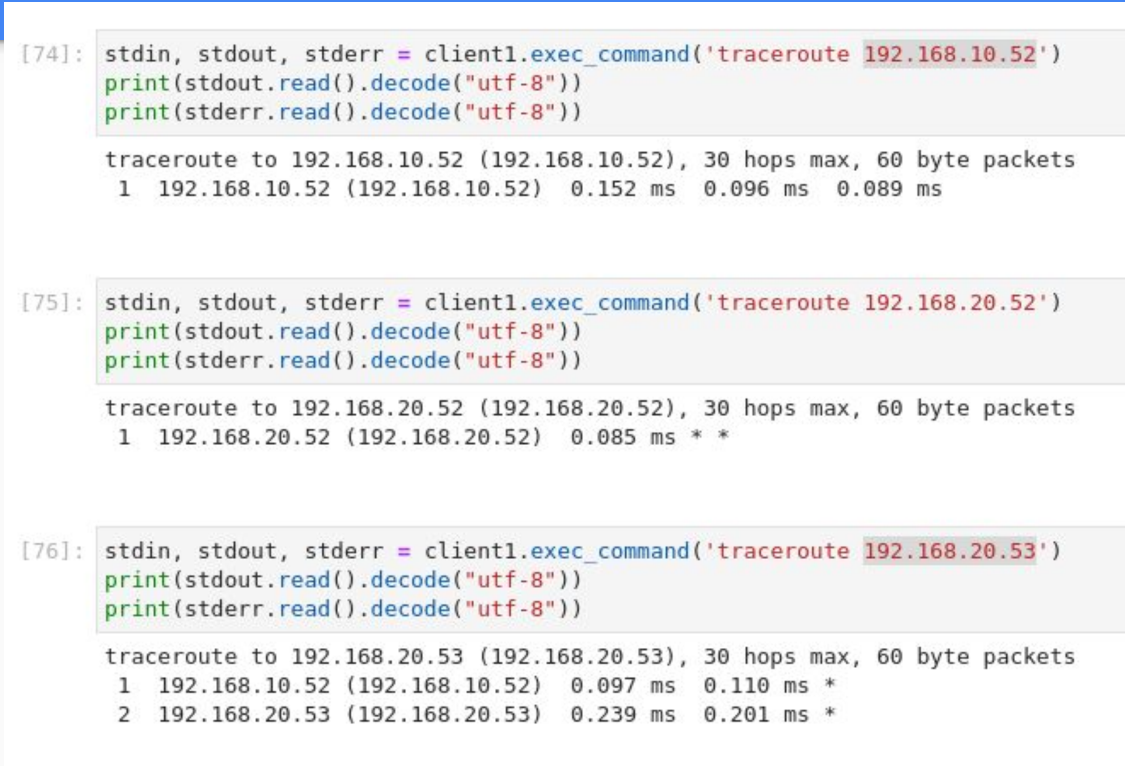


```
[68]: stdin, stdout, stderr = client1.exec_command('sudo ip route add 192.168.20.0/24 via 192.168.10.52')
      print(stdout.read().decode("utf-8"))
      print(stderr.read().decode("utf-8"))
```

```
[69]: stdin, stdout, stderr = client3.exec_command('sudo ip route add 192.168.10.0/24 via 192.168.20.52')
      print(stdout.read().decode("utf-8"))
      print(stderr.read().decode("utf-8"))
```

```
[70]: stdin, stdout, stderr = client2.exec_command('sudo sysctl -w net.ipv4.ip_forward=1')
      print(stdout.read().decode("utf-8"))
      print(stderr.read().decode("utf-8"))

      net.ipv4.ip_forward = 1
```

Now everything is configured. We can do a traceroute to see the paths packets take. Now the nodes can ping each other.



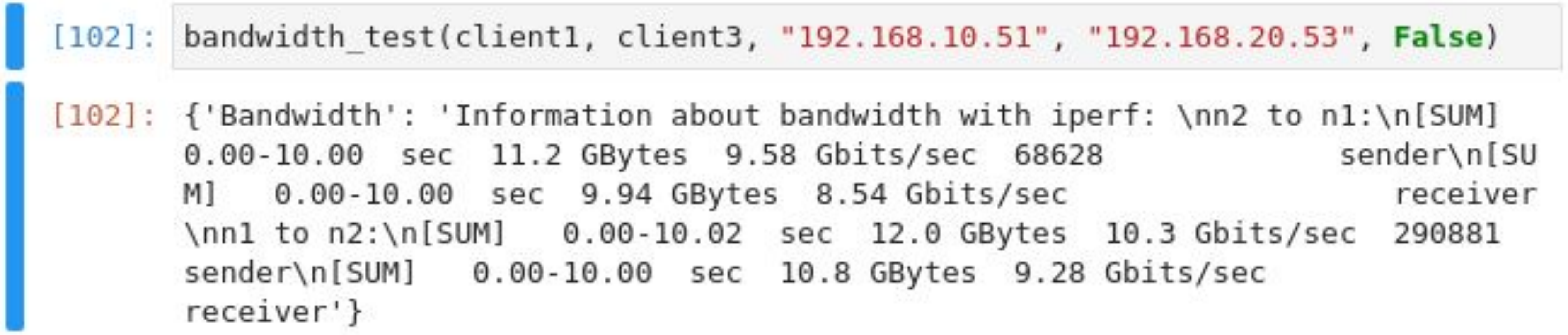
```
[74]: stdin, stdout, stderr = client1.exec_command('traceroute 192.168.10.52')
      print(stdout.read().decode("utf-8"))
      print(stderr.read().decode("utf-8"))

      traceroute to 192.168.10.52 (192.168.10.52), 30 hops max, 60 byte packets
       1  192.168.10.52 (192.168.10.52)  0.152 ms  0.096 ms  0.089 ms
```

```
[75]: stdin, stdout, stderr = client1.exec_command('traceroute 192.168.20.52')
      print(stdout.read().decode("utf-8"))
      print(stderr.read().decode("utf-8"))

      traceroute to 192.168.20.52 (192.168.20.52), 30 hops max, 60 byte packets
       1  192.168.20.52 (192.168.20.52)  0.085 ms * *
```

```
[76]: stdin, stdout, stderr = client1.exec_command('traceroute 192.168.20.53')
      print(stdout.read().decode("utf-8"))
      print(stderr.read().decode("utf-8"))

      traceroute to 192.168.20.53 (192.168.20.53), 30 hops max, 60 byte packets
       1  192.168.10.52 (192.168.10.52)  0.097 ms  0.110 ms *
       2  192.168.20.53 (192.168.20.53)  0.239 ms  0.201 ms *
```

We can also do a bandwidth test with a code module that we made.

We get a performance of around 10 Gbits/sec.



```
[102]: bandwidth_test(client1, client3, "192.168.10.51", "192.168.20.53", False)

[102]: {'Bandwidth': 'Information about bandwidth with iperf: \nn2 to n1:\n[SUM]
       0.00-10.00  sec  11.2 GBytes  9.58 Gbits/sec  68628             sender\n[SU
       M]   0.00-10.00  sec  9.94 GBytes  8.54 Gbits/sec                  receiver
       \nn1 to n2:\n[SUM]   0.00-10.02  sec  12.0 GBytes  10.3 Gbits/sec  290881
       sender\n[SUM]   0.00-10.00  sec  10.8 GBytes  9.28 Gbits/sec
       receiver'}
```

Experiment 2

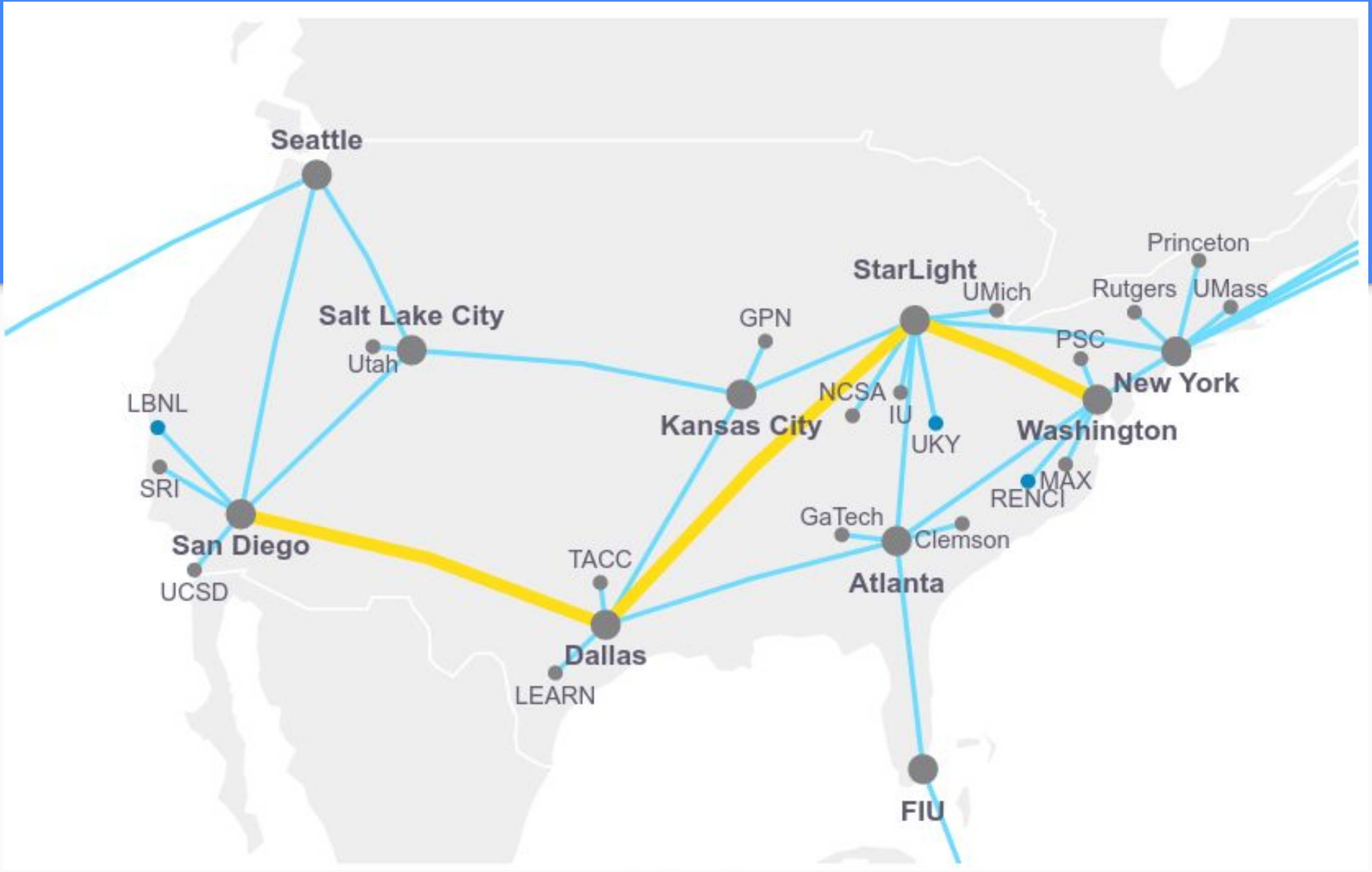
Motivation

We want to collect metrics about network paths between sites.

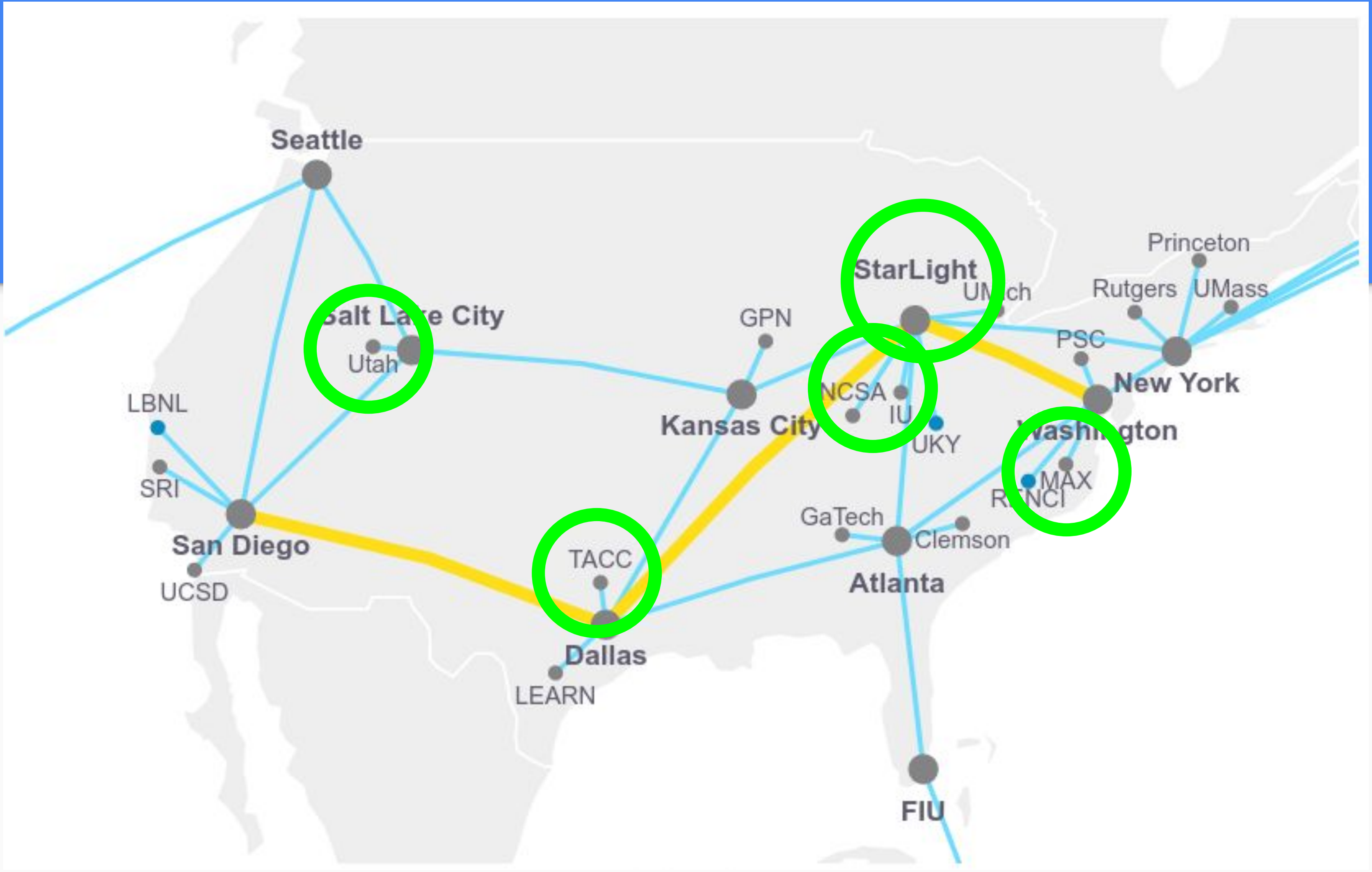
We will reserve nodes at the site “MAX”, and nodes at the site “UTAH”.

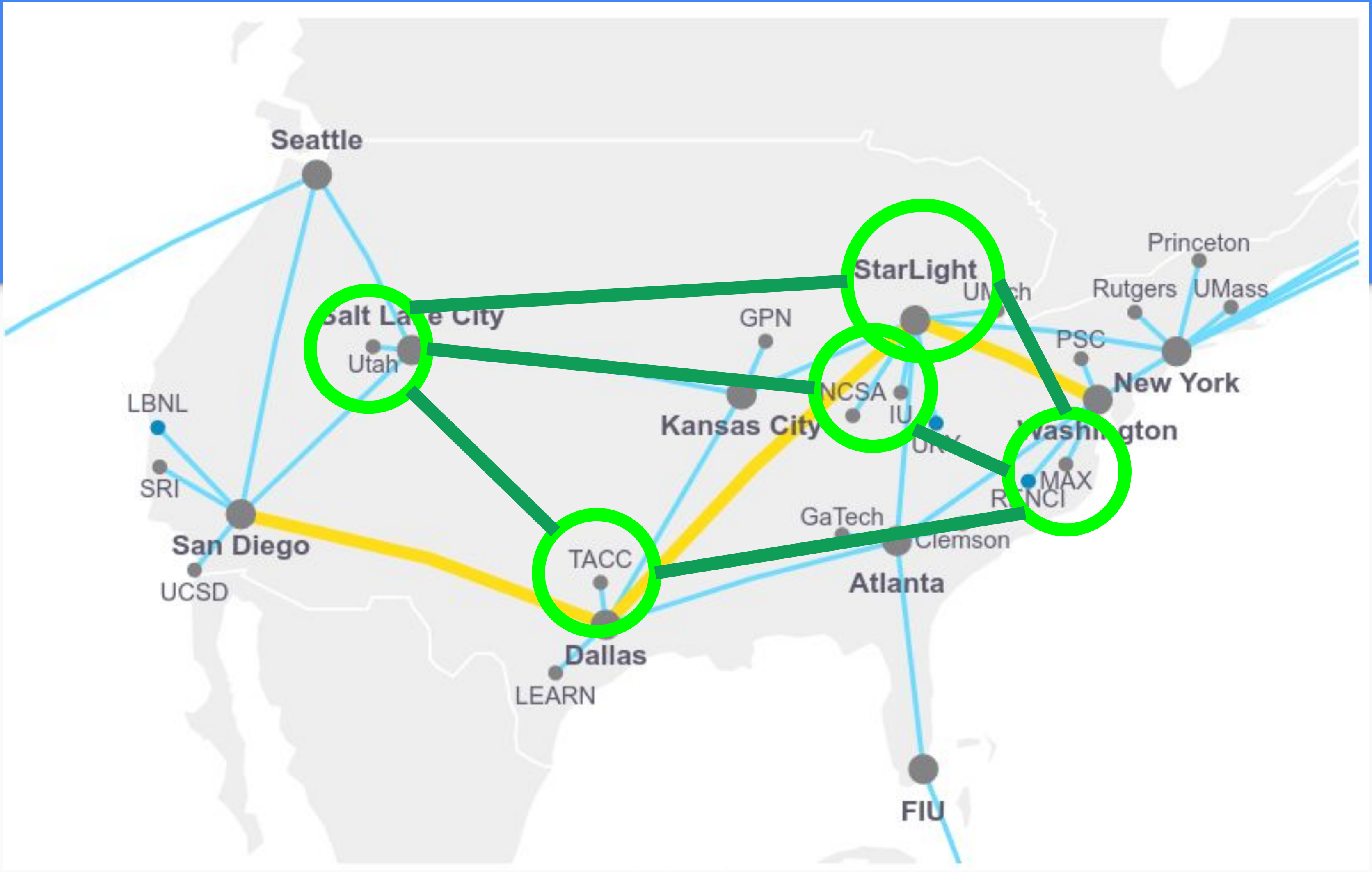
We are going to connect MAX to UTAH through three different sites in the middle. “STARLIGHT”, “NCSA” and “TACC”.

We will use **layer 2 point to point links** to connect MAX to one of the three sites to UTAH.

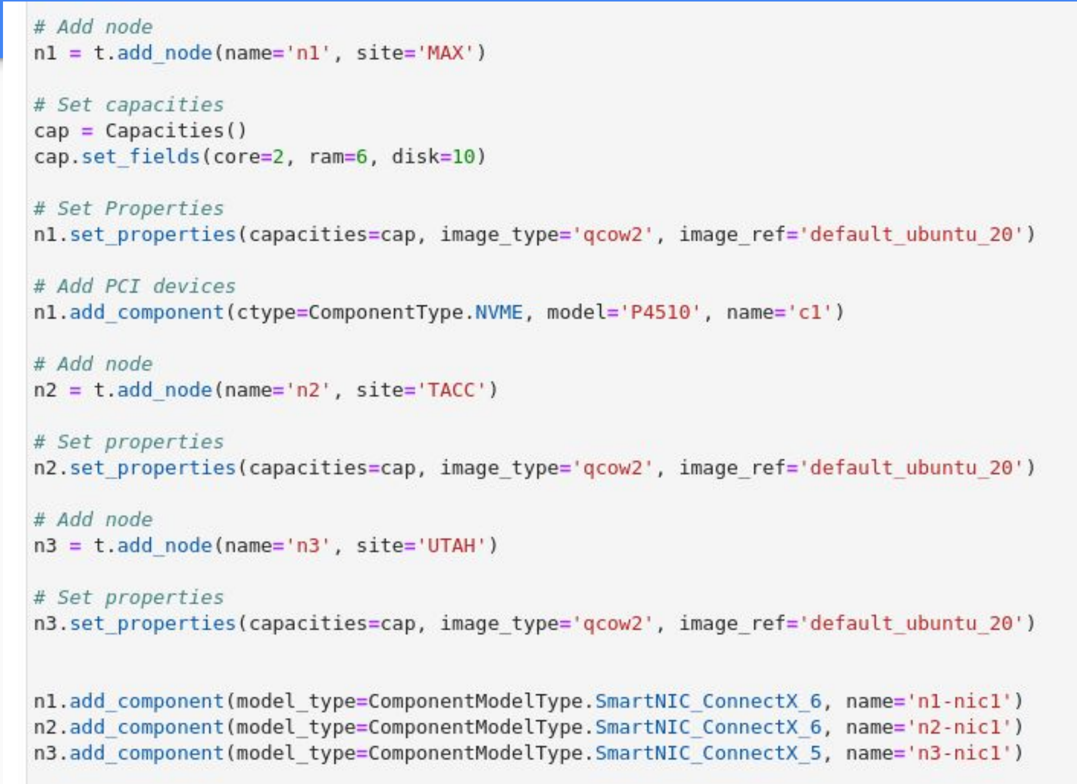


Seattle
Salt Lake City
Utah
LBNL
SRI
San Diego
UCSD
GPN
Kansas City
TACC
Dallas
LEARN
StarLight
UMich
NCSA
IU
UKY
GaTech
Atlanta
Clemson
FIU
Princeton
Rutgers
UMass
PSC
New York
Washington
MAX
RENCI


Seattle
Salt Lake City
Utah
LBNL
SRI
San Diego
UCSD
GPN
Kansas City
TACC
Dallas
LEARN
StarLight
UMich
NCSA
IU
UKY
Princeton
Rutgers
UMass
PSC
New York
Washington
MAX
RENCI
GaTech
Clemson
Atlanta
FIU


Seattle
Salt Lake City
Utah
LBNL
SRI
San Diego
UCSD
GPN
Kansas City
StarLight
UMich
NCSA
IU
UK
Princeton
Rutgers
UMass
PSC
New York
Washington
MAX
RENCI
GaTech
Clemson
Atlanta
TACC
Dallas
LEARN
FIU

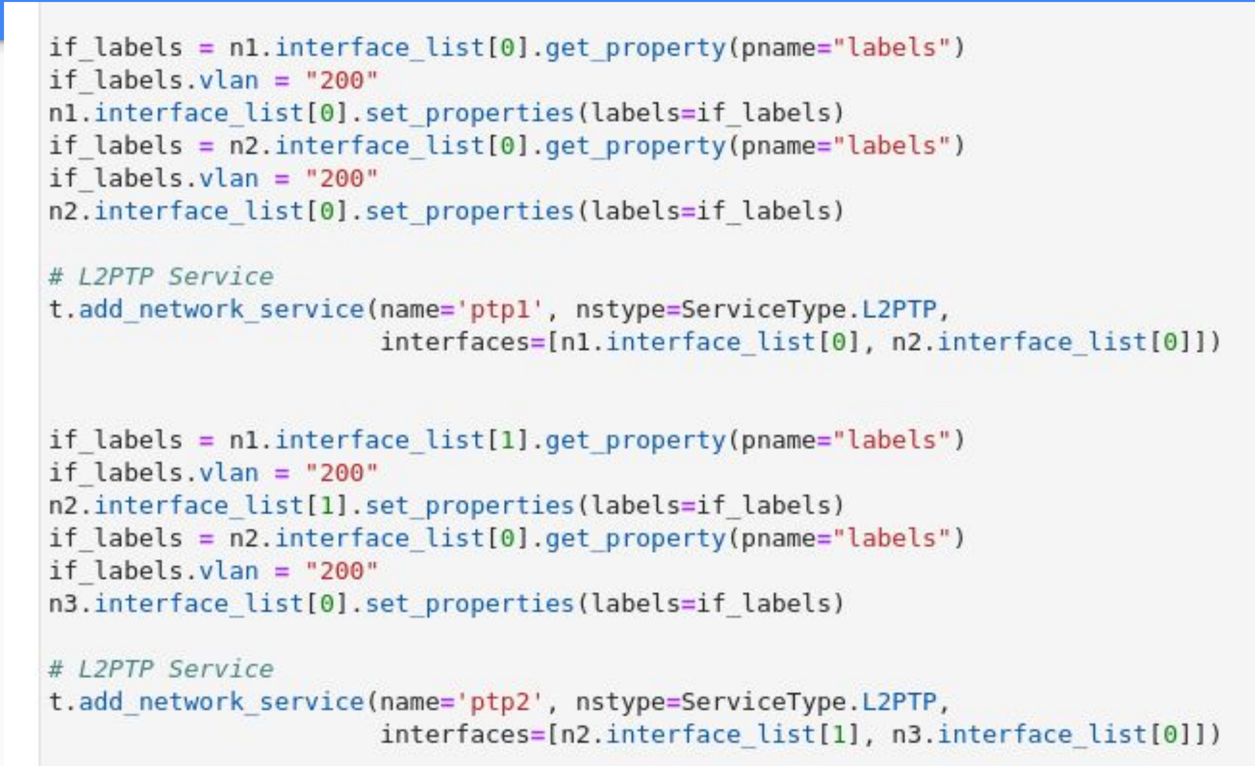
The code is very similar to the previous experiment. We reserve nodes with NICs.



```
# Add node
n1 = t.add_node(name='n1', site='MAX')

# Set capacities
cap = Capacities()
cap.set_fields(core=2, ram=6, disk=10)

# Set Properties
n1.set_properties(capacities=cap, image_type='qcow2', image_ref='default_ubuntu_20')

# Add PCI devices
n1.add_component(ctype=ComponentType.NVME, model='P4510', name='c1')

# Add node
n2 = t.add_node(name='n2', site='TACC')

# Set properties
n2.set_properties(capacities=cap, image_type='qcow2', image_ref='default_ubuntu_20')

# Add node
n3 = t.add_node(name='n3', site='UTAH')

# Set properties
n3.set_properties(capacities=cap, image_type='qcow2', image_ref='default_ubuntu_20')


n1.add_component(model_type=ComponentModelType.SmartNIC_ConnectX_6, name='n1-nic1')
n2.add_component(model_type=ComponentModelType.SmartNIC_ConnectX_6, name='n2-nic1')
n3.add_component(model_type=ComponentModelType.SmartNIC_ConnectX_5, name='n3-nic1')
```

We add vlan tags and point to point links

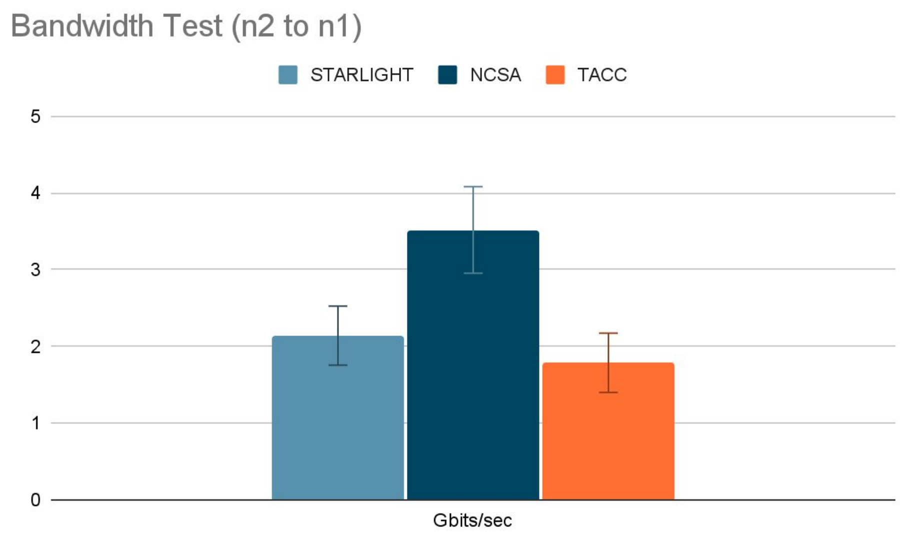
Then we can connect to the nodes, activate the network interfaces, assign IPs to them and setup routes and port forwarding.

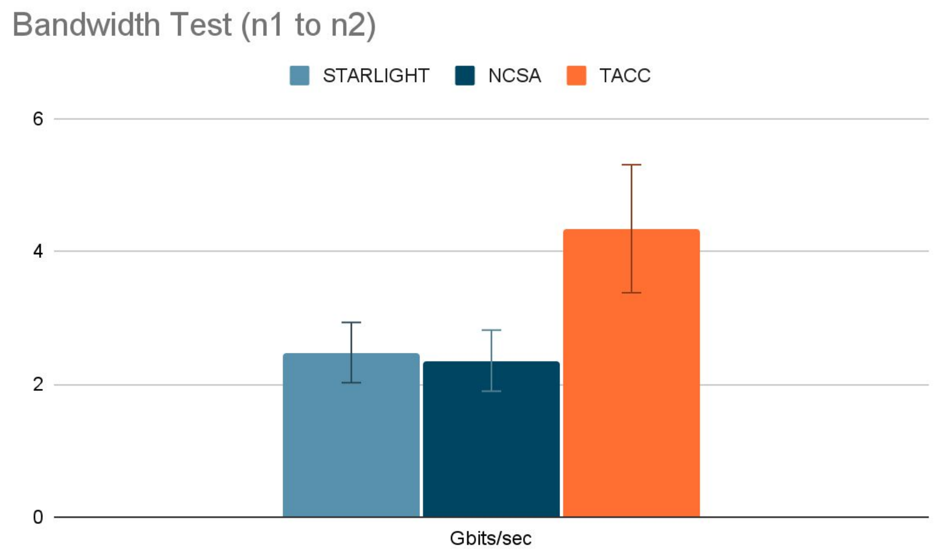


```
if_labels = n1.interface_list[0].get_property(pname="labels")
if_labels.vlan = "200"
n1.interface_list[0].set_properties(labels=if_labels)
if_labels = n2.interface_list[0].get_property(pname="labels")
if_labels.vlan = "200"
n2.interface_list[0].set_properties(labels=if_labels)

# L2PTP Service
t.add_network_service(name='ptp1', nstype=ServiceType.L2PTP,
                      interfaces=[n1.interface_list[0], n2.interface_list[0]])


if_labels = n1.interface_list[1].get_property(pname="labels")
if_labels.vlan = "200"
n2.interface_list[1].set_properties(labels=if_labels)
if_labels = n2.interface_list[0].get_property(pname="labels")
if_labels.vlan = "200"
n3.interface_list[0].set_properties(labels=if_labels)

# L2PTP Service
t.add_network_service(name='ptp2', nstype=ServiceType.L2PTP,
                      interfaces=[n2.interface_list[1], n3.interface_list[0]])
```

Bandwidth Test

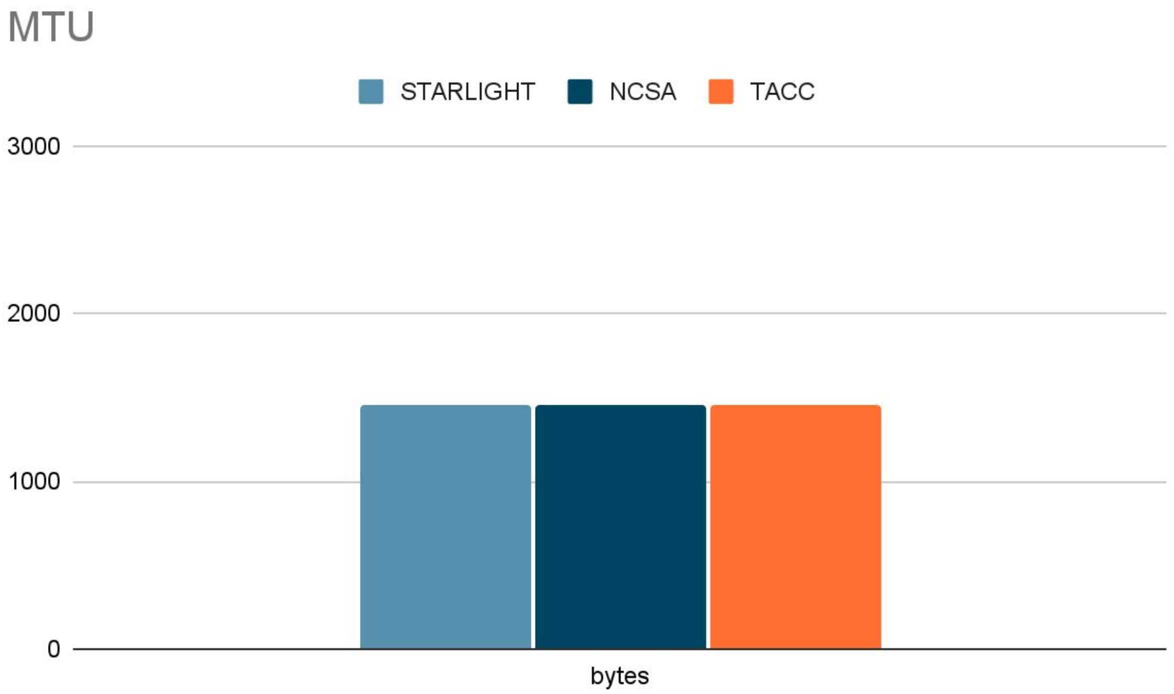
This shows the bandwidth over both directions of the paths. 10 measurements were taken.





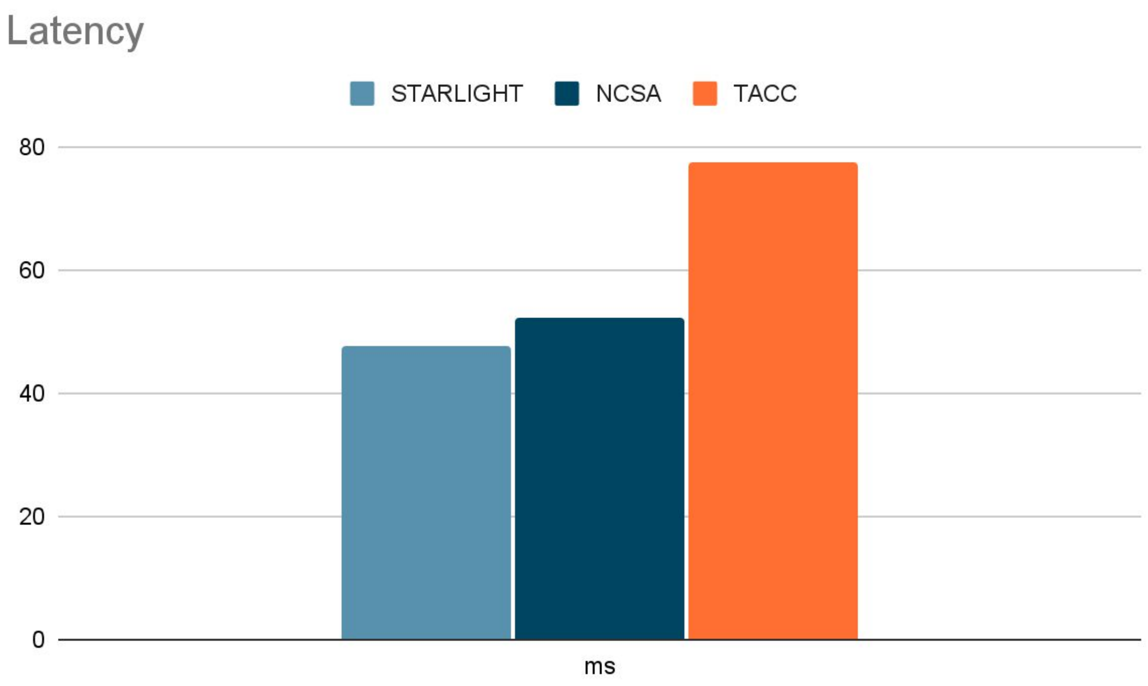
MTU

All paths have the same MTU size. The standard 1500 bytes MTU.



Latency

These are latency measurements with ping. 25 measurements were taken. No significant variability.



Conclusion

We reserved different topologies using VMs, bridges and point to point links.

We can use bridges to connect nodes on one site. We can isolate connections with vlan tags.

Point to point links allow us to connect nodes over multiple sites. We were able to collect data about the paths between these sites.

Code modules for the latency test, mtu test and bandwidth test were used to design a test harness that we can use internally periodically to check the hardware and the network paths.

Thank you.