**EE / Cpre / SE 492 - sdmay 36**

**Self Healing 5-G**

**Week 6 Report**

*March 30 - April 12*
*Faculty Advisor: Mohamed Selim*

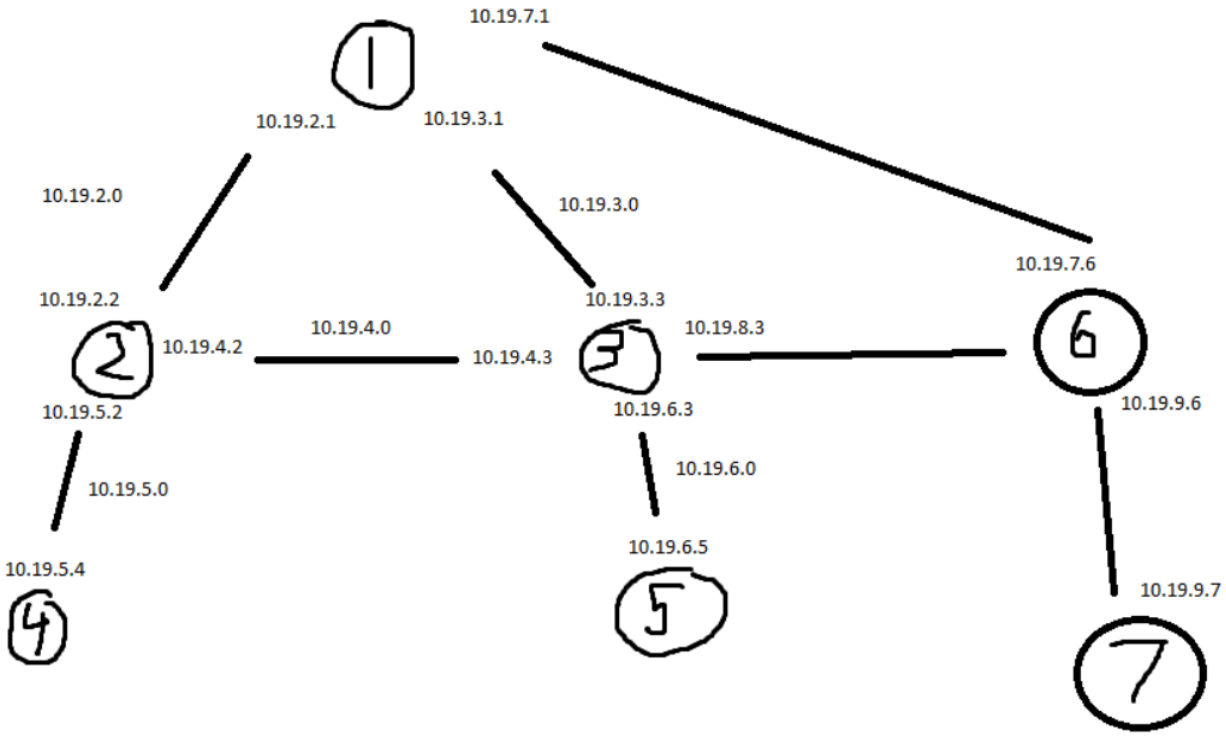
**Team Members:**
Matthew Johnson
Adam Tiedeman
Parnumart Hanthiradej (Nick)
Amber Chin
Yuin-Choon Ng (Austin)

***Weekly summary***

This week we worked on testing how to use the iperf experiment on the ORBIT testbed. In the previous report we had discussed that we would like to simulate more realistic scenarios. As a result this week we added more nodes to our network. This adds more complexity to our system but allows us to simulate more realistic scenarios. We also began implementing a bandwidth dimension to our artificial server nodes. Each server is assigned an arbitrary bandwidth and once servers go out, we are only able to serve up to our server's capacity. Finally, we changed our existing routing failure script to work on the fly. This means the central controller detects the location of the routing failure and adjusts the network to regain connectivity. We had also had discussions on how we would like to finish out our project for the semester. This includes how we would present and also what we think we can accomplish with our time left.

***Past Week Accomplishments***

- Added extra nodes to our system
- Worked on fixing our scripts so that they work more consistently
- Wrote a script to automate the entire setup process (imaging, route configuration, tool installation)
- Upgraded our initial self healing script (can now detect more than one scenario and heal it)
- Our self-healing script now works on the fly, this means it detects the failure location and dynamically plans a path to recovery using frequent icmp flooding of the network




- Expanded the network, by adding one base station and user or client. The idea of the experiment is to disconnect one base station, in this case for example, node 2, and the user which is on node 4 in the diagram above would be redirected to node 3, which is another base station whose connection is not disrupted.

```
#!/bin/bash
if [[ $1 == "1" ]]
then
    echo "!!!!!!!!!!!!!!!!!Setting UP NODES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
    bash LoadBaseline.sh
fi

if [[ $2 == "1" ]]
then
    echo "!!!!!!!!!!!!!!!!!Installing Tools!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
    bash InstallTools.sh
fi
sleep 15
if [[ $3 == "1" ]]
then
    echo "!!!!!!!!!!!!!!!!!CONFIGURING ROUTES!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"
    bash ConfigureRoutes.sh
fi
```

Above is a script that uses all written scripts written to fully automate the setup process.

```
#!/bin/bash

echo "Begin script"


echo "Turning node off--------------------------"
omf tell -a offh -t node1-1
omf tell -a offh -t node1-1

echo "Loading baseline-------------------------------"
omf load -i baseline.ndz -t node1-1

echo "Turning node on--------------------------------"
omf tell -a on -t node1-1
omf tell -a on -t node1-1
echo "Next node---------------------------"


echo "Turning node off--------------------------"
omf tell -a offh -t node1-2
omf tell -a offh -t node1-2

echo "Loading baseline-------------------------------"
omf load -i baseline.ndz -t node1-2
```

The script above is expansion of our original beginning script which will warm up all the nodes.

```
#!/bin/bash

ssh -o "StrictHostKeyChecking no" -tt root@node1-1 <<- EOT
sudo bash -c 'apt-get -y install net-tools >/dev/null 2>&1 & disown';
sudo bash -c 'apt-get -y install traceroute >/dev/null 2>&1 & disown';
exit;
EOT

ssh -o "StrictHostKeyChecking no" -tt root@node1-2 <<- EOT
sudo bash -c 'apt-get -y install net-tools >/dev/null 2>&1 & disown';
sudo bash -c 'apt-get -y install traceroute >/dev/null 2>&1 & disown';
exit;
EOT

ssh -o "StrictHostKeyChecking no" -tt root@node1-3 <<- EOT
sudo bash -c 'apt-get -y install net-tools >/dev/null 2>&1 & disown';
sudo bash -c 'apt-get -y install traceroute >/dev/null 2>&1 & disown';
exit;
EOT

ssh -o "StrictHostKeyChecking no" -tt root@node1-4 <<- EOT
sudo bash -c 'apt-get -y install net-tools >/dev/null 2>&1 & disown';
sudo bash -c 'apt-get -y install traceroute >/dev/null 2>&1 & disown';
exit;
EOT

ssh -o "StrictHostKeyChecking no" -tt root@node1-5 <<- EOT
sudo bash -c 'apt-get -y install net-tools >/dev/null 2>&1 & disown';
sudo bash -c 'apt-get -y install traceroute >/dev/null 2>&1 & disown';
exit;
EOT

ssh -o "StrictHostKeyChecking no" -tt root@node1-6 <<- EOT
sudo bash -c 'apt-get -y install net-tools >/dev/null 2>&1 & disown';
sudo bash -c 'apt-get -y install traceroute >/dev/null 2>&1 & disown';
exit;
EOT
```

Above is a script to install all the necessary tools on every node. We had issues when we tried to configure nodes and install tools at the same time so we separated the process.

```
#!/bin/bash

echo "Client 4's Route to internet";
traceroute 10.19.5.4 -i enp94s0f0;
echo "Client 5's Route to internet";
traceroute 10.19.6.5 -i enp94s0f0;
echo "Client 7's Route to internet";
traceroute 10.19.9.7 -i enp94s0f0;
while true
do
echo "Connected";
ping -c 1 -I enp94s0f0 10.19.6.5 >ping5.txt;
ping -c 1 -I enp94s0f0 10.19.5.4 >ping4.txt;
ping -c 1 -I enp94s0f0 10.19.9.7 >ping7.txt;
cat ping4.txt;
cat ping5.txt;
cat ping7.txt;
A=$(wc -l ping4.txt)
B=$(wc -l ping5.txt)
C=$(wc -l ping7.txt)
if [ "$A" == "5 ping4.txt" ]
then
echo "Fail!";
echo "Rerouting";
sudo route del -net 10.19.5.0 netmask 255.255.255.0 gw 10.19.2.2 enp94s0f0;
sudo route add -net 10.19.5.0 netmask 255.255.255.0 gw 10.19.3.3 enp94s0f0;
traceroute 10.19.5.4 -i enp94s0f0;
fi
if [ "$B" == "5 ping5.txt" ]
then
echo "Fail!";
echo "Rerouting";
sudo route del -net 10.19.6.0 netmask 255.255.255.0 gw 10.19.3.3 enp94s0f0;
sudo route add -net 10.19.6.0 netmask 255.255.255.0 gw 10.19.2.2 enp94s0f0;
traceroute 10.19.6.5 -i enp94s0f0;
fi
if [ "$C" == "5 ping7.txt" ]
then
echo "Fail!";
echo "Rerouting";
sudo route del -net 10.19.9.0 netmask 255.255.255.0 gw 10.19.7.6 enp94s0f0;
sudo route add -net 10.19.9.0 netmask 255.255.255.0 gw 10.19.3.3 enp94s0f0;
traceroute 10.19.9.7 -i enp94s0f0;
fi
sleep 1
done
```

The self healing script run on the central controller now monitors the entire network's connectivity using ICMP packets and reroute traffic based on the failing route.

```
#!/bin/bash
if [[ $1 == "1" ]]
then

ssh -o "StrictHostKeyChecking no" -tt root@node1-1 <<- EOT
bash Node1_RoutingFailure.sh &
exit
EOT

sleep 15

if [[ $2 == "2" ]]
then
ssh -o "StrictHostKeyChecking no" -tt root@node1-2 <<- EOT
sudo route del -net 10.19.2.1 netmask 255.255.255.255 gw 10.19.2.2 enp94s0f0;
exit
EOT
fi

if [[ $2 == "3" ]]
then
ssh -o "StrictHostKeyChecking no" -tt root@node1-2 <<- EOT
sudo route del -net 10.19.3.1 netmask 255.255.255.255 gw 10.19.3.3 enp94s0f0;
exit
EOT
fi
```

This script now automates the destruction of the network connectivity based on user input. It will be expanded to include other cases.

***Pending Issues***

- We still need to implement a system to read and distribute bandwidth, iperf has a means of distributing bandwidth so the plan is to map or hardcode bandwidth amounts for server client connections
- Add more fault scenarios and be able to handle multiple faults at once
- We want to try to completely knock a network node out and plan a path of recovery based on arbitrary(user configurable) bandwidth requirements of specific nodes, right now when this happens the network entirely fails

***Individual Contributions***

***Adam Tiedeman : Helped troubleshoot setup scripts and troubleshoot self healing script***
***Parnumart Hanthiradej : Tried to run our adjusted scripts***
***Yuin Choon Ng : Run self healing script to test out the system***
***Amber Chin : Tested out the connections between the nodes, to see if there is packet loss***
***Matthew Johnson : Wrote overall configuration script that functions based on user input. Rewrote network topology script to account for added network nodes and existing bugs in our previous network topology. Reconfigured central controller self-healing script to work on the fly based on the specific failing node as well as created failure script based on user input. Began experimenting with network bandwidth requirements on specific server nodes.***

| ***Team Member*** | ***Contribution*** | ***Weekly hours*** | ***Total Hours*** |
|---|---|---|---|
| ***Adam Tiedeman*** | Helped troubleshoot setup scripts and troubleshoot self healing script | ***15*** | ***77*** |
| ***Amber Chin*** | Tested out connections of the nodes, to see if there is packet loss | ***10*** | ***70*** |
| ***Matthew Johnson*** | Wrote overall configuration script that functions based on user input. Rewrote network topology script to account for added network nodes and existing bugs in our previous network topology. Reconfigured central controller self-healing script to work on the fly based on the specific failing node as well as created failure script based on user input. Began experimenting with network bandwidth requirements on specific server nodes. | **20** | ***85*** |
| ***Parnumart Hanthiradej*** | *Tried to run our adjusted scripts* | ***10*** | ***70*** |
| ***Yuin-Choon Ng*** | Run self healing script to test out the system | ***10*** | ***64*** |

**Counted total hours for only this semester*

***Plan for coming week***

- Write a more complex script for different routing situations
- Work on traffic generation that more closely simulates actual data transmission
- Work through and create a hardcoded map of bandwidths between different nodes to simulate connection strength
- Work out a way to distribute bandwidth via iperf and poll the bandwidth of nodes
- Finalize testing of self-healing scripts
- Work on knock-out test of server node
- Get bandwidth requirements working