Internetworking-EM Report

(NWworkstation1, Nwrouter, Isrouter, Swrouter, SWworkstation1)

Set Up

Login into DETERLAB, create a new experiment and interact with nodes using double ssh commands: `ssh username@isi.deterlab.net` and `ssh nodeID.ExperimentID.projectID.isi.deterlab.net`, opening 5 terminals with each representing a node. Use command 'sudo su -' to ensure you are root.

Implementation and Tasks

Task1

Use command `'ifconfig'` to view and place addresses on network interfaces and command `'ifconfig -a'` to view additional nodes. Then run command: `'/share/shared/Internetworking/showcabling Internetworking-EM OffTech2019'` from users.isi.deterlab.net to help identify cabled and cableless nodes.

```
[offtecaa@users ~]$ /share/shared/Internetworking/showca
bling Internetworking-EM OffTech2019

NWrouter eth1  <- is "wired" to ->  NWworkstation1 eth1
ISrouter eth4  <- is "wired" to ->  NWrouter eth2
ISrouter eth2  <- is "wired" to ->  SWrouter eth4
SWrouter eth1  <- is "wired" to ->  SWworkstation1 eth4
```

Fig1. Subcabling command

Choose 4 networks and 2 addresses. With reference to the notes, for the external and internal nodes, I chose the following IP addresses from public and private address ranges:

NWworksation network address: `172.16.1.0`

NWworkstation nodes: `172.16.1.2, 172.16.1.2`

NWrouter network address: `131.20.20.0`

NWrouter nodes: `131.20.20.1, 130.20.20.2`

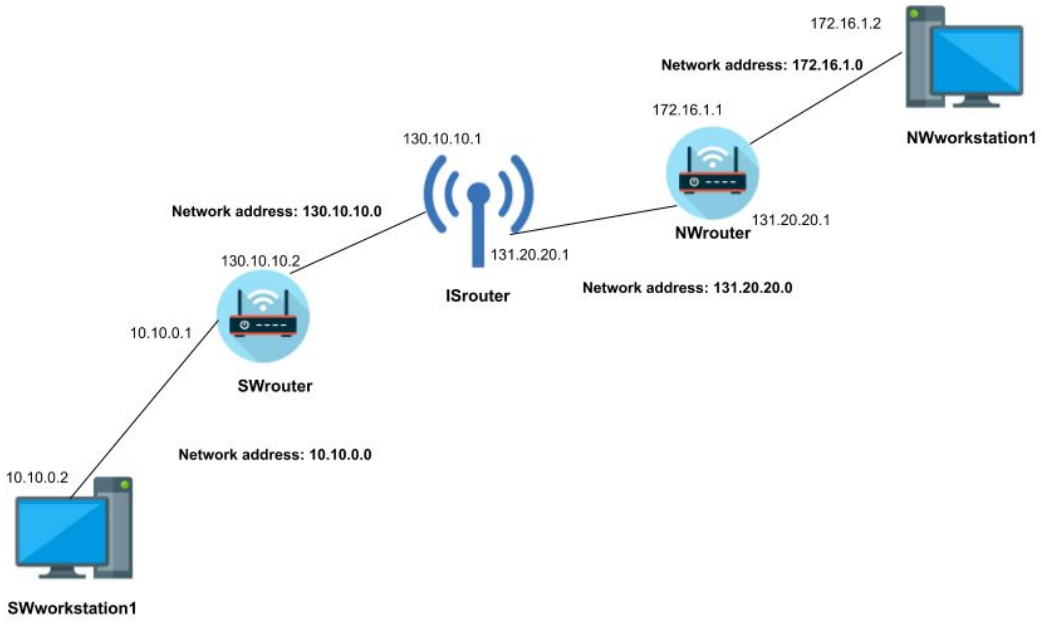
ISrouter network address: `130.10.10.0`

ISrouter nodes: `130.10.10.1, 130.10.10.2`

SWrworkstation network address: `10.10.0.0`

SWworkstation nodes: `10.10.0.1, 10.10.0.2`

The diagram below helps in understanding how the network addresses are assigned to the nodes.



Commands for assigning Network addresses using the subcabling output

NWworksation1

```
ifconfig eth1 172.16.1.2 netmask 255.255.255.248
```

NWrouter

```
ifconfig eth1 172.16.1.1 netmask 255.255.255.248

ifconfig eth2 131.10.10.1 netmask 255.255.255.248
```

```
root@nwrouter:~# ifconfig eth1 172.16.1.1 netmask 255.255.255.248
root@nwrouter:~# ifconfig eth2 131.20.20.2 netmask 255.255.255.248
root@nwrouter:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=64 time=0.512 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=64 time=0.254 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=64 time=0.401 ms
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.254/0.389/0.512/0.105 ms
root@nwrouter:~#
```

ISrouter

```
ifconfig eth4 131.20.20.1 netmask 255.255.255.248

ifconfig eth2 130.10.10.1 netmask 255.255.255.248
```

SWrouter

```
ifconfig eth4 130.10.10.2 netmask 255.255.255.248
```

```
ifconfig eth1 10.10.0.1 netmask 255.255.255.248
```

```
root@swrouter:~# ifconfig eth4 130.10.10.2 netmask 255.255.255.248
root@swrouter:~# ifconfig eth1 10.10.0.1 netmask 255.255.255.248
root@swrouter:~# ping 10.10.0.2
PING 10.10.0.2 (10.10.0.2) 56(84) bytes of data.
64 bytes from 10.10.0.2: icmp_seq=1 ttl=64 time=1.00 ms
64 bytes from 10.10.0.2: icmp_seq=2 ttl=64 time=0.306 ms
64 bytes from 10.10.0.2: icmp_seq=3 ttl=64 time=0.475 ms
^C
--- 10.10.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.306/0.595/1.005/0.298 ms
root@swrouter:~# s
```

SWworkstation1

```
ifconfig eth1 10.10.0.2 netmask 255.255.255.248
```

Then to make sure all the links are working, on ISrouters, in both directions, ping the work stations using: `ping 131.20.20.1 and ping 130.10.10.1`

On NWrouter: `ping 172.16.1.2`

```
root@nwrouter:~# ifconfig eth1 172.16.1.1 netmask 255.255.255.248
root@nwrouter:~# ifconfig eth2 131.20.20.2 netmask 255.255.255.248
root@nwrouter:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=64 time=0.512 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=64 time=0.254 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=64 time=0.401 ms
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.254/0.389/0.512/0.105 ms
root@nwrouter:~#
```

OnSWrouter: `ping 10.10.0.2`

The figures showing the status of the pings are seen alongside the routing table commands in the next task.

Task 2 (Inserting routes in route tables)

Now, we want to set routing tables for our 4-LANs so they know what to do with packets addressed to networks remote to them. Here we route only gatewayed network for each machines that is not a member of our 4LANs, since the local ones are set automatically.

NWworstation1

```
route add -net 131.20.20.0 netmask 255.255.255.248 gw 172.16.1.1

route add -net 130.10.10.0 netmask 255.255.255.248 gw 172.16.1.1

route add -net 10.10.0.0 netmask 255.255.255.248 gw 172.16.1.1
```

SWworkstation1

```
route add -net 130.10.10.0 netmask 255.255.255.248 gw 10.10.0.1

route add -net 131.20.20.0 netmask 255.255.255.248 gw 10.10.0.1

route add -net 172.16.1.0 netmask 255.255.255.248 gw 10.10.0.1
```

NWrouter

```
route add -net 130.10.10.0 netmask 255.255.255.248 gw 131.20.20.1
```

```
route add -net 10.10.0.0 netmask 255.255.255.248 gw 131.20.20.1
```

SWrouter

```
route add -net 131.20.20.0 netmask 255.255.255.248 gw 131.10.10.1
```

```
route add -net 172.16.1.0 netmask 255.255.255.248 gw 131.10.10.1
```

ISrouter

```
route add -net 172.16.1.0 netmask 255.255.255.248 gw 131.20.20.2
```

```
route add -net 10.0.0.0 netmask 255.255.255.248 gw 130.10.10.2
```

Now we confirm if we are able to ping every interface from every node

NWworkstation1: `ping 10.10.0.2`

```
root@nwworkstation1:~# route add -net 131.20.20.0 netmask 255.255.255.248 gw 172.16.1.
1
root@nwworkstation1:~# route add -net 130.10.10.0 netmask 255.255.255.248 gw 172.16.1.
1
root@nwworkstation1:~# route add -net 10.10.0.0 netmask 255.255.255.248 gw 172.16.1.1r
oot@nwworkstation1:~# ping 10.10.0.2
PING 10.10.0.2 (10.10.0.2) 56(84) bytes of data.
64 bytes from 10.10.0.2: icmp_seq=1 ttl=62 time=3.48 ms
64 bytes from 10.10.0.2: icmp_seq=2 ttl=62 time=1.90 ms
64 bytes from 10.10.0.2: icmp_seq=3 ttl=62 time=1.85 ms
64 bytes from 10.10.0.2: icmp_seq=4 ttl=62 time=1.60 ms
^C
--- 10.10.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 1.607/2.211/3.482/0.744 ms
root@nwworkstation1:~#
```

SWwrokstation1: `ping 172.16.1.2`

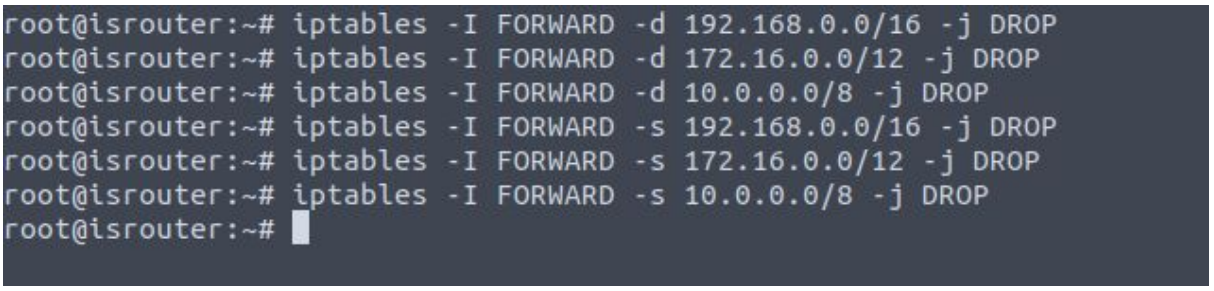
```
root@swworkstation1:~# ifconfig eth4 10.10.0.2 netmask 255.255.255.248
root@swworkstation1:~# route add -net 130.10.10.0 netmask 255.255.255.248 gw 10.
10.0.1
root@swworkstation1:~# route add -net 131.20.20.0 netmask 255.255.255.248 gw 10.
10.0.1
root@swworkstation1:~# route add -net 172.16.1.0 netmask 255.255.255.248 gw 10.1
0.0.1
root@swworkstation1:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=61 time=2.44 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=61 time=2.29 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=61 time=2.10 ms
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 2.105/2.281/2.441/0.143 ms
root@swworkstation1:~# s
```

Task 3 (Blocking private addresses)

What we do here is to create a prohibition against private addresses (SW and NW) that is, all addresses that begin with 10 and 172. Since the internet labels them as private addresses they cannot be used on out 'internetwork', so we must filter them out. Here, the IS router is serving as our internet surrogate

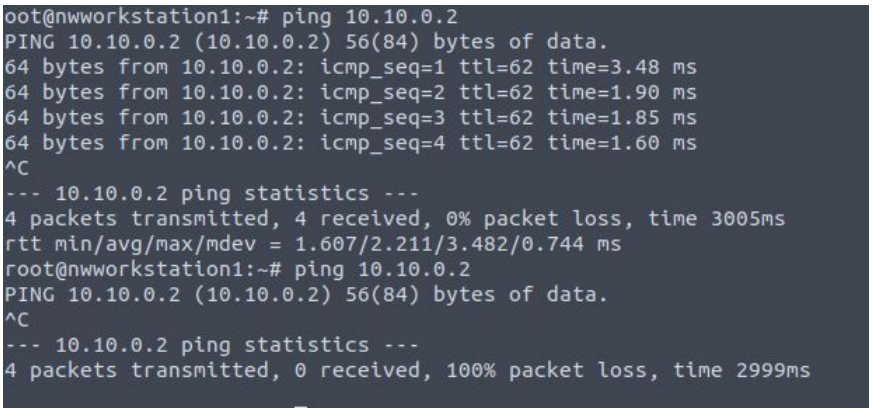
ISrouter

```
iptables -I FORWARD -d 192.168.0.0/16 -j DROP

iptables -I FORWARD -d 172.16.0.0/12 -j DROP

iptables -I FORWARD -d 10.0.0.0/8 -j DROP

iptables -I FORWARD -s 192.168.0.0/16 -j DROP

iptables -I FORWARD -s 172.16.0.0/12 -j DROP

iptables -I FORWARD -s 10.0.0.0/8 -j DROP
```



We are filtering out packets assigned to a destination from a one of the disapproved ranges will be discarded when it is noticed in the ISrouter. This will make pinging NW and SW workstations impossible:

NWworkstation: `ping 10.10.0.2`



SWworkstation: `ping 172.16.1.2`

```
root@swworkstation1:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=61 time=2.44 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=61 time=2.29 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=61 time=2.10 ms
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 2.105/2.281/2.441/0.143 ms
root@swworkstation1:~# ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2015ms
```

Task 4 (Configuring NAT in LAN routers)

Here, we try to fix the communication disabilty of the private networks in our internetwork. We do this by lending the public IP addresses of the internet-facing routers (SW and NW routers) to packets coming from workstations that are contaminated with private addresses. So we are typically trying to connect the workstations with the external internet. Simply put, we are swapping private addresses with public addresses because private networks wont work on the external internet.

Commands

NWrouter:

```
iptables -t nat -A POSTROUTING -o eth2 -s 172.16.1.0/29 -j SNAT --to 131.20.20.2
```

```
root@nwrouter:~# iptables -t nat -A POSTROUTING -o eth2 -s 172.16.1.0/29 -j SNAT
 --to 131.20.20.2
root@nwrouter:~#
```

SWrouter:

```
iptables -t nat -A POSTROUTING -o eth4 -s 10.10.0.0/29 -j SNAT --to 130.10.10.2
```

```
root@swrouter:~# iptables -t nat -A POSTROUTING -o eth4 -s 10.10.0.0/29 -j SNAT --to 130.10.10.2
root@swrouter:~#
```

Note: eth2 and eth4 are the interfaces facing the public networks.

To observe the effect, we pig at both endpoints, opposite to the router:

NWworksation1 → SWrouter: `ping 130.10.10.2`

```
root@nwworkstation1:~# ping 130.10.10.2
PING 130.10.10.2 (130.10.10.2) 56(84) bytes of data.
64 bytes from 130.10.10.2: icmp_seq=1 ttl=62 time=1.10 ms
64 bytes from 130.10.10.2: icmp_seq=2 ttl=62 time=1.05 ms
64 bytes from 130.10.10.2: icmp_seq=3 ttl=62 time=0.830 ms
64 bytes from 130.10.10.2: icmp_seq=4 ttl=62 time=1.24 ms
^C
--- 130.10.10.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.830/1.058/1.246/0.151 ms
root@nwworkstation1:~#
```

Simultaneously on ISrouter: `tcpdump -nnti eth2`

```
root@isrouter:~# tcpdump -nnti eth2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
IP 131.20.20.2 > 130.10.10.2: ICMP echo request, id 6481, seq 36, length 64
IP 130.10.10.2 > 131.20.20.2: ICMP echo reply, id 6481, seq 36, length 64
IP 131.20.20.2 > 130.10.10.2: ICMP echo request, id 6481, seq 37, length 64
IP 130.10.10.2 > 131.20.20.2: ICMP echo reply, id 6481, seq 37, length 64
IP 131.20.20.2 > 130.10.10.2: ICMP echo request, id 6481, seq 38, length 64
IP 130.10.10.2 > 131.20.20.2: ICMP echo reply, id 6481, seq 38, length 64
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
root@isrouter:~#
```

Here, we notice that packets are being transmitted but they do not bear the address of their originating node (NWworkstation) but they appear to be originating from the intermediate router, NWrouter (`131.20.20.2`), which is void of offending private addresses. But this really doesn't overcome private IP prohibition but rather avoids it.We still have the end-to-end problem. We have only solved the source half of the problem but left with the destination half.

Task 5 (Port Forwarding)

In order to solve the end-to-end problem mentioned in task 4 above, we use port forwarding. Port forwarding simply alters IP values in packets. Here, we need to install apache server on the SWworksattion1 and lynx on NWworkstation.

SWworkstation1: `sudo apt-get install apache2`

NWworkstation1: `sudo apt-get install lynx`

To ensure apache is running on SWworkstation, do: `sudo systemctl status apache2`

Now on running lynx 10.10.0.1 on NWworkstation1, we notice that it has a valid destination instead of a poisoned one. This is a serverlessness problem. What we do here is to ask the SWrouter to recognize packets addressed to it seeking a webserver that is not originally meant for it but for SWworkstation1. so we ask the SWrouter to reissue packets with SWworkstation's addresses on them. Note that, `10.10.0.2` is not poison, so there will be nor obstruction.

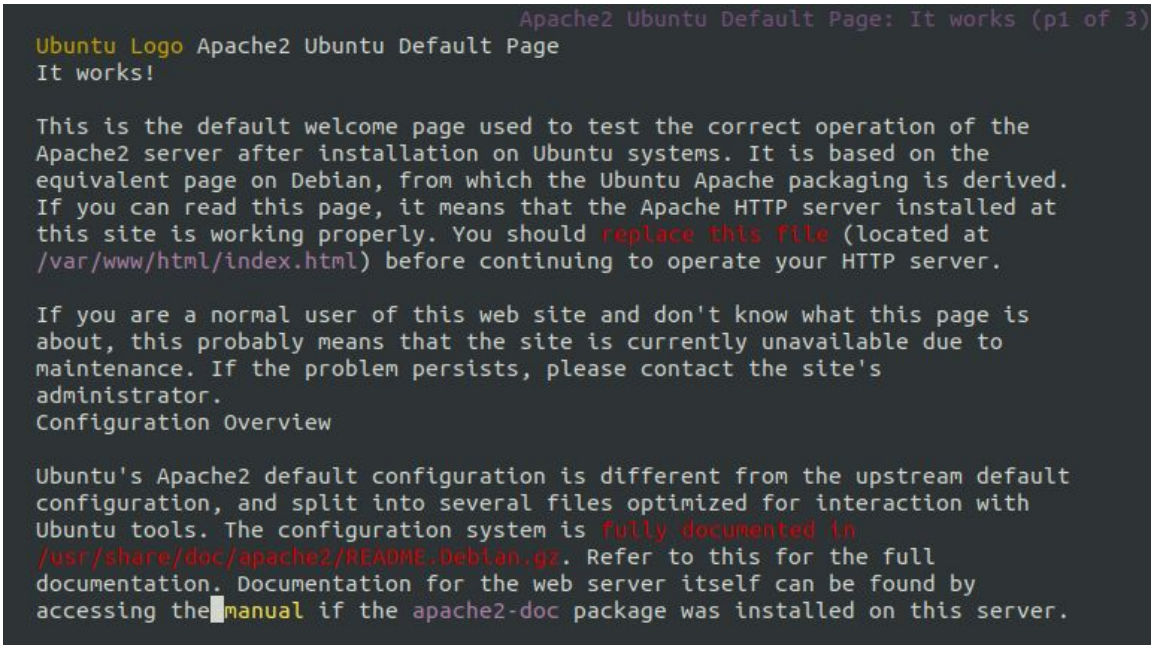
SWrouter:

```
iptables -t nat -A PREROUTING -i eth4 -d 130.10.10.2/32 -p tcp --dport 80 -j DNAT --to 10.10.0.2
```

```
root@swrouter:~# iptables -t nat -A PREROUTING -i eth4 -d 130.10.10.2/32 -p tcp --dport 80 -j DNAT --to 10.10.0.2
root@swrouter:~#
```

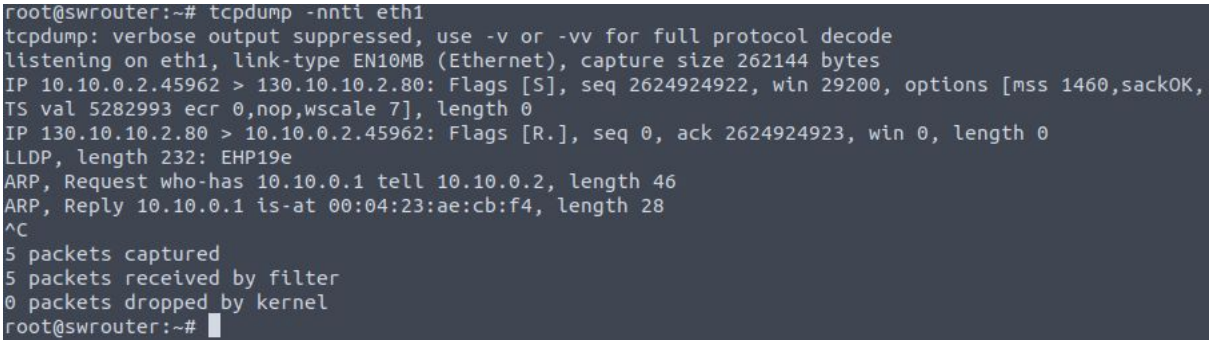
This says that for every packet arriving `-i eth4` packets, containing tcp segments `-p tcp` with destination port 80, should have their IP headers' destination address changed to this machine's `10.10.0.2` instead. So, simply put, we are swapping destination addresses. After this, the altered packet will be taken to a place where they are newly assigned to.

NWworkstation: `lynx 130.10.10.2`



Here we can notice the browser receives a default page, not from the node addressed (`130.10.10.2`) but from the node where apache server is running (SWworkstation →`10.10.0.2`).

SWrouter: `tcpdump -nnti eth1`



Simultaneously on NWworkstation1: `lynx 130.10.10.2`
Here, we noticed that the source is NWrouter and the destination address is SWworkstation1 due to port forwarding.