OLSR Guide for Linux

1.1 Installation:

1. The latest release is 0.6.0. Fetch the latest release from http://www.olsr.org/?q=download.
2. Unpack, compile and install the source code:

```
# tar jxvf olsrd-x.y.z
# cd olsrd-x.y.z
# make build_all
# make install_all
```

3. The **olsrd** gets installed to /usr/bin/
4. Follow the instructions at the end of the installation.

Check out the /etc/olsrd.conf config file, and change values to fit your system. All values in this file can be overridden with command line options to **olsrd**. The main options to change are:

```
# Debug level(0-9)
# If set to 0 the daemon runs in the background
Debug           1
# IP version to use (4 or 6)
Ipversion       6
# A list of whitespace separated interface names
Interfaces      eth1
```

Later on, when you know OLSRd is configured correctly, you may set "DEBUG" to **0** to make it run in the background. You may then also add it to your init scripts. But to test that everything first, set this to at least **1** (setting this higher will produce a lot more info messages on APM, forwarding, parsing of the config file etc.)

2.1 Using OLSRd

2.1.1. On one host

When OLSRd is installed and configured, it can be started as root with:

```
# olsrd
```

All the settings in /etc/olsrd.conf can be overridden by command line options:

```
# olsrd -i eth1 -d 1 -ipv6
```

Would start **olsrd** listening on interface **eth1** using IPv6 and with debug messages.

We start olsrd:

```
# olsrd -i eth1 -d 1 -ipv6

   *** olsrd - 0.6.0 ***
```

```
hello interval = 2.00       hello int nonwireless: = 4.00 ❶
tc interval = 5.00          polling interval = 0.10
neighbor_hold_time = 6.00   neighbor_hold_time_nw = 12.00
topology_hold_time = 15.00  tos setting = 16
hna_interval = 15.00         mid_interval = 5.00
Willingness set to 3 - next update in 20.000000 secs
Using IP version 6
Using multicast address ff05::15

 ---- Interface configuration ----

 eth1:                                              ❷
        Address: fec0:106:2700::10
        Multicast: ff05::15
        Interface eth1 set up for use with index 0


 Main address: fec0:106:2700::10                    ❸

 NEIGHBORS: l=linkstate, m=MPR, w=willingness

 Thread created - polling every 0.10 seconds        ❹
 neighbor list: 11:43:17.214807
 neighbor list: 11:43:19.194967
 neighbor list: 11:43:21.395046
 neighbor list: 11:43:23.604800
 neighbor list: 11:43:25.694875
```

❶ This shows all the settings OLSRd is using. You may override these by either specifying it in the config file (`/etc/olsrd.conf`) or specify it at the command line. Read the OLSR RFC for a description on what all these settings means.

❷ OLSRd found our interface. If you are using OLSRd with multiple interfaces, "Multiple Interface Declaration" (MID) messages will be generated.

❸ If you are using OLSRd with multiple interfaces, it will pick the first interface specified as the "main" address.

❹ Since no other hosts are running OLSRd, this list is empty.

Another thing worth noticing, is that an entry is added to our routing table:

```
# route -A inet6
Destination:   Next Hop   Flags  Metric  Ref  Use Iface
...
ff05::15/128   ff05::15   UAC    0       1    1   eth1
...
```

This is the IPv6 multicast address OLSR is using to talk to other nodes running OLSR.

2.1.2. Adding other hosts

There is no point in using OLSRd on only one node, so we add some nodes. You will then see the "neighbor list" gets updated:

```
neighbor list: 12:55:14.733586
```

```
neighbor list: 12:55:18.803585
Willingness for fec0:106:2700::11 changed from 0 to 3 - UPDATING  ❶
neighbor list: 12:55:22.763585
fec0:106:2700::11:l=0:m=0:w=3[2hlist:]                            ❷
neighbor list: 12:55:26.833589
fec0:106:2700::11:l=1:m=0:w=3[2hlist:]
Willingness for fec0:106:2700::12 changed from 0 to 2 - UPDATING  ❸
neighbor list: 12:55:30.903585
fec0:106:2700::12:l=0:m=0:w=2[2hlist:]
fec0:106:2700::11:l=1:m=0:w=3[2hlist:]
neighbor list: 12:55:34.863585
fec0:106:2700::12:l=0:m=0:w=2[2hlist:]
fec0:106:2700::11:l=1:m=0:w=3[2hlist:]
neighbor list: 12:55:39.153586
fec0:106:2700::12:l=1:m=0:w=2[2hlist:fec0:106:2700::11:]          ❹
fec0:106:2700::11:l=1:m=0:w=3[2hlist:fec0:106:2700::12:]          ❺
neighbor list: 12:55:43.443605
fec0:106:2700::12:l=1:m=0:w=2[2hlist:fec0:106:2700::11:]
fec0:106:2700::11:l=1:m=0:w=3[2hlist:fec0:106:2700::12:]
```

❶ Another node detected (node B). This specifies the willingness of a node to carry and forward traffic for other nodes. Here the new node **`fec0:106:2700::11`** is willing to forward traffic. A host with low battery may not be willing to forward large amount of traffic, - so it will proclaim a lower willingness value (routing based on powerstatus is available as a plugin).

❷ The node has been added to our routing table. We can not (yet) reach any other node by way of this node, since the 2-hop neighbor list (**`[2hlist:]`**) is empty. A 2-hop neighbor is a node heard by a neighbor.

❸ Here is a third node (node C) running OLSRd.

❹ After a short time, when all nodes have been updated and routes calculated, we may also reach any of the other nodes via the other. The 2-hop neighbor list (**`[2hlist:]`**) is populated: We can reach node **`B`** via **`C`**.

❺ Here we can reach node **`C`** via **`B`**.

You will also see the routing table is updated with the new hosts:

```
# route -A inet6
Destination:           Next Hop   Flags  Metric  Ref  Use Iface
...
fec0:106:2700::11/128  ::         UH     1       0    0   eth1
fec0:106:2700::12/128  ::         UH     1       0    0   eth1
...
```

The real beauty of OLSR is when you add a bunch of nodes and move them around. You can still reach each one of them either directly (if they are close), or through other nodes.

2.1.3 Movement

When every node can reach every other node, it's no fun. Let's start moving the nodes, so that node "A" and "B" are out of (radio) range of each other. So when we move node "A" far enough away so that it can't hear node "C", all traffic must go through node "B":

We move our three nodes so that node **A** and **C** must speak through node **B** to reach each other.

Tip: Instead of physically moving the nodes around, you can use **ip6tables**. You can drop all packet using the MAC-address. You just need to block on one node:

```
# ip6tables -A INPUT -m mac --mac-source XX:XX:XX:XX:XX:XX -j DROP
```

The output from OLSRd on host A is then:

```
neighbor list: 13:22:35.693587
fec0:106:2700::11:l=1:m=1:w=3[2hlist:fec0:106:2700::12:]  (1)
neighbor list: 13:22:40.093588
fec0:106:2700::11:l=1:m=1:w=3[2hlist:fec0:106:2700::12:]
neighbor list: 13:22:44.053594
fec0:106:2700::11:l=1:m=1:w=3[2hlist:fec0:106:2700::12:]
neighbor list: 13:22:48.233594
fec0:106:2700::11:l=1:m=1:w=3[2hlist:fec0:106:2700::12:]
neighbor list: 13:22:52.193605
fec0:106:2700::11:l=1:m=1:w=3[2hlist:fec0:106:2700::12:]
```

(1) We can reach node **B** directly, and via node **B** we can reach node **C**.

The routing table also gets updated. For node **A** to reach node **C** it must go through node **B**:

```
# route -A inet6
Destination:             Next Hop           Flags  Metric  Ref  Use Iface
...
fec0:106:2700::11/128    ::                 UH      1      1    0   eth1
fec0:106:2700::12/128    fec0:106:2700::11  UGH     2      0    0   eth1
...
```

3.1 What about HNA messages?

" In order to provide this capability of injecting external routing information into an OLSR MANET, a node with such non-MANET interfaces periodically issues a Host and Network Association (HNA) message, containing sufficient information for the recipients to construct an appropriate routing table."

" An example of such a situation could be where a node is equipped with a fixed network (e.g., an Ethernet) connecting to a larger network as well as a wireless network interface running OLSR." --- RFC3626: OLSR, section 12 (page 51).

OLSR with a gateway (GW), that sends out HNA messages. All the other nodes may then be accessing the "Internet"

To have one node, act as a gateway and send out HNA messages, you must change the **HNA6** in

/etc/olsrd.conf:

```
# HNA IPv6 routes
# syntax: netaddr prefix
# Example Internet gateway
HNA6 0:: 0
```

When you start **OLSRd**, you will see the node is sending out HNA messages periodically:

```
...
Sending HNA (48 bytes)...
...
```

When the other nodes receives a HNA message, they update their routing table:

```
# route -A inet6
Destination:             Next Hop           Flags  Metric  Ref  Use Iface
...
::/0                     fec0:106:2700::1   UG     1       0    0   eth1
...
```

You may also have multiple nodes in a MANET to act as gateways (sending out HNA messages). Each mobile node then use the nearest gateway.