1 User Manual for Deploying and Testing MEADcast

This document shows the components used for the experiments in the thesis and how they are deployed on the virtual machines.

1.1 Components

These applications and files are the components deployed on the virtual machines:

- mc_functions.py
- mc_file_sender.py
- mc_controller.py
- udp_file_sender.py
- udp_file_receiver5005.py and udp_file_receiver5006.py
- ndp.py

mc_functions.py is a python file that contains functions related to crafting MEADcast packets. It gets imported and used by both **mc_file_sender.py** and **mc_controller.py**

mc_file_sender.py is the application that generates traffic on the sender machine. It has two functions. The first one sends out MEADcast discovery request packets and the other one waits for MEADcast discovery response packets, generates the topology viewpoint and sends data to all the receivers.

mc_controller.py is an application that acts as a controller in a SDN. It only handles MEADcast packets and packets related to the Neighbor Discovery Protocol (NDP).

udp_file_sender.py is a simple application that sends a file to a list of receivers via unicast.

udp_file_receiver5005.py and **udp_file_receiver5006.py** are both applications that are run on the receiver machines. **udp_file_receiver5005.py** listens on UDP destination port 5005 and is used for MEADcast testing while **udp_file_receiver5006.py** uses UDP destination port 5006 and is used for unicast testing. **udp_file_receiver5006.py** is being used to avoid the flow installed on every switch, that sends every packet with UDP destination port 5005 to the controller. An alternative to avoid that flow, would be to delete every installed flow on all switches, before testing with unicast packets.

ndp.py is the NDP application written by Cuong Ngoc Tran and is used to transmit unicast packets. It acts as the controller and installs flows on the switches. It is used instead of **simple_switch_13.py** provided by RYU, because it can handle the loops in topology 2.

1.2 Sending and receiving MEADcast packets

To transmit data it is necessary to first run **mc_controller.py** on the controller machine:

ryu-manager mc_controller.py –observe-links

–observe-links is necessary for the controller, to view the links between the switches.

For every host that wants to receive the data sent via MEADcast, it is necessary to run this command on every receiving host machine:

python udp_file_receiver5005.py

The sender has to start **udp_file_sender.py** twice with different parameters. The first instance listens to discovery response packets and sends the data:

python mc_file_sender.py senddata $< argument1 > < argument2 >$

where $< argument1 >$ is the name of the file that is intended to get sent. $< argument1 >$ can be any file as long as it is in the same directory as the application. This instance will wait for discover response packets. It will build the topology viewpoint after every incoming discovery response packet and will start sending out the data, once 5 seconds have passed after the last discovery response packet.

The second argument $< argument2 >$ is the delay in seconds between between each packet that is being sent. It is being used to regulate the speed at which the sender sends out the packet. A value of 5 translates to 1 packet being sent every 5 seconds, while a value of 0.02 results in 50 packets being sent out per second.

To send the discovery request packets to each receiver:

python mc_file_sender.py senddisco $< argument1 >$

where $< argument1 >$ is the amount of receivers that the sender wants to send discovery request packets to. Currently 2, 4, 6 and 9 can be chosen as arguments. 2 and 4 only work for topology 1, while 6 and 9 only work on topology 2.

To start sending the file *10MB.txt* from pc1 to 6 receivers, with a delay of 0.02 seconds, the commands on pc1 would look like this:

python mc_file_sender.py senddata 10MB.txt 0.02
python mc_file_sender.py senddisco 6

1.3 Sending and receiving unicast packets

In order to send unicast packets, it is required to start **ndp.py** instead of **mc_controller.py** on the controller machine.

ryu-manager ndp.py –observe-links

To start the application that receives can receive data on the receiver host machines:

python udp_file_receiver5006.py

This application listens on UDP port 5006 for incoming packets.

To start sending unicast packets from the sender:

python udp_file_sender.py $< argument1 > < argument2 > < argument3 >$

The first argument $< argument1 >$ is the amount of receivers the sender wants to address. Available options are 2, 4, 6 and 9. 2 and 4 only work for topology 1, while 6 and 9 only work for topology 2.

The second argument $< argument2 >$ is the name of the file.

The third argument $< argument3 >$ is the delay in seconds between each packet that is being sent. It is being used to regulate the speed at which the sender sends out the packet. A value of 5 translates to 1 packet being sent every 5 seconds, while a value of 0.02 results in 50 packets being sent out per second.

If the sender wants to send the file *10MB.txt* to 4 receivers in topology 1 with a speed of 100 packets per second(delay of 0.01), the command would look like this:

python udp_file_sender.py 4 10MB.txt 0.01