Computer Networks
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320301
Date: 2015-10-28
Due: 2015-11-04

Problem Sheet #4

Mininet (http://mininet.org/) is a network emulator that uses Linux namespaces to emulate a complete network within a single Linux kernel. While you can install and run mininet inside your Linux kernel, it is easier to download a mininet virtual machine image (http://mininet.org/download/) and to run that image in VirtualBox (https://www.virtualbox.org/).

While mininet has been primarily designed to experiment with new networking technologies such as OpenFlow, we will use mininet here to create some simple network topologies and to do some basic experiments on it. Next to mininet, you will use tools such as `tcpdump`, `wireshark`, `iperf`, and the routing daemon `bird`.

Problem 4.1: *mininet network emulation* (3+1 = 4 points)

a) Download the mininet virtual machine image and run it in VirtualBox. Configure VirtualBox such that the mininet virtual machine has a NATed network interface to reach the Internet and a host network interface that allows you to login from your computer into the mininet virtual machine using ssh.

Within the mininet virtual machine, run mininet by typing the following command:

```
sudo mn
```

Run `iperf` within mininet to determine the throughput between the two nodes `h1` and `h2`. Next, run mininet with different link configurations:

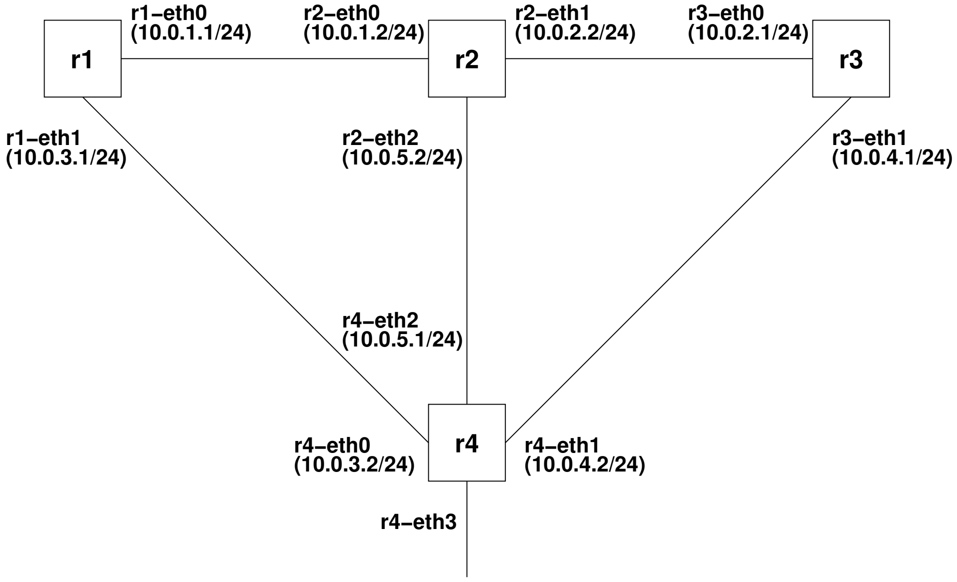
```
sudo mn --link tc,bw=10
sudo mn --link tc,bw=10,delay='10ms'
sudo mn --link tc,bw=10,delay='10ms',loss=1
sudo mn --link tc,bw=10,delay='10ms',loss=5
sudo mn --link tc,bw=10,delay='10ms',loss=10
```

Document the results you measure in a table.

b) For one of the runs, capture the packets in `h1` by running `tcpdump -w /tmp/h1.pcap &` in the background. Load the `h1.pcap` file into `wireshark` in order to study the TCP exchange. Which endpoint is initiating the teardown of the `iperf` TCP connection?

Problem 4.2: *mininet OSPF routing using bird* (2+2+2 = 6 points)

a) The power of mininet is the Python API provided. Write a Python script to generate a topology consisting of four nodes (r1, r2, r3, and r4).



Use the `addHost()` and `addLink()` API functions. Establish connectivity between neighbors using the IPv4 addresses shown above and using static routes.

b) Install the `bird` Internet routing daemon and write a configuration file for each router that enables OSPF. Enable IP forwarding on each router and let the routing daemons establish connectivity between all nodes. Show the content of the forwarding tables of all nodes (`ip route`).

c) Use the `birdc` command line client to inspect the state maintained by the `bird` daemons. Take down the link between `r1` and `r2`. How do the routing tables look like after OSPF converges? (Verify that all nodes can still reach each other.)

Note that you have to make sure that the `bird` configuration files and the `bird` control sockets are all kept separate.