Appendix A

Appendix A

This chapter covers the infrastructure and operation aspects of the testbed. It provides the hardware details along with basic structure and elements of the testbed. It also presents a stepwise procedure to install and configure the testbed as well as configure FlexFCAPF emulation.

A.1 Infrastructure

Figure 3.1 shows the overview of the current testbed infrastructure. The testbed consists of four physical machines (PC1, PC2, PC3, and PC4). All these four machines are connected to Paderborn University's network with an ethernet switch (20Gbits) using their internal Network Interface Cards (NICs). A terminal is connected to the first machine. A user can access the testbed using the terminal or using ssh through the university network. PC1 is used as MaxiNet frontend and the other three machines are used as MaxiNet worker. FlexFCAPF and the Ryu Controller Framework also run on PC1. Altogether, these machines build the MaxiNet Cluster. The system configuration of the testbed is described in Table A.1.

A.2 Software Installation and Configuration

All machines run on Ubuntu 14.04 LTS OS. The testbed has two user accounts configured, *crowd* and *maxinet*. The *crowd* user can be used for running any application, including SDN Controller. On the other hand, the *maxinet* user can execute commands as root with

Table A.1: System configuration of testbed machine

Details	CPU	Core	RAM	OS	Host Name	IP Address
PC1	3.00GHz	2	4GB	Ubuntu 14.04 LTS	fg-cn-crowd-1.cs.upb.de	131.234.250.30
PC2	3.00GHz	2	8GB	Ubuntu 14.04 LTS	fg-cn-crowd-2.cs.upb.de	131.234.250.31
PC3	3.00GHz	2	8GB	Ubuntu 14.04 LTS	fg-cn-crowd-3.cs.upb.de	131.234.250.32
PC4	3.00GHz	2	8GB	Ubuntu 14.04 LTS	fg-cn-crowd-4.cs.upb.de	131.234.250.33
All are Intel(R) Core(TM)2 Duo CPU E8400 processor						

sudo without a password and is needed to run the MaxiNet emulation.

The following softwares are installed in the MaxiNet Frontend (PC1):

1. Mininet v2.2.1
2. Maxinet v1.0
3. Metis v5.1
4. Pyro v4.37
5. Ryu v4.13

Meanwhile, the following softwares listed below are installed in the MaxiNet Worker (PC2, PC3, and PC4):

1. Mininet v2.2.1
2. Maxinet v1.0
3. Metis v5.1
4. Pyro v4.37
5. iperf v2.0.5 (Customized)

I assumed that the version control utility *git*, programming language *Python v2.7.6*, and package manager *pip* are already installed in all the machines. Below, I describe the installation instruction of the software mentioned above:

Metis: The archived source code of *Metis v5.1* was downloaded and installed (see Listing A.1). There is no specific configuration necessary for *Metis*.

Listing A.1: Metis nstallation

```
wget http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/metis-5.1.0.tar.gz
tar -xzf metis-5.1.0.tar.gz
cd metis-5.1.0
make config
make
sudo make install
```

Pyro: The archived source code of *Pyro v4.37* was downloaded and installed (see Listing A.2). There is no specific configuration necessary for *Pyro*.

Listing A.2: Pyro installation

```
git clone git://github.com/mininet/mininet
cd mininet
git checkout -b 2.2.1 2.2.1
cd util/
./install.sh
```

Mininet: The latest source code from the *Mininet* repository was downloaded and installed (see Listing A.3). There is no specific configuration necessary for *Mininet*.

Listing A.3: Mininet installation

```
git clone git://github.com/mininet/mininet
cd mininet
git checkout -b 2.2.1 2.2.1
cd util/
./install.sh
```

MaxiNet: The latest source code from the *MaxiNet* repository was downloaded and installed (see Listing A.4).

Listing A.4: MaxiNet installation

```
git clone git://github.com/MaxiNet/MaxiNet.git
cd MaxiNet
git checkout v1.0
sudo make install
```

Since I am running Ubuntu, I have to set up a user (*maxinet*) to use *sudo* without password (see Listing A.5). This is done by adding the following line to in the */etc/sudoers* file.

Listing A.5: Set no password

```
maxinet ALL=(ALL) NOPASSWD: ALL
```

The next thing I did was, copied the *MaxiNet.cfg* to */etc/* and modified it (see Listing A.6 and A.7).

Listing A.6: Copy MaxiNet configuration

```
sudo cp ~/MaxiNet/share/MaxiNet-cfg-sample /etc/MaxiNet.cfg
```

Listing A.7: Edit MaxiNet configuration

```
[all]
password = HalloWelt
controller = 131.234.250.30:6633
logLevel = INFO          ; Either CRITICAL, ERROR, WARNING, INFO  or DEBUG
port_ns = 9090           ; Nameserver port
port_sshd = 5345         ; Port where MaxiNet will start an ssh server on each
    worker
runWith1500MTU = True ; Set this to True if your physical network can not
    handle MTUs >1500.
```

```
useMultipleIPs = 0      ; for RSS load balancing. Set to n > 0 to use multiple
     IP addresses per worker. More information on this feature can be found
    at MaxiNets github Wiki.
deactivateTSO = True    ; Deactivate TCP-Segmentation-Offloading at the
    emulated hosts.
sshuser = maxinet           ; On Debian set this to root. On ubuntu set this to
     user which can do passwordless sudo
usesudo = True          ; If sshuser is set to something different than root
    set this to True.

[FrontendServer]
ip = 131.234.250.30

[fgcn-crowd-2]
ip = 131.234.250.31
share = 1

[fgcn-crowd-3]
ip = 131.234.250.32
share = 1

[fgcn-crowd-4]
ip = 131.234.250.33
share = 1
```

Ryu: The latest source code from the *Ryu* repository was downloaded and installed (see Listing A.8). There is no specific configuration necessary for *Ryu*, but it is good to check if the */usr/local/etc/ryu/ryu.conf* has been configured for expected host and port. For example, FCAPF Network Controller is running on PC1 and it uses WSGI port 8080 and OpenFlow listen port 6633 and host as localhost, these are default Ryu settings. It is important to note that the OpenFlow listen host and port should match the *MaxiNet* configuration, and WSGI host and port should match the FlexFCAPF configuration. FlexFCAPF uses this host and port information to call REST APIs provided by the FlexFCAPF Network Controller.

Listing A.8: Ryu installation

```
git clone git://github.com/osrg/ryu.git
cd ryu
sudo pip install .
```

Iperf: Standard *iperf* is not suitable for the testbed I built because I need to add a *bucket_id* in the *iperf* packet. The modified *iperf* code is available in the *FlexFCAPF* repository. The steps for building the *iperf* from source code is as follows (see Listing A.9):

Listing A.9: Iperf installation

```
git clone git@github.com:tarunsarkar/flexfcapf.git
cd flexfcapf/iperf2
./configure
make
cd ../
cp iperf2/src/iperf .
```

All the following scripts or executables are expected to be present inside *flexfcapf* directory of the home directory of the *maxinet* user. If not present, then the hard coded path in FlexFCAPF code referring them should be modified. They are all available in the FlexFCAPF repository:

1. fcapf_ryu_controller.py, the network controller.
2. cleanfront.sh, script to clean up and restart MaxiNet Frontend.
3. cleanworker.sh, script to clean up and restart MaxiNet Worker.
4. configure_delay.sh, script to configure processing delay in the LCAs.
5. iperf, the customized built *iperf* to generate traffic with given *bucket_id*.

A.3 System Operation

It is recommended to clone the *FlexFCAPF* repository in all the machines inside the home directory of *maxinet* user, which would create a sub-directory named *flexfcapf*. This way, all the required code, scripts, and executable are available inside *~/flexfcapf* directory. Login as *maxinet* user in all the Workers (PC2, PC3, and PC4) and execute the following command; this will clean the environment and start the worker (see Listing A.10):

Listing A.10: Clean worker

```
cd ~
git clone git@github.com:tarunsarkar/flexfcapf.git
cd flexfcapf
bash cleanworker.sh
```

Login as *maxinet* user in the Frontend (PC1) and execute the following command (see Listing A.11):

Listing A.11: Clean frontend

```
cd ~
git clone git@github.com:tarunsarkar/flexfcapf.git
cd flexfcapf
bash cleanfront.sh
```

The steps above will start the MaxiNet Frontend and will start the FCAPFNetwork-Controller in the background. Next, a Topology description file needs to be generated in the Frontend itself. The following command will generate a mesh topology description file with 36 nodes and 1000 flows (see Listing A.12):

Listing A.12: Generate topology description

```
cd ~/flexfcapf
python Generator_mesh.py 6 6 1000 Test_36_mesh.dat
```

Next execute the following command to start the emulation test (see Listing A.13).

Listing A.13: Start emulation exeution

```
cd ~/flexfcapf
python csimpfo_fcfs.py MaxiNet 7200 Test_36_mesh.dat
```

This will execute the test for 2 hours and test results will be saved in */flexfcapf/results* directory.

Execute the following command to setup the network without the emulation (see Listing A.14).

Listing A.14: Setup network without emulation

```
cd ~/flexfcapf
python crun_flexfcfs.py MaxiNet Test_36_mesh.dat
```

This will set up a network with 36 nodes and will also run the algorithm once so that forwarding entries are added in the underlying network elements.