Batman-Adv Mesh network emulator

José Daniel Britos[1], Silvia Arias[1], Nicolás Echáñiz[3], Guido Iribarren[3], Lucas Aimaretto[1], Gisela Hirschfeld[2]

[1] Laboratorio de Redes y Comunicaciones (LaRyC) Facultad de Ciencias Exactas, Físicas y Naturales, Universidad Nacional de Córdoba Av. Vélez Sarsfield 1611 - Ciudad Universitaria - CP: X5016GCA Córdoba.
[2] Departamento Universitario de Informática, Universidad Nacional de Córdoba, Valparaíso s/n - Ciudad Universitaria - CP: X5016GCA
[3] Fundación Altermundi, José de la Quintana, Córdoba.

Resumen We introduce a new network emulator environment, developed by our research group, called BAMNE. Our emulator is designed specifically to work with B.A.T.M.A.N. Adv. Mesh Protocols. This mesh network emulator, allow to test B.A.T.M.A.N. Adv. protocol and evaluate and debug mesh network. The emulated wireless equipment run in a virtual machine Virtualbox, and the wireless links are simulated with Vde-switch. Vde-switch allow simulate impediments in the link transmission such as lost of bits lost of packets, delay. To construct the emulation environment python language was used.

1. Introduction

The purpose of this network emulator is to test evaluate and debug B.A.T.M .A.N. Adv mesh network protocols. This network emulator is a front end for Openwrt machines running on Virtualbox connected through a Vde-switch and Wirefilter, simulating a wireless link, this is able to emulate delays, noise factors, channel bandwidth and packet loss on virtual wireless channel. The front end is written in python with "pygtk" graphics user interface. The python program don't introduces delay in the emulation due that the only purpose of this python software is to set up the emulation environment, parameters setup and monitor the Openwrt machines with the SNMP protocol. The main window of the program show transmitted packets for each interface, and originators tables for B.A.T.M.A.N. Adv protocol.

The Vde-switches have tap interfaces connected with the host machine, this allow to monitor the packet traffic with the Wire-shark program, the Openwrt eth0 interface are connected to the host via the Vboxnet interface of the host, in this way is possible to access to the Openwrt console for management purpose.This work is structured as follows: we start in Section 2 with the background on the Batman Adv. protocols. Section 3 describe the architecture of the emulation system. This is followed by Section 4 which describes the use of the emulator. Finally conclusions are drawn in Section 5.

2. Batman Adv.

BATMAN-Adv [11] is a proactive routing protocol for Wireless Mesh Networks (WMNs). It uses a distance-vector approach and a routing metric which incorporates the reliability of the radio links. Despite being developed and publicly available since 2006, BATMAN, especially its newer batman-adv variant, has received sparse attention in the scientific community. Batman is a simple and robust algorithm for establishing multi-hop routes in ad hoc networks. As explained by Johnson, D., et al [14] Batman does not maintain tables with full routes to the destination, each node along the route only maintains the information about the next link through which the node can find the best route. The objective is to maximize the probability of delivering a message. Batman does not check the quality of each link, it checks its existence. The protocol does these checks broadcasting periodically hello packets to its neighbours, these packets are known as originator messages (OGM). Broadcasting is when a single source sends messages to all available nodes in the broadcast domain/network. This is in contrast to unicast where a node sends messages to one specific node in the network. The structure of the OGM packet periodically sent is presented here:

- Originator address
- Sending node address: This is changed by receiving nodes and then the packet is re-broadcasted
- Unique sequence number: The sequence number is used to check the concurrency of the message
- Bidirectional link flag: Used when the OGM packet received is its own and the sender is someone else
- Time to leave (TTL)

When a node receives an OGM packet there are two possibilities, either the originator is or is not already in its routing table. If the originator is not in the routing table then a new entry is made for it and the sender node is added as a one hop neighbour to it and its count incremented. If the originator is in the routing table and the sender is a new, the sender is added as a one hop neighbour to the originator and count incremented. If the originator is in the routing table and the sender is not new, the senders count is incremented. The count is the amount of received OGMs packets of an originator through a specific one hop neighbour. The links are compared in terms of the number of originator messages that have been received within the current sliding window this value is called the transmission quality (TQ) and is the routing metric used by Batman. The sliding window is a fixed value that defines a range of the unique sequence numbers afforded to each OGM packet sent by a node [10]. Batman advanced (Batman-adv) [2] only uses the MAC address for addressing it neighbours. The result of working in layer two is that Batman-adv is able to emulate an Ethernet bridge, so that all nodes appear to be connected by a direct link. As cause all protocols above layer two are not aware of multi hop links. Batman's routing technique causes low processing and traffic cost. This makes it an attractive option for use on devices Batman's routing technique causes low processing and

traffic cost. In this work we focus on Batman-adv.That run in small processors such as the ARM MP.

3. Architecture

The emulator architecture is basically composed of two elements Nodes(OpenWrt) and Links (Wirefilter) [**?**] as shown in the figure 1. The nodes are shown in figure 2 this are more complex and have the following elements:

- OpenWrt, kamikaze trunk version for x86 with minimal modifications (see below)
- VirtualBox (unmodified) the version must support vde-switch [4].
- Vde_switch must run two instance for node to support 2.4 GHz and 5.0 GHz networks interface. The vde-switch have a patch colourful see below [13].

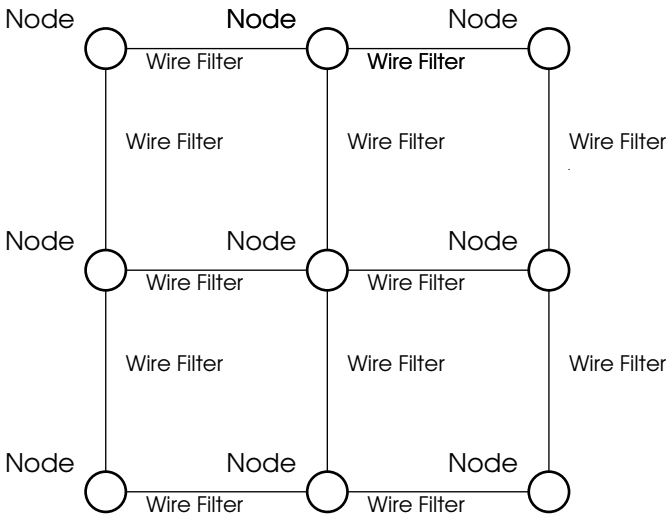

Figura 1. Network architecture

3.1. OpenWrt

A standard OpenWrt can be downloaded and configured for X86 [1]. Once that the virtual machine is running some packages must be download (ip, snmpd, tcpdump, netcat, kmod-batman-adv, batctl). For an automatic configuration of the network interfaces devices a setup script must be used on boot and stores it in */files/etc/rc.local* in your local OpenWrt build directory:

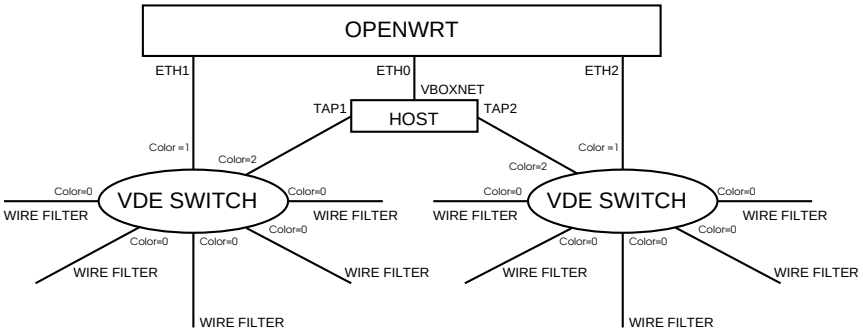

Figura 2. Node Diagram

```
#!/bin/sh
# pass ip config trough ethernet mac address
RED=$(ifconfig eth1 | sed '1,1!d' | sed 's/.*HWaddr //'| sed
's/.\{11\}://'| sed 's/.\{5\}$//')
NUM=$(ifconfig eth1 | sed '1,1!d' | sed 's/.*HWaddr //'| sed
's/.*://'| sed 's/[\n\ ].*//')
ip link delete eth0
ip addr add 192.168.100.$NUM/24 dev eth0
ip link set dev eth1 mtu 1532 up
ip link set dev eth2 mtu 1532 up
batctl -m bat0 interface add eth1
batctl -m bat0 interface add eth2
ip addr add 192.168.$RED.$NUM/24 dev bat0
ip link set dev bat0 address 90:$NUM:$NUM:$NUM:$NUM:$NUM
ip link set dev bat0 up
batctl -m bat0 originators
```

In VirtualBox is difficult to send the IP address for the interfaces of the virtual machines, this is accomplished setting the mac address in VirtualBox and in the *rc.local"* script read the mac address and set up the IP address in the interfaces.

3.2. SNMPD

When the "SNMP" is installed in the Openwrt machine we need to proceed to set up the MIB for batman-adv. To add custom records to the batman-adv MIB shell script was conducted running and returning to *stdout* what it takes from SNMP [5]. To request originators table, the next script was made: Script name *batctl_o.sh* (for originator list)

```
#!/bin/sh
BAT=$(batctl o | sed -n 's/^\(..:..:..:..:..:..\).*/\1/p')
echo $BAT
```

Then to add entries in the configuration file SNMPD */etc/snmp/snmpd.conf* using the following uci command in a terminal.

```
uci add snmpd exec
uci set snmpd.@exec[-1].name=.1.3.6.1.4.1.32.1.1
uci set snmpd.@exec[-1].prog=batctlo
uci set snmpd.@exec[-1].args=/batctl_o.sh
uci commit snmpd
/etc/init.d/snmpd restart
```

This append the following line to the files */etc/snmp/snmp.conf*.

```
exec  .1.3.6.1.4.1.32.1.1 batctl_o /batctl_o.sh
```

From the host the snmp request, can be tested with the followings command.

```
$ snmpget -v 1 -c public 192.168.100.11 iso.3.6.1.4.1.32.1.1.101
.1iso.3.6.1.4.1.32.1.1.101.1 = STRING: "80:03:00:00:07:41 80:03:
00:00:07:31 80:02:00:00:07:31 80:02:00:00:07:21 80:03:00:00:07:21"
```

We repeat the same for the next SNMP commands.
In the Git Hub *(https://github.com/dbritos/Network-mesh-emulator/blob/ master/openwrt.ova)* repository there is a fully configured virtual machine. Download openwrt.ova in VirtualBox go to:

```
File menu -> Import Appliance
```

3.3. Ip assignations in openwrt

To assign the IP address to the VM, first the MAC address is assigned to the VM. Each VM have three interfaces *nic1, nic2* and *nic3* this interfaces in the *openwrt* is shown as *eth0, eth1* and *eth2*.

```
nic1 (eth0) mac 80:01:00:00:07 + nodenumber(nn)
nic2 (eth1) mac 80:02:00:00:07 + nodenumber(nn) the 2 for 2.4GHz)
nic3 (eth2) mac 80:05:00:00:07 + nodenumber(nn) the 5 for 5.0GHz)
```

To configure the VM with this mac address the following commands are used:

```
VBoxManage modifyvm openwrtnn --nic1 generic --nicgenericdrv1 VDE
--nicproperty1 network=/tmp/c24GHznn[2] --macaddress1 8001000007nn
VBoxManage modifyvm openwrtnn --nic2 generic --nicgenericdrv2 VDE
--nicproperty2 network=/tmp/c24GHznn[2] --macaddress2 8001000007nn
VBoxManage modifyvm openwrtnn --nic3 generic --nicgenericdrv3 VDE
--nicproperty3 network=/tmp/c24GHznn[2] --macaddress3 8001000007nn
```

Where: *nn* is the Node number.
The script in the openwrt in */etc/rc.*local read the mac address of the interface *eth1* and configure the IP of the interfaces:

```
ip address eth0  = 192.168.100.nn
ip address bat0  = 192.168.7.nn
mac address bat0 = 90:nn:nn:nn:nn:nn
```

With this convention of IP Address and MAC address is easy to follow the packets trough the nodes. With the IP address of *eth0* interface is possible to access to the nodes via ssh command to the IP address 192.168.100.nn. (where nn is de node number). The host has the *vboxnet0* interface with the IP address 192.168.100.1. Each vde-switch have a tap interface through the network protocol analyzer "Wireshark" [12] can sniff the packets that traverse the vde-switch.

3.4. VirtualBox

The VirtualBox version must be 4.3 or higher [3]. To verify VDE-Switch support in the network windows select in Attached to: "Generic Driver" in the Name box, verify that exist VDE. In VirtualBox is difficult to set up the IP address for the interfaces of the virtual machines before starting the VM, this is accomplished setting the mac address in VirtualBox and in the *rc.local* script read the mac address and set the IP address in the interfaces.

3.5. Vde switch

The main advantage of vde-switch [7, 13] over uml switch is that any clients can be attached to this virtual switch: VirtualBox, UML, tap interfaces, virtual interconnections, and not just UML instances. If the vde-switches were just connected with wirefilter [9] "patch cables" without modification, we would end up creating a broadcast domain and switch loops which we don't want: The goal is to allow the packets to travel only from one host to it neighborhood, not farther. To accomplish this, the vde-switch needs to be modified to have *coloured* ports. The idea is that each port has a *colour* (an integer number), packets are only passed from ports to others with different *colours*. Packets are dropped on outgoing ports if this have the same*colour* (same number) as the incoming port. In this concept, the host port can have *colour 1* the TAP port *colour 2*, while the interconnection ports have *colour 0*. In this way, packets can only travel from the host to (all of) the interconnection ports, or from one interconnection port to the host port. However packets can not travel between the interconnection ports, thus only allowing "one hop" connections and avoiding switch loops and shared broadcast domains. The concept is illustrated in figure 2.

The patch against vde2-2.3.2 (current latest stable version) to add this colour patch can be find here:

```
(http://www.open-mesh.org/attachments/download/152/vde2-2.3.2_
colour.patch).
```

The vde-switch patched can be download from here:

`(https://github.com/dbritos/Network-mesh-emulator/blob/master/vde2-2.3.2-patch.tar).`

3.6. Wirefilter

The wirefilter program [6] is a tool where it is possible to simulate various link defects and limits as example: packet loss, burst loss, delay, duplicates, bandwidth, Interface speed, Channel capacity, Noise (damage to packets), mtu. However as the links are only set up bidirectional, interferences can unfortunately not be simulated with this system. For advanced testing it might be necessary to apply the aforementioned link defects to some packets only whereas other packets are able to traverse the emulated environment unharmed. Once you applied the "ethertype" patch you can specify an ethertype which wirefilter will simply forward. To apply a packet loss of 50 % to all packets except batman-adv packets, run:

```
wirefilter --ether 0x4305 -l 50
```

This patch also allows to filter batman-adv packet types. To apply a packet loss of 50 % to all packets except batman-adv ICMP packets, run:

```
wirefilter --ether 0x4305:02 -l 50
```

You can specify up to 10 packet types (separated by colon). The patch against vde2-2.3.1 (current latest stable version) can be found here:

`http://www.open-mesh.org/attachments/download/106`

4. Using BAMNE

In this section, we present a simple example using BAMNE to create a topology of five groups of nine nodes each. This example of BAMNE emulation is shown in the figure **??**, and discussed in detail. In this emulation, there are 45 nodes running Batman Adv. protocol in a computer with an Intel i7 CPU and 16GB of RAM.

The software have two modes of operation, when we run the program it begins in edition mode in this mode we can build the network topology to change to emulation mode, in the main menu we can choose "RUN" mode and the emulation begin until we choose "STOP" from main menu to finish the emulation.

In Execution mode we can't create nodes but we can destroy nodes or link and modify links to see how the Batman Protocol reacts to the case of nodes or links failures.

The information shown in main windows is the following: In the first line of the screen we can see the transmitted and received packets of each interfaces of the highlighted node *lo*, *eth0*, *eth1*, *eth2* and *bat0* Interface. In the second and third

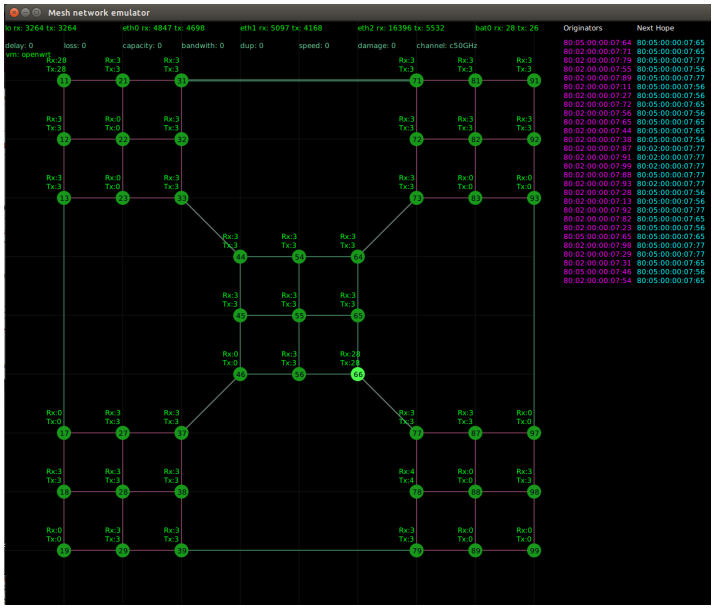

Figura 3. Emulator screen

line is shown the properties of the links wire filter, packets loss, channel capacity, damage packets, delay, bandwidth of the channel, duplicate packets and channel frequency for each frequency 2,4 and $5GHz$ respectively. Above each node there are the number of transmitted and received packets. Inside the green circle of each nodes is shown the node number. The links between nodes in red are $2{,}4GHz$ links and the links in green are the $5GHz$ links.

In the top right of the screen there are the originators and next hop list for the highlighted node number, in this example node 66 is shown.

When the virtual machine of the OpenWrt is created a console of this machine appears in the main windows from we can manage an run test, for example in the figure 4 is shown the console of VM node 66 making a ping to VM node number 71, with five hops, the delay is good and bellow $150ms$, the jitter are bellow $30ms$ showing that the network can support VoIP due to the values of delay fall well within the boundaries recommended by the ITU-Recommendation G. 114 [8].

In the figure 5 it is shown the CPU and memory usage for 45 nodes running all together, in this chart it is possible to see that the CPU usage is less than 35 % and the memory usage is less than a 35 % of $16GB$ of RAM.

5. Conclusions

In this paper we propose an emulator for Batman-adv protocol, with the capacity to evaluate the performance and the convergence of the protocol buil-

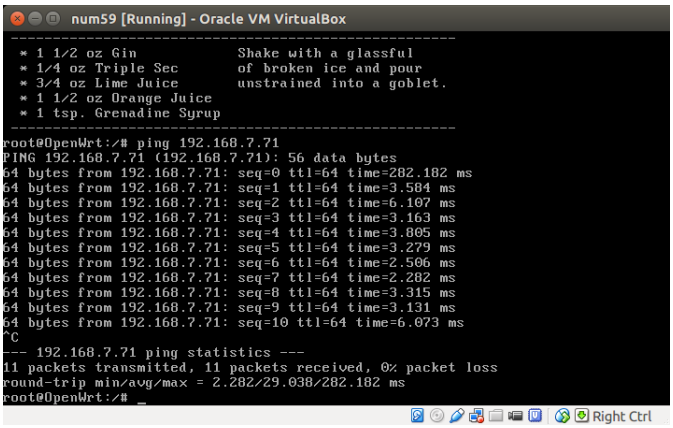

Figura 4. Delay with ping

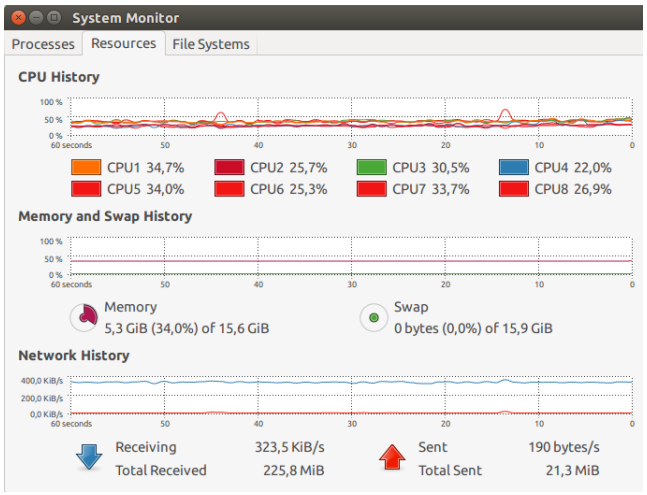

Figura 5. CPU usage

ding the next hop table for many topologies of the network. This program has a graphical interface to build the network and send commands to virtualbox, vde-switch and to show graphically the originators, next hop tables, interface properties and show the amount of transmitted and received packets for each one. The program fulfilled the expectations proposals, it can simulate various impediments on the transmissions links as lost packets, delay, bandwidth , showing a good performance up to 90 nodes in Intel i7 processors with 16 GBs. of RAM. The principals goal to add to this program is to show graphically the trace route at level two of the OSI model.

Referencias

1. Batman-adv-openwrt-config - batman-adv - Open Mesh, http://www.open-mesh.org/projects/batman-adv/wiki/Batman-adv-openwrt-config
2. Doc-overview - batman-adv - Open Mesh, http://www.open-mesh.org/projects/batman-adv/wiki/Doc-overview
3. OpenWrt in VirtualBox [OpenWrt Wiki], http://wiki.openwrt.org/doc/howto/virtualbox
4. Oracle VM VirtualBox, https://www.virtualbox.org/
5. SNMPD [OpenWrt Wiki], http://wiki.openwrt.org/doc/howto/snmp.server
6. Ubuntu Manpage: wirefilter - Wire packet filter for Virtual Distributed Ethernet, http://manpages.ubuntu.com/manpages/natty/man1/wirefilter.1.html
7. VDE - Virtualsquare, http://wiki.virtualsquare.org/wiki/index.php/VDE
8. ITU T.: One-way transmission time. recommendation G.114 (Feb 1996)
9. Caini, C., Firrincieli, R., Davoli, R., Lacamera, D.: Virtual Integrated TCP Testbed (VITT). In: Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities. pp. 36:1–36:6. TridentCom '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium (2008), http://dl.acm.org/citation.cfm?id=1390576.1390620
10. Cerda-Alabern, L., Neumann, A., Escrich, P.: Experimental Evaluation of a Wireless Community Mesh Network. In: Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems. pp. 23–30. MSWiM '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2507924.2507960
11. Chissungo, E., Blake, E., Le, H.: Investigation into Batman-adv Protocol Performance in an Indoor Mesh Potato Testbed. In: 2011 Third International Conference on Intelligent Networking and Collaborative Systems (INCoS). pp. 8–13 (Nov 2011)
12. Combs, G., et al.: Wireshark-network protocol analyzer. Version 0.99 5 (2008)
13. Davoli, R.: VDE: Virtual distributed Ethernet. In: First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. pp. 213–220 (Feb 2005)
14. Johnson, D., Ntlatlapa, N., Aichele, C.: Simple pragmatic approach to mesh routing using BATMAN (Oct 2008), http://researchspace.csir.co.za/dspace/handle/10204/3035