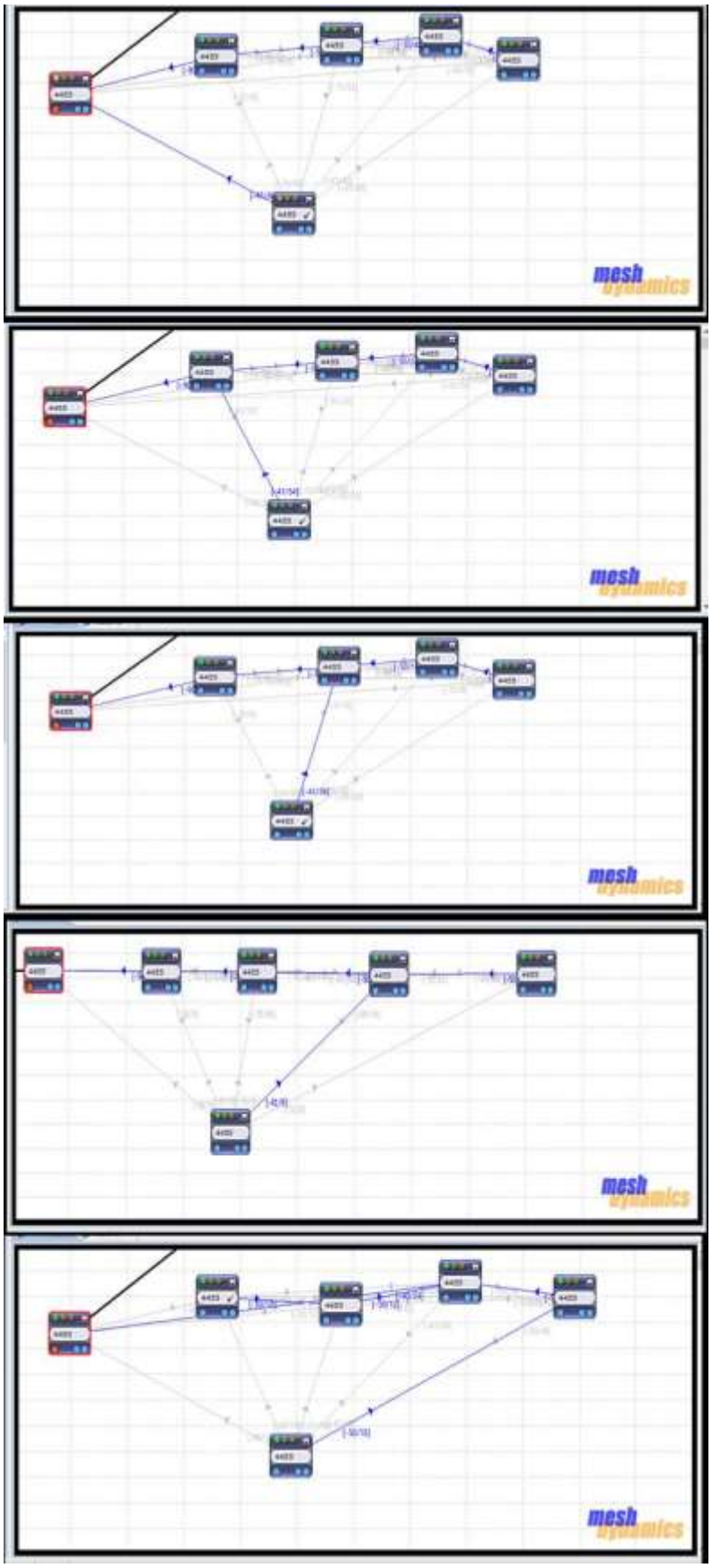
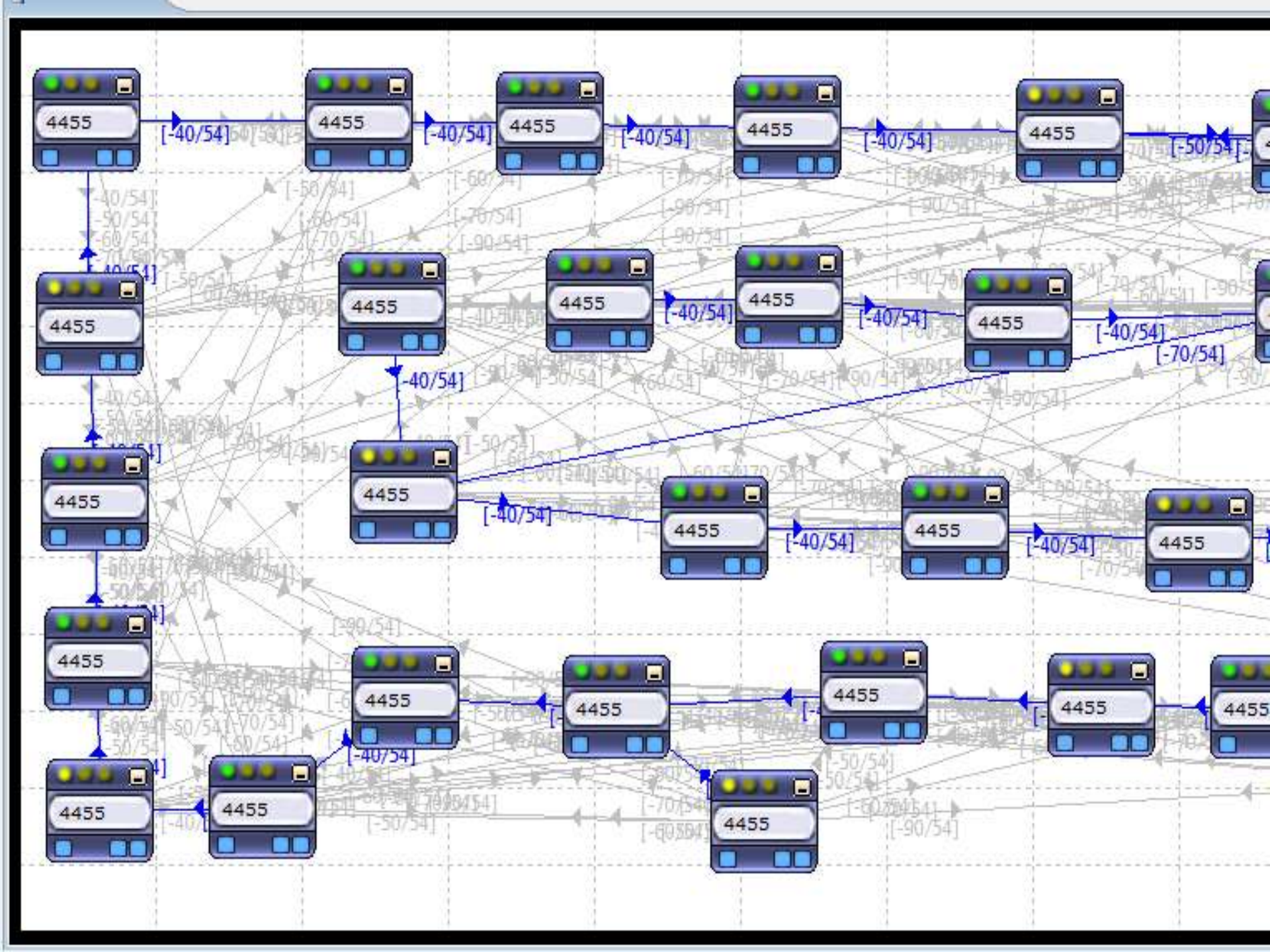
Meshdynamics Wi-Fi Simulator User Guide




4455
[-40/54]
[-50/54]
[-70/54]

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR MESHDYNAMICS REPRESENTATIVE FOR A COPY.

The performance testing tool in the MD4000 family of products and the Network Viewer uses Iperf, a program developed by the University of Illinois. Copyright © 1999-2014, The board of trustees of the University of Illinois. For more information please visit http://dast.nlanr.net/Projects/Iperf/ui_license.html

The Network Viewer uses geographic map data from OpenStreetMap.
OpenStreetMap is a free editable map of the whole world. Fore more information please visit
http://www.openstreetmap.org

The Network Viewer uses Java™.
Java™ is a registered trademark of Sun Microsystems. For more information please visit
http://www.oracle.com/technetwork/java/javase/downloads/index.html

THIS COMPUTER PROGRAM AND UNDERLYING CONCEPTS ARE PROTECTED BY PATENT AND COPYRIGHT LAWS. UNAUTHORIZED REPRODUCTION OR DISTRIBUTION OF THIS PROGRAM, OR ANY PORTION OF IT, MAY RESULT IN SEVERE CIVIL AND CRIMINAL PENALTIES AND WILL BE PROSECUTED TO THE MAXIMUM EXTENT.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

All other trademarks mentioned in this document or the Meshdynamics website are the property of their respective owners.
Structured Mesh™, Modular Mesh™, MeshControl™, P3M™, PBV™ are rademarks of Meshdynamics, Inc.

Audience

This guide is for the networking professional who designs, installs and/or manages Meshdynamics MD4000 mesh nodes.
To use this guide, you should be familiar with the concepts and terminology of wireless local area networks.
You should also be familiar with Meshdynamics multi-radio tree based mesh networking architecture and model configs.

Purpose

This guide provides the information to simulate and validation network design layouts for static and mobile networks.

Related Publications

Meshdynamics MD4000 Hardware Installation Guide
Meshdynamics MD4000 FIPS Security Policy
MD4000 4.9 GHz US Public Safety Management Guide
MD4000 Channel Management

Obtaining Documentation

Meshdynamics documentation and additional literature are available on Meshdynamics.com.

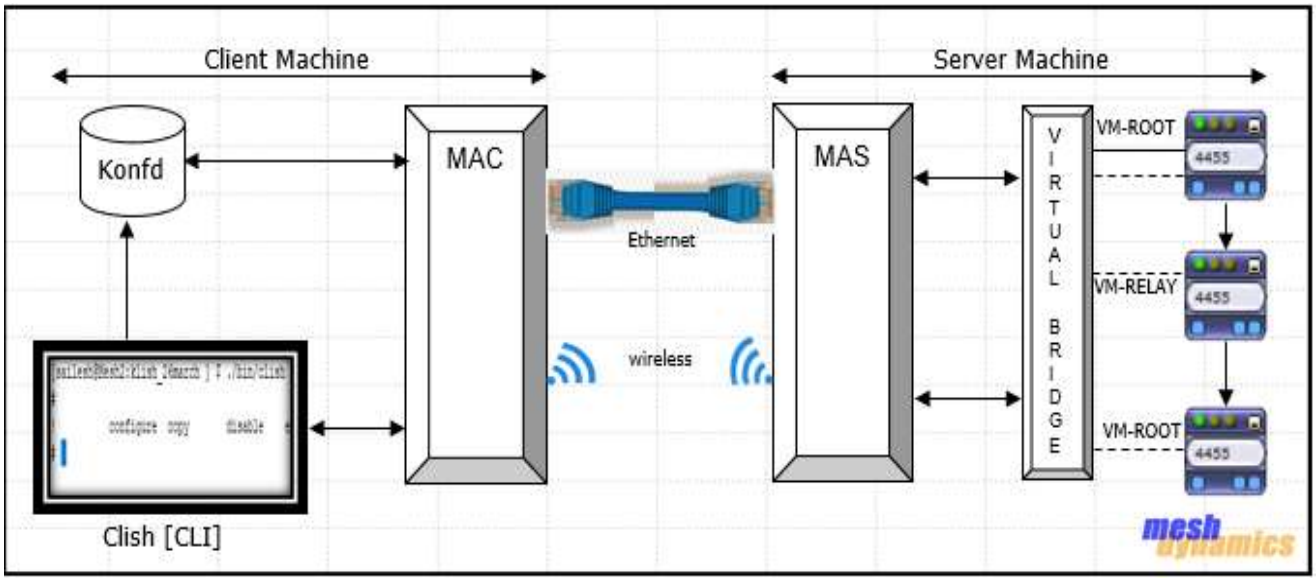
You can access the most current Meshdynamics documentation at the following URLs:
http://support.meshdynamics.com
http://www.meshdynamics.com/tech-presentations.html

You can access the Meshdynamics website at this URL:
http://www.meshdynamics.com

Documentation Feedback

You can rate and provide feedback about Meshdynamics technical documents by sending us comments to techsupport@meshdynamics.com.

We appreciate your comments.



Root and Relay Nodes

Mesh Networks provide long range connectivity by relaying packets from one mesh node to another, like a bucket brigade. The end of the bucket brigade terminates at the root – which connects to the Ethernet. Relays connect to the root or other relay nodes to form a wirelessly linked chain.

Virtual Bridge

A virtual Bridge is a piece of software used to unite two or more network segments. It behaves like a virtual switch, working transparently between real device and Virtual machine devices.

Virtual Machine

Virtual machine is an emulation of physical system which are based on system architecture and provides functionality of the real device. It provides functionality needed to execute entire mesh framework which uses the hypervisor to allow multiple virtual machines to run in an isolated environment.

Libvirt

Libvirt is an open-source API, daemon and management tool for managing platform virtualization. A primary goal of libvirt is to provide a common infrastructure to manage multiple different virtualization providers/hypervisors, such as KVM/QEMU.

Master Application Client

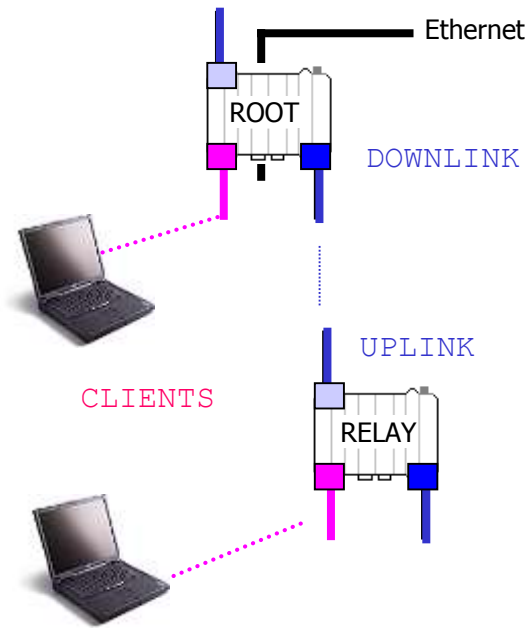
MAC is a python based entity used to retrieve the configuration of virtual machine nodes from command line interface(CLI), prepare a configuration in the appropriate format and send this information to MAS(master application server) via socket mechanism.

Master Application Server

MAS is a python based entity, responsible to handle the request from MAC, aid in the creation/deletion of virtual machine and provides configuration to virtual machine nodes.

Upstream & Downstream

Upstream implies closer to the Ethernet. The root is upstream of relay 1.



Wireless Uplinks and Downlinks.

The Ethernet link is the **uplink** (upstream link) connection for the root.

The **root** has is a wired uplink. Its 'backhaul' is the wired network.

Relays have wireless uplinks through a upstream backhaul radio.

Downlink radios act like Access Points (AP) : they send out a beacon. Uplink radios act like clients – they do not send out a beacon. A wireless radio card in the laptop can inform you of the presence of downlinks but not of uplinks.

The uplink and downlink radios form a wireless backhaul path.

Client AP radios operate in the 2.4GHz band to service 11b/g clients. 802.11a wireless devices may be serviced by the 5.8GHz downlink. Thus, both 802.11a and 802.11b/g client access is supported.

Backhaul radios operate in 802.11a 5.8GHz band to avoid interference with the 802.11b/g 2.4GHz AP radios (pink, right).

To summarize, there are 5 types of 'links' in **Structured Mesh™** products:

One or more wired uplink to provide Ethernet connectivity. This connects the ROOT node to the wired network.

One or more wireless downlink to provide wireless connectivity. Acts like an AP for the uplink. Typically 5.8GHz.

Always only one wireless uplink to connect to upstream mesh nodes. This is a 'client' to the downlink. Typically 5.8GHz.

One or more AP radio for clients. Typically 2.4GHz with support for both b and g clients.

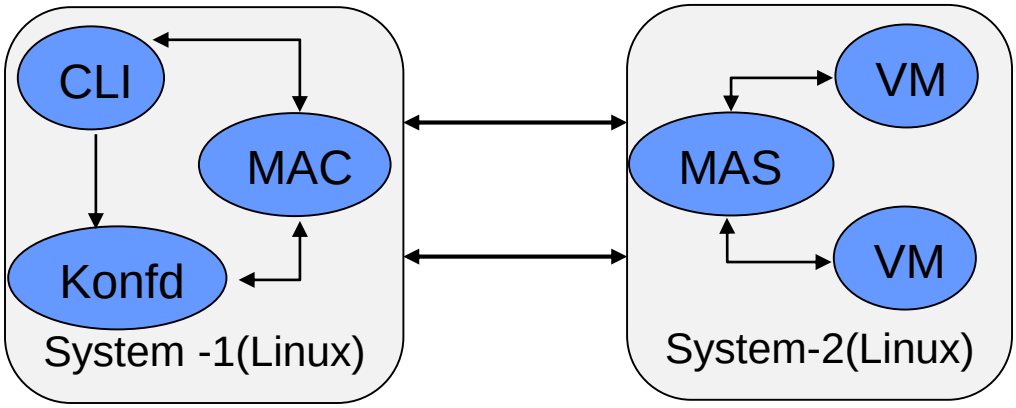
Always only one scanning/probe radio that is continuously scanning the RF environment – needed in fast mobility applications.

The Wi-Fi Simulator is a Software entity to setup a virtual framework for Mesh nodes to test and verify its features, stability and scalability with an extensible large number of nodes.

This Software Entity can be run on single system (Linux Machine) or two Different systems which are connected in Same network.

This Software entity structure contains four Major parts which are –

- CLI (klish/clish)
- MAC
- MAS
- VM node



Wi-Fi Simulator Software Structure

CLI(klish/Clish)

Clish is a command line interface utility to configure network and virtual machine with its required parameters . User has to provide configuration parameter data for every VM node through CLI. The entire information of every vm node is stored in konfd daemon.

MAC (Master Application Client)

MAC is an Python based utility responsible to get the information from konfd daemon. It prepares data of every vm with its required parameter and provides the data to master application server.

MAS (Master Application Server)

MAS is an python based utility responsible for creation and deletion of virtual machine, power-on and power-off of virtual machine. It is also responsible for providing configuration to virtual machine node and status message to master application client.

VM (Virtual Machine Node)

VM is an instance of virtual machine image which consists four major parts

Mesh dynamics framework supports both multi-radio and single radio modules.

Wmediumd is responsible for transmission and reception of packet between virtual machine node. Its create a real environment between nodes by providing signal strength with the help of distance factor.

Hw_Sim is a software simulator for 802.11 radios. It's a Linux kernel module used to simulate radios and can interface with user space application(like hostapd, wpa_supplicant) without any change in mac80211.

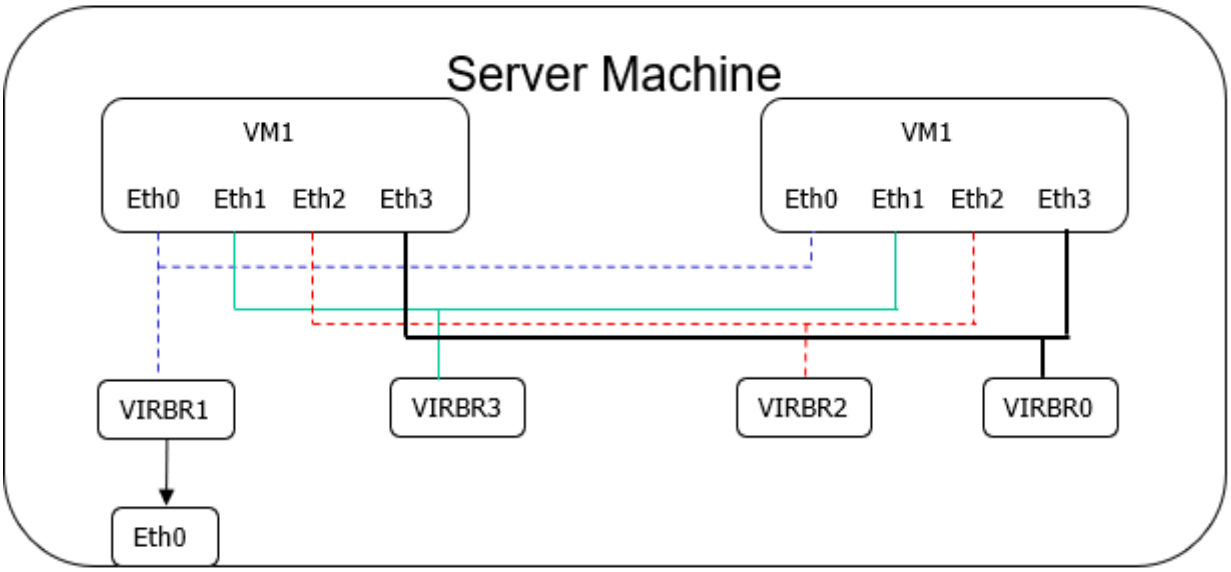
Vconfigd is a daemon utility responsible for getting configuration parameter from master application server. This utility is also responsible to start the mesh by providing configuration and provides the status message to MAS.

Virtual Machine node contains 4 interfaces which are

- **Eth0, Eth1** interfaces are representing regular eth0 and eth1 interface of Mesh nodes.
- **Eth2** interface represents the tunneling interface, used by wmediumd to send and received tunneled packets between virtual machine nodes.
- **Eth3** interface represents Management gateway between master application server and vconfigd. It is used to retrieve configuration form MAS to Vconfigd and provides status message to MAS.

Bridge Mappings

Virtual Machines uses virtual bridge to communicate with other VM's. There is a virtual bridge mapping which helps VM's to talk to each other, and also provide VM's to connect to the outside world. [For ex - to view VM node's in NMS, the virtual bridge is used to map the server physical interface to the VM node's eth0 interface.]



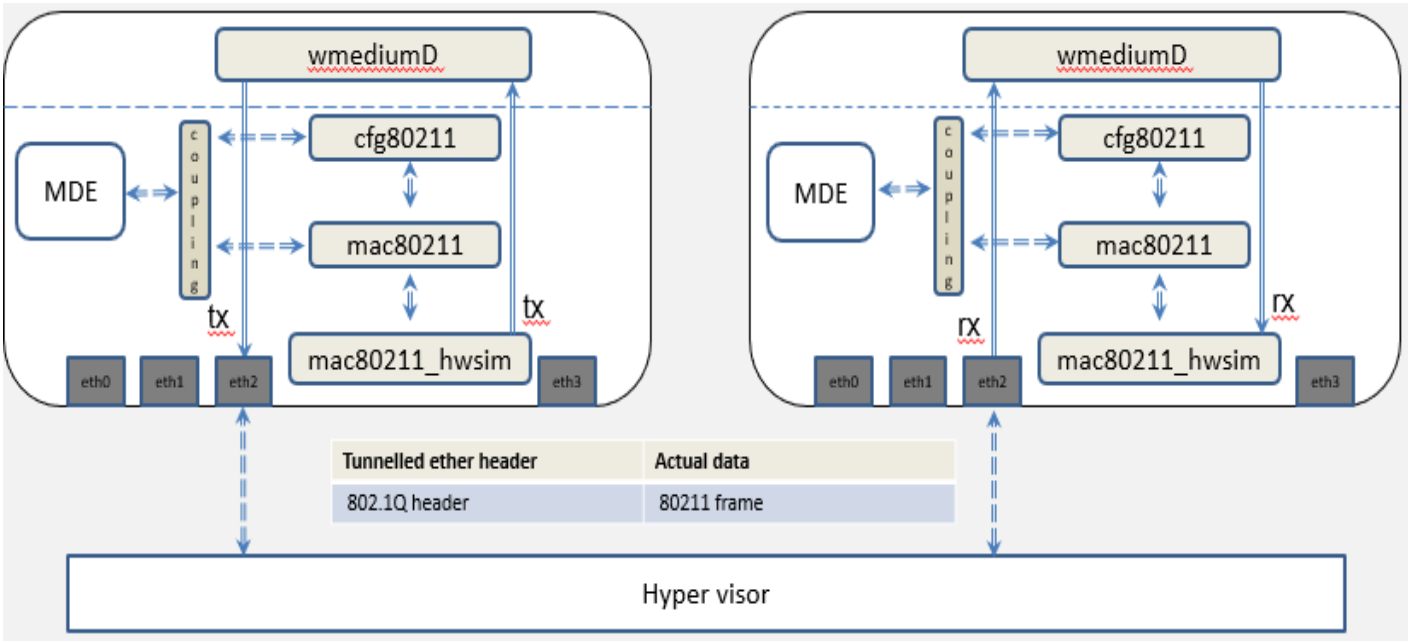
Bridge Mapping b/w VM and Server Machine

Here Eth0 and Eth1 are Mesh Uplink and Mesh Downlink Interface, Eth2 is tunneling interface for wmediumd and Eth3 is Mgmt Gateway interface for MAS and Vconfigd.

VM Interface	Virtual Bridge	Server machine Physical interface	Description
Eth0	VIRBR1	Eth0	Used to view vm node inside NMS
Eth1	VIRBR3		
Eth2	VIRBR2		Used as tunnelled interface between VM's to send and receive packets.
Eth3	VIRBR0		Used as management interface between MAS and vconfigd.

Bridge Mapping table b/w VM and Server Machine

Virtual Machine uses Mesh Framework with **hwsim** module and **wmediumd** to send and receive packets via tunneled interface eth2.



VM's Packet Flow

Tx Case

Packets transmitted from the MDE Coupling layer travels through cfg80211 and mac80211 layers and reaches the mac80211_hwsim module. From here the packet is sent to Wmediumd (user space application). Wmediumd will add simulator related information and released the packet towards tunneling interface eth2 mapped to virbr2 bridge interface.

Rx Case

In this case packet received at tunneling interface eth2 are picked up by wmediumd. it will add simulator related information and releases the packet towards mac80211_hwsim module. The hwsim module will then send the packet towards MDE via mac80211,cfg80211 and coupling layers.

Prerequisite modules for Wi-Fi Simulator

Please ensure that the following prerequisite modules are installed in Ubuntu (server) machine -
Version – [ubuntu-14 or ubuntu -16] Use the apt-get to install the following modules

sudo apt-get install [package-name]

- qemu-kvm
- virt-manager
- libvirt-bin
- virt-viewer
- kvm
- bridge-utils
- Virtinst
- liblua5.1-0-dev
- libexpat1-dev
- Expat

once the package installation is completed, restart your server machine and verify the below directories and interface are in your ubuntu server machine (with root permission).

• Run command – **"ls /var/lib/libvirt/"**

```
root@Mesh-Sim-Server:# ls /var/lib/libvirt/
boot  dnsmasq  images  network  qemu  sanlock
```

• Run command – **"brctl show"**

```
root@Mesh-Sim-Server:# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.52540015ba29       yes             virbr0-nic
```

Note: if there is no virbr0 entry in brctl show command, run the below commands and restart your machine and verify it again by running "brctl show" command

"virsh net-start default"
"virsh net-autostart default"

• Run command – **"ifconfig virbr0"** [the virbr0 interface ip is "192.168.122.1"]

```
root@Mesh-Sim-Server:# ifconfig virbr0
virbr0    Link encap:Ethernet  HWaddr 52:54:00:15:ba:29
          inet addr:192.168.122.1  Bcast:192.168.122.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

•Run command **"virsh net-list"** [the state of default should be in ative state]

```
root@Mesh-Sim-Server:# virsh net-list
Name                 State      Autostart     Persistent
----------------------------------------------------------
default               active         yes           yes
```

The result of above commands is same as shown. It confirms that the entire package has been successfully installed in your ubuntu (server) machine.

- Create three virtual bridge interfaces using below commands –

```
sudo brctl addbr virbr1
sudo brctl addbr virbr2
sudo brctl addbr virbr3
```

- Once the 3 virtual interfaces are created ensure all the 3 interfaces are up and running, to make interfaces up and running use below commands –

```
sudo ifconfig virbr1 up
sudo ifconfig virbr2 up
sudo ifconfig virbr3 up
```

- Verify the virtual bridge creation by running this command – **"brctl show"**

```
root@Mesh-Sim-Server:# brctl show
  bridge name     bridge id               STP enabled     interfaces
  virbr0          8000.52540015ba29       yes             virbr0-nic
  virbr1          8000.9c5c8e8856ce        no
  virbr2          8000.fe540000a001        no
  virbr3          8000.fe5400482606        no
```

- To Represent your VM nodes in NMS, you have to map your virbr1 interface with your physical interface of ubuntu (server)machine. To map virbr1 interface with physical interface use below command –

```
brctl addif virbr1 <physical interface device>
```

For ex- `brctl addif virbr1 eth0`

Here eth0 is ubuntu (server) machine physical interface

- To verify mapping of virtual interface with physical interface , run command – **"brctl show"**

```
root@Mesh-Sim-Server:# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.52540015ba29       yes             virbr0-nic
virbr1          8000.9c5c8e8856ce        no                eth0
virbr2          8000.fe540000a001        no
virbr3          8000.fe5400482606        no
```

Installing the Wi-Fi Simulator setup Environment

- The Wi-Fi Simulator setup Environment can be done by two ways –

1. Single system (ubuntu Machine)
2. Multiple systems (ubuntu Machine)
3. Multiple System with Multiple Server (ubuntu Machine)

Single System

The Wi-Fi Simulator setup can be run on single Machine(client and server both will run on single machine).

Please follow steps given below –

• Install and verify all the prior information for simulator setup environment.

• Create a new directory on your system(ubuntu machine) by running below command

```
mkdir wifi-sim
```

• Create two new directory "mas" and "image" on system inside wifi-sim directory.

```
cd wifi-sim
mkdir mas
mkdir image
```

• Download latest tar file of klish package inside the wif-sim directory. untar the klish package by running below command

```
tar xf klish_package.tar.xz
```

• Go inside the klish directory and run below commands to build the klish package

```
"./configure --prefix=/usr --with-lua --with-expat=builtin --disable-shared ac_cv_header_dlfcn_h=no"
  "make"
```

• Once the klish package build is done , binaries will be available inside klish package directory. To verify , run below command

```
ls bin

user@Mesh2:klish_26march ] $ ls bin
clish  clish.c  clish.o  konf  konf.c  konfd  konfd.c  konfd.o  konf.o module.am
sigexec sigexec.c  sigexec.o
```

• Copy master application server utility inside wifi-sim/mas directory.

• Copy your latest image inside wifi-sim/image directory.

• Once the above steps completed, directory contents can be seen by below command

```
ls wifi-sim

root@Mesh-Sim-Server:# ls wifi-sim/
image klish_pkg  mas
```

•Please verify bridge mappings also by running below command **"brctl show"**

```
root@Mesh-Sim-Server:# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.52540015ba29       yes             virbr0-nic
virbr1          8000.9c5c8e8856ce        no                eth0
virbr2          8000.fe540000a001        no
virbr3          8000.fe5400482606        no
```

•Now the wi-fi-simulator setup is completed and ready to run.

Multiple System

The Wi-Fi Simulator setup can be run on multiple Machine, where one machine is working as client and another one is working as server. Please follow steps given below

- Install and verify all the prior information for simulator setup environment into server machine.
- Create a new directory on your system(client machine) by running command **`mkdir wifi-sim`**
- Download latest tar file of klish package inside the wif-sim directory.
- Untar the klish package by running below command **`tar xf klish_package.tar.xz`**
- Go to inside klish directory and run below commands to build the klish package –

"./configure --prefix=/usr --with-lua --with-expat=builtin --disable-shared ac_cv_header_dlfcn_h=no"
"make"

- Once the klish package build is done , binaries will be available inside klish package directory.
- To verify , run command **`ls bin`**

```
[user@Mesh2:klish_26march ] $ ls bin
clish  clish.c  clish.o  konf  konf.c  konfd  konfd.c  konfd.o  konf.o  module.am  sigexec
 sigexec.c  sigexec.o
```

- Create a new directory on your system(server machine) by running command **`mkdir wifi-sim`**
- Create two new directory "mas" and "image" on your system inside wifi-sim directory(server machine).

```
cd wifi-sim
mkdir mas
mkdir image
```

- Copy master application server utility inside wifi-sim/mas directory.
- Copy your latest image inside wifi-sim/image directory.
- Once the above steps completed, your directory contents can be seen by command **ls wifi-sim**

At client machine

```
root@Mesh-Sim-Server:# ls wifi-sim/
klish_pkg
```

At server machine

```
root@Mesh-Sim-Server:# ls wifi-sim/
image  mas
```

- Please verify bridge mappings(at server machine) by running command **`brctl show`**

```
root@Mesh-Sim-Server:# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.52540015ba29       yes             virbr0-nic
virbr1          8000.9c5c8e8856ce        no                eth0
virbr2          8000.fe540000a001        no
virbr3          8000.fe5400482606        no
```

- Now Wifi-simulator setup is completed and ready to run.

Multiple System with Multiple Server

The Wi-Fi Simulator setup can be run with multiple servers - where one machine is working as client and remaining machines working as server. The steps for installing wifi-simulator on Multiple server machine is same as a single server machine which was described in Multiple System section. Each server machine structure is like this –

At server machine

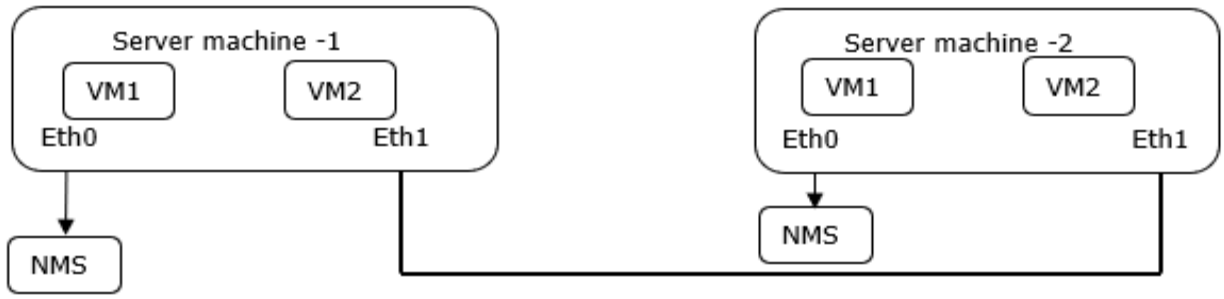
```
root@Mesh-Sim-Server:# ls wifi-sim/
image  mas
```

Multiple sever machines node talks each other via virtual bridge which are mapped to physical interface of server machines.

• Bridge mappings(at server machine) can be verified by running below command

"brctl show"

```
root@Mesh-Sim-Server:# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.52540015ba29       yes             virbr0-nic
virbr1          8000.9c5c8e8856ce        no                eth0
virbr2          8000.fe540000a001        no                eth1
virbr3          8000.fe5400482606        no
```



Multiple Server machine Connection

Here Eth1 [physical interface of server machine -1] is connected to Eth2[Physical interface of server machine -2]. These interfaces are internally mapped to virbr2 used to send and receive tunneled packet.

Eth0 of Both server machines is used to see nodes on NMS.

VM Interface	Virtual Bridge	Server machine Physical interface	Description
Eth0	VIRBR1	Eth0	Used to see vm node inside NMS
Eth2	VIRBR2	Eth1	Used to send tunnelled packets from one server machine to another server machine

Multiple Server machine Connection

Wi-fi-Simulator uses Mesh framework inside Virtual machine nodes to verify its feature and scalability.

For this Wi-fi simulator uses Mesh configuration file given below –

- MD4455-AAIA
- MD4250-AAxx
- .. Per customer requirements

Each node uses approx 128 Mb of RAM.

Wi-Fi simulator Framework Creates virtual machine node at server machine with the help of libvirt API's and virtual bridge. a hypervisor mechanism uses native execution to share and manage hardware, allowing multiple virtual machine which are isolated with each other, yet exist on the same physical machine(server machine).

In real environment, Mesh Framework uses root and relay topology to frame a network between nodes where each nodes are placed at some distance. As a distance factor is available in real environment , the signal strength varies from one node to another node.

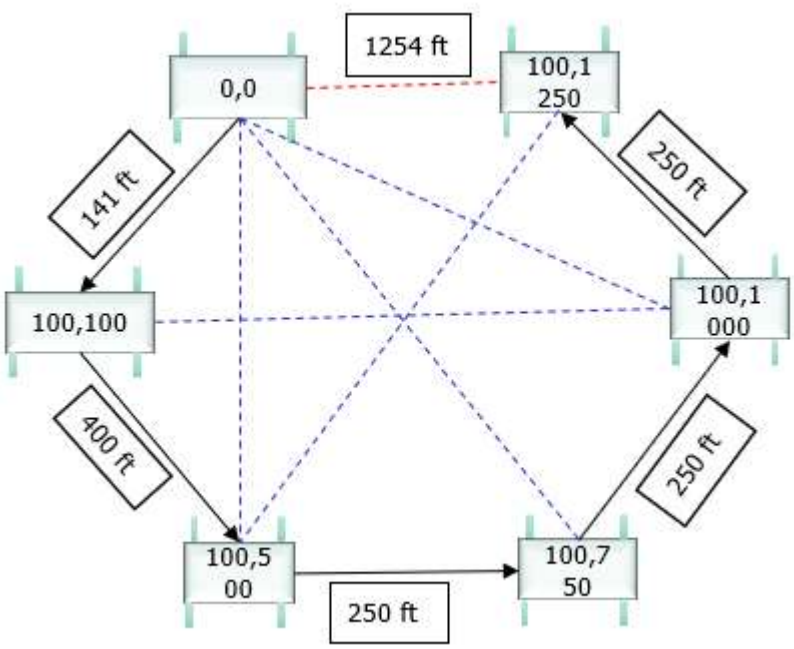
In Virtual environment, the entire virtual nodes created at server machine, where each node is isolated from its nearby nodes. To verify Mesh framework with its feature and scalability, a signal to distance mechanism introduced in wifi-simulator framework .

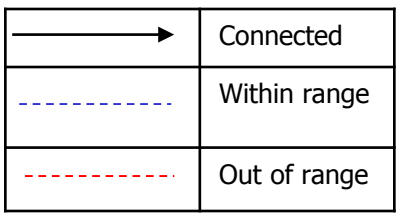
User will provides the coordinates of each node in the format of x,y,z (assume distance will be in feet). Once the configuration had reached at each virtual machine, wmediumd will calculate the actual distance of that vm with its nearby nodes and provides signal strength based on signal to distance mapping table.

Distance (feet)	Signal strength at Wmediumd (db)	Signal strength value at VM	Type
0 – 250	-40	56	EXCELLENT
250 – 500	-50	46	GREAT
500 – 750	-60	36	GOOD
750 – 1000	-70	26	FAIR
1000 - 1250	-90	6	POOR

Signal to Distance Mapping

VM Nodes in 3-D Plane





Wi-Fi Simulator provides CLI Interface to Configure Virtual machine with its required Parameters. User has to Provide Appropriate configuration for each vm nodes. CLI Uses nested structure format to stored configuration in konfd(daemon) utility. to configure a node we have to follow Basic steps given below –

- Run clish command to get CLI terminal
- Run configure command to get config terminal
- Enter Configuration for Network
- Enter configuration for Virtual Machine(vm)
- Run enable/disable command to on/off of a single vm node or network.
- Run show command to verify the status of a single vm or network
- Run do show running-config command to see the configuration
- Run no command to deletion of a single vm or network
- Run exit command to close config and clish terminal.

CLI with Configure Terminal commands

Run konfd daemon by running below command [at client machine (ubuntu) **./bin/konfd**

```
[user@Mesh2:klish_pkg ] $ ./bin/konfd
[user@Mesh2:klish_pkg ] $ ps -aef | grep konfd
user      9071     1  0 Feb03 ?        00:00:00 ./bin/konfd
```

Export the path of xml-simulator by running below command [at client machine (ubuntu)]

"export CLISH_PATH=xml-simulator/"

Run clish command to get CLI Terminal [at client machine (ubuntu)]

"./bin/clish"

```
[user@Mesh2:klish_pkg ] $ ./bin/clish
#
```

Run configure command to get config terminal **"configure terminal"**

```
# configure terminal
(config)#
```

To see supported command on config terminal , use TAB command, result shown below-

```
# configure terminal
(config)#
!        do       exit     no       nw_name  show
(config)#
```

do - show to stored configuration in konfd [for ex – do show running-config]

exit - to close config terminal

no - delete network [for ex – no network1

nw_name - create network [for ex – nw_name network1]

show - show network status [show network1]

Note: Here network1 is network name given by user to create network

Network Configuration commands

Creation , deletion and status of a network is done by network configuration commands.

Currently you are in config terminal –

```
# configure terminal
(config)#
!        do       exit     no       nw_name show
(config)#
```

Run below command to create network **"nw_name <network_name>"**

```
(config)# nw_name Meshdynamics
(config-Meshdynamics)#
```

Run below command to delete network **"no <network_name>"**

```
(config)# no Meshdynamics
```

Run Command to show network status

"show <network_name>" - it will show the status of network with its vm node status

```
(config)# show Meshdynamics
```

Run Command to see configuration stored in konfd

"do show running-config"

Once you entered network name , you will get new config terminal associated with your network.
Press TAB key to see the supported by network config terminal

```
!              do             exit           network_status no              show           vm_name
```

do -to show current configuration stored in konfd utility
exit - to close network config terminal
network_status - to Enable /Disable of your network [for ex- network_status enable/disable]
no-to delete virtual machine node with its configuration from konfd utility
[for ex- no node1]
show-to show the status of your virtual machine node [for ex- show node1]
vm_name- to create virtual machine node [for ex- vm_name node1]

Run Command to create/off/on the whole network

"network_status enable/disable" -> by default network_status is disable

```
(config-Meshdynamics)# network_staus enable  - to create/ on network

(config-Meshdynamics)# network_staus disable  - to disable/off network
```

VM Configuration commands

Creation , deletion and status of a vm is done by VM configuration commands.

Currently you are in network config terminal

```
!               do              exit            network status no               show            vm name
```

Run command to create vm node

"vm_node <vm name>"

```
(config-boobalan)# vm_name node1
```
-> node1 is the name of vm

Once vm name is created , it entered into new config terminal associated with vm node. Press TAB command to see supported command by vm config terminal

```
(config-node1)#
!               exit            mgmt_ip         mgmt_mac        mip_ip          model_number    server_ip       server_port
vm_coordinates  vm_imag_path    vm_mobile_distance vm_mobile_speed vm_role      vm_status
```

exit -to close vm config terminal

mgmt-ipmanagement ip for vm node [it should be in range of 192.168.122.x]
it should be unique

mgmt-macManagement mac for vm node [it should be in range of 52:54:00:xx:xx:xx]
[there will be difference of 20 mac address range between two vm nodes]

mip-ipmip ip of vm node(used by Mesh)

Model_numbermodel number of configuration file used by mesh

server_ip ip address of server machine where vm node will going to be create
[for single system – 127.0.0.1]

server_portport number of server machine [by default 4955]

vm_coordinatesdistance between vm nodes [xxxx:yyyy:zzzz] it should be in four digits

vm_imag_path path of your vm image

vm_roleRole of the virtual machine
it can be ROOT/RELAY/MOBILE

vm_statusto create/off/on of virtual machine node
by default vm_status is disable
it has two parameters [enable/disable]

If vm_role is **MOBILE** , then you to also configure below two parameters –

vm_mobile_distance total distance covered by vm node
vm_mobile_speedvm node spped

Note – to creation/on/off of vm node depends on both network_status and vm_status command. If your network is disable, then vm configuration will be stored in konfd utility.

If network_status command is enable , then vm configuration will be stored in konfd and also vm will be created on server machine.

By running network_status enable/disable command , you can control entire vm nodes inside that network.

Wi-Fi Simulator setup can be run on Single system or Multiple Systems. Single system always uses loopback interface IP as a server IP. Multiple system uses IP address of server machine for creation of virtual machine.

To Run Wi-Fi simulator, Follow the below given steps-

•Please verify that all the perquisite is installed in the system.

•According to Wi-Fi Simulator setup, there are 3 directories available in system.

Client machine :

```
user@Mesh-Sim:~/wifi-sim$ ls
klish_pkg
```

Server Machine:

```
user@Mesh-Sim-Server:~/wifi-sim$ ls
image  mas
```

Note- If you are using only single sytem to run wifi-simulator setup, the directories structure will be shown below

```
user@Benisnon-sytem:~/wifi-sim$ ls
image  klish_pkg  mas
```

Here Directory contains –

klish_pkg contains clish cli, konfd daemon utility , Master app client utility and xml-simulator directory

mas contains Master app sever utility.

image contains virtual machine image file.

•Please verify (in case of Multiple system), client and server Machines are connected and pingable each other.

•Go to inside mas directory at server machine and run master application server by running below command [at new Terminal]

"sudo python master_app_server.py"

NOTE: To run MAS(master application server), you have to be required root permissions.

```
user@Mesh-Sim-Server:~/wifi-sim/mas$ python master_app_server.py
****************************************
        Master Application Server
****************************************
virsh list --all
 Id    Name                           State
----------------------------------------------------
```

•Go to inside klish_pkg/mac directory at client machine and run master application client by running below command [at new Terminal]

python master_app_client.py"

```
[user@Mesh:mac ] $ python master_app_client.py
****************************************
        Master Application Client
****************************************
{}

<MAC>: Waiting To Receive Action from klish
```

- Go inside klish_pkg directory at client machine and run clish(CLI) by running below command

  "./bin/konfd"

- Go inside klish_pkg directory at client machine and run clish(CLI) by running below command

  "export CLISH_PATH=xml-simulator/"

  "./bin/clish"

```
[user@Mesh2:klish_pkg ] $ ./bin/konfd
[user@Mesh2:klish_pkg ] $ ./bin/clish
#
```

- Clish Utility provides you command line interface, where configurations done for virtual machines. Steps are given below

  •Go to configure terminal by running below command
  "configure terminal"

  •Create a new network by running below command
  "nw_name <nw_name>"

  •Create a new virtual machine by running below command
  "vm_name <vm_name>"

  •Fill all the configuration parameter one by one by running below commands

```
mgmt_ip <192.168.122.x>
mgmt_mac <52:54:00:xx:xx:xx>
mip_ip <mip_ip of mesh node>
model_number <Mesh node configuration file name>
server_ip <server machine IP address>
server_port <4955>
vm_coordinates <xxxx:xxxx:xxxx>
vm_imag_path <path of vm image>
vm_role <ROOT/RELAY/MOBILE>
vm_status <enable/disable>
```

  •Closing vm configuration terminal by running below command
  "exit"

Configuration example with clish

```
[user@Mesh2:klish_pkg ] $ ./bin/clish
# configure terminal
(config)# nw_name Meshdynamics
(config-Meshdynamics)# vm_name node1
(config-node1)# mgmt_ip 192.168.122.100
(config-node1)# mgmt_mac 52:54:00:00:A0:00
(config-node1)# mip_ip 192.168.0.100
(config-node1)# model_number MD4455-AAIA
(config-node1)# server_ip 192.168.128.25
(config-node1)# server_port 4955
(config-node1)# vm_coordinates 0000:0000:0000
(config-node1)# vm_imag_path /home/user/simulator/openwrt-x86-generic-combined-ext4.img
(config-node1)# vm_role root
(config-node1)# vm_status enable
 Sending the cmd to MAC : " Meshdynamics node1 vm_status enable
                "
 Klish response : " "Given vm_status configuration saved." "
(config-node1)#exit
(config-Meshdynamics)#
```

- Once the configuration is completed, below command is used to create and start the network with its VMs.

"network_status enable"

```
(config-Meshdynamics)# network_status enable
['node1']
 Sending the cmd to MAC : " Meshdynamics network_status enable
                "
 Klish response : " {"Meshdynamics": {"node1": "ENABLED"}} "
```

- To verify created node, Run below command on server machine

"virsh list -all"

```
user@Mesh-Sim-Server:~/wifi-sim/mas$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 2190  Meshdynamics-node1                 running
```

- VM node console can be obtained by running below command

"virsh console <vm_node name shown in list> "

```
user@Mesh-Sim-Server:~/wifi-sim/mas$ virsh console Meshdynamics-node1
Connected to domain Meshdynamics-node1
Escape character is ^]



BusyBox v1.24.1 () built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 DESIGNATED DRIVER (Bleeding Edge, unknown)
 -----------------------------------------------------
  * 2 oz. Orange Juice         Combine all juices in a
  * 2 oz. Pineapple Juice      tall glass filled with
  * 2 oz. Grapefruit Juice     ice, stir well.
  * 2 oz. Cranberry Juice
 -----------------------------------------------------
root@OpenWrt:~#
```

- The complete Network with its VMs can be poweroff by running below command on clish CLI at client Machine

"network_status disable"

```
(config-Meshdynamics)# network_status disable
 Sending the cmd to MAC : " Meshdynamics network_status disable
                "
 Klish response : " {"Meshdynamics": {"node1": "DISABLED"}} "
```

- Single VM can be enable(on)/disable(off) by running below command

"vm_status enable"
"vm_status disable "

Note - single vm can be on/off from vm config terminal

```
(config-Meshdynamics)# vm_name node1
( (config-node1)# vm_status disable
 Sending the cmd to MAC : " Meshdynamics node1 vm_status disable
                  "
 Klish response : " "Domain Meshdynamics-node1 destroyed\n\n" "
(config-node1)# vm_status enable
 Sending the cmd to MAC : " Meshdynamics node1 vm_status enable
                  "
 Klish response : " "Domain Meshdynamics-node1 started\n\n" "
(config-node1)#
```

- Single VM can be removed form the network by running below command

"no <vm name>"

```
(config-Meshdynamics)# no node1
 Sending the cmd to MAC : " Meshdynamics node1 delete
                  "
 Klish response : " "given vm_name configuration and machine successfully deleted." "
```

- Single VM status can be show by running below command

"show <vm name>"

```
(config-Meshdynamics)# show node1
 Sending the cmd to MAC : " Meshdynamics node1 show
          "
 Klish response : " "enable" "
```

Note - single vm deletion and status command will be run from network config terminal

- Network with its VMs can be removed by running below command

"no <nw_name>"

```
(config)# no Meshdynamics
 Sending the cmd to MAC : " Meshdynamics delete
                  "
 Klish response : " "Given Network name configuration and vm successfully deleted" "
```

- Network with its VMs status can be shown by running below command

"show <nw_name>"

```
(config)# show Meshdynamics
 Sending the cmd to MAC : " Meshdynamics show
          "
 Klish response : " {"node1": "enable"} "
```

Note - network deletion and status command will be run from clish configure terminal.

Wi-Fi Simulator Framework used to create virtual machine which consists Mesh Framework. Each Virtual machine works as a Mesh node or as an Access point.

access point (AP), is a networking hardware device that allows a Wi-Fi device to connect to a wired network. An AP connects directly to a wired local area network, typically Ethernet, and the AP then provides wireless connections using wireless LAN technology to its stations.

To test and verify station (created as Virtual machine) connectivity with AP, Following procedure is used –

•Get the VM station image and copy it to inside wifi-sim/image directory at server machine.

•Get a terminal on server machine for creation of virtual machine station.

•Run Below Command to create virtual machine station node.

```
sudo virt-install --connect qemu:///system --name Meshdynamics-sta --ram 256 --vcpus=1 --disk
path=/home/user/wifi-sim/image/openwrt-x86-generic-combined-ext4_sta.img,size=12,bus=ide --os-type linux
--vnc --noautoconsole --network bridge=virbr1,model=e1000 --network bridge=virbr3,model=e1000 --network
bridge=virbr2,model=e1000 --network bridge=virbr0,model=e1000 --import
```

Here
Station name - **Beniosn-sta**
Station image name - **openwrt-x86-generic-combined-ext4_sta.img**

•Verify the virtual machine station by running below command

"virsh list --all"

```
user@Mesh-Sim-Server:~$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 2229  Meshdynamics-sta                    running
```

•To get the console of virtual machine, run below command and Press Enter.

"virsh console <vm-station name >"

```
user@Mesh-Sim-Server:~$ virsh console Meshdynamics-sta
Connected to domain Meshdynamics-sta
Escape character is ^]

BusyBox v1.24.1 () built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 DESIGNATED DRIVER (Bleeding Edge, unknown)
 -----------------------------------------------------
  * 2 oz. Orange Juice         Combine all juices in a
  * 2 oz. Pineapple Juice      tall glass filled with
  * 2 oz. Grapefruit Juice     ice, stir well.
  * 2 oz. Cranberry Juice
 -----------------------------------------------------
root@OpenWrt:/#
```

- To connect vm station with vm node, SSID parameter of wlan2 (wireless) interface of vm node is required. To get the ssid of wlan2 interface of vm node, open the below file inside vm node.

at vm node

"vi /etc/hostapd_wlan2.conf "

```
root@OpenWrt:~# vi etc/hostapd_wlan2.conf
# Hostapd config file


driver=nl80211
ssid=StructuredMesh_wlan2
rts_threshold=2347
fragm_threshold=2346
preamble=1
beacon_int=100
country_code=00
interface=wlan2
channel=1
hw_mode=g
```

Here wlan2 interface **ssid=StructuredMesh_wlan2**

- open wpa_supplicant config file at vm station node and write the ssid of vm node by running below steps.

at vm station node

vi etc/config/wpa_supplicant.conf

```
root@OpenWrt:/# vi etc/config/wpa_supplicant.conf

ctrl_interface=/var/run/wpa_supplicant

network={
    ssid="StructuredMesh_wlan2"
    key_mgmt=NONE
}
```

- Run below sequence of commands on vm station

"ifconfig eth2 up"

```
root@OpenWrt:/# ifconfig eth2 up
[ 3217.607565] IPv6: ADDRCONF(NETDEV_UP): eth2: link is not ready
[ 3217.609405] 8021q: adding VLAN 0 to HW filter on device eth2
```

"ifconfig wlan1 up"

```
root@OpenWrt:/# ifconfig wlan1 up
[ 3233.991047] IPv6: ADDRCONF(NETDEV_UP): wlan1: link is not ready
```

"wmediumd -c etc/config/wmediumd_4.conf &"

"wpa_supplicant -i wlan1 -c etc/config/wpa_supplicant.conf & "

Here , **wlan1** is used as station interface , which will be going to connect with vm node.

```
root@OpenWrt:/# wpa_supplicant -i wlan1 -c /etc/config/wpa_supplicant.conf &
root@OpenWrt:/# Successfully initialized wpa_supplicant
wlan1: SME: Trying to authenticate with 52:54:00:00:a0:37 (SSID='StructuredMesh_wlan2' freq=2412 MHz)
[ 3291.790627] wlan1: authenticate with 52:54:00:00:a0:37
[ 3291.791358] wlan1: send auth to 52:54:00:00:a0:37 (try 1/3)
[ 3291.800091] wlan1: authenticated
wlan1: Trying to associate with 52:54:00:00:a0:37 (SSID='StructuredMesh_wlan2' freq=2412 MHz)
[ 3291.810031] wlan1: associate with 52:54:00:00:a0:37 (try 1/3)
[ 3291.820082] wlan1: RX AssocResp from 52:54:00:00:a0:37 (capab=0x421 status=0 aid=1)
[ 3291.821378] wlan1: associated
[ 3291.821887] IPv6: ADDRCONF(NETDEV_CHANGE): wlan1: link becomes ready
wlan1: Associated with 52:54:00:00:a0:37
wlan1: CTRL-EVENT-CONNECTED - Connection to 52:54:00:00:a0:37 completed [id=0 id_str=]
wlan1: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
```

•To verify the connection between vm station and vm node, provide ip to the wlan1 interface [in same range which have given on vm node mip0 interface] . Run below commands.

"ifconfig wlan0 <ip_address>"
"ping <ip_address_of_vm node mip0 interface >"

```
root@OpenWrt:/# ifconfig wlan1 192.168.0.50
root@OpenWrt:/# ping 192.168.0.231
PING 192.168.0.231 (192.168.0.231): 56 data bytes
64 bytes from 192.168.0.231: seq=0 ttl=64 time=19.565 ms
64 bytes from 192.168.0.231: seq=1 ttl=64 time=9.414 ms
```

•To destroy or delete vm station, use below command on server(ubuntu) machine.

"virsh destroy <vm station name>"
"virsh undefine <vm station name>"

```
user@Mesh-Sim-Server:~$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 2229  Meshdynamics-sta               running
 2230  Meshdynamics-node1             running

user@Mesh-Sim-Server:~$ virsh destroy Meshdynamics-sta
Domain Meshdynamics-sta destroyed

user@Mesh-Sim-Server:~$ virsh undefine Meshdynamics-sta
Domain Meshdynamics-sta has been undefined

user@Mesh-Sim-Server:~$ virsh list --all
 Id    Name                           State
----------------------------------------------------
 2230  Meshdynamics-node1             running
```

Wi-Fi Simulator framework can be used to test mobility feature between vm nodes. Where one node is working as moving nodes and others are working as stationary nodes.

Stationary node contains vm_role parameter as "ROOT" or "RELAY" , where Moving node has vm_role parameter as "MOBILE" nodes .

Note: First create stationary nodes and provide them preferred parent feature , then you have to create mobile node.

For Mobile node – [during Clish configuration]

vm_role -> MOBILE

vm_coordinates-> 0000:0000:0000

vm_mobile_distance -> total distance has to be covered by mobile node [in feet]

vm_mobile_speed -> spped to the mobile (moving) node

Note: - vm coordinates of mobile node will be 0000:0000:0000 , because to test mobility case you have to kept ROOT vm node coordinates as 0000:0000:0000.

For Stationary node – [during clish configuration]

vm_role -> ROOT/RELAY

Once all the stationary vm node is up , provide preferred parent feature to each vm node [which are relay].

The total time taken by mobile node to cover distance is –

time = (distance/speed)

The coordinates will be provided to the stationary vm nodes based on distance to signal mapping table.

For Ex -

Here , 5 –stationary node , 1- moving node

vm_mobile speed = 36 km/pr

vm_mobile_distance = 1600 mtr

Time = ((1600*60*60)/(36*1000)) seconds

Time = 160 seconds.

The mobile node will take 160 seconds to cover total distance.

Note : Mobility test case is given in Sample test case to test and verify.

Spawn 5 Wi-Fi-Simulator Nodes Where one as Root and Others are as Relay

Description

Create 5 Wi-Fi-simulator nodes with One as Root and Other 4 nodes are as relay.

Execution Steps

•Run klish, MAC and MAS utility to configure and creation of virtual machine.
•On clish Prompt , enter vm configuration command and fills appropriate data.
•Once configuration done , give network_status command to create and start of virtual machine.
•Use "virsh list --all" command to show the list of running virtual machine.
•Use "Virsh console <vm name>" command to get the console of virtual machine.
•Verify mode under cat /proc/net/meshap/mesh/adhoc_mode.
•Verify the entry of child nodes under cat /proc/net/meshap/mesh/table of parent node
•verify the data path between the nodes.
•Verify that NMS has same topology.

Configurations (Given Through CLI)

```
vm_name NODE101
mgmt_ip 192.168.122.200
mgmt_mac 52:54:00:00:00:10
mip_ip 192.168.0.200
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0000:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role root
vm_status enable
exit
vm_name NODE102
mgmt_ip 192.168.122.201
mgmt_mac 52:54:00:00:00:20
mip_ip 192.168.0.201
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0400:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE103
mgmt_ip 192.168.122.202
mgmt_mac 52:54:00:00:00:30
mip_ip 192.168.0.202
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0800:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE104
mgmt_ip 192.168.122.203
mgmt_mac 52:54:00:00:00:40
mip_ip 192.168.0.203
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1200:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
Exit
```

```
vm_name NODE105
mgmt_ip 192.168.122.204
mgmt_mac 52:54:00:00:00:50
mip_ip 192.168.0.204
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1600:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
```

Expected Results
one node becomes as FFR and others are as FFN.
Data path working between nodes.
Nodes is shown in NMS and also configurable through NMS.

Result:
`cat /proc/net/meshap/mesh/adhoc Output`

```
root@OpenWrt:~# cat /proc/net/meshap/mesh/adhoc
NODE OPERATING AS FFN STEP SCAN INDEX 0

-------------------------------------------------------------------------
PARENT ADDR       |CHN|SIG|RATE|TRATE|FLAGS       |FNC|R-VAL|
-------------------------------------------------------------------------
52:54:00:00:00:35|044|046|0000|00000|- M - - - -|FFN|00000|
52:54:00:00:00:15|048|046|0000|00100|- - - - - -|FFR|00000|
52:54:00:00:00:45|153|026|0000|00000|- M - - - -|FFN|00000|
52:54:00:00:00:55|048|006|0000|00000|- M - - - -|FFN|00000|
root@OpenWrt:~# cat /proc/net/meshap/mesh/table
-------------------------------------------------------------------------
STATION ADDR : RELAY AP ADDR -> IMMEDIATE AP ADDR
-------------------------------------------------------------------------
```

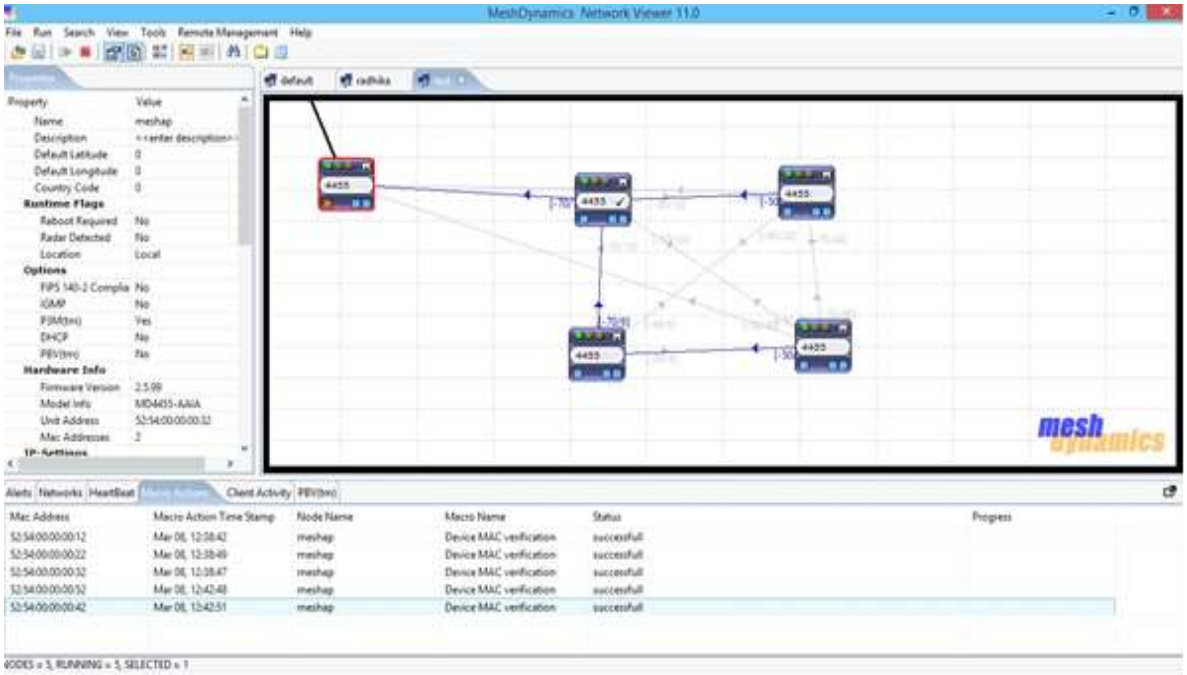
`cat /proc/net/meshap/mesh/table Output =`

```
root@OpenWrt:~# cat /proc/net/meshap/mesh/table
-------------------------------------------------------------------------
STATION ADDR : RELAY AP ADDR -> IMMEDIATE AP ADDR
-------------------------------------------------------------------------
52:54:00:00:00:32 : 52:54:00:00:00:32 -> 52:54:00:00:00:32
        52:54:00:00:00:22 : 52:54:00:00:00:32 -> 52:54:00:00:00:32
        52:54:00:00:00:52 : 52:54:00:00:00:32 -> 52:54:00:00:00:32
                52:54:00:00:00:42 : 52:54:00:00:00:32 -> 52:54:00:00:00:52
```

\\

```
root@OpenWrt:~# fping -l -s -f fping
192.168.0.204 : [0], 84 bytes, 0.15 ms (0.15 avg, 0% loss)
192.168.0.203 : [0], 84 bytes, 18.9 ms (18.9 avg, 0% loss)
192.168.0.202 : [0], 84 bytes, 29.9 ms (29.9 avg, 0% loss)
192.168.0.200 : [0], 84 bytes, 59.6 ms (59.6 avg, 0% loss)
192.168.0.201 : [0], 84 bytes, 59.2 ms (59.2 avg, 0% loss)
```

NMS Output



Spawn 5 Wi-Fi-Simulator Nodes Where Two nodes as Root and Others are as Relay

Description

Create 5 Wi-Fi-simulator nodes with two nodes as Root and Other 3 nodes are as relay.

Execution Steps

- Run klish, MAC and MAS utility to configure and creation of virtual machine.
- On clish Prompt , enter vm configuration command and fills appropriate data.
- Once configuration done , give network_status command to create and start of virtual machine.
- Use "virsh list --all" command to show the list of running virtual machine.
- Use "virsh console <vm name>" command to get the console of virtual machine.
- Verify mode under cat /proc/net/meshap/mesh/adhoc_mode.
- Verify the entry of child nodes under cat /proc/net/meshap/mesh/table of parent node
- verify the data path between the nodes.
- Verify that NMS has same topology.

Configurations (Given Through CLI)

```
vm_name NODE101
mgmt_ip 192.168.122.200
mgmt_mac 52:54:00:00:00:10
mip_ip 192.168.0.200
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0000:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role root
vm_status enable
exit
```

```
vm_name NODE102
mgmt_ip 192.168.122.201
mgmt_mac 52:54:00:00:00:20
mip_ip 192.168.0.201
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0400:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role root
vm_status enable
exit
vm_name NODE103
mgmt_ip 192.168.122.202
mgmt_mac 52:54:00:00:00:30
mip_ip 192.168.0.202
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0800:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE104
mgmt_ip 192.168.122.203
mgmt_mac 52:54:00:00:00:40
mip_ip 192.168.0.203
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1200:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
Exit
vm_name NODE105
mgmt_ip 192.168.122.204
mgmt_mac 52:54:00:00:00:50
mip_ip 192.168.0.204
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1600:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
```

Expected Results

two nodes become as FFR and others are as FFN.
Data path working between nodes.
Nodes is shown in NMS and also configurable through NMS.

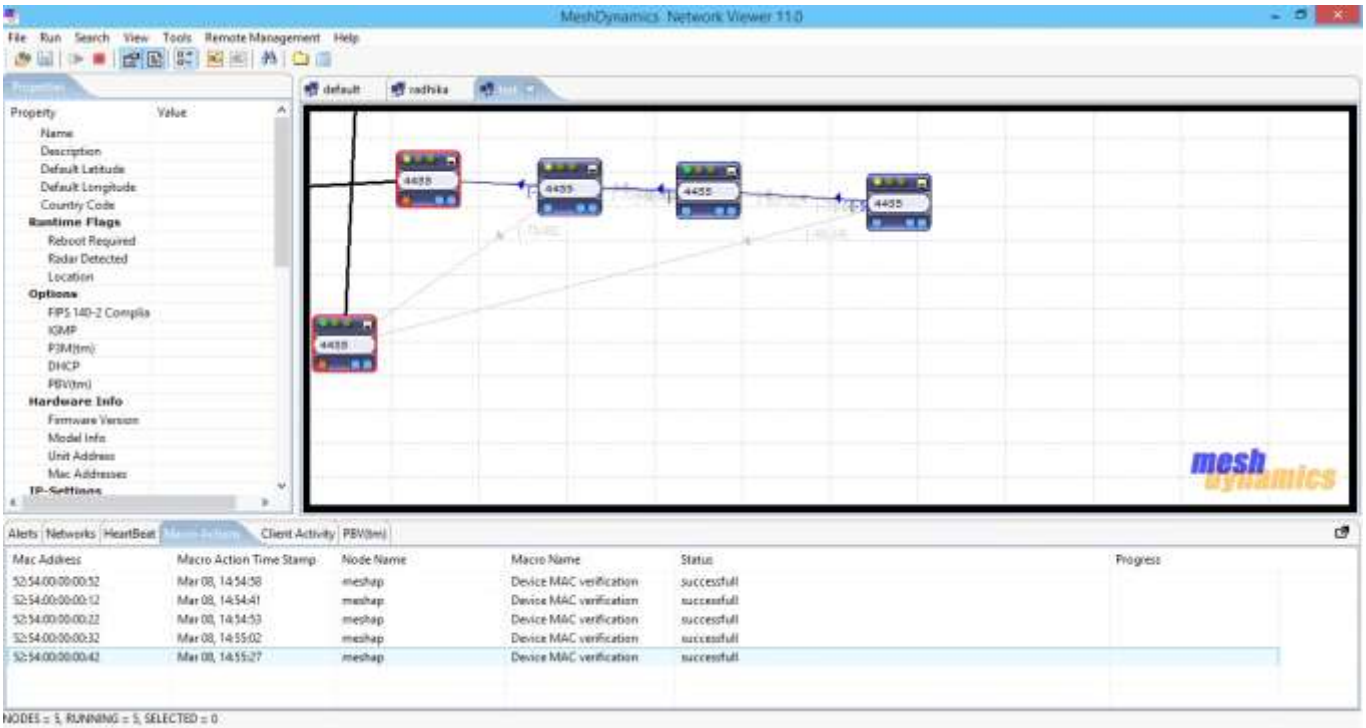
Result

`cat proc/net/mishap/mesh/adhoc`

```
root@OpenWrt:~# cat proc/net/meshap/mesh/adhoc
NODE OPERATING AS FFN STEP SCAN INDEX 0

-------------------------------------------------------------
PARENT ADDR       |CHN|SIG|RATE|TRATE|FLAGS      |FNC|R-VAL|
-------------------------------------------------------------
52:54:00:00:00:25|048|046|0000|00100|-  -  -  -  -  -|FFR|00000|
52:54:00:00:00:45|153|046|0000|00000|-  M  C  -  -  -|FFN|00000|
52:54:00:00:00:15|040|026|0000|00100|-  -  -  -  -  -|FFR|00000|
52:54:00:00:00:55|040|026|0000|00000|-  M  C  -  -  -|FFN|00000|
```

NMS output



Data path Logs (fping Command)

```
root@OpenWrt:~# fping -l -s -f fping
192.168.0.204 : [0], 84 bytes, 0.15 ms (0.15 avg, 0% loss)
192.168.0.203 : [0], 84 bytes, 18.9 ms (18.9 avg, 0% loss)
192.168.0.202 : [0], 84 bytes, 29.9 ms (29.9 avg, 0% loss)
192.168.0.200 : [0], 84 bytes, 59.6 ms (59.6 avg, 0% loss)
192.168.0.201 : [0], 84 bytes, 59.2 ms (59.2 avg, 0% loss)
```

Spawn 5 Wi-Fi-Simulator Nodes Where all nodes are relays

Description

Create 5 Wi-Fi-simulator nodes with one node as Root and Other 4 nodes are as relay. Later on Make eth0 down of root interface and nodes becomes relays.

Execution Steps

•Run klish, MAC and MAS utility to configure and creation of virtual machine.
•On clish Prompt , enter vm configuration command and fills appropriate data.
•Once configuration done , give network_status command to create and start of virtual machine.
•Use "virsh list --all" command to show the list of running virtual machine.
•Use "virsh console <vm name>" command to get the console of virtual machine.
•Verify mode under cat /proc/net/meshap/mesh/adhoc_mode.
•Verify the entry of child nodes under cat /proc/net/meshap/mesh/table of parent node
•Make eth0 down of root nodes and verify their adhoc modes.
•Verify the nodes in NMS.

Configurations (Given Through CLI)

```
vm_name NODE101
mgmt_ip 192.168.122.200
mgmt_mac 52:54:00:00:00:10
mip_ip 192.168.0.200
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0000:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role root
vm_status enable
exit
vm_name NODE102
mgmt_ip 192.168.122.201
mgmt_mac 52:54:00:00:00:20
mip_ip 192.168.0.201
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0400:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE103
mgmt_ip 192.168.122.202
mgmt_mac 52:54:00:00:00:30
mip_ip 192.168.0.202
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0800:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
```

```
vm_name NODE104
mgmt_ip 192.168.122.203
mgmt_mac 52:54:00:00:00:40
mip_ip 192.168.0.203
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1200:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
Exit
vm_name NODE105
mgmt_ip 192.168.122.204
mgmt_mac 52:54:00:00:00:50
mip_ip 192.168.0.204
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1600:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
```

Expected Result

one node becomes as FFR and others are as FFN.
Making eth0 interface of FFR down and node becomes as LFR and others become as LFN
Data path working between nodes.
Nodes is shown in NMS and also configurable through NMS.

Result:

`Cat /proc/net/mishap/mesh/adhoc` -> Node changed from FFR to LFR when eth0 becomes down

```
root@OpenWrt:~# cat /proc/net/meshap/mesh/adhoc
NODE OPERATING AS FFR STEP SCAN INDEX 0

-------------------------------------------------------------
PARENT ADDR      |CHN|SIG|RATE|TRATE|FLAGS      |FNC|R-VAL|
-------------------------------------------------------------
root@OpenWrt:~# ifconfig eth0 down
root@OpenWrt:~# cat /proc/net/meshap/mesh/adhoc_mode
LFR
```

`Cat /proc/net/mishap/mesh/table`

```
root@OpenWrt:~# cat /proc/net/meshap/mesh/adhoc
NODE OPERATING AS LFN STEP SCAN INDEX 0

-------------------------------------------------------------
PARENT ADDR      |CHN|SIG|RATE|TRATE|FLAGS      |FNC|R-VAL|
-------------------------------------------------------------
52:54:00:00:00:35|044|046|0000|00000|- M - - - L|LFN|00000|
52:54:00:00:00:15|048|046|0000|00100|- M - - - L|LFR|00000|
52:54:00:00:00:45|153|026|0000|00000|- M - - - L|LFN|00000|
52:54:00:00:00:55|048|006|0000|00000|- M - - - L|LFN|00000|
```

NMS Output-

Spawn 5 Wi-Fi-Simulator Nodes with Preferred Parent Where one node as root and others are as relay

Description

Create 5 Wi-Fi-simulator nodes with one node as Root and Other 4 nodes are as relay. Nodes will associated with its preferred parent.

Execution Steps

•Run klish, MAC and MAS utility to configure and creation of virtual machine.
•On clish Prompt , enter vm configuration command and fills appropriate data.
•Once configuration done , give network_status command to create and start of virtual machine.
•Use "virsh list --all" command to show the list of running virtual machine.
•Use "Virsh console <vm name>" command to get the console of virtual machine.
•Verify mode under cat /proc/net/meshap/mesh/adhoc_mode.
•Verify the entry of child nodes under cat /proc/net/meshap/mesh/table of parent node
•Verify the nodes in NMS.

Configurations (Given Through CLI)

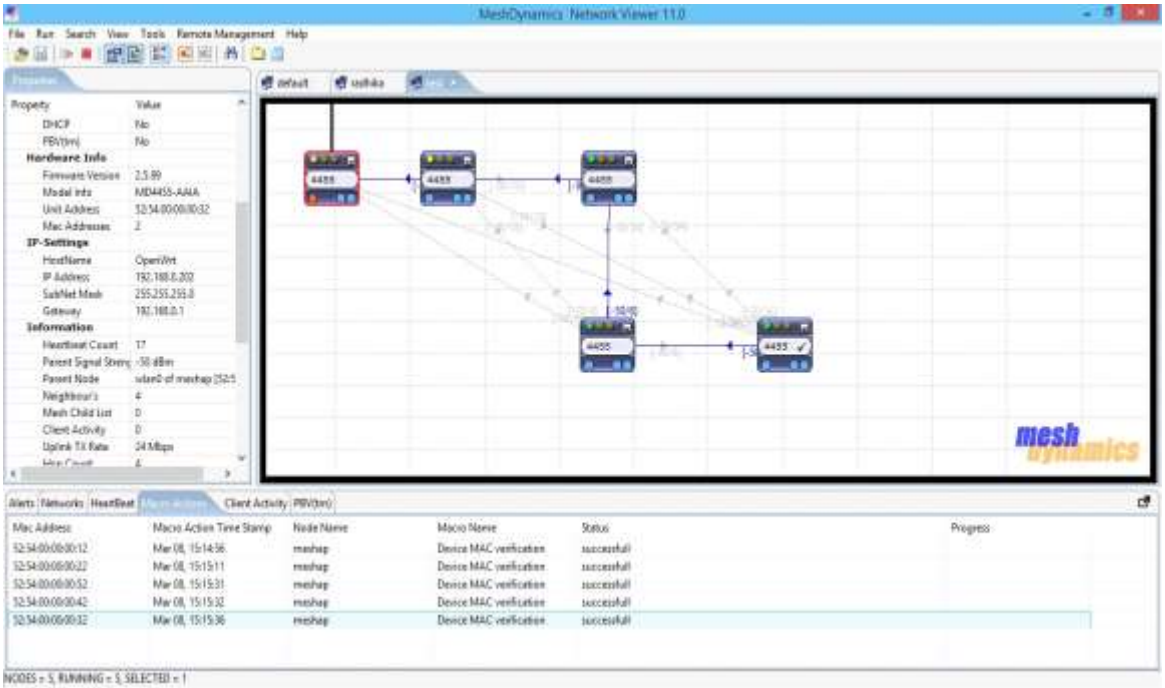
```
vm_name NODE101
mgmt_ip 192.168.122.200
mgmt_mac 52:54:00:00:00:10
mip_ip 192.168.0.200
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0000:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role root
vm_status enable
exit
```

```
vm_name NODE102
mgmt_ip 192.168.122.201
mgmt_mac 52:54:00:00:00:20
mip_ip 192.168.0.201
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0400:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE103
mgmt_ip 192.168.122.202
mgmt_mac 52:54:00:00:00:30
mip_ip 192.168.0.202
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0800:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
Exit
vm_name NODE104
mgmt_ip 192.168.122.203
mgmt_mac 52:54:00:00:00:40
mip_ip 192.168.0.203
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1200:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
Exit
vm_name NODE105
mgmt_ip 192.168.122.204
mgmt_mac 52:54:00:00:00:50
mip_ip 192.168.0.204
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1600:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
```

Expected Result

one node becomes as FFR and others are as FFN.
Every nodes associated with its parent node.
Data path working between nodes.
Nodes is shown in NMS and also configurable through NMS.

NMS output



Mobility Test Case : one Root node and 4 relay node with Preferred Parent and 1 mobile node.

Description

Create 5 simulator nodes , one node working as root and other 4 nodes working as relay . Provide preferred parent to all the relay nodes.

Execution Steps

Root and Relay node

- Run klish, MAC and MAS utility to configure and creation of virtual machine.
- On clish Prompt , enter vm configuration command and fills appropriate data.
- Once configuration done , give network_status command to create and start of virtual machine.
- Use "virsh list --all" command to show the list of running virtual machine.
- Use "Virsh console <vm name>" command to get the console of virtual machine.
- Verify mode under cat /proc/net/meshap/mesh/adhoc_mode.
- Verify the entry of child nodes under cat /proc/net/meshap/mesh/table of parent node
- Verify the nodes in NMS.

Mobile Node

- On clish prompt, enter vm configuration for mobile node with appropriate data.
- provide virtual machine speed and total distance(meter) coverd by mobile nodes.
- Once configuration done, provide "network_status enable" or "vm_status enable" command to create node.
- Verify mobile nodes creation by " virsh list --all"
- Use "Virsh console <vm name>" command to get the console of virtual machine.
- Verify mode under cat /proc/net/meshap/mesh/adhoc_mode.
- Verify the nodes in NMS.

Configurations (Given Through CLI)

ROOT and Relay Nodes

```
vm_name NODE101
mgmt_ip 192.168.122.200
mgmt_mac 52:54:00:00:00:10
mip_ip 192.168.0.200
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0000:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role root
vm_status enable
exit
vm_name NODE102
mgmt_ip 192.168.122.201
mgmt_mac 52:54:00:00:00:20
mip_ip 192.168.0.201
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0400:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE103
mgmt_ip 192.168.122.202
mgmt_mac 52:54:00:00:00:30
mip_ip 192.168.0.202
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0800:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE104
mgmt_ip 192.168.122.203
mgmt_mac 52:54:00:00:00:40
mip_ip 192.168.0.203
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1200:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
vm_name NODE105
mgmt_ip 192.168.122.204
mgmt_mac 52:54:00:00:00:50
mip_ip 192.168.0.204
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 1600:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role relay
vm_status enable
exit
```

Mobile Node

```
vm_name NODE106
mgmt_ip 192.168.122.205
mgmt_mac 52:54:00:00:00:70
mip_ip 192.168.0.205
model_number MD4455-AAIA
server_ip 127.0.0.1
server_port 4955
vm_coordinates 0000:0000:0000
vm_imag_path /home/Meshdynamics/wifi-sim/images/openwrt-x86-generic-combined-ext4.img
vm_role mobile
vm_mobile_distance 1600
vm_mobile_speed 36
vm_status enable
Exit
```

Here in Mobile Node Configuration , the total distance is given 1600 meter and vm speed is 36km/hr. Based to time and distance concept , the total time taken by mobile node from first node to last node is -

time = (1600*3600)/(36*1000) = 160 seconds [2.40 minute]

Expected Result

•one node become FFR and others will FFN based on given Preferred parent .
•Once mobile node starts , it will connect form its nearest node , FFR node.
•Based on vm speed, it will change the parent and move towards last node.
•Once it reached last node , it will come back towards first nodes with vm speed.
•Data path will be working between Root, relay and mobile nodes.
•Transitions form first node to last node and vice-versa can be seen inside NMS.

NMS OUTPUT

(next page)