OLSR and IPv6 using Mininet

Álvaro Silva, Afonso Queirós, Bruno Maurício

April 2020

1 Questions

Q1)

Imagine that we would use IPv4, and that PC1 address was for instance 169.254.1.1/16. What would be the destination IP address of the HELLO packets sent by PC1? Assume that the Ipv4Broadcast configuration of the OLSR daemon is not set.

R): As it says in the olsrd.conf configuration file, if IPV4Broadcast address is not set, the broadcast address configured in the interface will be used. By default, the broadcast address in an interface is the directed broadcast address of that interfaces' network. Therefore, PC1 would send a broadcast to "169.254.255.255". Experimental tests verify this.

Q2)

Based on the logs collected determine how much bandwidth-per-node does the OLSR protocol consume in this scenario.

R): According to wiresharks' statistics, the average bandwidth (bitrate) per node is the following:
For the line topology:
Node 1: 122b/s
Node 2: 285b/s
Node 3: 296b/s
Node 4: 120b/s
For the star topology:
Node 1: 121b/s
Node 2: 298b/s
Node 3: 120b/s
Node 4: 119b/s

Q3)

When node 4 moves, there's a period during which there is no network connectivity. Measure this time period and find a relation between its duration and the HelloInterval and TcInterval parameters.

R): To evaluate this time, we set a ping with 500ms interval from PC1 to PC4. The difference in time between the first packetping with no response and the first ping with a response is of about 14 seconds.
The time to reestablish the connection should be approximately equal to the bidirectional test of the links between the PC1 and PC2. Since it was the PC4 to move, this time would have a maximum time of one HelloInterval (from H4, so that H2 detects H4), then a TcInterval from H2 to both H4 (now H4 knows where H1 is) and to H1. This means that in the worst case scenario, this interval would be of TcInterval+HelloInterval= 10+6 = 16 seconds.

Q4)

In the mobility scenario, the topology of the network changes after node 4 moves. Identify which nodes act as Multi-Point Relayers before and after the topology change, based on the log information.

R): Before the topology change:

Node 2 considers only node 3 as an MPR node, while node 1's link is just a Symmetric Link (packet 78, Line capture).
Node 3 considers node 2 as an MPR node, and node 4 as a normal node with a Symmetric Link (packet 79, Line capture).
Node 4 only sees node 3 and considers it it's MPR node (packet 80, Line capture).
Node 1 only sees node 2 and considers it it's MPR node (packet 81, Line capture).

After the topology change:
Node 2 is the only MPR node. 2 considers all links Symetric (packet 69, Star capture), while all other nodes consider node 2 their MPR.
This conclusion (that was taken from observing the Hello messages), could have also been taken from the bandwidth per node evaluated in Q2).

Q5)

Throughout the OLSR guide (section 2), several network configurations had to be made, including routing flags, IPv6 settings and addresses. Taking advantage of the Mininet Python API (see the links in the next section), explain how you can make these network configurations directly on the mininet_olsr_topology.py Python script.

R): Instead of running the command mininet, it is possible to use Mininet's python API to directly run shell scripts on the hosts:
(h1.cmd("echo a >/home/netedu/Desktop/proof")).
It is necessary to launch the script directly, and creating a function that calls Mininet with the correct topology and can control the hosts. For example, we can add the line:
h1.cmd("ifconfig h1-eth0 inet6 add fc00::1/64"))
to set an IPV6 address to h1. We can also run all the commands that are in the bashscripts in this way.

Q6)

Taking advantage of Mininet's API, is it possible to change the network topology at runtime (i.e., without stopping and restarting Mininet)? Briefly explain how you would do this, or why it is not possible.

R): In the same sense that in the previous question, we could communicate with all the hosts inside the network, and since in this work we had to change the topology "on the fly" with the provided bash script, it is indeed possible to change the topology without restarting Mininet, as we have done previously.
We can also "physically" change the topology in the following way: In the python script that we provided we initialize the Mininet topology inside another class. The creation of the Topology is done with the start method, but isn't necessarily final. We can for instance use an initial topology, and follow it with a loop that awaits for user input and then we can call Mininets' net class methods like, net.addHost() and others, effectively changing the topology, never needing to rerun the script nor issue a start() or stop(). So in the same script there can be two or more topologies. One before the user input and another one after.