# Soporte para prueba y análisis de redes WMN

**Guillermo Rigotti**
Facultad de Ciencias Exactas Universidad Nacional del Centro de la Pcia. de Bs. As.
grigotti@exa.unicen.edu.ar

**Resumen**

Las redes malladas inalámbricas (WMN) han tenido una amplia difusión en los últimos años debido a su bajo costo, facilidad de instalación y su versatilidad para adaptarse a una gran variedad de aplicaciones, entre las que se cuentan, situaciones imprevistas de desastres, aplicaciones militares, y conformación de redes en lugares en los que no se dispone de otra infraestructura de comunicaciones. En particular, el trabajo presentado aquí surge como una necesidad que se observó durante la implementación de una red inalámbrica comunitaria. El objetivo es el desarrollo de una herramienta que permita emular el comportamiento de una red WMN en una única PC utilizando sistemas operativos reales. Esto permite adquirir experiencia en la configuración de los nodos y en el análisis y evaluación de los protocolos y sistemas operativos a utilizar, sin necesidad de contar con la red real instalada. Además, se hace posible evaluar de qué manera influye la calidad de los enlaces de comunicación y la movilidad de los equipos en el servicio ofrecido por la red a los usuarios. El trabajo realizado permite incorporar a una herramienta de emulación de redes, CORE, máquinas virtuales corriendo sistemas operativos reales que se utilizan en los nodos mesh, en particular, openwrt.

***Palabras clave**:* Redes inalámbricas malladas, emulación, B.A.T.M.A.N.

## Introducción

Las redes inalámbricas malladas (Wireless Mesh Networks -WMN-), son redes compuestas por nodos conectados entre sí mayoritariamente por vínculos wireless (por ejemplo 802.11, WiMax, etc.). Generalmente poseen una estructura basada en un backbone constituido por los nodos con mayor capacidad de comunicación, disponibilidad, estabilidad, etc.; a este backbone se conectan (por medios inalámbricos o cableados) los equipos de los usuarios, cuyo objetivo es utilizar la capacidad de comunicación y posible acceso a Internet que ofrece la red WMN.

En general, es posible distinguir los nodos por su función, aquellos que operan como ruteadores, denominados nodos mesh, los que ofrecen además capacidad para brindar acceso a la Internet o a otras redes (funcionalidad de gateways) y aquellos nodos que son los de los usuarios de la red (clientes), que pueden o no actuar con ruteadores, según como estén configurados.

Las WMNs se han popularizado en los últimos años debido a diversos factores. Entre ellos podemos mencionar facilidad de instalación, bajo costo, resistencia a fallas, capacidad de auto organización, y versatilidad en cuanto a su campo de aplicación. Este último aspecto ha llevado a utilizarlas en situaciones de emergencias provocadas por desastres naturales, en aplicaciones militares, y como medio de acceso a Internet y de comunicación en zonas que no cuentan con la infraestructura adecuada de comunicaciones. En este último aspecto, se han popularizado las redes comunitarias, que proveen comunicación y acceso gratuito a Internet en zonas sin infraestructura de comunicación. Entre estas redes podemos citar Buenos Aires Libre, Quintana Libre, en Argentina, la iniciativa denominada Freifunk en Alemania, etc. La motivación de este trabajo se debe a una necesidad detectada durante el desarrollo del

proyecto Red Inalámbrica Educativa Comunitaria (RIEC) [1], realizado en el marco de la 22 convocatoria de proyectos de Extensión Universitaria y Vinculación Comunitaria 2014: Universidad, Estado y Territorio. En el mencionado proyecto se instaló y configuró una red comunitaria inalámbrica. Durante el desarrollo del proyecto se hizo evidente la utilidad de que los encargados de la instalación, configuración y mantenimiento de la red se familiarizaran, con algunos aspectos básicos de su funcionamiento. Entre ellos se puede mencionar el protocolo de comunicación utilizado, la configuración de los nodos y su relación con la calidad de servicio provista, los sistemas operativos a utilizar (el más común openwrt), y aspectos relacionados con la escalabilidad de la red.

El objetivo perseguido por el trabajo que aquí se presenta fue crear una herramienta que, integrada en una misma PC, permita emular redes mesh, incluyendo características de las líneas de comunicación y movilidad de los nodos, incorporando nodos reales a dicha emulación y también el acceso a servidores locales y a la Internet. De esta manera, es posible replicar la red a instalar y las aplicaciones para las cuales es diseñada.

El resto del trabajo se organiza de la siguiente manera: en la sección 2 se hace referencia al software utilizado, en la sección 3 se describen los aspectos relevantes relativos a la integración de ese software, cuáles fueron los problemas encontrados, sus soluciones y las limitaciones resultantes, En la sección 4 se describe un ejemplo simple de uso del ambiente resultante, que incluye la configuración y la visualización del tráfico generado. Finalmente, en la sección 5 se hace referencia a las conclusiones a las que se arribó luego de finalizar el trabajo, y a futuros trabajos relacionados. La sección 6 detalla la bibliografía y recursos utilizados.

**Software utilizado**

Se decidió implementar la solución buscada integrando la plataforma de emulación CORE con el soporte de virtualización provisto por VirtualBox. Para el análisis del tráfico generado se utilizó Wireshark. El sistema operativo virtualizado utilizado durante el desarrollo fue Ubuntu-14.10 Server, y el protocolo de ruteo en la red mesh emulada, batman advanced. El trabajo se desarrolló en un sistema operativo Ubuntu 14.04 LTS.

CORE [2] es un emulador de redes basado en la creación de nodos virtuales para representar los nodos a emular; utiliza el soporte namespace provisto por Linux para realizar la virtualización. Esto permite que cada nodo virtualizado sea una réplica (limitada) del sistema operativo anfitrión. CORE está estructurado en dos partes principales, un servidor o daemon que se encarga de crear y administrar los nodos virtuales y una interfaz gráfica que permite a los usuarios definir emulaciones e interactuar con ellas en la etapa de ejecución de las mismas. El servidor está desarrollado en su mayor parte en Python, y se basa en la utilidad bridge-utils provista por Linux para emular las redes que conectan a los nodos emulados. La interfaz gráfica, desarrollada en un principio para el proyecto IMUNES (Integrated Multiprotocol Network Emulator/Simulator) [3], está desarrollada en Tcl/Tk, y provee un entorno gráfico muy simple de utilizar, que integra tanto la parte de diseño de la red como la parte de ejecución, permitiendo en esta última etapa, acceder a las consolas de cada nodo virtual y producir el desplazamiento geográfico de los nodos móviles. Dos características decisivas para la elección de CORE fueron: 1-su compatibilidad con VirtualBox, ya que son esquemas de virtualización que soportan bridge-utils, lo cual permite integrar sus interfaces de red, y 2-CORE tiene integrado a EMANE (Extendable Mobile Ad-hoc Network Emulator) [4], un soporte que permite la emulación de redes inalámbricas con un alto grado de detalle. En el desarrollo se utilizó la version 4.8 de CORE, que soporta EMANE 0.9.2.

El protocolo de ruteo utilizado en las redes WMNs, es batman (better approach to mobile ad-hoc networking) advanced [6]. Es un protocolo de uso muy difundido en las redes WMN , y ha evolucionado desde una versión

arquitecturalmente ubicada sobre IP, a una que se ubica sobre el nivel IEEE.802.11 o similar, convirtiendo a la red en un switch que comprende a la totalidad de los nodos. Esta característica lo hace independiente de cualquier protocolo de nivel 3, como IPv4 o IPv6, pero sin embargo plantea algunos interrogantes acerca de su escalabilidad. Batman advanced se distribuye con Linux en forma de módulo kernel; este módulo se encarga del manejo de las interfaces y de los aspectos de ruteo y control del nodo. Se utilizó también el utilitario batctl, que permite a través de comandos, configurar el comportamiento del nodo y acceder a tablas internas y estadísticas.
Para el análisis del trafico en la red se utilizó la versión 2.0.1 de Wireshark [5], que, además de poder invocarse desde la interfaz gráfica de CORE, cuenta con un disector para el protocolo batman: esta característica, no soportada por versiones anteriores de Wireshark, es de suma importancia ya que permite comprender el funcionamiento del protocolo y analizar su comportamiento en situaciones especiales.
El sistema operativo donde se realizó el trabajo es Ubuntu Desktop 14.04 LTS; los nodos virtualizados fueron Ubuntu server 14.04 durante el desarrollo, y luego se utilizó Openwrt [5], este ultimo de gran importancia ya que es el que comúnmente corren los routers wireless (nodos mesh).

**Aspectos relevantes de la integración del software utilizado**

A continuación se describe de qué manera se integró el software mencionado en la sección anterior, cuáles fueron los problemas encontrados y cómo se resolvieron. Una primera incompatibilidad fue la de batman advanced con CORE. Debido a que el código de batman-adv no tiene en cuenta el uso de namespaces, fundamentales para la creación de nodos emulados por parte de CORE, se le aplicó el patch correspondiente obtenido de [8]. Luego de la recopilación e instalación del módulo batman-adv fue posible que los nodos emulados (sus namespaces) fueran reconocidos por el módulo batman. Otro problema derivado de que batman-adv utiliza debugfs y éste no reconoce namespaces, es que las interfaces virtuales creadas por batman (por defecto llamadas bat0), producen inconsistencia si son utilizadas con este nombre en más de un nodo emulado; la solución fue utilizar diferentes nombres en cada nodo virtual, haciendo corresponder el nombre de interfaz **i** para el nodo i (suponiendo que en cada nodo virtual se define una única interfaz bat**x**). Una consecuencia adicional del uso de debugfs por parte de batman es que en los nodos emulados no puede accederse a la información estadística y de control de diferentes aspectos de ruteo almacenados por cada nodo (por ejemplo translation table, nodos originadores, tabla arp distribuída, etc). Tampoco puede ejecutarse los comandos específicos de batman (a nivel 2) ping y traceroute. Este inconveniente planteaba la alternativa de reemplazar el uso de debugfs en batman, lo cual excedía los objetivos actuales del trabajo. Sin embargo, estas limitaciones no existen en los nodos incorporados a la emulación en forma de máquinas virtuales, por lo que no resulta significativa ya que podemos observar el estado y comportamiento de la red desde estos últimos (en un caso extremo, dependiendo de la capacidad del equipo anfitrión, todos los nodos de la red mesh podrían ser nodos implementados como máquinas virtuales.
Como se mencionó antes, una característica de gran interés fue el hecho de lograr que sistemas operativos reales corriendo en máquinas virtuales pudieran incorporarse a la emulación como nodos emulados, pero sin las limitaciones que éstos presentan. De esta manera, se tiene la posibilidad de simular WMNs con un numero considerable de nodos (por ejemplo para chequear su escalabilidad), de los cuales algunos pueden ser máquinas"reales" virtualizadas o máquinas independientes, por ejemplo un router wireless corriendo openwrt.

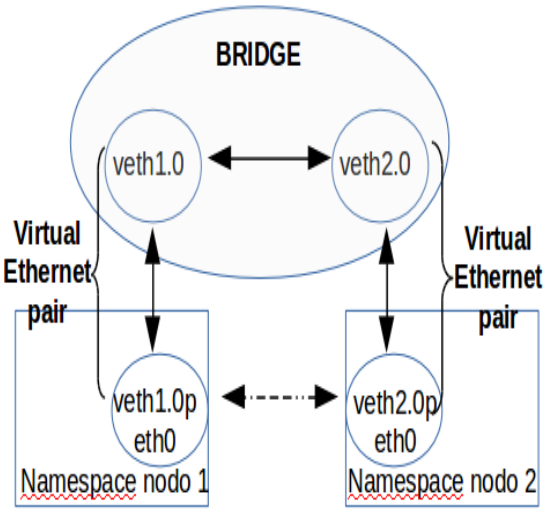


Fig.1 Esquema de comunicación entre namespaces utilizado por CORE

Para lograr incorporar una máquina virtual a la emulación se recurrió a definir nodos emulados cuya función es representar en la emulación a la máquina virtual (proxys de la máquina virtual). De esta manera no se requiere ningún tipo de modificación del código CORE. Estos nodos se limitan a reflejar lo recibido desde la máquina virtual en una interfaz hacia la red mesh.
Para que un nodo emulado pueda conectarse con una interfaz del anfitrión, CORE define un elemento denominado RJ45, que es capaz de ser asociado a un nodo emulado, y por otra parte, se asocia a una interfaz existente del host anfitrión. VirtualBox permite definir adaptadores de red conectados como adaptadores puente, también sobre una interfaz del host anfitrión. De esta manera, la conexión nodo emulado-maquina virtual se obtiene conectando ambas interfaces a traves de un bridge. Dese el lado de VirtualBox, es necesario utilizar un adaptador puente debido a que la interfaz debe estar visible para CORE. No es posible utilizar una red interna de VirtualBox, ya que esto no implica la visibilidad de alguna interfaz para conectar al RJ45 de CORE; tampoco es conveniente utilizar una interfaz física del anfitrión para el adaptador puente, porque en el caso de que se quiera conectar mas de un nodo core a las máquinas virtuales, necesitaríamos disponer de varias interfaces físicas, ya que CORE no permite utilizar mas de una vez la misma interfaz debido a que la asocia a un bridge, y bridge-utils no permite que una interfaz sea asociada a más de un bridge.
Para implementar la conectividad de los nodos emulados (cada uno en un namespace propio), CORE utiliza interfaces virtuales tipo veth conectadas por bridges provistos por el soporte bridge-utils. Por el momento no está pensado actualizar este soporte a plataformas de mayor performance como OVS (Open Virtual Switch). En particular, por cada interfaz definida en un nodo emulado, CORE crea un par de interfaces tipo veth; la característica de este tipo de interfaces es que ambas resultan comunicadas entre sí, pudiendo estar en diferentes namespaces. Estas interfaces son llamadas en el código de CORE vethx.y.s, donde ”x” hace referencia al número de nodo, ”y” al número de interfaz dentro del nodo, y ”s” al número de sesión al que pertenece el nodo (el servidor core soporta sesiones simultáneas independientes entre sí). Por ejemplo, veth1.0.55 hace referencia a la interfaz eth0 del nodo 1. En el momento en que (por ejemplo) desde la interfaz gráfica se solicita al servidor CORE iniciar una emulación, se crea un par de veths por cada una de las interfaces a emular, y una de ellas es movida por CORE al namespace correspondiente al nodo, cambiándole el nombre por uno adecuado al caso usual (eth0, eth1, etc.). Desde el namespace del nodo veremos sólo esta interfaz, pero lo que enviamos y recibimos por ella es exactamente igual en su par. Para construir un link entre dos interfaces emuladas, CORE crea un bridge en su propio espacio (visible desde una consola del anfitrión, no desde el nodo simulado) conteniendo las dos interfaces que quedaron en su espacio. De esta forma, cada vez que un nodo envía información por su interfaz (en su propio namespace), esta información pasa a su interfaz par, en el namespace del daemon core: alli, como consecuencia del bridge, se propaga a la interfaz par de la del otro nodo, que de manera similar a la anterior, se refleja en la interfaz

del segundo nodo, en su namespace. Esto puede verse en la Figura 1, donde se indica la comunicación real con líneas enteras, y la comunicación entre namespaces (los dos nodos) con línea punteada.

En la figura 2 se muestra el esquema de conexión de CORE con una máquina virtual. Debido al manejo de interfaces que realizan CORE y VirtualBox, se propone la creación de una interfaz lógica por cada conexión entre un nodo CORE y una máquina virtual, antes del inicio de la ejecución de las máquinas virtuales y de CORE. Se asocia esa interfaz al adaptador puente de la máquina virtual, y por otro lado se asocia la interfaz al RJ45 del nodo CORE. Como se ve en la figura, CORE asocia dicha interfaz a un bridge, que tiene como componente a la interfaz par de la asociada al nodo en su namespace. Para casos de conectividad más complejos, es recomendable utilizar un par de interfaces tipo veth, destinando una a CORE y la otra al adaptador puente de la máquina virtual, ya que como VirtualBox no asocia esta interfaz a ningún bridge, queda la posibilidad de hacerlo para construir esas conexiones de mayor complejidad. En nuestro caso, se utilizó un par de interfaces.

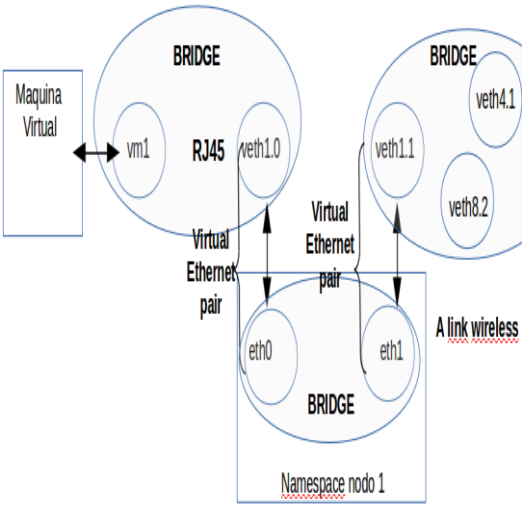


Fig 2. Conexión de una máquina virtual con su nodo proxy

**Ejemplo**

En esta sección se presenta un ejemplo de uso de la herramienta. Se conformó una red WMN compuesta de 8 nodos, 5 de los cuales son nodos emulados (n4 a n8) y se muestran en verde, mientras que los 3 restantes (n1 a n3) son máquinas virtuales incorporadas a la emulación a traves de nosdos proxys. En la figura 3 puede verse la topología de la red y las conexiones resultantes entre los nodos como consecuencia del alcance de cada uno de ellos. Cada link se configura de acuerdo a EMANE; se muestra para cada nodo asociado con una máquina virtual, el elemento RJ45 que permite la asociación.

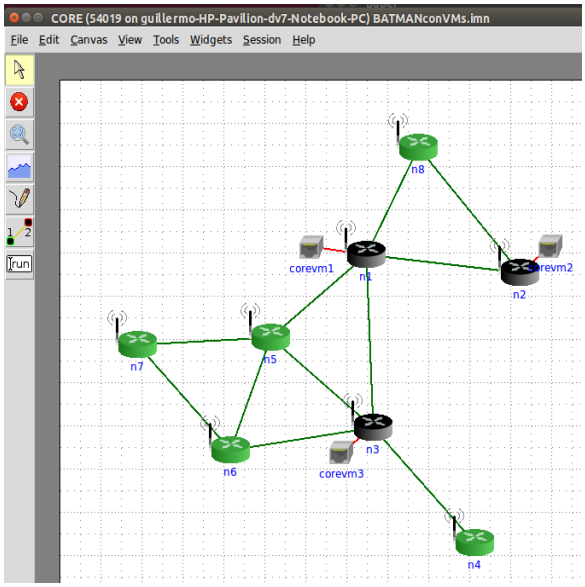


Fig 3. Topología utilizada para el ejemplo

En la máquina anfitrión, antes de realizar la ejecución, se han creado los tres pares de interface tipo veth, destinadas a conectar cada máquina virtual con su proxy en CORE. Por otro lado, se ha activado el módulo batman-adv, que (como consecuencia del patch mencionado), soporta namespaces y por lo tanto puede correr en los nodos CORE emulados. En paralelo con la emulación, se ejecuta VirtualBox, que virtualiza tres Ubuntu server asociados a los nodos proxy. En las figura 4 se muestran los comandos de arranque de los nodos proxy (creacion del bridge uniendo las interfaces) y en los nodos que corren batman directamente desde su namespace (incorporan su interfaz eth0 al protocolo batman)

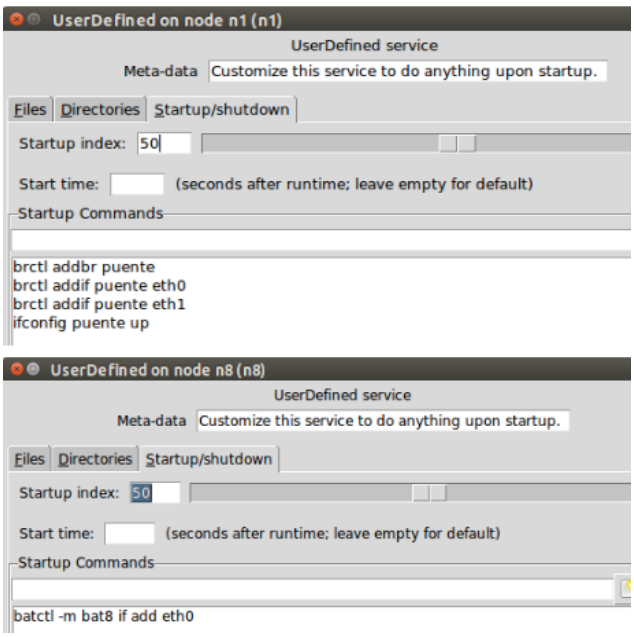


Fig 4. Configuración inicial de los nodos emulados. Arriba: nodos ejecutando batman-adv. Abajo: nodos proxys.

Por otro lado, en cada una de las máquinas virtuales se ha cargado el módulo batman-adv y también se ha incorporado la interfaz eth0 (correspondiente al adaptador puente conectado a CORE) al protocolo batman. Durante la ejecución realizamos un ping y un traceroute desde el nodo 2 (máquina virtual 2) hasta el nodo 3 (estas funciones, ping y traceroute se realizan a nivel MAC, y son provistas por batctl). Luego solicitamos a batctl información almacenada en el kernel, acerca de los nodos vecinos. La consola de la máquina virtual puede verse en la figura 5. Finalmente, en la figura 6, se muestra una captura sobre la interfaz de la máquina virtual asociada al nodo 3; en ella es posible visualizar los frames intercambiados por los nodos en forma periódica (OGM - Originator Messages-) para anunciarse a los demás, y el echo request y echo reply generados por un ping del nodo 1 al nodo 3. Notar que, por razones de claridad, hemos configurado las direcciones MAC de los nodos para que se correspondan con el número de nodo

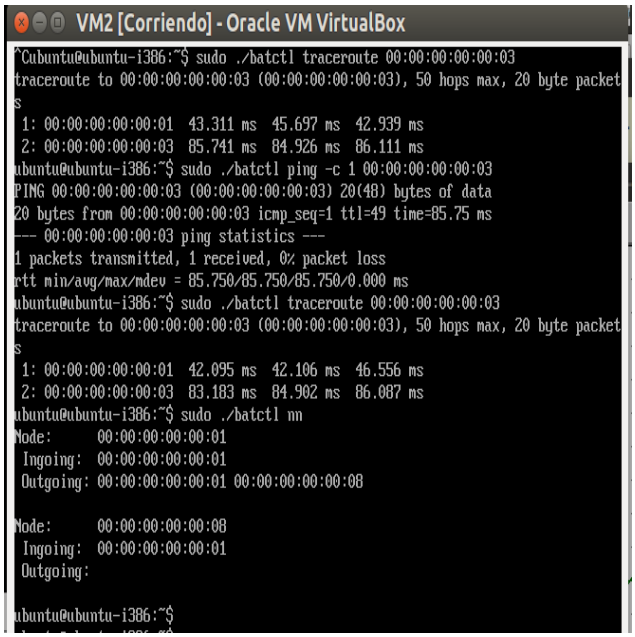


Fig 5. Uso de batctl en una maquina virtual

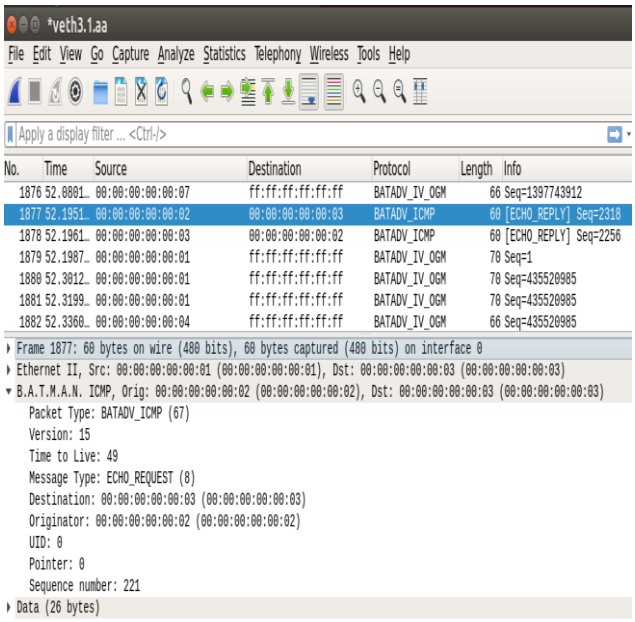


Fig 6. Captura de tráfico realizada con Wireshark

## Conclusiones

La motivación de este trabajo fue la necesidad de contar con un ambiente que permitiera familiarizarse con la configuración del protocolo batman-adv operando en redes mesh. Si bien la plataforma de emulación CORE provee la capacidad de construir redes mesh, se encontró que debido a incompatibilidad parcial del emulador con el

protocolo, es imposible configurar totalmente un nodo emulado y también poder acceder a los datos de la red que el nodo almacena. Otro impedimento que presenta el emulador es que al estar basado en la utilidad bridge-util, no es capaz de soportar redes locales virtuales (VLANs) que sí son soportadas y utilizadas por batman-adv. En este punto, cabe aclarar que en una emulación con nodos emulados y máquinas virtuales, solo estas últimas lo soportan. Por lo tanto, para poder familiarizarse plenamente con los componentes de la red mesh, es necesario incorporar sistemas reales a la emulación, lo que se logró con éxito. A través de la información obtenida con el utilitario batctl, se adquirió experiencia relativa a las posibles configuraciones de los nodos y el efecto que estas tienen en la performance de la red, y en la interpretación de los datos estadísticos que batman-adv provee. En la actualidad, se está evaluando de qué manera influye el hecho de incorporar máquinas virtuales remotas y routers reales corriendo Openwrt a la emulación. El análisis del código de CORE que fue necesario para comprender la creación de nodos, redes e interfaces, abrió un camino para considerar la posibilidad de reemplazar el soporte bridge-utils por Open Virtual Switch, lo que permitiría, entre otras cosas, el soporte de VLANs en CORE.

En el futuro, se espera poder trabajar en un análisis más exhaustivo de algunas de las características de batman advanced aún no exploradas, básicamente su escalabilidad en cuanto a la cantidad de clientes y al roaming de los nodos (que no necesitan ser máquinas virtuales). Otro aspecto de interés es utilizar la herramienta para comparar la performance y características de batman operando en un entorno IPv4 y otro IPv6.

**Referencias**

[1] https://docs.google.com/presentation/d/1pQ-peoMRMrhaw2jYQkgyiz5RhXKGKo0o6T_dQjZAEfg/edit?usp=sharing
[2] Common Open Research Emulator (CORE), www.nrl.navy.mil/itd/ncs/products/core
[3] Integrated Multiprotocol Network Emulator/Simulator, imunes.net
[4] Extendable Mobile Ad-hoc Network Emulator http://www.nrl.navy.mil/itd/ncs/products/emane
[5] Wireshark 2.0.1 https://www.wireshark.org/docs/relnotes/wireshark-2.0.1.html
[6] https://www.open-mesh.org/projects/batman-adv
[7] htps://openwrt.org
[8] https://lists.open-mesh.org/pipermail/b.a.t.m.a.n/2014-May/012055.html.