Computer Networks
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CO20-320301
Date: 2018-03-02
Due: 2018-03-09

CN 2018 Problem Sheet #2

Problem 2.1: *analysis of packet traces* (1+2+1 = 4 points)

One way to learn about computer networks is to inspect packet traces. A powerful tool to analyze packet traces is wireshark, `https://www.wireshark.org/`. A common file format for packet traces are `pcap` files (packet capturing files). To answer the following questions, you have to get the `pcap` linked to this assignment on the course web page.

a) Look at the capture file properties. How many packets and bytes have been captured? Then look at the endpoint statistics to find details about Ethernet broadcasts. How many packets and bytes have been broadcasted? What is the percentage of broadcast packets and broadcast bytes?

b) Filter the packets for bridge PDUs. The wireshark filter expression is simply 'stp' (short for spanning tree protocol). Which MAC address is sending bridge PDUs? To which destination address are bridge PDUs sent? How frequently are bridge PDUs sent? What is the bridge identifier (priority plus MAC address) of the root bridge?

c) The spanning tree protocol used by bridges uses LLC encapsulation. Are there any other protocols in the trace that use LLC encapsulation? Try to filter for LLC and then select the names from the 'Protocol' column of the main window.

Problem 2.2: *basic IPv6 network in mininet* (4+2 = 6 points)

Mininet by default automatically assigns IPv4 addresses to hosts and establishes connectivity between the hosts by emulating a local area network. (Mininet by default uses OpenFlow switches with an external controller.)

Creating an IPv6 network with the current version of Mininet requires to manually configure network interfaces and forwarding rules. The Linux command line tool of choice to configure the IP networking stack is called `ip` (you will often find explanations online how to use tools like `ifconfig` or `route` but we do not use them).

a) Write a Mininet script that establishes the following toplogy:

```
  2001:638:709:a::/64 2001:638:709:f::/64 2001:638:709:b::/64
 .------------------..------------------..------------------.

h1 ------ s1 ------ r1 ------ s0 ------ r2 ------ s2 ------ h2
  eth0          eth0  eth1          eth0  eth1          eth0
```

Host `h1` is connected via switch `s1` to router `r1` and host `h2` is connected via switch `s2` to router `r2`. The routers `r1` and `r2` are connected via switch `s0`. The IPv6 networks use the prefixes shown above. Assign the following IPv6 addresses and prefixes to the hosts and routers:

`h1-eth0`	`2001:638:709:a::1/64`
`r1-eth0`	`2001:638:709:a::f/64`
`r1-eth1`	`2001:638:709:f::1/64`
`r2-eth0`	`2001:638:709:f::2/64`
`r2-eth1`	`2001:638:709:b::f/64`
`h2-eth0`	`2001:638:709:b::1/64`

Use the ip(8) command to configure the IPv6 addresses and the IPv6 forwarding table entries that are necessary to establish connectivity. Make sure that the routers have forwarding enabled (sysctl -w net.ipv6.conf.all.forwarding=1). It is recommended to use the 'Mid-level API' of Mininet.

Send the configuration commands via h1.cmd() etc. to the hosts before you call net.start(). Verfiy that your IPv6 network works by running ping and traceroute (or mtr) from all nodes to each other. Make sure you use IPv6 and not accidentally IPv4. (You can disable the automatic assignment of IPv4 addresses by passing `ip=None` to `addHost()`.)

b) Add a second network 2001:638:709:e::/64 consisting of `r3`, `s3`, `r4`

```
     2001:638:709:a::/64 2001:638:709:f::/64 2001:638:709:b::/64
    .-------------------..-------------------..-------------------.

   h1 ------ s1 ------ r1 ------ s0 ------ r2 ------ s2 ------ h2
     eth0     |   eth0   eth1          eth0  eth1    |     eth0
              |                                      |
              '------- r3 ------ s3 ------ r4 -------'
                  eth0   eth1          eth0  eth1

                        '-------------------'
                         2001:638:709:e::/64
```

Use the following IPv6 addresses:

```
  r3-eth0  2001:638:709:a::e/64
  r3-eth1  2001:638:709:e::1/64
  r4-eth0  2001:638:709:e::2/64
  r4-eth1  2001:638:709:b::e/64
```

Configure the forwarding rules such that traffic from `h1` flows via `r1` and `r2` to `h2` while traffic from `h2` flows via `r4` and `r3` to `h1`. Show that the traffic between `h1` and `h2` is using an asymmetric path.

Here is a Mininet script that you may use as a starting point. It also includes some hints how to debug your script if things do not work out on first try.

```
#!/usr/bin/env python
"""p2-template.py:

    This mininet script creates a linear routed multi-hop topology:

        2001:638:709:a::/64 2001:638:709:f::/64 2001:638:709:b::/64
       .-------------------..-------------------..-------------------.

      h1 ------ s1 ------ r1 ------ s0 ------ r2 ------ s2 ------ h2
        eth0           eth0  eth1          eth0  eth1          eth0

     Assign the following IPv6 addresses and prefixes:

     h1-eth0  2001:638:709:a::1/64
     r1-eth0  2001:638:709:a::f/64
     r1-eth1  2001:638:709:f::1/64
     r2-eth0  2001:638:709:f::2/64
     r2-eth1  2001:638:709:b::f/64
     h2-eth0  2001:638:709:b::1/64

     Use the ip(8) command to configure the IPv6 addresses and the
     IPv6 forwarding table entries that are necessary to establish
     connectivity. Make sure that the routers have forwarding enabled
     (sysctl -w net.ipv6.conf.all.forwarding=1).

     Send the configuration commands via h1.cmd() etc. to the hosts
     before you call net.start(). Verfiy that your IPv6 network works by
     running ping and traceroute (or mtr) from all nodes to each other
     (make sure you use IPv6 and not accidentally IPv4).
     Debugging hints:

     - If you get a 'network unreachable' error, then most likely a
       forwarding table entry is missing somewhere or wrong.

     - If you get a 'destination unreachable' error, then most likely
       neighbor discovery failed somewhere.

     - Use the 'ip -6 address', 'ip -6 route', and 'ip -6 neigh'
       commands to inspect the forwarding tables and the neighbor
       mapping tables.

     - If pings from h1 to h2 do not work, try to ping r1 from h1 and
       try to ping r1 from h2. (Test whether the links work first.)

     - You can print the result of the configuration commands to see
       whether any errors occurred.

     - You can run tcpdump (or even wireshark) to capture packets in
       order to see what is going on.
"""

from mininet.cli import CLI
from mininet.net import Mininet
from mininet.nodelib import LinuxBridge
from mininet.log import setLogLevel

if __name__ == '__main__':
    setLogLevel('info')

    net = Mininet(switch=LinuxBridge, controller=None)

    h1 = net.addHost('h1', ip=None)
    # create rest of the topology here

    # configure IPv6 addresses and forwarding table entries here
    print h1.cmd("ip -V")

    net.start()
    CLI(net)
    net.stop()
```