HIP MANET

Anonimato en redes ad-hoc mediante integración de los protocolos HIP y OLSR

Proyecto Final de Carrera
Ingeniería Informática
Redes y Sistemas Operativos

Autor: Francisco Javier Campos Berga
Directores: Pietro Manzoni y Carlos Tavares Calafate

Valencia, 1 de julio de 2009

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Las redes ad-hoc son redes inalámbricas que no requieren ningún tipo de infraestructura fija ni administración centralizada. En ellas todos los nodos deben ofrecer servicios de encaminamiento, retransmitiendo paquetes entre aquellas máquinas que no tienen conexión inalámbrica directa. Además, estas redes se caracterizan por los frecuentes cambios de topología.

La naturaleza de las redes ad-hoc hace que aparezcan los siguientes problemas. Los datos intercambiados entre dos estaciones deben atravesar otras máquinas. Estos datos podrían ser recopilados o modificados por los nodos intermedios. Asimismo, los nodos intermedios sabrán qué dos estaciones mantienen una comunicación. Además, cualquier nodo puede suplantar la identidad de otro, generando paquetes con direcciones IP origen diferentes a la suya. Por otro lado, los constantes cambios de topología y la frecuencia con la que los nodos abandonan temporalmente la red, hace menos recomendable si cabe el uso de las direcciones IP como identificador de cada estación, ya que estas direcciones pueden cambiar con frecuencia.

Por tanto, es recomendable un nuevo sistema para identificar los nodos, para que estos no pierdan o cambien su identidad cuando cambien de dirección IP. También se requieren mecanismos que garanticen la integridad, la confidencialidad y la autenticación de los paquetes, ya que estos atraviesan otras estaciones. Además es necesario poder distinguir entre paquetes actuales y ataques de reenvío. Por último, puede resultar interesante en algunos escenarios ocultar la identidad de las 2 estaciones implicadas en una comunicación, para que los nodos intermedios no dispongan de ninguna información. Denominaremos a esta última característica “anonimato”.

En este trabajo se propone una modificación del protocolo HIP (Host Identity Protocol) [1] para poder utilizarlo en redes ad-hoc. Con ello se cubren todas las carencias descritas anteriormente, incluido el anonimato, que no se encuentra contemplado en la versión original del protocolo HIP.

HIP es un protocolo experimental para Internet que permite establecer conexiones seguras entre hosts, y mantener estas conexiones aunque la localización (dirección IP) de los hosts cambie. La movilidad se consigue con una nueva capa en la pila de protocolos, en particular entre las capas de red y transporte. HIP está diseñado para ser resistente a ataques de denegación de servicio y de hombre-en-medio. Durante el establecimiento de la conexión, los hosts intercambian sus claves públicas y acuerdan una clave secreta de sesión mediante el mecanismo Diffie-Hellman [4]. Tras establecer la asociación, puede utilizarse el protocolo ESP [3] para garantizar la integridad de los datos y cifrarlos. HIP asume que inicialmente se conoce (o se puede consultar en un DNS) la IP y la identidad del host con el que se quiere establecer la comunicación. Esto no se puede asumir en redes ad-hoc, donde los hosts pueden cambiar constantemente de ubicación y donde un servidor DNS no es común.

Para lograr el anonimato será necesario, además de ocultar los identificadores del protocolo HIP, utilizar seudónimos en la capa de red, así como modificar el protocolo de

encaminamiento para que estos seudónimos sean interpretados como nuevos nodos de la red.

1.1. Objetivos

El objetivo de este trabajo es diseñar e implementar una modificación del protocolo HIP que pueda ser utilizada en las redes ad-hoc. Esta modificación pretende resolver los principales problemas existentes en este tipo de redes:

- **Problemas de seguridad:** los datos son enviados a través de nodos intermedios, los cuales pueden leer información no destinada a ellos, incluso modificarla. También existe la posibilidad de que un atacante se haga pasar por otro usuario, enviando paquetes cuya dirección IP origen no sea la suya. Tampoco existe anonimato, ya que los nodos intermedios pueden leer las direcciones IP que participan en una comunicación.

- **Problema de localización de servicios:** puede ocurrir que un nodo de la red quiera comunicarse con otra máquina en concreto, pero desconozca su dirección IP. En redes con infraestructura, este problema se resolvería utilizando un servidor centralizado de resolución de nombres. Sin embargo, las redes ad-hoc se caracterizan por no requerir ningún tipo de infraestructura fija ni administración centralizada, por lo que no dispondrán de servidores de este tipo.

- **Doble uso de las direcciones IP:** actualmente las direcciones IP se utilizan tanto para enrutar los paquetes como para identificar a los usuarios. Este segundo uso tiene el inconveniente de que cuando un equipo cambia su localización en la red (y por tanto cambia de dirección IP) pierde su identidad y pasa a tener una nueva. Este problema es frecuente en las redes ad-hoc, donde los nodos pueden cambiar con frecuencia de dirección IP.

La implantación del protocolo HIP en una red ad-hoc resolvería los problemas de seguridad citados, a excepción del anonimato. El doble uso de las direcciones IP también quedaría resuelto, ya que HIP proporciona una identidad a cada usuario, independiente de su dirección IP y que no varía aunque se cambie de ubicación o de dirección IP. El anonimato y el problema de la localización del nodo destino serán resueltos a lo largo de este proyecto.

Para alcanzar el objetivo del anonimato será necesario modificar, además del protocolo HIP, el protocolo de encaminamiento dinámico. Medir las prestaciones de todas estas modificaciones también es uno de los objetivos de este proyecto. Se estudiará el posible coste introducido por las modificaciones en el protocolo HIP original y en el protocolo de encaminamiento. También se medirá el coste de utilizar el protocolo HIP modificado en una red ad-hoc.

La aplicación resultante tendrá que ser compatible con la versión original. El usuario podrá indicar en el fichero de configuración (o en la orden de invocación) si desea que la aplicación se comporte como la versión original o como la versión modificada. Esto evita que el usuario tenga que tener instaladas dos aplicaciones en lugar de una.

Finalmente se desea obtener una aplicación fiable, fácil de utilizar, documentada, lista para ser distribuída y que resuelva los problemas citados de las redes ad-hoc.

1.2. Estructura de la memoria

La estructura de este documento es la siguiente. A continuación se hace referencia a algunos trabajos relacionados. En el capítulo 3 se analizan las redes ad-hoc y el protocolo de encaminamiento OLSR. En el capítulo 4 se estudia el protocolo HIP. A continuación, en el capítulo 5 se presenta la propuesta realizada en este proyecto, y que consiste en la integración de los protocolos HIP y OLSR, logrando solucionar los problemas estudiados de las redes ad-hoc y ofreciendo anonimato. En el capítulo 6 se explica en detalle el funcionamiento de la aplicación y cómo configurarla correctamente. El capítulo 7 recoge las pruebas realizadas para verificar el correcto funcionamiento y medir el rendimiento de la aplicación. Finalmente se presentan las conclusiones del proyecto en el capítulo 8.

2. TRABAJOS RELACIONADOS

En los últimos años, el interés por las redes ad-hoc ha aumentado hasta convertirse en un campo de investigación muy activo. Actualmente se trabaja para ofrecer seguridad en este tipo de redes, ya que la naturaleza de la transmisión inalámbrica las hace más inseguras que las redes cableadas. En las redes inalámbricas, cualquier usuario situado próximo a otras máquinas puede recibir tráfico no destinado a él. Además, en las redes ad-hoc los usuarios realizan tareas de encaminamiento, por lo que un atacante puede leer y modificar información destinada a otros usuarios.

La información que se transmite por una red ad-hoc puede ser de dos tipos: tráfico de control de la red ad-hoc y tráfico de datos entre usuarios. El primero de ellos se utiliza para encontrar y mantener actualizadas las rutas de encaminamiento. En [10] se propone una solución para hacer seguro el tráfico de control del protocolo de encaminamiento OLSR. Dicha propuesta añade una firma digital a los paquetes de control para garantizar su integridad y verificar la identidad del emisor.

Muchos de los trabajos publicados recientemente se han centrado en desarrollar protocolos de encaminamiento que garanticen el anonimato. Diremos que una comunicación es anónima si los nodos intermedios no pueden conocer qué dos máquinas participan en ella

ANODR [11] es un protocolo de encaminamiento bajo demanda en el que cada nodo intermedio cifra con su clave los paquetes de descubrimiento de ruta antes de reenviarlos, y realiza el proceso inverso con los paquetes de respuesta. Es decir, cada nodo intermedio añade una capa de cifrado, logrando así anonimato en el descubrimiento de la ruta (los nodos intermedios no sabrán qué dos nodos desean comunicarse). La comunicación posterior se hará utilizando pseudónimos, manteniendo así el anonimato. Sin embargo, ANODR no ofrece seguridad en los datos enviados por los usuarios, por lo que estos pueden ser leídos y modificados por otros nodos de la red.

A3RP [12] también es un protocolo de encaminamiento que ofrece anonimato, pero extiende las características de ANODR incluyendo autenticación en los paquetes de control. Por desgracia, su utilización en redes ad-hoc está limitada porque requiere la existencia de una Autoridad Certificadora que gestione las claves de todos los nodos de la red. El mismo requisito se repite en MASK [13]. En la práctica, este requisito es difícil de conseguir, ya que las redes ad-hoc se caracterizan por no necesitar una administración centralizada. A3RP sólo autentica el tráfico de control y no el tráfico generado por los usuarios.

ASRPAKE [14] soluciona el problema de la Autoridad Certificadora estableciendo grupos de confianza dentro de la red, que utilizarán firmas de grupo. Este esquema requiere que existan relaciones de confianza entre los usuarios de la red, algo más fácil de conseguir que una Autoridad Certificadora.

En cuanto a la movilidad ofrecida por HIP, existe otro protocolo que también permite el cambio de ubicación de los usuarios de forma transparente: Mobile IP (MIP) [20]. Pero este protocolo, a diferencia de HIP, requiere una infraestructura y no podrá utilizarse en una red ad-hoc. En Mobile IP unos routers serán los encargados de redirigir el tráfico desde

la antigua ubicación del nodo nómada hasta la nueva red en la que se encuentre. MIP tampoco ofrece el nivel de seguridad de HIP, ya que los datos sólo se cifrarán para ser transportados desde una antigua ubicación hasta la ubicación actual del nodo nómada, pero no desde el nodo emisor a la antigua ubicación del destino.

Desde 2005, HIP es utilizado con éxito en la fábrica que la compañía Boeing posee en Everett (Washington, EEUU) [21]. En estas instalaciones se construyen los aviones Boeing 777 a partir de las piezas recibidas desde otros lugares (Figura 1). HIP es utilizado para conectar de forma segura los robots encargados de transportar las piezas con los controladores de los robots. Además, los dispositivos móviles empleados por los trabajadores de la planta utilizan HIP para poder cambiar de ubicación de forma transparente (movilidad). Concretamente, estos terminales pueden utilizar indistintamente la red inalámbrica de la fábrica o la conexión 3G/GPRS. Gracias a HIP, el cambio de una red a otra no supondrá la pérdida de la identidad ni de las conexiones existentes.

Figura 1. Fábrica de Boeing donde se ha implantado HIP para ofrecer seguridad y movilidad

Actualmente no existen propuestas para la utilización del protocolo HIP en redes ad-hoc. Las únicas referencias que aparecen en la literatura con respecto al protocolo HIP en redes ad-hoc hablan de redes ad-hoc conectadas a redes fijas, y comentan la necesidad de disponer de un servidor central de resolución de nombres [22]. En concreto se comenta el uso del protocolo HIP para la comunicación con máquinas de la red fija, y no para comunicaciones entre nodos de la red ad-hoc.

3. LAS REDES AD-HOC

3.1. Introducción

Las redes ad-hoc, también conocidas como MANETs, son redes inalámbricas que no requieren ningún tipo de infraestructura fija ni administración centralizada. En ellas los nodos, además de funcionar como estaciones finales en comunicaciones punto a punto, deben ofrecer también servicios de encaminamiento, retransmitiendo paquetes entre aquellas máquinas que no tienen conexión inalámbrica directa.

Las redes ad-hoc deben adaptarse dinámicamente a los continuos cambios que se dan en este tipo de redes: cambios en la posición de los nodos, en la potencia de la señal, en el tráfico de la red y en la distribución de la carga. De entre estas características, los frecuentes cambios en la topología de la red suponen el mayor reto al que deben hacer frente las redes ad-hoc.

En comparación con las redes cableadas, las redes ad-hoc presentan cambios de topología frecuentes e impredecibles debido a la movilidad de sus nodos. Esto impide que en las redes ad-hoc se puedan utilizar los protocolos de encaminamiento desarrollados para redes cableadas. De hecho, para solventar los problemas de estas redes, existe dentro del *Internet Engineering Task Force* (IETF), un grupo de trabajo denominado *Mobile Ad hoc Networking group* (MANET) [16], cuyo objetivo es motivar la investigación en las redes ad-hoc.

Los protocolos de encaminamiento desarrollados para redes cableadas no se adaptan al entorno de las redes ad-hoc. Dichos protocolos envían mensajes de actualización de rutas, ya sea periódicamente o cuando se detecta un cambio en la topología. Estos mensajes generan una elevada sobrecarga incluso en redes con poco tráfico. Este mecanismo hace que en las redes ad-hoc, donde los cambios de topología son muy frecuentes, dichos protocolos produzcan una sobrecarga excesiva.

A continuación se explica brevemente el funcionamiento del protocolo Link-State, uno de los principales protocolos de encaminamiento en redes cableadas. Además, este protocolo fue optimizado para funcionar eficientemente en redes ad-hoc, dando lugar a un nuevo protocolo, conocido como OLSR, al cual se dedica el apartado 3.3. Por último, en el apartado 3.4, se detallan los problemas de seguridad más críticos en las redes ad-hoc.

3.2. Link-State Routing Protocol (LSRP)

LSRP es uno de los principales protocolos de encaminamiento para redes cableadas. La optimización de este protocolo para su uso en redes ad-hoc dio lugar a un nuevo protocolo, conocido como OLSR. El objetivo del protocolo LSRP es que todos los routers de la red tengan una visión global de la red que les permita enrutar correctamente los paquetes.

LSRP es un protocolo proactivo. Esto quiere decir que los routers envían periódicamente información sobre sus vecinos. Esta información se envía por difusión y llega a todos los

demás routers. Esto consume parte del ancho de banda de la red, pero tiene como ventaja que las rutas ya están calculadas en el momento de tener que enrutar un paquete.

A continuación se describe brevemente el funcionamiento del protocolo.

Primero, cada router de la red debe descubrir qué otros routers son vecinos suyos. Nótese que en una red ad-hoc todos los nodos serán routers, pues cumplen funciones de encaminamiento de paquetes. Decimos que 2 nodos son vecinos si pueden comunicarse mutuamente y directamente, es decir, sin necesidad de que sus mensajes sean reenviados por otros nodos intermedios.

Para descubrir a sus vecinos, cada nodo envía un mensaje *Hello*. Los nodos que reciben el mensaje son los vecinos del emisor, y contestan identificándose. Pasado un tiempo, todos los nodos sabrán quiénes son sus vecinos. En ese momento, cada nodo creará un mensaje de control de topología. Dicho mensaje contiene el identificador del emisor y una lista con los identificadores de todos sus vecinos.

A diferencia de los mensajes *Hello*, los mensajes de control de topología son reenviados por toda la red. Esto quiere decir que los nodos que los reciben deberán retransmitirlos por difusión. Como consecuencia pueden aparecer bucles: un mismo nodo retransmite varias veces el mismo mensaje. Para evitarlo, cada mensaje de control lleva un número de secuencia. Este número no se modifica durante el reenvío de los mensajes, pero sí se incrementa cada vez que el emisor genera un nuevo mensaje. Cada nodo deberá mantener un historial con el número de secuencia y el identificador del emisor de los últimos mensajes que ha recibido. Sólo reenviará los mensajes de un determinado emisor si su número de secuencia se ha incrementado con respecto al del historial.

Al cabo de un tiempo, todos los nodos han recibido los mensajes de control de topología de todos los demás nodos. A partir de esos mensajes es posible construir el grafo de la red. En este grafo los vértices representan los nodos, y las aristas unen a los vecinos (los nodos adyacentes). El siguiente paso consiste en que cada nodo calcule los caminos mínimos entré él y cada uno de los nodos de la red. Para ello se puede utilizar el algoritmo de *Dijkstra*, también conocido como el algoritmo de los caminos mínimos. Hecho esto, cada nodo sólo deberá almacenar cuál es el primer salto para llegar a cada uno de los nodos.

Como hemos visto, se trata de un protocolo distribuido, por lo que podría utilizarse en redes inalámbricas ad-hoc. Sin embargo, los mensajes de control de topología, que se envían por broadcast cada vez que se detecta un cambio, suponen una importante sobrecarga. En redes ad-hoc, donde los cambios de topología son muy frecuentes, esta sobrecarga es prohibitiva.

Con el fin de reducir la sobrecarga introducida por la difusión de los mensajes de control de topología, se desarrolló el protocolo OLSR. Sus características y funcionamiento se detallan a continuación.

3.3. Protocolo OLSR

El protocolo *Optimized Link State Routing* (OLSR) [6] es una optimización del algoritmo *Link State* para redes móviles ad-hoc. La idea clave de esta optimización son los llamados *Multipoint Relays* (MPRs).

Los MPRs son los nodos de la red encargados de retransmitir los paquetes broadcast. El número de MPRs que retransmiten un paquete broadcast es siempre menor que el número total de nodos, por lo que esta técnica reduce considerablemente el coste de las inundaciones de paquetes broadcast. Sin embargo, el protocolo sigue proporcionando las rutas óptimas, pues los paquetes broadcast siguen llegando a todos los nodos. Con la utilización de los MPRs se reduce la información redundante que se envía por difusión.

OLSR es un protocolo proactivo. Esto quiere decir que el intercambio de información sobre la topología de la red se realiza periódicamente, en lugar de bajo demanda, como ocurriría en un protocolo reactivo. Los protocolos proactivos tienen la ventaja de ser más rápidos, ya que la topología de la red se conoce previamente y no es necesario calcularla antes de enviar cada paquete. La desventaja de los protocolos proactivos es que requieren que se envíe periódicamente paquetes de control, lo que supone una sobrecarga para la red. La sobrecarga también aparece en los protocolos reactivos, pero sólo cuando el número de peticiones es relativamente alto.

OLSR es adecuado para redes en las que el tráfico se produce entre un gran conjunto de nodos, y no entre un pequeño grupo solamente. Cuanto mayor es la red (en número de nodos y en densidad), mayor es la optimización de OLSR con respecto al protocolo *Link State* clásico.

OLSR está diseñado para trabajar en un ambiente completamente distribuido, sin necesidad de una entidad central. Tampoco requiere que la transmisión de los paquetes del protocolo se realice de forma fiable. Dichos paquetes se envían periódicamente, y se asume que algunos se perderán, algo que ocurre con frecuencia en redes wireless.

3.3.2. MPR

Llamaremos “vecinos en 2 saltos de N” a los vecinos de los vecinos del nodo N. Llamaremos “vecinos estrictos en 2 saltos de N” a los “vecinos en 2 saltos de N” que además no son vecinos de N. Cada nodo puede calcular sus “vecinos estrictos en 2 saltos” del siguiente modo: realizar la unión de los vecinos de sus vecinos (analizando los mensajes *Hello*), y restar al resultado sus vecinos directos.

Cada nodo debe elegir, de entre sus vecinos directos, un subconjunto al que llamaremos MPRs. Con ese subconjunto tendrá que ser posible alcanzar a todos los “vecinos estrictos en 2 saltos”. Además, el número de MPRs tendrá que ser el menor posible (Figura 2).

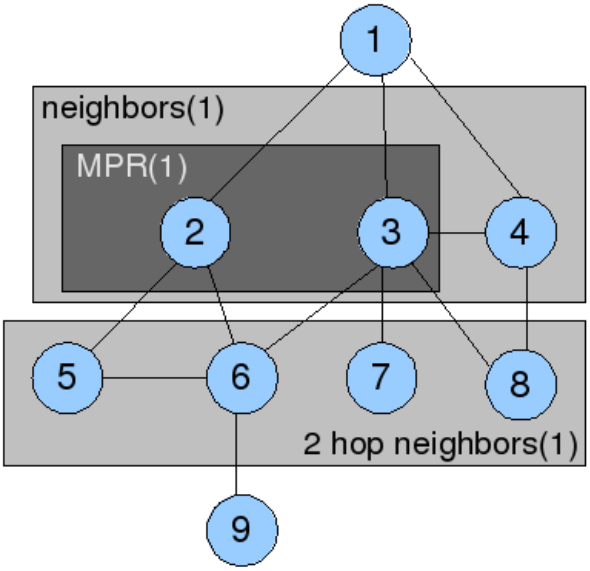


Figura 2. Elección de MPR en el nodo 1 de una red ad-hoc

Los MPRs de un nodo N serán los únicos vecinos de N que reenviarán los mensajes difundidos por N. Las condiciones que deben cumplirse al crear el conjunto de MPRs aseguran que los paquetes enviados por difusión siguen llegando a todos los nodos de la red pero lo hacen necesitando un menor número de reenvíos por parte de los nodos intermedios.

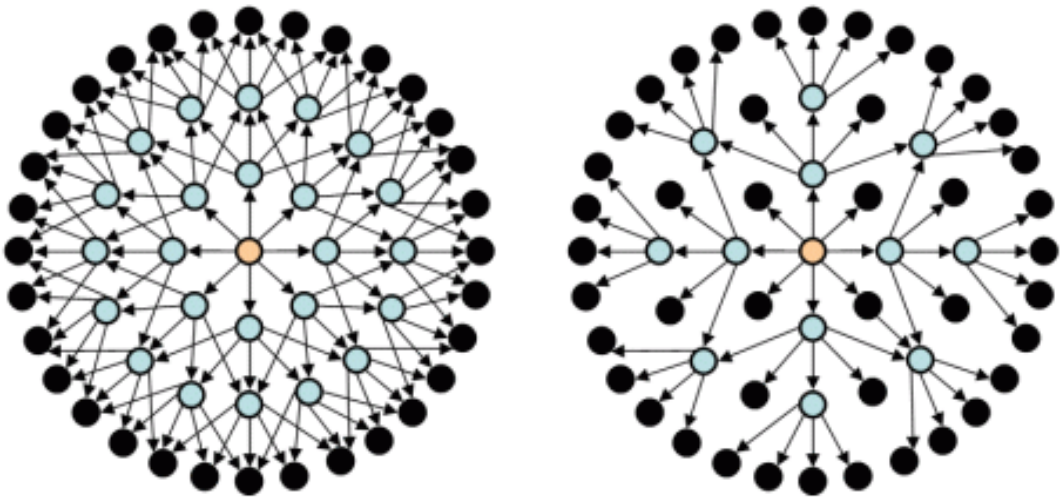


Figura 3. Ejemplo de red ad-hoc sin optimización (izda) y con MPRs (dcha)

En la Figura 3 se representa a la izquierda una red ad-hoc sin la optimización de los MPRs. En la red de la derecha sí que se hace uso de los MPRs. En ambos casos, el nodo central envía un paquete por difusión. En la red de la izquierda, todos los vecinos de dicho nodo reciben el paquete y lo reenvían, también por difusión. Finalmente, todos los nodos de la red han reenviado el paquete por difusión. En cambio, en la red de la derecha, sólo los MPR reenvían el paquete enviado por difusión. Por tanto, el número de nodos que reenvían

el paquete (representados en color claro) es menor al utilizar MPRs, lo que reduce la sobrecarga en la red.

3.3.3. Formato de los mensajes OLSR

Para ser transmitidos por la red, los paquetes OLSR son encapsulados en mensajes UDP (Figura 4). La IANA (Internet Assigned Numbers Authority) ha reservado el puerto 698 para uso exclusivo del protocolo OLSR.

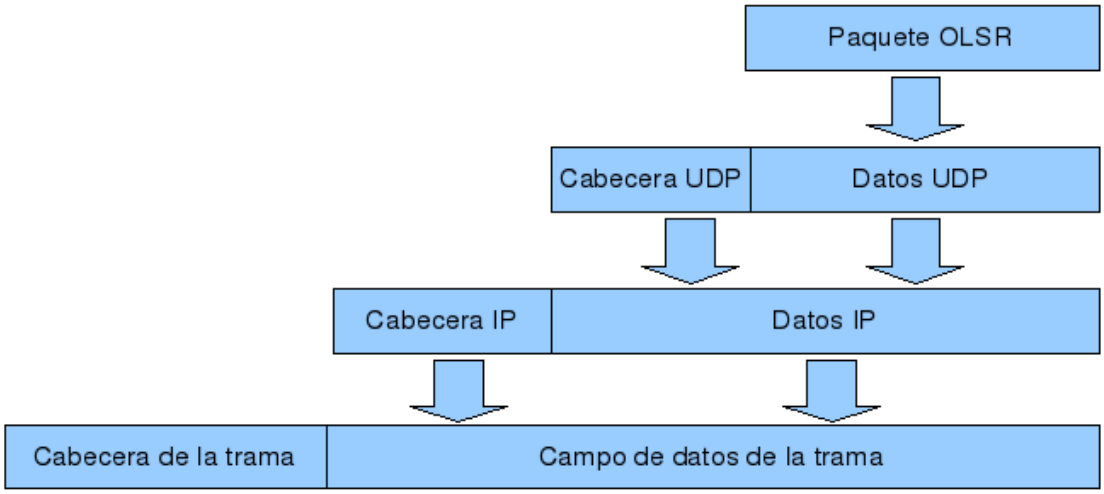


Figura 4. Encapsulación de paquetes OLSR

Los paquetes OLSR pueden contener básicamente dos tipos de información: mensajes de control de topología (TC en inglés), o mensajes HELLO. Todos los paquetes OLSR se envían por difusión, pero sólo los mensajes TC deben ser reenviados por los nodos MPR. Todo este tráfico del protocolo de encaminamiento supone una sobrecarga en la red, y por este motivo se ha trabajado para reducir el número de paquetes OLSR.

Podría pensarse que cada vez que se desea enviar un mensaje OLSR (TC o HELLO), se debe crear un paquete OLSR nuevo. La construcción de dicho paquete conllevaría que éste se encapsulase en un mensaje UDP, que a su vez se encapsularía en un datagrama IP. Este datagrama IP rellanaría el campo de datos de una trama. En una red inalámbrica ad-hoc bajo el estándar IEEE 802.11, el acceso al medio es por competición, y la capa de enlace tendría que esperar a que el medio quedase libre para poder transmitir la trama.

Con el fin de reducir la sobrecarga en la red y los tiempos de espera, el protocolo OLSR permite que varios mensajes OLSR sean encapsulados dentro de un único paquete OLSR. Así, un paquete OLSR tiene la siguiente estructura básica (Figura 5).

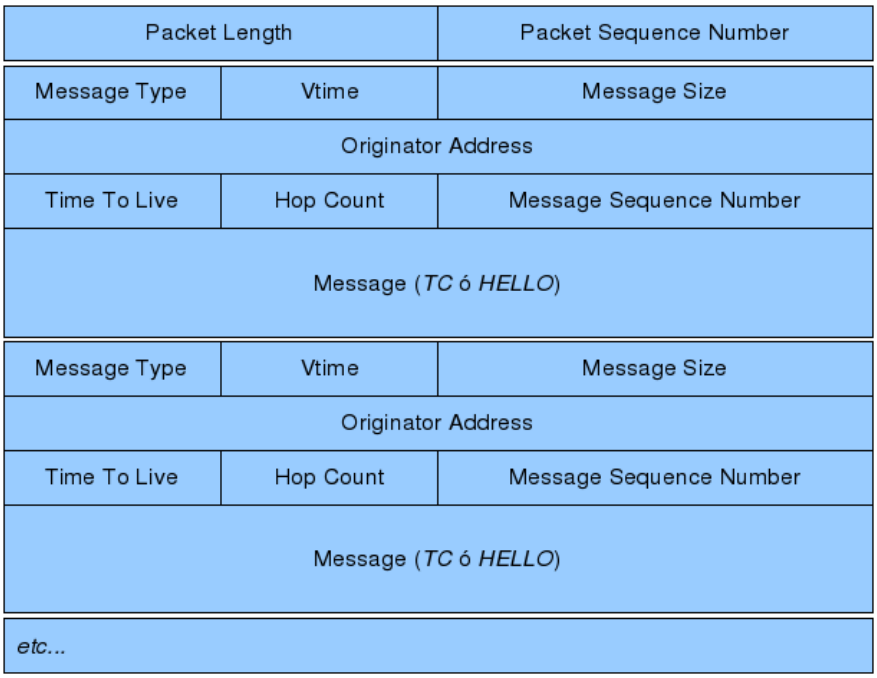


Figura 5. Formato de un paquete OLSR

La cabecera del paquete OLSR está formada sólo por dos campos: el primero indica el tamaño total del paquete y el segundo es un número de secuencia. Tras la cabecera, pueden adjuntarse tantos mensajes OLSR como sea necesario enviar simultáneamente (siempre que éstos quepan en una única trama). También se observa en la Figura 5 que cada mensaje OLSR tiene una cabecera. El significado de los campos de los mensajes OLSR se explica a continuación:

- Message Type: señala si se trata de un mensaje de tipo HELLO, TC (Topology Control) o MID. Los mensajes MID indican que un nodo tiene múltiples interfaces de red. No se ha profundizado en los mensajes MID porque se ha trabajado con nodos que sólo tienen una interfaz de red configurada con OLSR y por tanto no generan mensajes MID.
- Vtime: Este campo indica cuánto tiempo, desde el instante de la recepción, hay que considerar válida la información contenida en el mensaje.
- Message Size: es el tamaño del mensaje en bytes. Se mide desde el principio del campo "Message Size" hasta el final del mensaje.
- Originator Address: contiene la dirección principal del nodo que originalmente generó dicho mensaje OLSR. Este campo no es modificado por los nodos intermedios
- Time To Live: máximo número de saltos que el mensaje será retransmitido. El valor de este campo es reducido en una unidad cada vez que un nodo recibe el mensaje y lo retransmite. No se retransmitirán mensajes con TTL igual a 1 ó 0.
- Hop Count: contabiliza el número de veces que un mensaje ha sido encaminado. Inicialmente su valor es cero.
- Message Sequence Number: cada nodo numera sus propios mensajes mediante este campo. Los demás nodos pueden utilizar este campo para descartar mensajes duplicados.

Mensajes HELLO

Los mensajes OLSR de tipo HELLO tienen el formato que se muestra en la Figura 6. Se ha omitido la ya explicada cabecera común de los mensajes OLSR. En dicha cabecera los mensajes HELLO tienen el campo TLL igual a 1, ya que no tienen que ser reenviados. El significado de los campos de los mensajes HELLO es el siguiente:

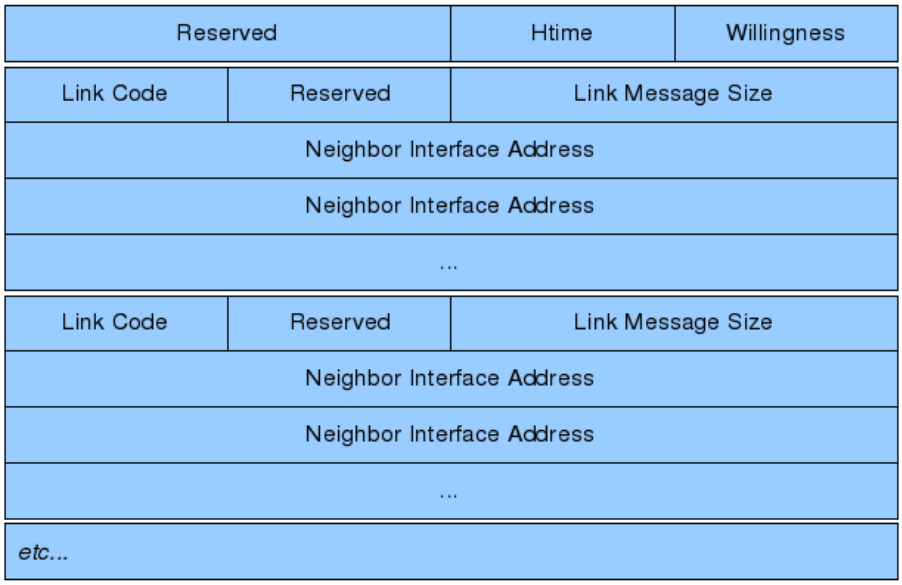


Figura 6. Formato del mensaje OLSR HELLO

- Reserved: este campo debe tener siempre valor cero para cumplir la especificación del protocolo.
- Htime: indica cada cuánto tiempo se envía un mensaje HELLO desde dicho nodo. Si transcurrido ese tiempo no se recibe un nuevo mensaje de este tipo, es posible que la conexión directa entre el emisor del mensaje y el receptor se haya perdido.
- Willingness: informa del grado de disposición del nodo emisor para retransmitir mensajes OLSR de otros nodos. Según este valor, un nodo podrá ser elegido como MPR siempre, nunca, o a veces.
- Link Message Size: es el tamaño, medido en bytes, desde el principio de este campo hasta el inicio del siguiente campo "Link Code".
- Neighbor Interface Address: dirección de un vecino. Las características del enlace con dicho vecino se extraen del campo "Link Code".
- Link Code: contiene información sobre los enlaces con los vecinos que se listan a continuación. Concretamente, los 2 bits de menor peso indican el tipo de enlace entre el emisor del mensaje y los vecinos a los que precede este campo. Los tipos de enlace contemplados son "no especificado", "enlace asimétrico", "enlace simétrico", y "enlace perdido". El tercer y cuarto bit de menor peso especifican el tipo de dichos vecinos, que puede ser "vecino alcanzado simétricamente", "vecino es MPR" y "no vecino". El resto de bits deben permanecer a cero.

Tal y como se observa en la Figura 6, los vecinos de características iguales (tipo de enlace y tipo de vecino) se agrupan bajo un "Link Code" y "Link Message Size" común.

Mensajes TC

Los mensajes de Control de Topología o TC tienen el siguiente formato (Figura 7). El significado de cada campo se explica a continuación.

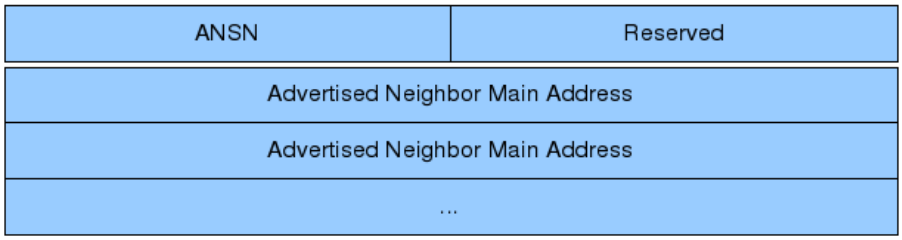


Figura 7. Formato del mensaje OLSR TC

- ANSN (Advertised Neighbor Sequence Number): Cada vez que los vecinos de un nodo cambian (se descubren nuevos vecinos o se pierden otros), dicho nodo incrementa el valor de este campo en los mensajes que genera. De este modo, cuando otro nodo recibe el mensaje, puede saber si la información recibida es válida o, por el contrario, ya ha recibido mensajes más actuales (ANSN mayor) desde el mismo nodo origen. Equivaldría a un número de secuencia, salvo que no es necesario incrementarlo si los vecinos no cambian.
- Advertised Neighbor Main Address: Contiene la dirección principal de un nodo vecino. Todos los mensajes TC deben contener las direcciones principales de todos los vecinos del nodo que origina el mensaje. Si el tamaño máximo de la trama impide incluir todas las direcciones, se generarán tantos paquetes OLSR como sea necesario.
- Reserved: al igual que en otras ocasiones, este campo debe tener siempre valor cero para cumplir la especificación del protocolo.

Tras explicar el formato de los paquetes y de los mensajes OLSR, cabe enfatizar en el hecho de que muchos de los paquetes OLSR enviado/recibidos contendrán más de un mensaje OLSR, y esto supone un mejor aprovechamiento del medio y una reducción de la sobrecarga en la red.

3.3.4. Funcionamiento

Al igual que ocurre en el protocolo LSRP, con OLSR cada nodo debe conocer qué nodos son sus vecinos. Para ello, cada nodo envía periódicamente un mensaje OLSR de tipo HELLO. Estos mensajes se envían por difusión, pero indicando un TTL igual a 1. Es decir, todos los vecinos de un nodo recibirán el mensaje HELLO, pero no lo reenviarán. Los mensajes HELLO contienen información sobre los vecinos del nodo emisor. De esta forma los nodos vecinos se intercambian información.

Puede ocurrir que un nodo A reciba los HELLO de un nodo B, pero B no reciba los de A. El nodo A puede detectar esta situación, ya que en los mensajes HELLO de B no aparecerá la dirección de A en la lista de vecinos.

Transcurrido un tiempo, todos los nodos habrán recibido los HELLO de sus vecinos. Analizando el contenido de estos mensajes, cada nodo calcula su lista de vecinos a 2 saltos

(es decir, los nodos alcanzables desde sus vecinos). Para obtener la lista de vecinos estrictos a 2 saltos, restará al conjunto anterior la lista de sus vecinos a 1 salto.

En la Figura 8, el nodo A recibe los mensajes HELLO de B, C, E y F (representados en color rojo). Estos son sus vecinos a 1 salto. Tras analizar estos mensajes, obtiene la lista de vecinos a 2 saltos (representados en azul): D, E, F, G y H. Puede ocurrir, como en este ejemplo, que algunos vecinos de A sean también vecinos entre ellos (nodos E y F). La lista de vecinos estrictos a 2 saltos será: D, G y H.

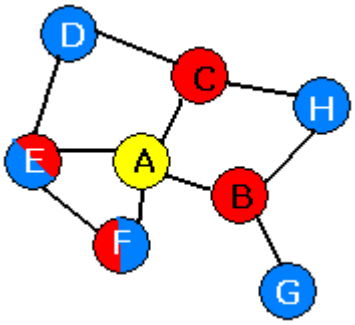


Figura 8. Ejemplo de red ad-hoc

A continuación, cada nodo debe obtener su lista de MPRs. Los MPRs serán los únicos vecinos a 1 salto que reenviarán los mensajes enviados por difusión. El número de MPRs será lo menor posible, pero garantizará que los mensajes difundidos sigan llegando a todos los nodos de la red. En la Figura 8, los vecinos escogidos por A para actuar como MPRs serán B y C, o B y E (según la implementación que se haga del algoritmo de selección de MPRs). En cualquier caso, no se escogería simultáneamente como MPRs a B, C y E, porque no es el subconjunto mínimo.

Cada nodo informa, por medio de los mensajes HELLO, de sus vecinos escogidos como MPRs. Si no informara, dichos vecinos MPR no sabrían que tienen que reenviar los mensajes difundidos por el citado nodo.

Paralelamente al envío periódico de mensajes HELLO, cada nodo envía mensajes de Control de Topología (TC). Estos mensajes, enviados por difusión, serán reenviados por los nodos MPR. Cada mensaje TC contiene la dirección del nodo que lo origina, así como las direcciones de sus vecinos.

Transcurridos unos segundos, todos los nodos habrán recibido el mensaje TC generado por cada uno de los nodos de la red. A partir de dichos mensajes, cada nodo construirá un grafo completo de la red ad-hoc. Del mismo modo que ocurre en el protocolo LSRP, cada nodo calculará los caminos mínimos entré él y el resto de nodos de la red. Para ello se utiliza el algoritmo de *Dijkstra*, conocido como el algoritmo de los caminos mínimos.

Tras calcular los caminos mínimos, cada nodo añadirá en sus tablas de encaminamiento una entrada por cada nodo de la red, a excepción de sus vecinos a 1 salto (con ellos la comunicación es directa). Cada entrada en la tabla indicará cuál es el primer salto para llegar a un determinado nodo. Esta información será consultada cuando se quiera

establecer comunicación con un nodo que no es vecino a 1 salto, o cuando haya que encaminar la comunicación entre otros 2 nodos.

En la Figura 9, el nodo A tendrá una entrada en su tabla de encaminamiento que le indicará que para comunicarse con el nodo D tendrá que encaminar la información hacia el nodo B. Por su parte, el nodo B tendrá una entrada en su tabla para encaminar el tráfico con destino D a través de C. El nodo C enviará la información a D directamente, sin consultar ninguna entrada en la tabla.

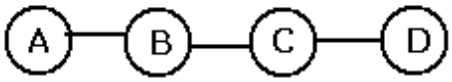


Figura 9. Ejemplo de red ad-hoc

3.4. Seguridad en redes ad-hoc

A diferencia de las redes cableadas, que pueden hacerse seguras hasta un alto grado, en las redes inalámbricas las intrusiones son más probables. En una red cableada, el número de puntos desde los que un intruso puede conectarse es limitado. Además, el acceso a estos puntos suele estar restringido. Por ejemplo, en una oficina con red local cableada, un intruso tendría que acceder primeramente al interior de las oficinas, ya que las conexiones a la red se encuentran allí. Además, si la infraestructura de la red es la adecuada, cada máquina sólo recibirá su tráfico, y no podrá capturar paquetes de otras comunicaciones.

Sin embargo, en las redes inalámbricas las intrusiones no pueden evitarse tan fácilmente. En este tipo de redes, cualquier máquina dentro del radio de cobertura de la red recibe paquetes de otras comunicaciones. El comportamiento de una máquina no atacante consistirá en descartar los paquetes no dirigidos a ella, pero un intruso podría capturar sin dificultad los datos intercambiados entre otras 2 máquinas.

En las redes inalámbricas ad-hoc, los ataques afectan tanto a los datos enviados como a los protocolos de encaminamiento que se utilizan. A continuación nos centraremos en las redes inalámbricas ad-hoc que utilizan OLSR como protocolo de encaminamiento.

En las redes ad-hoc, las comunicaciones se encaminan a través de otros nodos intermedios. En las redes con infraestructura, estos nodos intermedios son los routers, máquinas cuya única función es encaminar los paquetes hacia su destino. Sin embargo, en las redes ad-hoc, estos nodos intermedios son usuarios, y podrían capturar la información. Es decir, la **confidencialidad** de los datos no está garantizada. Además, las máquinas intermedias saben qué dos direcciones IP participan en cada comunicación.

Tampoco se asegura la **integridad** de los datos en OLSR. Se dice que un paquete está íntegro cuando el destinatario recibe una copia exacta del paquete que generó el nodo origen. En una red ad-hoc con OLSR, un usuario intermedio podría modificar el contenido de uno de los paquetes que debe encaminar, recalcular los campos *checksum* del mismo, y reenviar el paquete modificado. El destinatario del paquete no podría verificar si éste ha sido modificado. Este ataque, en el que un tercer usuario puede leer, insertar y modificar paquetes de una comunicación entre otros 2 nodos, es conocido como Man-in-the-Middle (MitM).

Los ataques de suplantación de identidad también pueden aparecer en estas redes. Ocurren cuando un atacante envía paquetes haciéndose pasar por otro nodo de la red. En las redes IP, esto se consigue rellenando el campo "IP origen" de los paquetes IP con la dirección de otro nodo. En el protocolo IP no existen mecanismos de **autenticación**, por lo que tampoco en una red ad-hoc con OLSR es posible verificar que el remitente del paquete (campo "IP origen") es real.

Las redes ad-hoc son especialmente vulnerables a los ataques de Denegación de Servicio (en inglés DoS, Denial of Service). Estos ataques causan que un recurso sea inaccesible durante un periodo de tiempo. En una red ad-hoc, el atacante intentaría consumir un ancho de banda elevado para que los demás nodos no pudieran acceder al medio, o intentaría sobrecargar computacionalmente el resto de nodos. Normalmente en este tipo de redes, este ataque se lleva a cabo mediante el envío de paquetes broadcast. Estos paquetes inundan toda la red, y muchos nodos de la red deben invertir recursos en leerlos y posiblemente reenviarlos. En las redes con infraestructura también hay ataques DoS, pero concretamente los ataques con broadcast no suponen un peligro, ya que la propagación de estos paquetes está limitada a pequeñas áreas de la red. En las redes ad-hoc pueden contenerse estos ataques limitando la frecuencia de las inundaciones broadcast.

Los ataques de retraso y repetición (Delay and Replay Attacks) suelen ser utilizados cuando sí existen mecanismos que controlan la integridad de los paquetes. En tal contexto, este ataque consiste en capturar paquetes de una comunicación y reenviarlos pasado un tiempo. Al no ser modificados, los paquetes superarán las comprobaciones de integridad en el nodo destino. El objetivo de este ataque es que el destinatario dé por válida una información desfasada, y por tanto errónea. Para detectar estos ataques son necesarios mecanismos que añadan una marca de tiempo en los paquetes.

Por otro lado, el propio protocolo de encaminamiento también es vulnerable a ataques. Estos protocolos parten de la idea de que las máquinas no actuarán maliciosamente. Sin embargo, los nodos pueden comportarse de forma diferente a la especificada en el protocolo, perjudicando al resto de la red. Algunos de los comportamientos incorrectos en el protocolo de encaminamiento son:

- Comportamiento egoísta: ocurre cuando un nodo no colabora encaminando paquetes. Por ejemplo, no reenvía los paquetes que recibe, o modifica los paquetes de control para no tener que encaminar tráfico de otros nodos.

- Comportamiento malicioso: un nodo genera intencionadamente paquetes de control erróneos, o reenvía paquetes con topologías obsoletas. Un ataque común consiste en anunciar que es vecino de todos los demás nodos. De esta forma, la mayor parte del tráfico se encamina hacia el nodo malicioso, lo que le permite interceptar más comunicaciones.

Para corregir las vulnerabilidades propias de un protocolo de encaminamiento como OLSR, existe una extensión que proporciona seguridad al tráfico relacionado con el protocolo (paquetes de control de topología), pero no a los datos que se envían los usuarios. Dicha extensión garantiza la integridad de los mensajes de control, pero no su confidencialidad [10].

En conclusión, sigue siendo necesaria una solución que ofrezca confidencialidad, autenticación, integridad y garantice la actualidad de los paquetes en una comunicación entre 2 nodos de una red ad-hoc. También hay que tener en cuenta los posibles ataques DoS que pueda sufrir dicha solución. Además, puede resultar interesante en algunos escenarios que las máquinas intermedias no puedan saber qué dos máquinas participan en la comunicación que están encaminando (anonimato).

En la Figura 10 se muestran algunas de estas vulnerabilidades. Los nodos A y B (en azul) participan en una comunicación. El nodo de la derecha (en rojo) realiza un ataque de suplantación de identidad, ya que el protocolo IP no garantiza la autenticación de los paquetes. El nodo que se encuentra entre A y B (también en rojo) realiza un ataque de Hombre en Medio (MitM): es capaz de leer los paquetes (no hay confidencialidad) y de modificarlos (no se garantiza su integridad). Tampoco hay anonimato, ya que el atacante sabe qué dos direcciones IP se están comunicando.

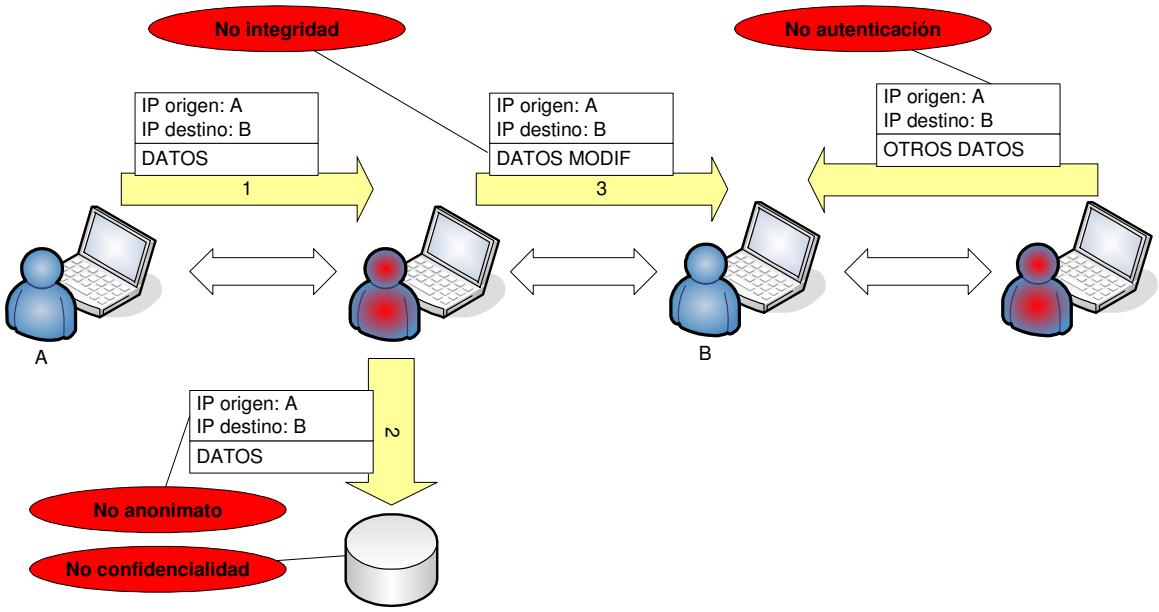


Figura 10. Vulnerabilidades en una red ad-hoc

4. Protocolo HIP

Internet se compone de equipos finales (end-points), una infraestructura que los intercomunica y una serie de servicios que ofrecen algunos end-points. Las direcciones IP, que se diseñaron para establecer una jerarquía con la que poder enrutar correctamente los paquetes, actualmente se utilizan también para identificar los end-points.

Este segundo uso tiene el inconveniente de que cuando un equipo cambia su localización en la red (y por tanto cambia de dirección IP) pierde su identidad y pasa a tener una nueva. Actualmente este problema también se da sin cambiar físicamente de posición, ya que los proveedores de Internet suelen ofrecer direcciones IP dinámicas. Además, en redes ad-hoc en las que no hay servidor DHCP es posible que un host se conecte de forma intermitente con diferentes direcciones IP. En la Figura 11 se muestra un error devuelto por un servicio web tras realizar un cambio de dirección IP en el cliente.

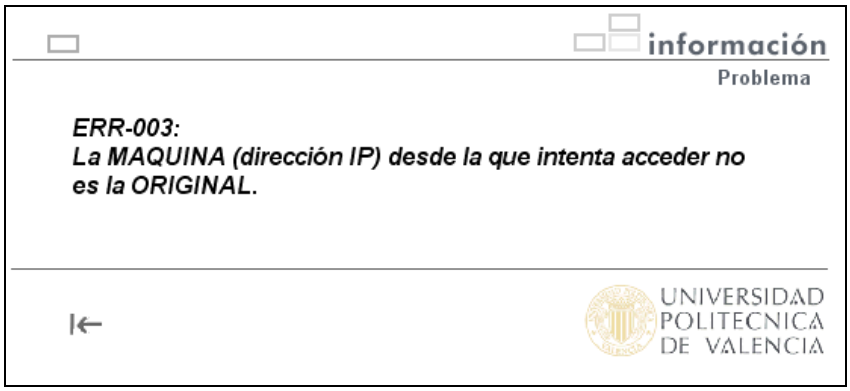


Figura 11. Error devuelto por servidor web tras un cambio de IP

El protocolo HIP (Host Identity Protocol) [2] propone una nueva capa en la pila de protocolos, entre los niveles de red y de transporte. Este nuevo nivel proporciona autenticación, con independencia de la dirección IP del host. Con esta nueva capa, la función de las direcciones IP pasa a ser únicamente la de enrutar los paquetes. Esta nueva pila de protocolos se muestra en la Figura 12.

Además, con el protocolo HIP se establece una asociación segura entre los dos hosts que participan en la comunicación. Esta asociación, que se crea como paso previo al intercambio de datos, proporciona autenticación, integridad, confidencialidad de los datos enviados y es resistente a ataques de Denegación de Servicio (DoS) y a ataques de reenvío de paquetes. Las asociaciones HIP pueden reutilizarse durante un tiempo, por lo que no es necesario crear una nueva asociación cada vez que dos nodos quieren comunicarse.

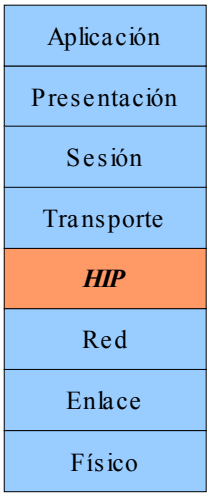


Figura 12. Pila de protocolos ISO/OSI con la nueva capa HIP

HIP también se encarga de que los usuarios no pierdan la identidad que poseen aunque cambien de ubicación en la red o cambien de dirección IP. Estos cambios son totalmente transparentes para el usuario si utiliza HIP.

Para el cifrado de los datos, HIP utiliza el protocolo ESP [3]. Este protocolo garantiza la confidencialidad e integridad de los datos, así como la autenticación. El protocolo ESP cifra los datos utilizando una clave simétrica, pero no especifica cómo acuerdan esta clave las dos partes implicadas en la comunicación. La asociación HIP se utiliza para este fin.

4.1. HIP, HIT y LSI

Cada host que implemente HIP tendrá al menos una identidad. Para crear una identidad se deberá generar un par de claves, una pública y otra privada, mediante criptografía asimétrica. HIP define su uso con el sistema de cifrado asimétrico RSA [15], aunque en un futuro podrían considerarse nuevos algoritmos. El identificador del host se obtiene a partir de la clave pública. Para hacer el protocolo HIP independiente del algoritmo de encriptación y conseguir un identificador de longitud fija, a la clave pública se le aplica una función Hash. De este modo se consigue una cadena de longitud fija 128 bits, denominada HIT (Host Identity Tag), y que aparecerá en la cabecera de los paquetes HIP para identificar al emisor y al receptor.

La longitud de los HIT coincide con la longitud de las direcciones IPv6, lo que hace a HIP compatible con las librerías desarrolladas para IPv6. Por ejemplo, una aplicación que haga un ping a una dirección IPv6 especificada, podría recibir en el campo 'dirección' un HIT. El demonio HIP que se ejecuta en la máquina, detectaría que se intenta enviar un paquete, y realizaría los pasos necesarios para obtener la dirección IP del host receptor, y así establecer la conexión HIP.

En la actualidad, las direcciones más comúnmente usadas son las direcciones IPv4, de longitud 32 bits. Para poder realizar pruebas con el protocolo HIP hasta que se desarrollen versiones de las librerías para IPv6, y para permitir la coexistencia de IPv4 y HIP, se ha alcanzado una solución de compromiso llamada Local Scope Identifier (LSI). LSI es un

identificador de 32 bits que se construye directamente a partir de un HIT. Su tamaño le permite ser utilizado en lugar de la dirección IPv4 en aplicaciones y librerías ya existentes. El demonio HIP se encargará de detectar los paquetes que pretenden enviarse por la red a una LSI en lugar de a una dirección IPv4, obtendrá la dirección IP de la máquina destino y establecerá la conexión HIP.

4.2. Convivencia entre IPv4 y HIP

La convivencia entre IPv4 y LSI es resuelta de la siguiente forma. Se asumirá que todas las direcciones IP en el rango 1.0.0.0/8 son en realidad LSI, y los paquetes que intentan enviarse a dichas direcciones tendrán que ser capturados por el demonio HIP, que buscará la dirección IP real del destinatario y establecerá la asociación HIP.

Las direcciones IP comprendidas entre 1.0.0.0 y 2.255.255.255, pese a ser direcciones IP válidas, están reservadas por la IANA (Internet Assigned Numbers Authority) [9] y actualmente no están en uso. Por tanto, no habrá ningún conflicto entre los paquetes que deben ser enrutados hacia Internet y los paquetes con destino 1.0.0.0/8 que deben ser manejados por el demonio HIP.

El protocolo HIP no especifica cómo construir el LSI. La implementación del protocolo HIP con la que se ha trabajado en este proyecto (OpenHIP [17]) lo construye copiando los 24 bits de menor peso del HIT en los 24 bits de menor peso del LSI. Los 8 bits de mayor peso del LSI serán siempre 00000001.

El demonio HIP intercepta los paquetes IP con una dirección destino en el rango 1.0.0.0/8, y también las consultas DNS con el sufijo ".hip" (por ejemplo, www.upv.es.hip). A continuación, busca concordancias en el fichero local de identidades conocidas. En la implementación OpenHIP, este fichero se ubica en `/usr/local/etc/hip/known_host_identities.xml`. La Figura 13 muestra un fragmento de dicho fichero.

```
<host_identity alg="RSA" alg_id="5" length="128">
  <name>un-dominio.com-1024</name>
  <addr>81.124.2.201</addr>
  <HIT>2001:19:cae1:2f79:8f5b:d8c6:4a2f:c1af</HIT>
  <LSI>1.23.55.233</LSI>
</host_identity>
```

Figura 13. Fragmento del fichero de identidades conocidas

Este fichero puede contener, para cada identidad HIP conocida (`<host_identity>`), su nombre de dominio (si tiene), su dirección IP, su HIT y su LSI. Si para alguno de los hosts se conoce su dominio pero no su dirección IP, el demonio HIP intentará obtenerla por consulta DNS al inicio de su ejecución. Si no se conoce el LSI, éste se calculará de forma directa a partir del HIT.

Cuando el demonio HIP intercepta una consulta DNS para *un-dominio.com.hip*, busca el dominio *un-dominio.com* en el fichero anterior. Si aparece junto con su LSI y la aplicación

que realiza la consulta está esperando una dirección IPv4, se le devolverá el LSI; si aparece junto con su HIT y la aplicación espera una dirección IPv6, se le devolverá el HIT. Si en el fichero no se encuentra concordancia, se intenta realizar una consulta al servidor DNS HIP. Este servidor es similar a un servidor DNS, pero en lugar de almacenar las correspondencias entre nombres de dominio y direcciones IP, almacena la relación entre nombres de dominios y su HIT. Si no hay servidor DNS HIP, se devuelve el mensaje NXDOMAIN (non-existant domain). Este comportamiento se resume en la Figura 14.

```
IF (un-dominio.com aparece en el fichero) THEN
   IF (parece junto con su LSI) THEN
         devuelve a la aplicación el LSI
   ELSE
         devuelve 1.X.X.X (X.X.X son 24 bits menor peso del HIT).
ELSE
   IF (hay servidor DNS HIP [parámetro dns_server en hip.conf])
         hace consulta.
   ELSE
         devuelve NXDOMAIN (non-existant domain).
```

Figura 14. Comportamiento de HIP al interceptar consulta DNS para un-dominio.com.hip

Cuando el demonio HIP intercepta un paquete IP con dirección IP destino 1.0.0.0/8, busca esa dirección en los campos LSI del fichero local de identidades conocidas (`known_host_identities.xml` en OpenHIP). Si aparece junto a su HIT y dirección IP real, se intenta establecer la asociación HIP. Una vez establecida la asociación HIP, se enviará el paquete interceptado. Si aparece en el fichero el LSI pero no junto a su HIT, se intentará establecer la asociación en modo oportunista. El modo oportunista es aquel que intenta establecer una asociación HIP con un host cuya identidad se desconoce. El host remoto podrá aceptar o rechazar las solicitudes oportunistas. También puede ocurrir que en el fichero no aparezca el LSI. En este caso podría hacerse una consulta a un servidor DHT. Este servidor almacena las correspondencias entre LSI y dirección IP actual. Este comportamiento aparece esquematizado en la Figura 15.

```
IF (1.X.X.X aparece en el fichero junto a su dirección IP) THEN
   IF (parece junto con su HIT) THEN
         Iniciar asociación HIP.
   ELSE
         Se intenta modo oportunista.
ELSE
   Puede hacerse consulta a DHT: IP = get(LSI)
```

Figura 15. Comportamiento de HIP al interceptar paquete IP con IP_destino = LSI

A continuación se expone con un ejemplo el funcionamiento descrito (Figura 16). Supongamos que un usuario quiere acceder a una página web desde su navegador. Este usuario aparece en la parte superior de la figura, mientras que el servidor lo hace en la inferior. Si se utilizase el protocolo HIP, la página web tendría una url de la forma http://un-dominio.com.hip (finalizado en .hip), o si se desea acceder mediante la dirección IP, http://1.X.X.X. El usuario teclearía una de las 2 opciones en su navegador (paso 1). Automáticamente, el demonio HIP capturaría la petición HTTP (o previamente la petición

DNS, si se ha introducido http://un-dominio.com.hip) y buscaría concordancias en el fichero de identidades conocidas (paso 2). En el caso de disponer en dicho fichero del identificador HIT del destino y de la dirección IP actual de éste, se iniciaría la asociación HIP entre el usuario y el servidor (paso 3). Esta asociación requiere el envío de 4 paquetes que se explicarán más adelante. Tras establecer la asociación HIP, la petición HTTP se envía cifrada (paso 4). El demonio HIP del servidor descifra la petición y la dirige a la aplicación servidor web (paso 5). La respuesta HTTP (paso 6) es cifrada y enviada por el demonio HIP del servidor (paso 7). El demonio HIP del usuario descifra la información recibida y el usuario finalmente visualiza la página web (paso 8). Las posteriores comunicaciones entre ambas máquinas reutilizarán la asociación HIP ya establecida (en paso 3).

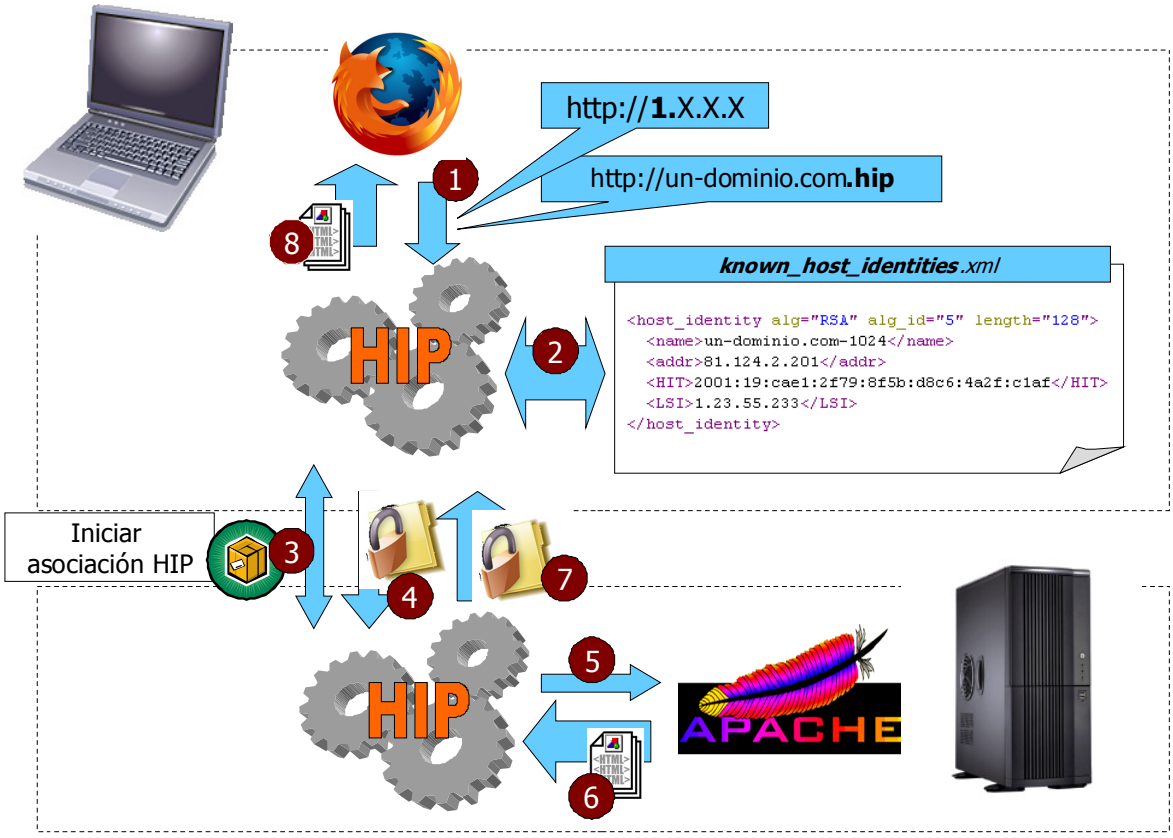


Figura 16. Ejemplo de aplicación de una asociación HIP

4.3. Asociación HIP

Para establecer una asociación HIP entre 2 hosts se envían 4 paquetes, denominados I1, R1, I2 y R2. Todos los paquetes HIP tienen una cabecera común, en la que se indica el tipo de paquete, el identificador de la máquina origen (*HIT_origen*) y el identificador del la máquina destino (*HIT_destino*).

A continuación se explica detalladamente el contenido de los paquetes involucrados en la asociación HIP. Llamaremos *Host1* y *Host2* a los 2 hosts que forman la asociación HIP, siendo el primero de ellos el que inicia la conexión.

4.3.1. Paquete I1

Inicialmente, *Host1* envía el paquete I1. Este paquete contiene básicamente el HIT del emisor y del receptor (*Host1* y *Host2* respectivamente). Cuando *Host2* recibe I1, comprueba que *HIT_destino* le identifica. Si no es así, borra directamente el paquete I1. Puede ocurrir que *Host1* y *Host2* envíen simultáneamente un paquete I1 a la otra parte, porque ambos quieran establecer una asociación HIP con el otro y en el mismo instante. Para que sólo prospere una de las 2 asociaciones, se hace lo siguiente: si un host A ha enviado I1 al host B, y recibe otro I1 de B, sólo contestará si *HIT_origen* es mayor que *HIT_destino*. De esta forma sólo contesta uno.

También puede ocurrir que el paquete I1 llegue con el campo *HIT_destino* vacío. Esto quiere decir que el emisor está intentando establecer una asociación con una máquina de la cual desconoce su identidad. Esto se conoce como modo oportunista. En este caso el receptor debe elegir si acepta dicho modo y contesta, o elimina el paquete directamente.

La Figura 17 contiene una captura de pantalla del analizador de protocolos de red Wireshark [19]. En ella aparece destacado el paquete I1 de una asociación HIP.

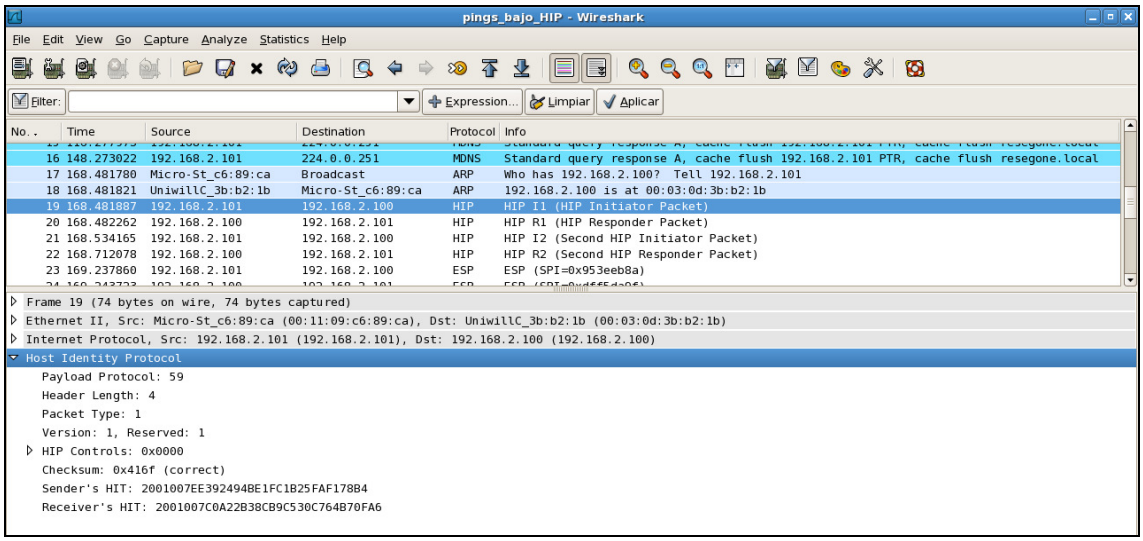


Figura 17. Paquete I1 visualizado con la herramienta wireshark

4.3.2. Paquete R1

El paquete R1 es la contestación a I1, y se enviará desde *Host2* hacia *Host1*. La dirección IP a la que se enviará será la IP origen del paquete IP que contenía a I1. El campo *HIT_destino* de R1 será el *HIT_origen* leído de I1. El *HIT_origen* será el *HIT_destino* que aparecía en I1, excepto en el modo oportunista, en el que será *Host2* quien decida cuál de sus identidades utilizar en *HIT_origen*.

R1 contiene un puzzle que *Host1* debe resolver. Se trata de una técnica para evitar ataques de denegación de servicio (DoS): para que la asociación HIP se complete es necesario que quien la inicia resuelva el puzzle, y esto le supone un coste computacional grande al posible atacante. Los puzzles están diseñados para que existan K niveles de dificultad, con $0<K<20$, siendo el nivel 1 el más fácil. *Host2* podrá elegir el nivel de dificultad del puzzle que va a ofrecer, en base a la confianza que tenga en *Host1*. Por ejemplo, si *Host1* está

intentando el modo oportunista o si ha resuelto fallidamente varios puzzles (posible ataque), la dificultad debería aumentar.

La forma de resolver el puzzle es la siguiente: *Host2* indica en el paquete R1 un número "I", una dificultad "K" y el tiempo límite en el que la solución del puzzle será aceptada. *Host1* deberá encontrar el valor "J" tal que los "K" dígitos de menor peso del resultado de la expresión E sean cero:

$$E = RHASH(I \mid HIT\_origen \mid HIT\_destino \mid J)$$

Donde el símbolo "|" denota concatenación, y RHASH es una función de resumen.

Además del puzzle, R1 contiene la clave pública del *Host2*, la firma digital del paquete, una lista con los modos de cifrado disponibles, y el primer paso del algoritmo Diffie-Hellman [4].

La firma digital es generada por *Host2* aplicando una función de resumen al resto del paquete y cifrando el resultado con su clave privada (K^-_{host2}). *Host1*, al recibir el paquete, descifrará la firma con la clave pública de *Host2* (K^+_{host2}, recibida en ese mismo paquete). El resultado obtenido tendrá que ser igual al resultado de la función resumen sobre el resto del paquete recibido.

A continuación se indica la igualdad que debe cumplirse en los paquetes R1 válidos.

$$\text{Firma R1} = K^-_{host2}(\text{HASH(paquete)})$$

$$K^+_{host2}(\text{Firma R1}) = \text{HASH(paquete)}$$

Esta firma garantiza que el paquete ha sido creado por el propietario de clave pública contenida en el mismo. Y como a partir de la clave pública es posible obtener directamente el HIT (mediante una función resumen), la firma autentica el paquete respecto al campo *HIT_origen*.

La lista de modos de cifrado permitidos, también incluida en el paquete R1, hace más flexible el protocolo HIP, de modo que puedan ir añadiéndose nuevas implementaciones de futuros algoritmos de cifrado. *Host1* deberá elegir una de las opciones que le indica *Host2* en R1.

Cuando un paquete R1 llega a *Host1*, éste comprobará que llega desde un host (*HIT_origen*) al que ha enviado I1 recientemente. En el caso de que *Host1* hubiese enviado I1 en modo oportunista, la comparación se hará con *IP_origen*. Si *Host1* ya ha recibido R1 pero aún no ha respondido con I2 (estado *I1_SENT*), debe seleccionarse el R1 con mayor *R1_COUNTER*. Si *Host1* ya había recibido R1 y había contestado con I2 (estado *I2_SENT*), se debe volver al estado *I1_SENT* y reenviar I2 (como respuesta al paquete R1 con mayor *R1_COUNTER*).

El campo *DIFFIE_HELLMAN* del paquete R1 se utiliza para acordar una clave privada de sesión entre los dos hosts. Diffie-Hellman es el nombre de un protocolo que permite, con el envío de sólo 2 mensajes, acordar una clave secreta entre 2 máquinas. La clave secreta pactada con Diffie-Hellman no puede ser descubierta por un atacante, aunque éste

intercepte los dos mensajes del protocolo. Sin embargo, Diffie-Hellman es vulnerable si no se utiliza conjuntamente con otro protocolo que ofrezca autenticación. Los 2 mensajes del protocolo Diffie-Hellman están contenidos en los paquetes R1 y I2. A modo de anexo, se incluye más información sobre este protocolo (ANEXO II: Algoritmo Diffie-Hellman).

Para concluir la explicación del paquete R1, hay que señalar que estos paquetes no son construidos cada vez que se recibe un paquete I1 válido, sino que son precalculados periódicamente. De esta forma, el protocolo HIP es resistente a ataques de Denegación de Servicio, ya que responder a un paquete I1 con un paquete R1 precalculado tiene un coste computacional mínimo. En cambio, el atacante deberá resolver el puzzle contenido en R1, y esto sí supone un importante coste computacional.

En la Figura 18 se observan los parámetros de un paquete R1. El campo *HOST_ID* contiene la clave pública de *Host2*, y los campos *TRANSFORM* informan de los modos de cifrado disponibles.

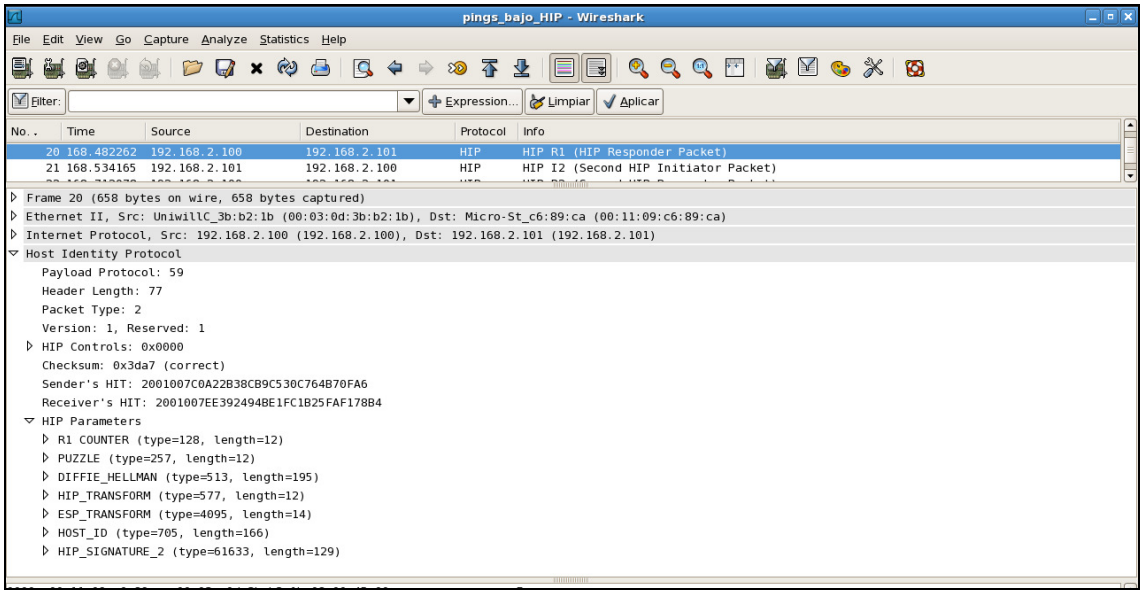


Figura 18. Paquete R1 visualizado con la herramienta wireshark

4.3.2. Paquete I2

Host1 responderá al paquete R1 con el paquete HIP I2. Este nuevo paquete contiene la solución al puzzle, el modo de cifrado elegido, el segundo paso del algoritmo Diffie-Hellman, un campo *HMAC* para garantizar la actualidad del paquete, la clave pública de *Host1* cifrada, la firma digital del paquete y un parámetro llamado SPI.

Al recibir I2 y extraer el parámetro del algoritmo Diffie-Hellman, *Host2* ya puede construir la clave simétrica. Esta clave se utilizará para descifrar la clave pública de *Host1*, que ha llegado cifrada en I2. Con la clave pública verificará la firma digital. La firma digital garantiza que el paquete no ha sido modificado y que fue realmente creado por quien dice el campo *HIT_origen* de la cabecera HIP. La firma digital de I2 se construye de forma análoga a la firma de R1:

$$\text{Firma I2} = K^{-}_{host1}(\text{HASH(paquete)})$$

El campo *HMAC* es un resumen del paquete, cifrado con la clave simétrica. *HMAC* garantiza que el paquete pertenece a la asociación HIP actual. Es decir, este campo protege contra ataques de reenvío de paquetes de otras asociaciones HIP.

SPI son las siglas de Índice de Parámetro de Seguridad, y es un parámetro que se utiliza para el cifrado de datos posterior a la asociación HIP, mediante el protocolo ESP. Este parámetro, junto con la dirección IP origen, será utilizado por el protocolo ESP para identificar la asociación segura (SA) a la que pertenecen los datos cifrados.

Host2 también tendrá que comprobar que la solución al puzzle es correcta y no ha caducado. Deberá comprobar que *Host1* ha elegido una de las codificaciones que *Host2* le ofreció previamente en el paquete R1. Tendrá que verificar que la clave pública recibida guarda relación con el campo *HIT_origen* del paquete, y también deberá comprobar que previamente él (*Host2*) ha recibido un paquete I1 desde ese mismo origen y respondió con un paquete R1.

La Figura 19 corresponde a un paquete I2 capturado con Wireshark. El parámetro *SOLUTION* contiene la solución al puzzle. *HIP_TRANSFORM* almacena el modo de cifrado elegido por *Host1*. *ENCRYPTED* contiene la clave pública de Host1, cifrada con la clave simétrica obtenida con Diffie-Hellman y *ESP_INFO* contiene el campo SPI.

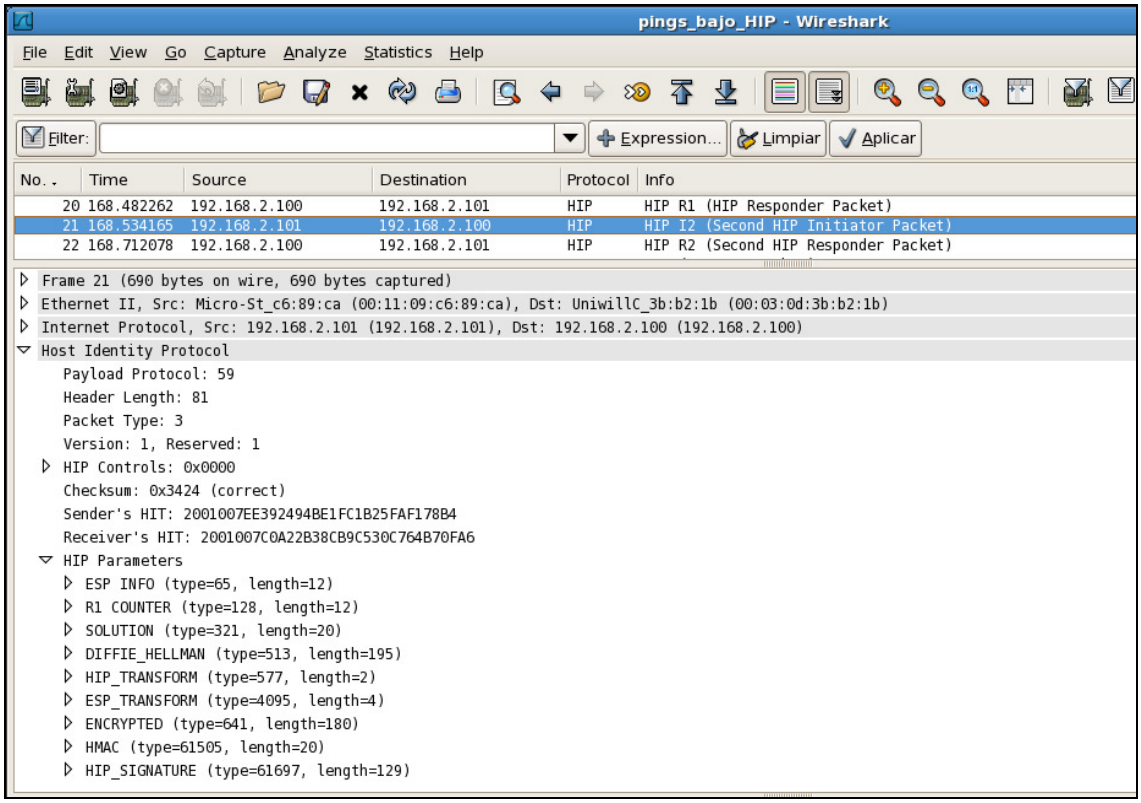


Figura 19. Paquete I2 visualizado con wireshark

4.3.4. Paquete R2

Si todo es correcto, *Host2* responderá al paquete I2 con un paquete R2. Este último paquete contiene, además de la cabecera HIP con *HIT_origen* y *HIT_destino*, el campo *SPI*, la

firma digital y el *HMAC*. *SPI* se utiliza para el cifrado posterior, mientras que la firma y el campo *HMAC* garantizan la integridad del paquete, la autenticidad del emisor y la pertenencia del paquete a la asociación HIP actual. Es decir, estos campos cumplen la misma función que en el paquete anterior I2. La Figura 20 corresponde a un paquete R2 capturado con Wireshark.

Cuando *Host1* recibe R2, debe comprobar si previamente envió el paquete I2 al host que aparece como *HIT_origen* en el paquete R2. También tiene que verificar el campo *HMAC* y la firma digital. Si todo es correcto, *Host1* pasará al estado "*ESTABLISHED*". HIP no especifica la forma de transportar los datos una vez establecida la asociación, pero permite usar ESP para transportar los datos. Así, se consigue una implementación de IPSec independiente de la dirección IP (ver paquetes *UPDATE*), y con establecimiento seguro.

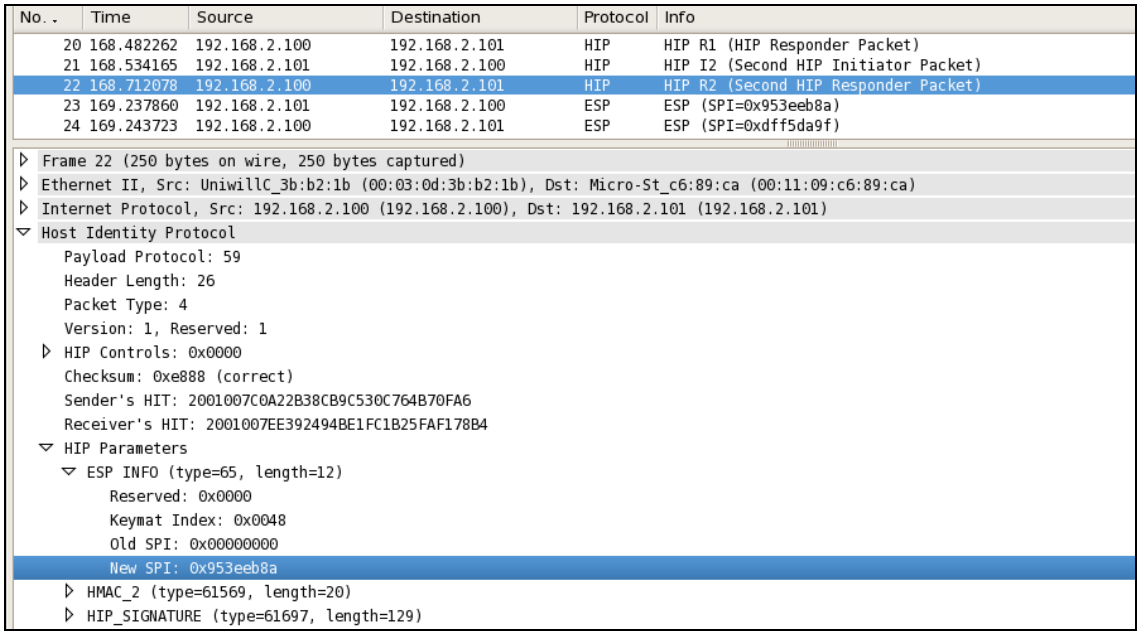


Figura 20. Paquete R2 visualizado con wireshark

A modo de resumen se presenta en la Figura 21 la secuencia de establecimiento de una asociación HIP. Para cada paquete se indican los campos que lo forman. La cabecera, formada por los campos *HIT_origen* y *HIT_destino*, es común a todos los paquetes y no se ha representado en la figura. En los laterales aparece el estado en el que se encuentra la máquina de estados en cada uno de los hosts. En la Figura 24 puede consultarse la máquina de estados simplificada del protocolo HIP.

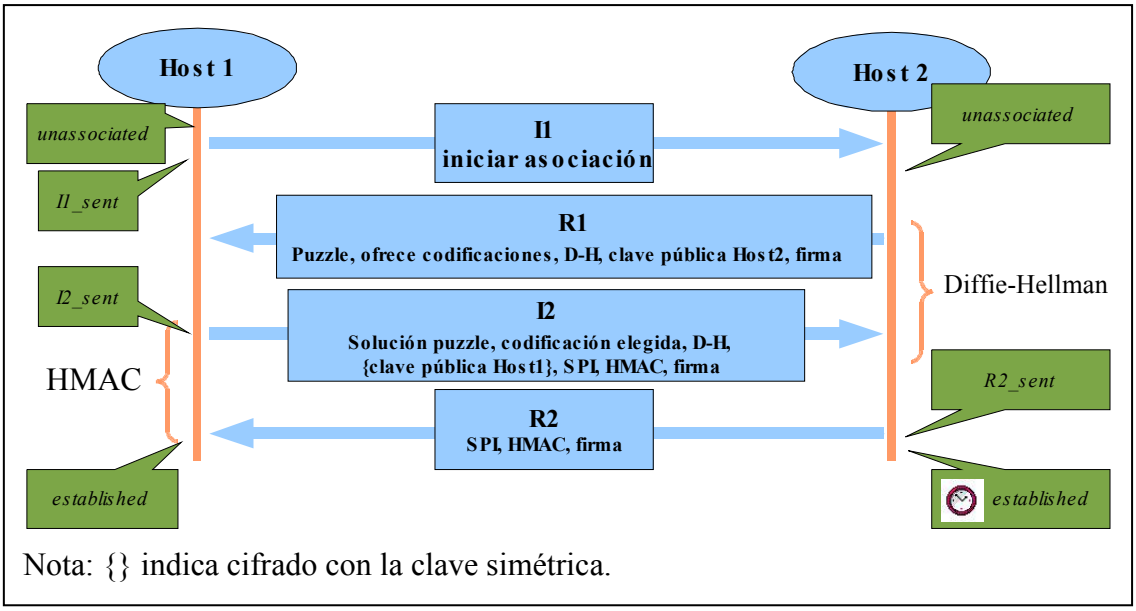


Figura 21. Resumen del establecimiento de una asociación HIP

4.4. Movilidad con HIP: paquetes UPDATE

Cuando una de las partes de una asociación HIP cambia su dirección IP, el protocolo HIP se encarga de informar a la otra parte para que la comunicación cifrada con ESP no se invalide. Para ello se utiliza un protocolo en 3 pasos con paquetes HIP de tipo *UPDATE*. Los tres paquetes están firmados digitalmente y tienen un parámetro *HMAC*, al igual que los paquetes I2 y R2.

El primer paquete *UPDATE* contiene un campo *ESP_INFO* con el antiguo *SPI* y el nuevo *SPI*. También contiene un valor contador (*SEQ*), que se incrementa con cada paquete *UPDATE* que envía ese host (salvo en las retransmisiones).

El receptor del paquete *UPDATE* debe verificar los campos firma y *HMAC*, además de comprobar que *HIT_origen* pertenece a una máquina con la que tiene una asociación HIP. Si todo es correcto, responderá con otro paquete *UPDATE*. Este nuevo paquete contiene el valor contador (*SEQ*) recibido en el paquete anterior, y un campo *ACK* indicando que es una contestación.

El tercer paquete *UPDATE* contiene el campo *ACK*. En la Figura 22 se resumen estos 3 paquetes. Si el primero de los paquetes no recibe respuesta, vencerá un timeout y el paquete será reenviado. Si se supera el número máximo de reenvíos se inicia el cierre de la asociación HIP (paquete *CLOSE*).

Host2 -> Host1
IP (HIP (**SEQ**, ESP_INFO, HMAC, SIGNATURE))

Host1 -> Host2
IP (HIP (**ACK**, **SEQ**, ESP_INFO, HMAC, SIGNATURE))

Host2 -> Host1
IP (HIP (**ACK**, HMAC, SIGNATURE))

Figura 22. Resumen del protocolo en 3 pasos de paquetes UPDATE

El campo contador es necesario porque los paquetes HIP no tienen garantizada la entrega ordenada (son paquetes sobre IP; no sobre TCP o similar). Gracias al contador, si un host recibe varios paquetes *UPDATE*, puede conocer el orden en el que fueron enviados e identificar retransmisiones de un mismo paquete, y según eso descartarlos o no.

4.5. Finalizar una asociación HIP: paquetes CLOSE

Cuando quiere finalizarse una asociación HIP explícitamente, o cuando una asociación HIP lleva tiempo sin ser utilizada, se envía un paquete HIP de tipo *CLOSE*. Este paquete contiene, además de la cabecera HIP con los HIT origen y destino, un campo *ECHO_REQUEST_SIGNED*, la firma digital y el parámetro *HMAC*. Los 2 últimos campos sirven para lo mismo que en los otros paquetes en los que aparecen: garantizar la integridad del paquete, la autenticación, y que el paquete pertenece a la asociación HIP actual.

El host que recibe el paquete *CLOSE* y verifica que todo es correcto, contesta con el paquete *CLOSE_ACK*. Éste contiene también una firma digital y el parámetro *HMAC*. Además, en lugar del campo *ECHO_REQUEST_SIGNED* contiene el campo *ECHO_REPLY_SIGNED*. Los dos campos *ECHO* deben tener el mismo valor. De esta forma se comprueba que el paquete *CLOSE_ACK* es realmente la contestación del paquete CLOSE.

En la Figura 23 se visualizan los paquetes *CLOSE* y *CLOSE_ACK* con la herramienta Wireshark. Tal y como se ha explicado, el parámetro *ECHO* contiene el mismo valor en ambos paquetes.

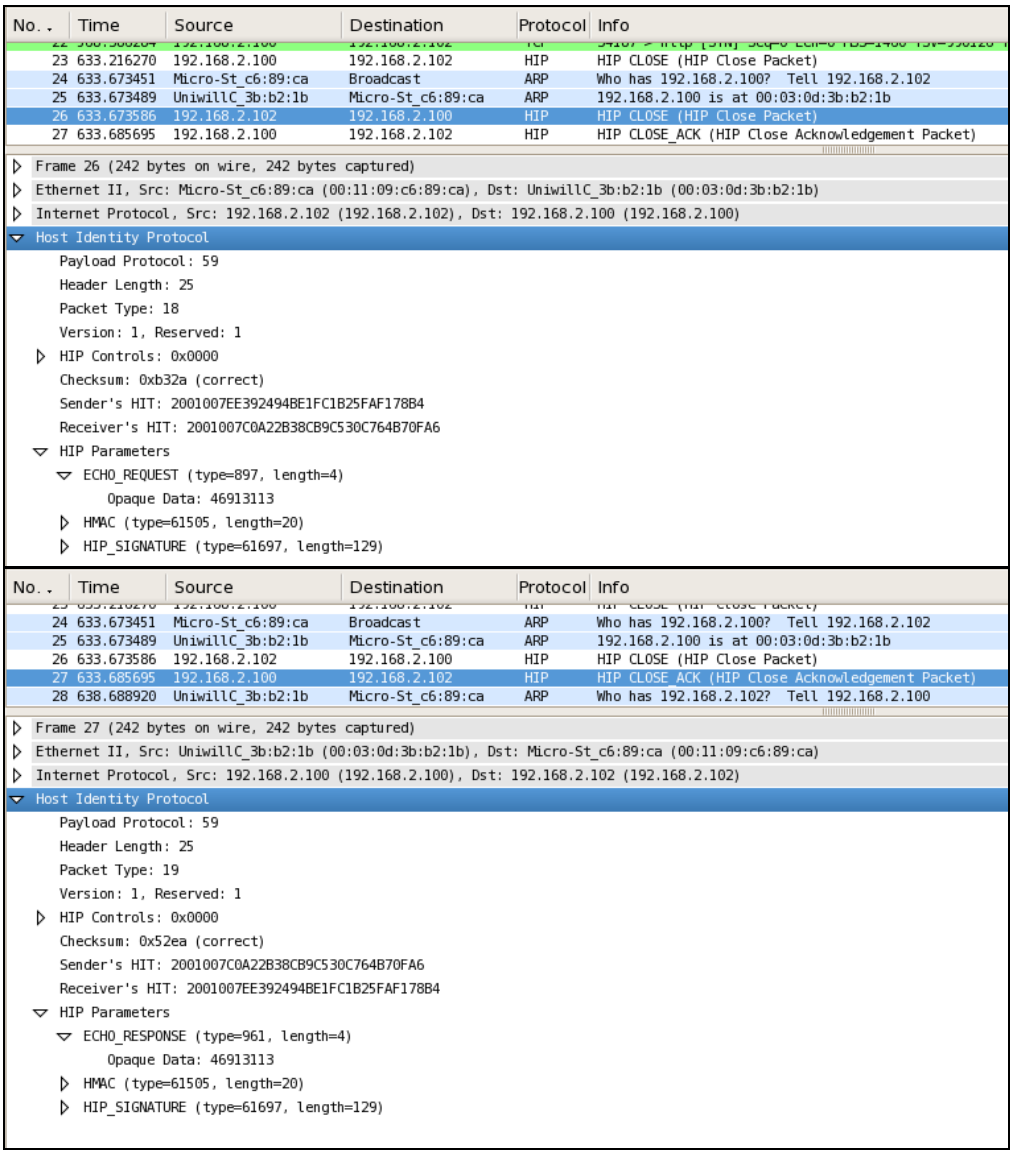


Figura 23. Paquetes CLOSE y CLOSE_ACK visualizados con wireshark

4.6. Notificaciones HIP

Una vez establecida la asociación HIP, pueden utilizarse los paquetes HIP de tipo *NOTIFICATION* para informar al otro Host sobre incidencias detectadas en la asociación HIP. Este tipo de paquete acepta el parámetro *NOTIFICATION*, del que se definen tantos tipos como errores tratables. Algunos ejemplos de notificaciones son: INVALID_SYNTAX, CHECKSUM_FAILED, HMAC_FAILED, SERVER_BUSY_PLEASE_RETRY.

Antes de establecer la asociación HIP no se puede informar de los errores con paquetes HIP. En este caso se propone utilizar mensajes ICMP para informar de los errores, aunque se recomienda un comportamiento silencioso en la mayoría de los casos, es decir, descartar los paquetes HIP erróneos sin enviar mensaje ICMP. El comportamiento silencioso consume menos recursos que el envío de un mensaje ICMP, lo que supone una alternativa más eficaz contra ataques de Denegación de Servicio (DoS).

Para finalizar la explicación de los paquetes HIP, se ha elaborado un diagrama con la máquina de estados simplificada de HIP (Figura 24). Todas las asociaciones HIP comienzan desde el estado *UNASSOCIATED*, y sólo llegan al estado *ESTABLISHED* tras completar satisfactoriamente el intercambio de paquetes I1, R1, I2 y R2. Las asociaciones HIP permanecerán en el estado *ESTABLISHED* hasta que sean finalizadas explícitamente o estén un largo periodo de tiempo sin ser utilizadas.

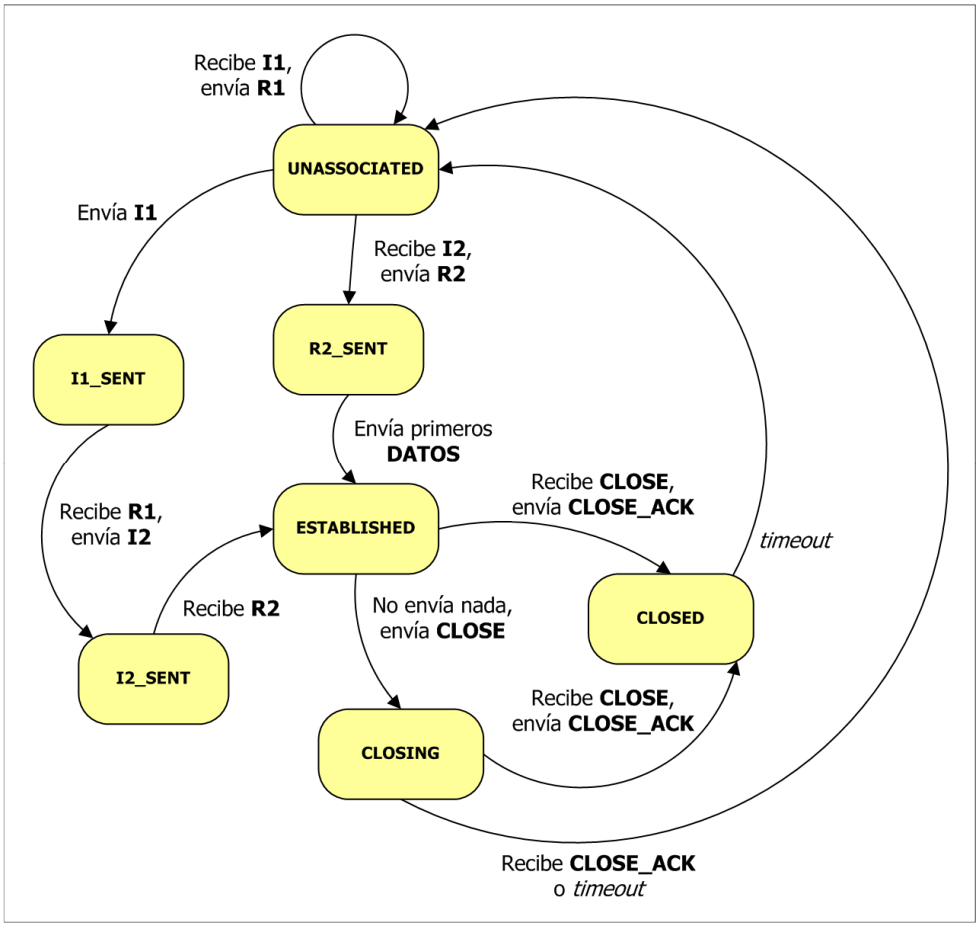


Figura 24. Diagrama de estados HIP simplificada

4.7. Seguridad ofrecida por HIP

HIP se ha diseñado para proporcionar autenticación entre hosts. También garantiza la integridad y la actualidad de los paquetes HIP, y junto con ESP ofrece confidencialidad, autenticación e integridad en los datos enviados. Además protege contra ataques de denegación de servicio (DoS). Es seguro ante ataques de hombre-en-medio (MitM) y de reenvío de paquetes. A continuación se describen algunos posibles escenarios de ataque y cómo son resueltos por HIP.

1. Ante una oleada de paquetes I1, el receptor puede reutilizar D-H y el puzzle. Así se reduce el coste computacional de generar un paquete R1. La reutilización supone un riesgo, pero ante un ataque es mejor esta solución de compromiso que generar datos nuevos para cada petición y sobrecargar el sistema.

2. Intento de DoS con I2. El atacante envía I1, recibe el puzzle en R1, lo resuelve, y envía una oleada de I2 con el campo FIRMA erróneo.

   Solución: al cabo de N paquetes I2 incorrectos, HIP descarta sistemáticamente los paquetes I2 que contengan el mismo *HIT_origen* que los incorrectos. El atacante tendrá que empezar de nuevo (enviar I1), y ahora se le puede asignar un puzzle más complicado (aumentando la dificultad K).

3. Simulación de reinicio de una de las partes de una asociación HIP ya existente: El atacante envía I1 con el *HIT_origen* de una de las partes (suplantación de identidad).

   Solución: Desde el estado *ESTABLISHED* no se transita si llega I1, por lo que el atacante no logra su objetivo.

4. Emulación de fin de estado (cierre de asociación HIP): El atacante envía *CLOSE* o *CLOSE_ACK* haciéndose pasar por una de las partes de una asociación HIP existente.

   Solución: Paquetes *CLOSE* y *CLOSE_ACK* contienen FIRMA y *HMAC*, por lo que un atacante no puede cerrar una asociación HIP si no forma parte de ella. La firma permite verificar la identidad del emisor del paquete y que dicho paquete no ha sido alterado (autenticación e integridad). El campo *HMAC* permite comprobar que el paquete se ha generado durante la asociación HIP actual, y no se trata de un ataque por reenvío.

Además, el campo FIRMA puede ser utilizado por estaciones intermedias (por ejemplo, routers) para rechazar paquetes incorrectos (para verificar la FIRMA sólo se necesita el paquete y la clave pública del emisor; esto está disponible en los paquetes R1).

El uso conjunto de los campos FIRMA y *HMAC* en un mismo paquete es una técnica criptográfica relativamente nueva. Su utilización ha dado lugar en los últimos años a una nueva familia de protocolos, denominada SIGMA ('SIGn-and-MAc') [5]. Estos protocolos persiguen el mismo objetivo que el protocolo Diffie-Hellman, es decir, acordar una clave secreta entre dos máquinas a través de un canal inseguro. Pero a diferencia del protocolo Diffie-Hellman, los protocolos SIGMA no son vulnerables a ataques de Hombre en Medio (MitM) porque ofrecen autenticación, integridad y resistencia a ataques de reenvío.

En el algoritmo Diffie-Hellman, un atacante podría situarse entre ambas máquinas y acordar una clave simétrica con cada una de las partes, haciéndose pasar por el host A de cara al host B y viceversa. Una vez establecidas las 2 claves simétricas, el atacante haría de puente entre los 2 hosts, descifrando toda la comunicación y volviéndola a cifrar para enviársela al otro host.

Con la firma digital de los dos paquetes que contienen la información intercambiada por D-H (en HIP, R1 y I2), se garantiza que el paquete no ha sido modificado (integridad) y que ha sido generado por el host cuya clave pública aparece también en dicho paquete (autenticación).

Además, los campos *HMAC* presentes junto al segundo mensaje D-H y en el paquete posterior (en HIP, los paquetes I2 y R2) son un resumen del paquete (Hash) cifrado con la clave simétrica obtenida por el algoritmo de Diffie-Hellman. Con este campo se garantiza que los paquetes pertenecen a la asociación actual y no corresponden a un ataque de reenvío.

Respecto al protocolo SIGMA básico, que requiere 3 paquetes, HIP añade un paquete adicional. Este paquete es I1, y su inclusión es necesaria para ofrecer un puzzle y ser así resistente a ataques de DoS, tal y como se comentó antes.

5. PROPUESTA DE INTEGRACIÓN DE HIP EN REDES MANET

En una red ad-hoc el protocolo HIP resultaría beneficioso, ya que admite la movilidad de los hosts sin que estos pierdan la identidad, y en las redes ad-hoc los hosts pueden estar constantemente en movimiento y pueden cambiar de dirección IP. Además HIP permite una comunicación cifrada, y esto es muy útil en las redes ad-hoc, ya que la información viaja a través de otros usuarios de la red y conviene proteger los datos confidenciales. Por último, HIP proporciona autenticación, integridad y el establecimiento de la asociación HIP es resistente a ataques DoS y MitM, como se ha comentado en el apartado 4.

Sin embargo, la versión original de HIP no puede utilizarse directamente en una red ad-hoc. La máquina que inicia una asociación HIP necesita conocer la dirección IP de la identidad con la que quiere comunicarse. En una red con infraestructura, esta dirección IP puede conocerse o puede consultarse en un servidor DNS que resuelva consultas HIP. En una red ad-hoc no habrá un servidor DNS y puede ocurrir que la dirección IP de los hosts cambie con más frecuencia. Para resolver este problema se ha modificado el protocolo HIP para que los paquetes I1 sean enviados por difusión. Así llegarán a su destinatario aunque se desconozca su dirección IP, y sin necesidad de disponer de un servidor central.

Además, para garantizar que los nodos intermedios no son capaces de leer qué 2 identidades se comunican en una asociación HIP, se propone cifrar los paquetes HIP y hacer uso de seudónimos.

Esta modificación del protocolo HIP y su integración con el protocolo de encaminamiento OLSR ha sido bautizada con el nombre de HIP MANET. A continuación se detallan las características que diferencian a HIP MANET de la versión original, así como las ventajas que supone.

5.1. Difusión de paquetes I1

En las redes de ordenadores puede ocurrir que una determinada máquina quiera acceder a un servicio concreto pero inicialmente no sepa cómo llegar a él. Esto sucede, por ejemplo, cuando un usuario teclea una página web en su navegador. En dicha situación, el ordenador del usuario recibe la url, que identifica inequívocamente la página a la que debe acceder, y conoce el protocolo que debe utilizar (HTTP, HTTPS, etc…). Sin embargo, para acceder finalmente al servicio, el ordenador debe realizar una consulta a un servidor DNS, que responderá con la dirección IP hacia la que se debe realizar la petición de la página web.

Con el protocolo HIP ocurre un comportamiento similar. Un determinado ordenador puede conocer la identidad HIP con la que desea comunicarse, pero deberá conocer también su actual dirección IP para iniciar la asociación HIP. La solución propuesta por el protocolo HIP consiste en almacenar las identidades HIP de los nodos de la red, junto con sus direcciones IP actuales, en servidores DNS modificados.

Sin embargo, la solución de los servidores DNS no se puede aplicar a las redes ad-hoc. Este tipo de redes se caracterizan por no requerir ningún tipo de infraestructura fija ni administración centralizada. Además, las redes ad-hoc suelen configurarse como tales ante la imposibilidad de disponer de servicios centralizados o infraestructura. Por tanto, si se añadiese un servidor DNS se estaría centralizando en parte la administración de la red, y esto en muchos escenarios ad-hoc puede no ser posible.

Como solución alternativa a los servidores DNS, se propone iniciar la asociación HIP con el envío del paquete I1 por broadcast, en lugar de unicast. Así la red ad-hoc, al no depender de ningún servidor central, sigue caracterizándose por su funcionamiento distribuido.

Tras enviar un paquete I1 por broadcast, todos los nodos de la red deberán recibir dicho paquete. La función de propagación de los paquetes broadcast puede no estar implementada en el protocolo de encaminamiento utilizado en la red ad-hoc. Por este motivo, el demonio HIP también será modificado para, opcionalmente, poder encargarse del reenvío de los paquetes I1 recibidos por broadcast.

La dirección de broadcast utilizada podrá ser configurada, ya que en algunos escenarios puede ser más interesante utilizar una dirección de broadcast dirigida a la red (por ejemplo, 192.168.2.255) que la dirección de broadcast limitado (255.255.255.255). Puede ocurrir que un protocolo de encaminamiento reenvíe los broadcast dirigidos a la red pero no los broadcast limitados, por este motivo se ofrece la posibilidad de configurar la dirección IP de broadcast.

En la Figura 25 se muestra el tráfico en la red cuando los paquetes I1 se envían por difusión. Se observa que la estructura del paquete no ha variado. Sólo se ha modificado la dirección IP destino, que ha pasado a ser 255.255.255.255 (dirección de broadcast). En esa misma figura, el iniciador de la asociación HIP es el host con dirección IP 192.168.2.101. El host que responde tiene la dirección IP 192.168.2.100. Contesta ese host y no otro porque al leer el campo *HIT_destino* verifica que va dirigido a él.

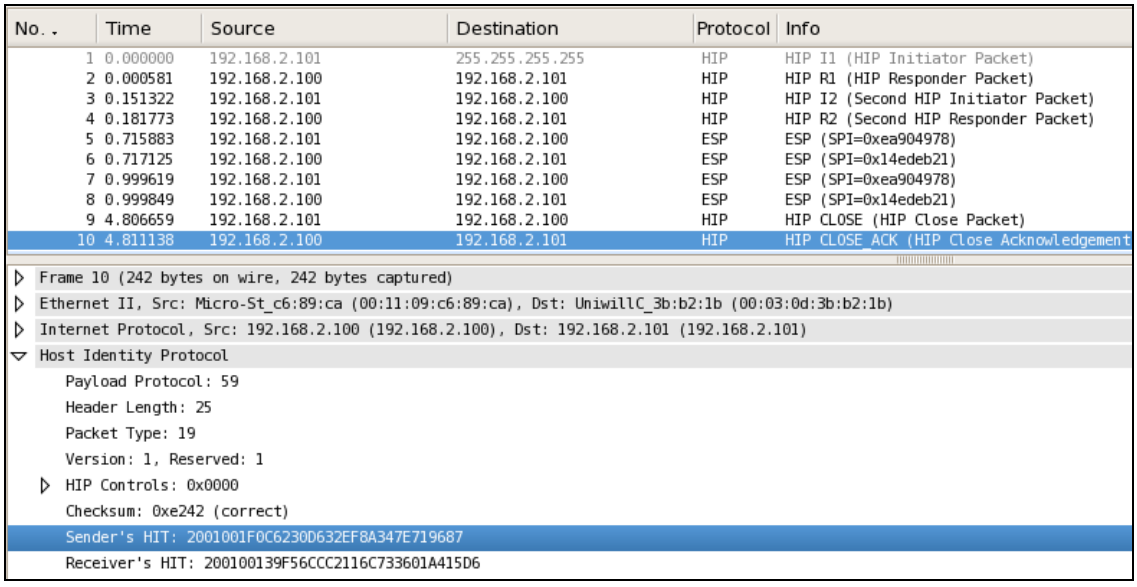


Figura 25. Difusión de paquetes I1

5.1.1. Uso de LSI como dirección IP

Si no se requiere anonimato en la red ad-hoc, la siguiente propuesta puede resultar también interesante. Esta solución evita el envío por broadcast del paquete I1, pero como contrapartida, los nodos intermedios pueden conocer qué dos identidades participan en cada comunicación (pérdida de anonimato).

En las redes con infraestructura, las direcciones IP se utilizan para crear una jerarquía entre los nodos, que facilita el enrutamiento de los paquetes. Así, las máquinas con prefijos comunes en sus direcciones IP, estarán geográficamente próximas y en la misma subred. Estos prefijos comunes identificarán a la red, mientras que el resto de la dirección identificará al host dentro de dicha red. Gracias a ello, los routers de Internet no almacenan en sus tablas de encaminamiento una entrada por cada ordenador de Internet (esto sería muy lento). En lugar de eso, los routers encaminan el tráfico atendiendo a direcciones de red. Es decir, cada entrada de sus tablas de encaminamiento encaminará paquetes destinados a un rango de direcciones IP, en lugar de una única máquina.

Sin embargo, en las redes ad-hoc esta jerarquía no existe. El número de nodos en este tipo de redes nunca es tan grande como en Internet, y los protocolos de encaminamiento ad-hoc como OLSR funcionan añadiendo entradas individuales en las tablas de enrutamiento. Es decir, OLSR añade una entrada por cada nodo de la red.

La nueva propuesta consiste en utilizar como dirección IP el identificador HIP de 32 bits, conocido como LSI. Este identificador tiene la forma de una dirección IP cuyo primer byte toma el valor uno: 1.X.X.X. El resto de la dirección se obtiene directamente con los 24 bits de menor peso del HIT. La utilización de esta dirección IP evitaría tener que enviar los paquetes I1 por broadcast, ya que desde el momento que se conociese la identidad (HIT) con la que queremos comunicarnos, podríamos deducir su dirección IP y comenzar la asociación HIP vía unicast.

Esta solución reduciría la sobrecarga en la red, ya que se eliminaría la inundación de paquetes I1. La desventaja de esta propuesta es la pérdida de anonimato. Un nodo intermedio leería la cabecera IP, obteniendo así los LSI que participan en la comunicación. A continuación el nodo atacante consultaría su listado de identidades conocidas. Si los nodos que se están comunicando se encuentran entre sus conocidos, la comunicación no será anónima. Esta situación aparece reflejada en la Figura 26.

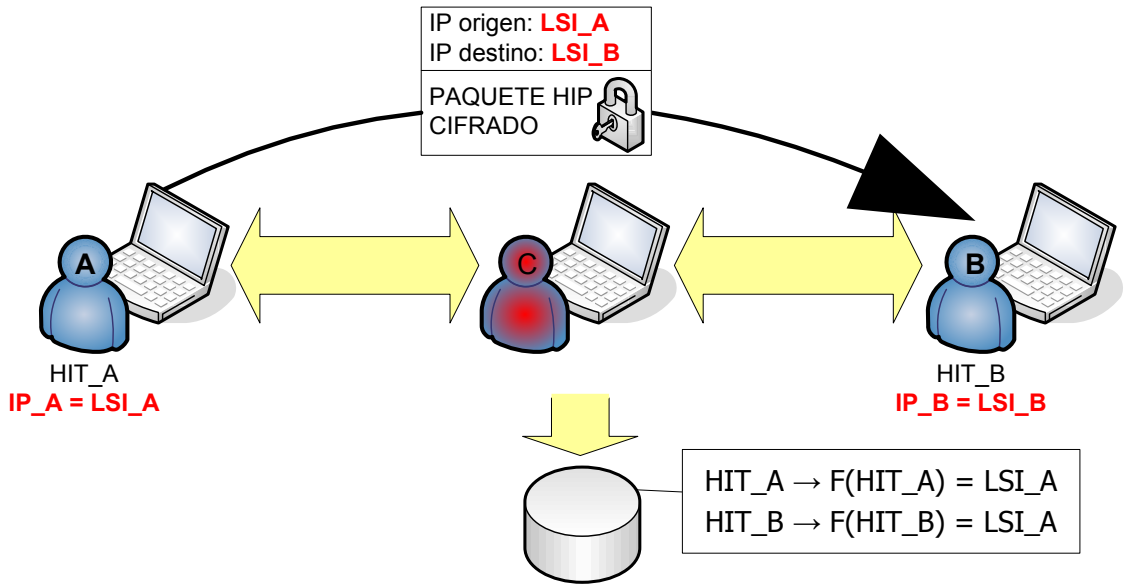


Figura 26. Pérdida de anonimato al utilizar LSI como dirección IP

La propuesta quedaría invalidada ante una colisión de direcciones IP. Resulta estadísticamente imposible que dos nodos tengan la misma identidad, ya que el campo HIT tiene una longitud de 128 bits, es decir, 2^{128} posibles valores. Por el contrario, una colisión en el espacio de nombres LSI sí podría ocurrir, ya que "sólo" existen 2^{24} posibles valores.

Con el fin de permitir la implantación de esta propuesta sin necesidad de modificar las implementaciones existentes del protocolo HIP, será necesario utilizar un rango de direcciones IP diferente al 1.0.0.0/8, ya que los paquetes IP con estas direcciones son interceptados por el demonio HIP. Como solución se sugiere utilizar como dirección IP el propio LSI, pero sustituyendo el 1 por un 2. Es decir, un nodo con LSI igual a 1.123.123.123 utilizará la dirección IP 2.123.123.123. Hay que recordar que el rango 2.0.0.0/8 está reservado por la IANA [9], al igual que el 1.0.0.0/8. El fichero de identidades conocidas tendrá que ser rellenado respetando la relación existente entre LSI y IP.

De aquí en adelante, si no se especifica lo contrario, se asumirá que los paquetes I1 son enviados por difusión en la versión modificada de HIP. La alternativa titulada "Uso de LSI como dirección IP" es interesante porque también permite el uso de HIP en redes ad-hoc, pero con ella no se logra alcanzar el anonimato en redes ad-hoc, que es el objetivo principal de este proyecto.

5.2. Asociación HIP cifrada

Otro problema presente en las redes ad-hoc ocurre durante el establecimiento de una asociación HIP. Los hosts intermedios pueden saber qué 2 identidades pretenden comunicarse, simplemente leyendo los campos *HIT_origen* y *HIT_destino* de los paquetes I1, R1, I2 y R2. Para evitarlo, la versión modificada de HIP permite cifrar los paquetes HIP con la clave pública del receptor.

Hasta ahora no era necesario tener almacenadas las claves públicas de los hosts con los que se deseaba establecer una asociación HIP, ya que estas claves se recibían en los paquetes R1 e I2. Pero ahora estas claves son necesarias antes de enviar I1, para poder enviarlo cifrado. Consecuentemente, las claves públicas se almacenarán en el fichero de identidades conocidas, junto a los campos HIT, dirección IP, LSI y nombre. Las claves públicas se añadirán al fichero automáticamente cada vez que se reciba un paquete R1 o I2. HIP utiliza el sistema RSA [15] para la criptografía asimétrica. Las claves públicas en este sistema se componen de un exponente público "E" y de un valor "N". "N" es el producto de los 2 números primos que debe generar el host.

En la Figura 27 se compara un fragmento del fichero de identidades conocidas. A la izquierda se muestra el posible contenido del fichero cuando se utiliza la versión original de HIP. A la derecha, se observan los nuevos campos añadidos en la versión modificada de HIP para almacenar la clave pública.

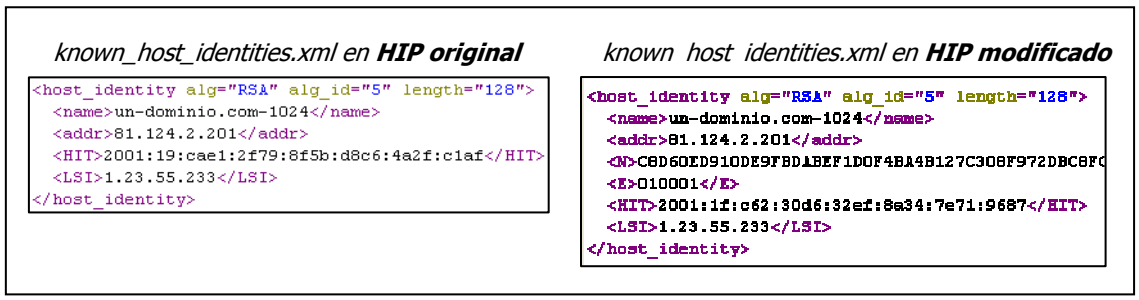
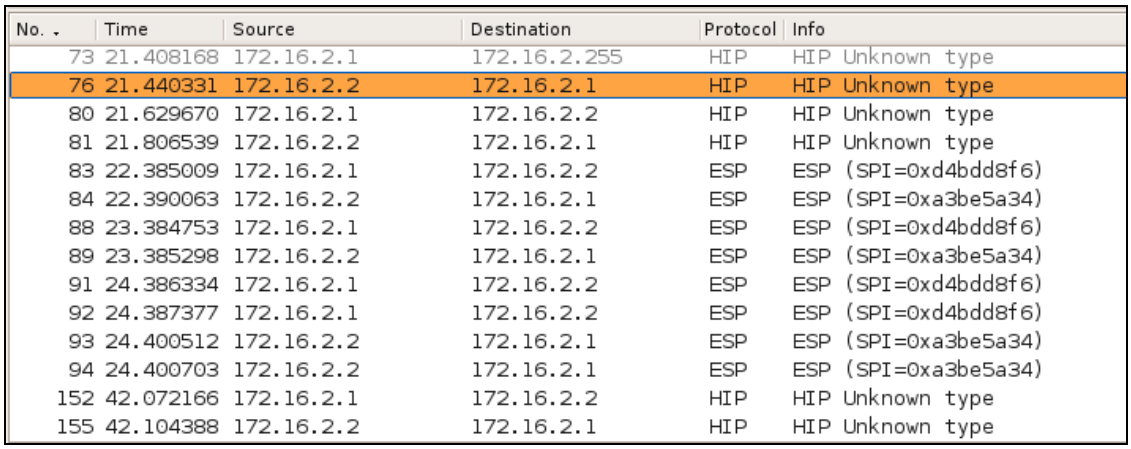


Figura 27. Comparativa de un fragmento del fichero known_host_identities.xml

Por tanto, si se desea enviar los paquetes HIP cifrados con la clave del receptor, habrá que conocer previamente la clave pública de éste. Esta clave nos la puede facilitar directamente la otra persona (por ejemplo, vía Bluetooth [23]), o podemos disponer de ella si previamente hemos establecido una asociación HIP sin cifrado de los paquetes HIP.

Cuando un paquete HIP cifrado llega a un host, éste lo intenta descifrar con su clave privada. Si la cabecera HIP resultante es incorrecta, dicho host descartará el paquete y, opcionalmente, procederá al reenvío del mismo (si el demonio HIP ha sido configurado para ello). Si la cabecera del paquete HIP descifrado es correcta, el contenido del paquete será interpretado como en la versión original de HIP.

En la Figura 28 se observa el estado de la red con los paquetes cifrados. El analizador de red no puede determinar el tipo de los paquetes HIP cifrados, ya que para ello debería descifrarlos con la clave privada del receptor.



No.	Time	Source	Destination	Protocol	Info
73	21.408168	172.16.2.1	172.16.2.255	HIP	HIP Unknown type
76	21.440331	172.16.2.2	172.16.2.1	HIP	HIP Unknown type
80	21.629670	172.16.2.1	172.16.2.2	HIP	HIP Unknown type
81	21.806539	172.16.2.2	172.16.2.1	HIP	HIP Unknown type
83	22.385009	172.16.2.1	172.16.2.2	ESP	ESP (SPI=0xd4bdd8f6)
84	22.390063	172.16.2.2	172.16.2.1	ESP	ESP (SPI=0xa3be5a34)
88	23.384753	172.16.2.1	172.16.2.2	ESP	ESP (SPI=0xd4bdd8f6)
89	23.385298	172.16.2.2	172.16.2.1	ESP	ESP (SPI=0xa3be5a34)
91	24.386334	172.16.2.1	172.16.2.2	ESP	ESP (SPI=0xd4bdd8f6)
92	24.387377	172.16.2.1	172.16.2.2	ESP	ESP (SPI=0xd4bdd8f6)
93	24.400512	172.16.2.2	172.16.2.1	ESP	ESP (SPI=0xa3be5a34)
94	24.400703	172.16.2.2	172.16.2.1	ESP	ESP (SPI=0xa3be5a34)
152	42.072166	172.16.2.1	172.16.2.2	HIP	HIP Unknown type
155	42.104388	172.16.2.2	172.16.2.1	HIP	HIP Unknown type

Figura 28. Paquetes HIP cifrados, visualizados con la herramienta wireshark

Con esta modificación, un host intermedio no puede leer los campos HIT origen y HIT destino en las cabeceras de los paquetes HIP, ya que todo el paquete HIP está cifrado. Sin embargo, cabe la posibilidad de que un atacante que haya establecido asociaciones HIP con al menos dos hosts, pueda descubrir las identidades HIT que participan en un paquete HIP cifrado. Este problema implica que el protocolo HIP modificado aún no garantiza el anonimato. En el siguiente apartado se profundiza en este problema y se propone una solución más adecuada.

5.3. Uso de seudónimos

Con la modificación de cifrado de los paquetes HIP, explicada anteriormente, puede ocurrir la situación que se representa en la Figura 29. Se observan 3 nodos de una red ad-hoc, etiquetados con las letras A, B y C. Tanto A como B han establecido una asociación HIP con el nodo C. En consecuencia, el nodo C es capaz de almacenar la relación existente entre las identidades de las otras 2 máquinas y sus respectivas direcciones IP actuales.

Cuando los nodo A y B intercambien un paquete HIP cifrado, dicho paquete atravesará el nodo intermedio C. Este último no podrá leer la cabecera HIP del paquete, pero sí mirará los campos IP origen e IP destino de la cabecera IP. A partir de estos campos, y consultando la información de la que dispone, el nodo C sería capaz de deducir las identidades HIT que se ocultan detrás del paquete cifrado.

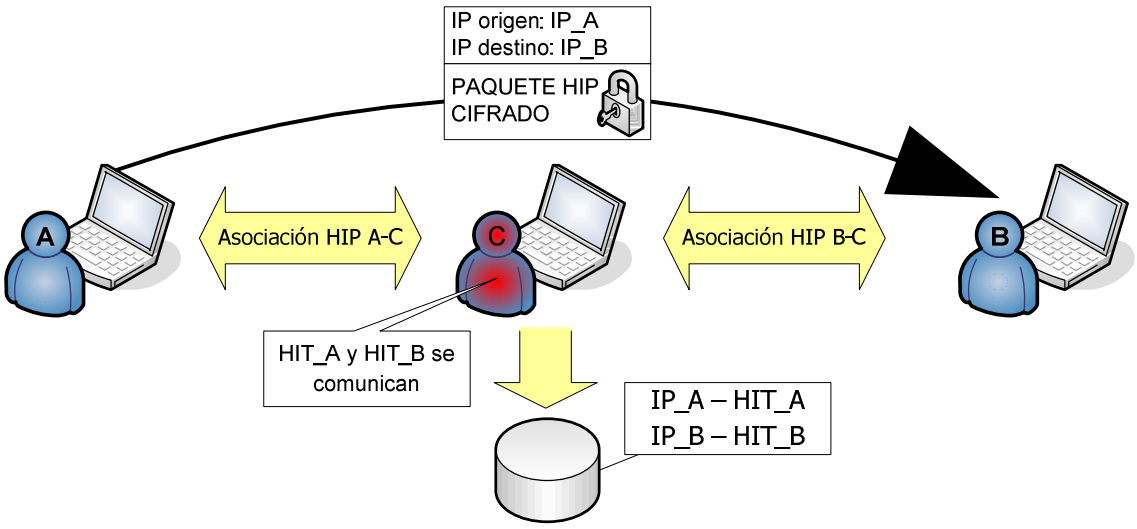


Figura 29. Pérdida de anonimato en una asociación HIP cifrada

Para resolver este problema, se propone el uso de seudónimos. Un seudónimo es un nuevo identificador que se utiliza en lugar del identificador propio. Concretamente, se propone la utilización de múltiples direcciones IP por cada nodo como solución al problema y garantizar así el anonimato.

5.3.1. Seudónimos en HIP

Cuando un nodo A desee establecer una asociación HIP con otro nodo C, elegirá una de sus múltiples direcciones IP, y la utilizará siempre que quiera comunicarse con C. Empezará utilizándola en el campo IP origen del paquete I1 enviado por broadcast. Cuando dicho paquete llegue a C, éste elegirá una de sus múltiples direcciones IP para contestar. Esta dirección IP escogida por C será la misma durante todas las comunicaciones con A.

Cuando el nodo A desee comunicarse con otro nodo B, escogerá una dirección IP diferente. De esta forma el nodo C, aunque recibiese los paquetes HIP cifrados enviados entre A y B, no sería capaz de asociar la identidad de A con ese paquete.

5.3.2. Seudónimos en OLSR

Para que la solución planteada funcione, los nodos de la red no deben tener forma de conocer las múltiples direcciones IP de un nodo. Cada dirección IP debe ser interpretada por el resto de nodos de la red como un nuevo nodo con una única dirección IP. Para conseguir esto es necesario modificar ligeramente el demonio del protocolo de encaminamiento, concretamente el protocolo OLSR [6].

En la Figura 30 se muestra el mismo escenario que en la Figura 29 pero con el uso de seudónimos. Gracias a la modificación del protocolo de encaminamiento, las direcciones IP utilizadas como seudónimos por los nodos A y B son interpretadas como nuevos nodos (en color verde). Estos nuevos nodos ficticios aparecerán en la topología de la red como vecinos de los nodos que los utilizan como seudónimos. Por ejemplo, el nodo A utiliza los seudónimos D y E.

Gracias a esta topología, cuando un nodo desee comunicarse con el seudónimo de otro nodo (sin saber que se trata en realidad de un seudónimo), el protocolo de encaminamiento se encargará de que los paquetes sean encaminados a través del destinatario real. Cuando los paquetes lleguen a dicho destinatario, en lugar de encaminarse hacia el vecino ficticio, serán procesados. En el ejemplo, si el nodo B envía paquetes a la dirección IP_E (un seudónimo de A), los paquetes serán encaminados hacia A y, al llegar a éste, se procesarán. El nodo A responderá a B utilizando la dirección IP_E.

También en la Figura 30 se comprueba cómo se ha logrado resolver el problema del anonimato. En la asociación HIP entre A y C, el nodo A ha elegido su dirección IP_D. En la asociación entre B y C, el nodo B ha elegido su dirección IP_F. Para la nueva asociación HIP entre A y B, los nodos han elegido IP_E e IP_G respectivamente. Los paquetes HIP cifrados de esta última asociación HIP son leídos por el nodo C, pero este no puede conocer las identidades HIT involucradas en dicha comunicación.

La modificación del demonio OLSR consiste en simular la llegada de mensajes de control OLSR desde los vecinos ficticios. En el ejemplo de la Figura 30, el demonio OLSR presente en el nodo A debe simular la recepción de mensajes de control OLSR provenientes de los nodos D y E. Estos mensajes, que serán de tipo HELLO y TC, contendrán la información necesaria para simular la topología que aparece en la figura, y seleccionarán al nodo A como MPR de D y E, con el fin de que el nodo A propague los mensajes TC de ambos vecinos ficticios. El resto de nodos de la red no tendrán forma de diferenciar los nodos reales de los ficticios.

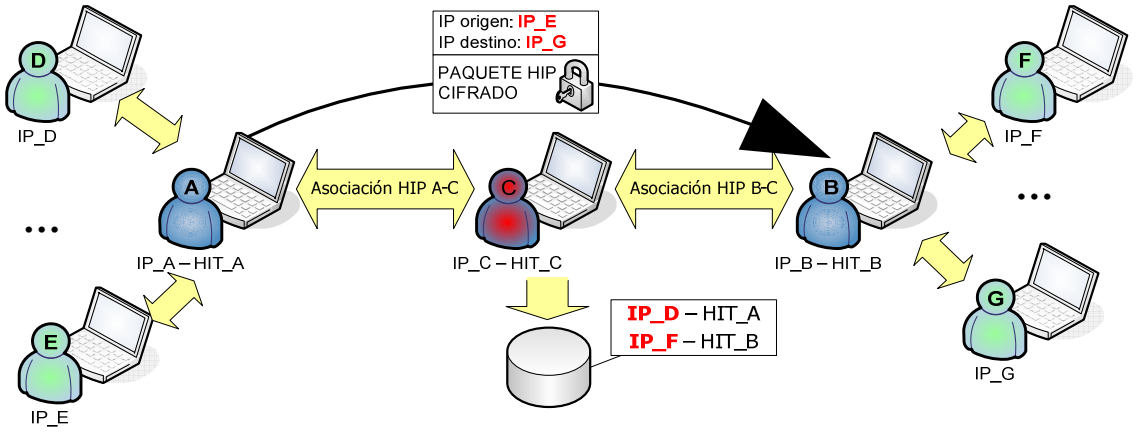


Figura 30. Utilización de seudónimos para lograr anonimato

El nuevo demonio OLSR también permitirá la elección automática de los seudónimos. Esto quiere decir que será el propio demonio (y no el usuario mediante un fichero de configuración) el que escogerá un subconjunto de direcciones IP libres para utilizarlas como seudónimos.

Según lo explicado, sólo será necesario utilizar la versión modificada de OLSR en los usuarios que deseen anonimato. Es decir, los nodos que implementen la versión modificada de OLSR dispondrán de anonimato incluso en una red en la que el resto de nodos utilice la versión original de OLSR.

6. Detalles de implementación

A continuación se detallan aspectos relacionados con la implementación de las modificaciones de HIP y OLSR. También se explica cómo debe ser configurada la interfaz de red para garantizar el anonimato. Finalmente se presenta HOP (Hip + Olsr + Pseudonyms), una utilidad para configurar automáticamente y ejecutar HIP y OLSR ofreciendo anonimato. En el Anexo III se facilita información sobre algunas de las herramientas utilizadas durante el desarrollo de este trabajo.

6.1. Modificaciones al protocolo HIP

Las modificaciones al protocolo HIP se han implementado sobre el código fuente del proyecto *OpenHIP* [17]. Este proyecto es libre, de código abierto, e implementa el protocolo experimental HIP. Cuenta con una excelente documentación en línea, y el código fuente puede ser compilado en multitud de sistemas operativos, tales como Linux, BSD, Windows XP y Mac OS X. En concreto se ha trabajado con la versión 0.5 de *OpenHIP*.

A la hora de analizar el código, hay que tener en cuenta que existen dos modos de ejecución en *OpenHIP*: el modo en espacio de usuario y el modo kernel. El segundo modo está disponible sólo para linux, mientras que el primero funciona en linux, Windows XP y Mac OS X. La versión en espacio de usuario recibe más atenciones por parte de los desarrolladores y se encuentra en una fase más avanzada. Hay ficheros del código que pertenecen sólo a uno de los dos modos.

Al descomprimir la distribución de *OpenHIP* se crea la siguiente estructura de ficheros:

HIP_0.5
- /conf: Configuración por defecto (*known_host_identities.xml* y *hip.conf*).
- /config: Ficheros utilizados en la instalación.
- /docs: Soporte Doxygen (documentación para desarrolladores).
- /src:

/src/include	ficheros .h
/src/linux	Fuentes específicos para linux. Función “main” para modo kernel (hip_linux_main.c) y para modo usuario (hip_linux_umh.c).
/src/mac	Funciones reescritas para Mac.
/src/protocol	Corazón de HIP. Funciones para la creación y comprobación de paquetes I1, R1, I2 y R2.
/src/username	Código que implementa ESP en el nivel de usuario (en lugar de en el kernel), y emula otras funcionalidades de la capa de red.
/src/util	Cálculo de cheksums, edición de XML, comparación de HITS, inicialización de estructuras, etc.
/src/win32	Código específico para Windows y su instalador.

La versión modificada del demonio HIP es completamente compatible con la versión original. Es decir, la versión modificada funcionará exactamente igual que la versión

original si no se especifican las nuevas opciones en el fichero de configuración o en la llamada desde la línea de comandos.

Los nuevos parámetros que pueden aparecer en el fichero de configuración (/usr/local/etc/hip/hip.conf) son los siguientes:

- **adhoc_mode**. Puede tomar como valor "no" o "yes". Indica si el paquete I1 se enviará por unicast (valor "no") o broadcast (yes). Si no se especifica este parámetro, su valor por defecto es "no".
- **anonymous_mode**. Indica si los paquetes HIP se cifran con la Clave Pública del destinatario. Puede tomar como valor "no" o "yes". Su valor por defecto es "no".
- **forwarding_broadcast**. Si toma valor "yes", el demonio HIP se encargará de retransmitir los I1 Broadcast. En caso contrario, esta tarea tendrá que ser realizada por otra aplicación (por ejemplo, el protocolo de encaminamiento). El valor por defecto es "no".
- **ip_broadcast**. Este campo indica la dirección broadcast a la que deben enviarse los paquetes I1 cuando el parámetro ad hoc_mode está en "yes". Por defecto, el valor de este parámetro es 255.255.255, pero puede ser interesante utilizar direcciones de broadcast dirigido a la red (por ejemplo, 172.16.2.255).

En la Figura 31 se muestra un ejemplo de fichero de configuración con estos nuevos parámetros:

```
<adhoc_mode>yes</adhoc_mode>
<anonymous_mode>no</anonymous_mode>
<forwarding_broadcast>no</forwarding_broadcast>
<ip_broadcast>192.168.2.255</ip_broadcast>
```

Figura 31. Fichero de configuración HIP con los nuevos parámetros

La versión modificada de HIP también dispone de nuevos argumentos para la invocación desde consola. A continuación se explican estos nuevos argumentos:

- **-V**. Nueva opción de depuración. Durante la ejecución aparecerán mensajes de depuración relacionados con las nuevas funcionalidades.
- **-pseudo fichero_seudonimos**. Este argumento sirve para utilizar los seudónimos. La ruta del fichero de seudónimos debe indicarse a continuación de "-pseudo". Este fichero contendrá las direcciones IP, una en cada línea, que deben utilizarse como seudónimos.
- **-auto_death.** Este argumento se utiliza para facilitar la medición del tiempo de establecimiento de asociación HIP. Con este argumento, el demonio HIP terminará la ejecución tras enviar o recibir un paquete R2 válido.
- **-time_assoc.** Cuando se utiliza este argumento, el tiempo requerido para establecer la asociación HIP es medido y almacenado al final del fichero de tiempos.

Tras enviar un paquete I1 por broadcast, todos los nodos de la red deberán recibir dicho paquete. Si el fichero de configuración tiene el parámetro forwarding_broadcast con valor "yes", cuando el host reciba I1 por broadcast posiblemente deberá reenviarlo. Para ello, comprueba el campo *HIT_destino*. Si éste no corresponde a una de sus identidades, restará

una unidad al campo TTL (Time to Live) del paquete IP que contiene I1 y lo reenviará (de nuevo por difusión, y sin modificar el campo *IP_origen*). Cuando I1 llegue al legítimo receptor, éste podrá contestar con R1 a la dirección *IP_origen* del paquete IP.

Para evitar inundar la red con paquetes I1, cada nodo, antes de reenviar un paquete I1 recibido por broadcast, verifica que es la primera vez que lo reenvía. Para controlar esta situación es necesaria una lista en la que se almacena la dirección IP origen y el campo ID de los paquetes IP que contienen los I1 reenviados últimamente. Además, los paquetes I1 recibidos por broadcast no son reenviados por el emisor inicial cuando éste los vuelve a recibir como consecuencia de un ciclo en la red. En este caso, el emisor original borra el paquete recibido directamente.

El protocolo HIP en el host origen se encarga de las retransmisiones de paquetes I1 que no reciben respuesta. Esto no se ha modificado de la versión original de HIP.

Cuando un paquete HIP cifrado llega a un host, éste lo intenta descifrar con su clave privada. Si la cabecera HIP resultante es incorrecta, dicho host procederá al reenvío del paquete, tal y como se explicó anteriormente: comprobar lista de paquetes enviados recientemente, restar una unidad al campo TTL, etc. Si el paquete HIP descifrado es correcto, no se reenviará el paquete porque es para él.

En cuanto al uso de seudónimos, la elección de un seudónimo *IP_X* para las comunicaciones con un destino *HIT_Y* se hace en base a una función de resumen. Gracias a ello, aunque la asociación HIP caduque o se cierre, la nueva asociación HIP seguirá utilizando el mismo seudónimo, garantizando así el anonimato.

En la Figura 32 se representa un diagrama del demonio HIP. En color se resaltan los módulos en los que se han centrado las modificaciones. El módulo de entrada/salida de paquetes HIP ha sido modificado para dar soporte al cifrado y descifrado de estos paquetes HIP. Además, el inicio de las asociaciones HIP cuenta ahora con la posibilidad de enviar los paquetes I1 vía broadcast. Estos paquetes pueden ser reenviados por el demonio HIP para que su propagación llegue a toda la red. Adicionalmente, el uso de seudónimos condiciona el campo IP origen de los paquetes HIP de una asociación, así como el de los paquetes ESP posteriores.

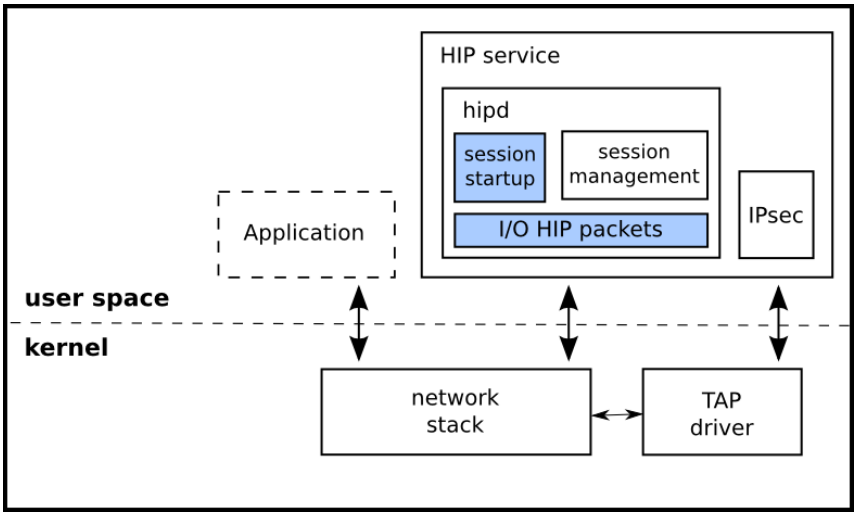


Figura 32. Diagrama de la aplicación modificada

En el Anexo I puede consultarse un manual de usuario para la versión modificada de HIP. Éste contiene información sobre la instalación, configuración y ejecución, así como soluciones a los problemas más frecuentes.

6.2. Modificaciones al protocolo OLSR

Las modificaciones que se han realizado en el demonio OLSR para dar soporte al uso de seudónimos han utilizado como implementación inicial *olsrd* [18]. Esta implementación empezó como parte de la tesis de Andreas Tønnesen en el UniK University Graduate Center (Universidad de Oslo) en el año 2004. Actualmente es un proyecto de código abierto, compatible con muchas arquitecturas y sistemas operativos. Además, *olsrd* cuenta con una amplia comunidad de usuarios y desarrolladores, por lo que ha sido testado en muchos escenarios reales y recibe actualizaciones periódicamente. *Olsrd* es una implementación rápida, que consume pocos recursos y altamente escalable. Actualmente se utiliza con éxito como protocolo de encaminamiento en una red de más de 2000 nodos en Atenas[1].

Se ha utilizado la versión más reciente de *olsrd* en el momento de realizar este trabajo: olsr 0.5.6-r4, publicada el 26 de marzo de 2009. La implementación modificada de *olsrd* (en adelante, *olsrdP*) es compatible con la versión original. Esto quiere decir que se comporta como la versión original si no se especifican los nuevos argumentos que puede aceptar.

olsrdP se instala de la misma forma que la versión original. Tras descomprimir la distribución, se ejecutará *make all* y *make install*. A continuación se editará el fichero de configuración (/etc/olsrd.conf) para indicar la interfaz de red que debe utilizar el demonio. Para ejecutar *olsrdP* como la versión original, se invocará de la siguiente forma:

```
/usr/sbin/olsrd
```

Si se desea simular una topología para los seudónimos, la invocación contendrá la opción "-pseudo", seguida del nombre del fichero que contiene las direcciones IP que se utilizarán como seudónimos (una dirección en cada línea).

```
/usr/sbin/olsrd –pseudo fichero_seudonimos.txt
```

Para facilitar la tarea al usuario, también se ha implementado la posibilidad de que los seudónimos sean elegidos automáticamente por *olsrdP*. En esta situación, *olsrdP* se ejecutará durante 30 segundos como la versión original, buscando direcciones IP en uso en la red ad-hoc. Transcurrido ese tiempo, escogerá tantas direcciones IP libres como desee el usuario y las utilizará como seudónimos, creando la topología ficticia. Además, los seudónimos elegidos automáticamente se almacenan en un fichero, para ser utilizados también por el demonio HIP y permitir configurar la interfaz de red. La ejecución con elección automática de seudónimos es la siguiente:

```
/usr/sbin/olsrd –auto num_seudonimos –pseudo fichero_destino.txt
```

[1] Athens Wireless Metropolitan Network (AWMN). Puede consultarse el mapa de la red en: http://wind.awmn.net/?page=nodes

En la ejecución anterior, *num_pseudonimos* debe sustituirse por el número de seudónimos diferentes que se quieren utilizar. El último argumento indica en qué fichero deben almacenarse los seudónimos elegidos.

La simulación de la recepción de los mensajes OLSR provenientes de los vecinos ficticios se realiza de la siguiente forma. Cada nodo con seudónimos creerá estar recibiendo mensajes de tipo HELLO y TC desde cada uno de los vecinos ficticios. Cada uno de estos vecinos ficticios representa uno de sus seudónimos. Cada 5 segundos se simula la recepción de un paquete de tipo HELLO desde cada uno de los vecinos ficticios. Estos mensajes informarán de la existencia de un enlace simétrico entre el vecino ficticio y el nodo que lo recibe. Además, el vecino ficticio selecciona al nodo como MPR, por lo que este último deberá reenviar los mensajes TC recibidos del primero. También cada 5 segundos se simula la recepción de los mensajes TC, y se marcan para ser reenviados.

Muchos de los mensajes TC simulados podrán ser agrupados y reenviados en un mismo paquete OLSR junto con otros mensajes OLSR. Podrán ser agrupados los mensajes OLSR que debían enviarse próximos entre sí. Esta optimización del protocolo OLSR reduce el número de paquetes OLSR en la red y por tanto el número de veces que un nodo debe competir por el acceso al medio.

En la Figura 33 se representa la recién comentada optimización. En un instante dado, la recepción de los mensajes simulados coincide con la generación del mensaje HELLO y TC del nodo A, por lo que varios mensajes OLSR son agrupados en un único paquete.

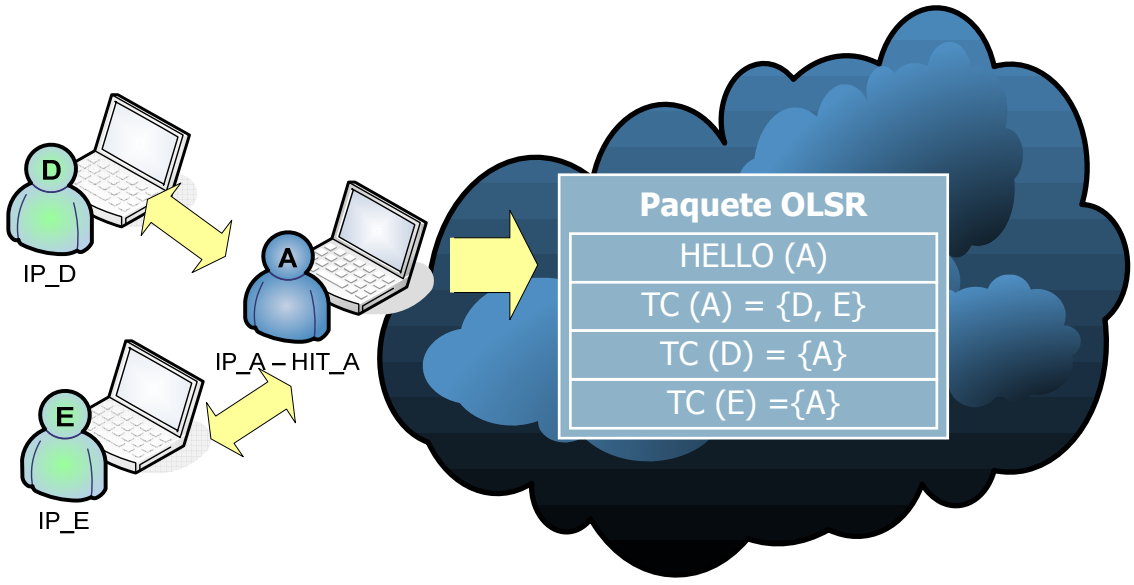


Figura 33. Optimización OLSR: múltiples mensajes OLSR en un único paquete

6.3. Configuración de la red

Cuando se desea hacer uso de los seudónimos, es necesario configurar la interfaz de red con múltiples direcciones IP. Para facilitar dicho proceso de configuración, se ha creado el script *pseudonym-manager.sh*. Puede consultarse el código del script en el Anexo IV. Este script sirve para añadir un conjunto de direcciones IP a una interfaz desde un fichero de

texto, o para eliminar múltiples direcciones IP de una interfaz de red. La llamada al script tiene la siguiente forma:

```
./pseudonym-manager.sh ra0 add fichero_ips.txt
./pseudonym-manager.sh ra0 del N
```

La primera llamada se utiliza para añadir las direcciones IP presentes en el fichero de texto (una dirección IP en cada línea) a la interfaz recibida como primer argumento (ra0). La segunda llamada debe utilizarse para borrar los seudónimos de una interfaz de red. El argumento "N" indica los seudónimos que se eliminarán: desde el seudónimo 0 (el primero que fue insertado) hasta el seudónimo N.

Además de la configuración de la interfaz de red con múltiples direcciones, es necesario filtrar algunos paquetes recibidos. En concreto, habrá que filtrar los paquetes ARP (Address Resolution Protocol) que realizan consultas sobre las direcciones IP utilizadas como seudónimos. El protocolo ARP se utiliza para obtener la dirección MAC (nivel de enlace) a la que debe enviarse un paquete cuya dirección IP destino (nivel red) se conoce. Para ello el nodo origen envía por difusión una trama ARP REQUEST (nivel 2) que contiene sus direcciones IP y MAC, y la dirección IP a consultar. Todos los nodos vecinos reciben dicha trama, pero sólo el nodo propietario de la dirección IP consultada responderá. Lo hará con una trama ARP REPLY que contendrá su dirección MAC.

En la Figura 34 se esquematiza el tráfico ARP entre un nodo atacante C (derecha) y otro nodo vecino A (izquierda) que está utilizando seudónimos. Si no se filtra ninguna trama ARP, el nodo A responderá a las consultas ARP para las direcciones IP_A, IP_D e IP_E, indicando en todas ellas la misma dirección MAC. El atacante descubriría rápidamente que se trata de un único nodo con múltiples direcciones IP, y se perdería el anonimato logrado con las modificaciones realizadas al protocolo HIP y OLSR.

Esta vulnerabilidad puede comprobarse rápidamente utilizando la herramienta *arping*[2] que permite enviar consultas ARP a la dirección IP indicada. En el ejemplo de la Figura 34, el nodo atacante ejecutaría las siguientes tres órdenes, obteniendo la misma dirección MAC en todas ellas:

```
arping IP_A -c 1 -I ra0
arping IP_D -c 1 -I ra0
arping IP_E -c 1 -I ra0
```

La solución radica en filtrar las consultas ARP dirigidas a las direcciones IP utilizadas como seudónimos. Estas tramas pueden filtrarse con la herramienta *arptables*. Siguiendo con el ejemplo de la Figura 34, el nodo A debería ejecutar la siguiente orden tantas veces como seudónimos utilice, y sustituyendo adecuadamente el campo IP_PSEUDONIMO:

```
arptables -A INPUT -d IP_PSEUDONIMO -j DROP
```

[2] *arping* forma parte del paquete *iputils*, disponible en los repositorios oficiales de Linux.

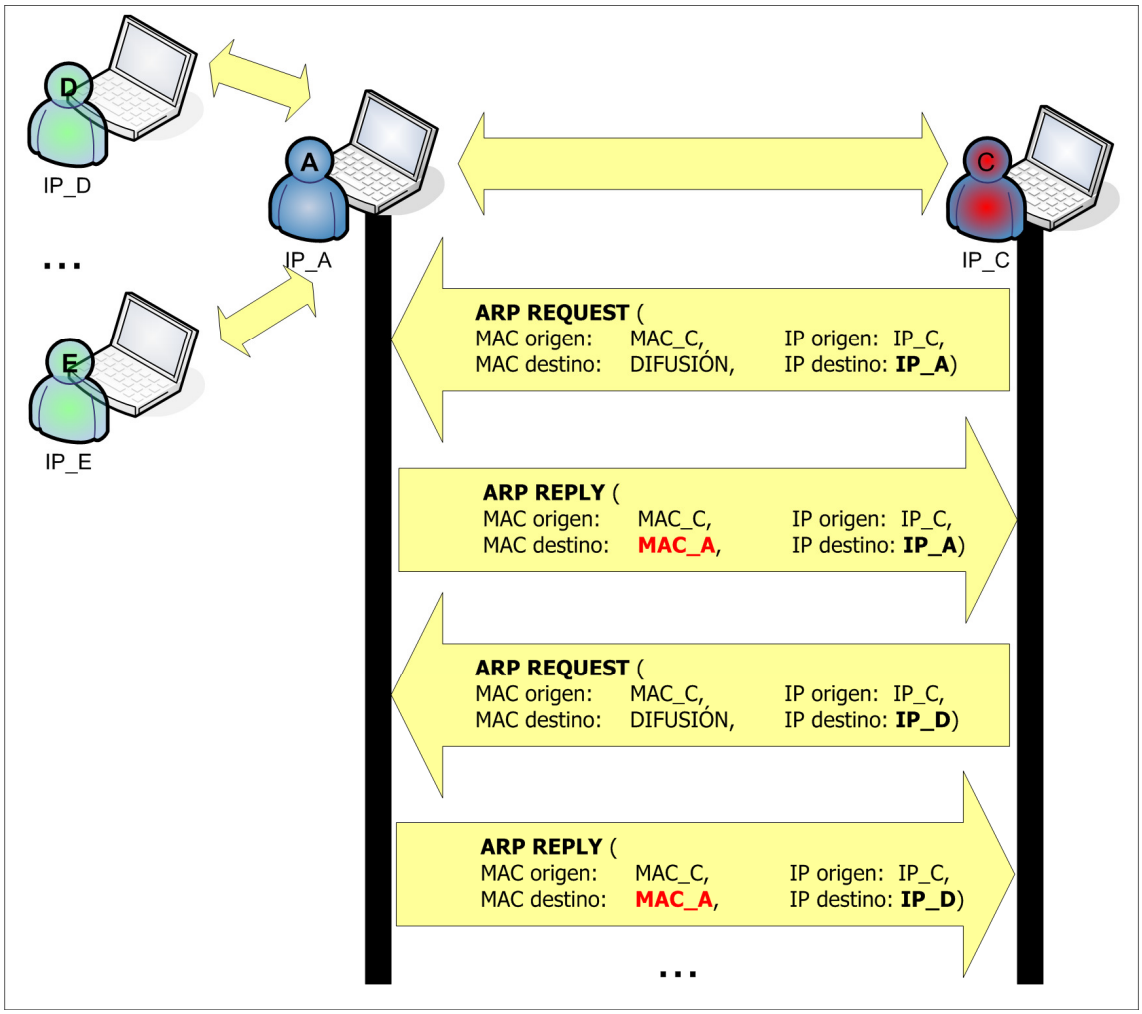


Figura 34. Ejemplo de tráfico ARP generado por un atacante

6.4. Aplicación HOP (Hip + Olsr + Pseudonyms)

Según lo explicado hasta ahora en el capítulo 6, un usuario que desee implantar la propuesta de este trabajo deberá realizar los siguientes pasos:

- **Paso 1**. Buscar direcciones IP libres para utilizarlas como seudónimos, o ejecutar *olsrdP* con la opción –auto.
- **Paso 2**. Configurar la interfaz de red con múltiples direcciones IP (tantas nuevas direcciones IP como seudónimos se hayan elegido).
- **Paso 3**. Filtrar los paquetes ARP que realicen consultas sobre las direcciones IP utilizadas como seudónimos, para evitar ataques al anonimato de las estaciones.
- **Paso 4**. Si se han escogido los seudónimos manualmente, ejecutar *olsrdP* con la opción –pseudo, indicando el nombre del fichero que contiene los seudónimos. Si las direcciones IP han sido elegidas automáticamente con *olsrdP* (en el paso 1), no es necesario realizar este paso.
- **Paso 5**. Ejecutar HIP con la opción –pseudo, indicando también el nombre del fichero que contiene los seudónimos.

Los pasos anteriores pueden resultar incómodos para el usuario final, que tendrá que llevar a cabo una configuración laboriosa cada vez que desee hacer funcionar nuestra propuesta. Además, cuanto mayor es el número de pasos necesarios para hacer funcionar una aplicación, mayor es la probabilidad de que alguno de ellos se olvide, obteniendo así una solución incorrecta.

Con el fin de agilizar la ejecución, se ha desarrollado un script que realiza automáticamente todos los pasos. Este script se ha empaquetado junto con las versiones modificadas de HIP y OLSR, y con la documentación correspondiente. El resultado es un fichero comprimido listo para ser distribuido. Este trabajo final ha recibido el nombre de HOP (Hip + Olsr + Pseudonyms).

El código fuente del script que realiza todos los pasos puede consultarse en el Anexo V. Gracias a él, el usuario sólo deberá ejecutar un script cada vez que quiera hacer funcionar el sistema. Este script, que se denomina hop_1.0, puede ejecutarse de dos formas:

```
./hop_1.0 < interfaz > –manual <nombre_fichero_pseudonimos>
./hop_1.0 < interfaz > –auto <número_pseudonimos>
```

En ambas ejecuciones hay que indicar como primer argumento el nombre de la interfaz de red. Con la primera orden de ejecución, los seudónimos son elegidos manualmente por el usuario, y se debe especificar el nombre del fichero que los contiene. En la segunda opción, la elección de los seudónimos es automática, y el usuario sólo debe indicar cuántos seudónimos desea que se elijan. Resulta más cómoda la segunda opción de ejecución.

7. Validación y Pruebas

Con el objetivo de verificar el correcto funcionamiento de la aplicación, así como medir sus prestaciones, se han llevado a cabo las pruebas que se describen a continuación.

7.1. Banco de pruebas

7.1.1. Hardware y Sistema Operativo

La red ad-hoc en la que se realizan las pruebas se compone de 5 ordenadores portátiles eeePC 901, cuyas características técnicas son las siguientes:

- Procesador Intel Atom N270: 1.60GHz, 512KB cache L2, tecnología Hyper-Threading.
- Memoria RAM: 1 GB DDR2.
- Conectividad:
  - WiFi 802.11 b/g/n (tarjeta Ralink RT2860 PCIe).
  - 10/100 Ethernet.
- Disco duro de estado sólido (SSD), con capacidad 20 GB.
- Pantalla de 8.9 pulgadas, con resolución 1024x600px.
- Peso: 1100 gramos.

Figura 35. Ordenador portátil eeePC 901

Estos ordenadores (Figura 35) se venden con una distribución Linux ya instalada, llamada Xandros. Esta distribución está pensada para sacar el máximo partido, a nivel de usuario, a un equipo de reducidas dimensiones: iconos grandes, accesos directos a aplicaciones web, etc. Sin embargo, el uso de esta distribución no está muy extendido, y en ella han sido limitadas algunas de las funcionalidades presentes en la mayoría de distribuciones de Linux, así como la instalación de nuevas aplicaciones.

Por estos motivos se ha optado por instalar otra distribución de Linux. Se han fijado los siguientes requisitos para la nueva distribución:

- fácil de instalar y de utilizar,
- compatible con el hardware de los equipos,
- de uso extendido entre los usuarios de Linux, y
- con un bajo consumo de recursos debido a las prestaciones de los equipos.

En un primer momento se pensó en la distribución Ubuntu, ya que su uso ha crecido en los últimos meses y actualmente es la distribución que goza de mayor popularidad[3]. Además, como se buscaba un sistema operativo que consumiera pocos recursos, se optó por instalar la distribución Xubuntu.

Xubuntu es una distribución oficial basada en Linux Ubuntu, que utiliza el entorno de escritorio Xfce, en lugar de GNOME. Xubuntu está diseñado para ser utilizado en ordenadores que poseen recursos limitados, o para usuarios que buscan un entorno de escritorio altamente eficiente.

Sin embargo, ni la versión de Xubuntu disponible en el momento de las pruebas (Xubuntu 8.04.1), ni la nueva versión candidata para ser lanzada (Xubuntu 8.10 rc) reconocían correctamente todo el hardware de los eeePC. Tampoco fueron satisfactorios los resultados con la versión actual de Ubuntu (Ubuntu 8.04).

Finalmente se encontró una distribución de Ubuntu diseñada específicamente para estos ordenadores, Ubuntu 8.04 eee. Esta versión, finalmente, cumple todos los requisitos anteriores.

En la Tabla 1 se muestran los sistemas operativos probados y el hardware reconocido correctamente por los mismos.

Tabla 1. Resumen de los Sistemas Operativos probados

Sistema Operativo	***ETHERNET***	***WiFi***	***SONIDO***	***T. GRÁFICA***
Xubuntu 8.04.1	NO	NO	NO	NO
Xubuntu 8.10 rc	SI	NO	NO	NO
Ubuntu 8.04	NO	NO	SI	SI
Ubuntu 8.04 eee	SI	SI	SI	SI

7.1.2. Simulación de la red ad-hoc

Las pruebas se han llevado a cabo en un laboratorio, por lo que las distancias entre los ordenadores eran mínimas. En un escenario así, todos los nodos son visibles entre ellos, se comunicarían directamente, y ninguna máquina realizaría tareas de encaminamiento de paquetes.

Sin embargo, la red ad-hoc que se quiere simular debe tener la topología que se muestra en la imagen (Figura 36). En dicha red es fácil medir tiempos de inicio de sesión, productividad, etc... en función del número de saltos que deben realizar los paquetes.

[3] Fuente: http://distrowatch.com. Datos sobre el uso de Linux en 2008.

Figura 36. Topología de red ad-hoc que se desea simular. Topología en cadena

Con el fin de obtener la topología deseada sobre la que realizar las mediciones, se ha configurado el cortafuegos de cada nodo para simular distancias. Concretamente, se ha utilizado IPTABLES [7]. Esta herramienta es un cortafuegos para Linux. Entre otras cosas, permite filtrar paquetes en base a los criterios que se le indiquen.

Nuestro objetivo es simular que hay nodos lo suficientemente alejados como para que algunas comunicaciones requieran ser encaminadas a través de otros nodos. Llamaremos "nodos no visibles desde X" a aquellos nodos con los que no podemos comunicarnos directamente desde el nodo X.

Inicialmente se podría pensar que para conseguir esto bastaría con filtrar en el nodo X los paquetes entrantes cuya dirección IP origen corresponde a un nodo no visible desde X. Esta solución es errónea, ya que los nodos intermedios, con funciones de encaminamiento, no modifican las cabeceras IP. El resultado obtenido sería un escenario en el que los nodos no visibles no pueden comunicarse aunque entre ellos haya nodos capaces de encaminar sus paquetes correctamente.

La solución correcta es filtrar a nivel de la capa de enlace (nivel 2). Es decir, se deben filtrar los paquetes entrantes cuya dirección MAC corresponda a un nodo no visible. En la Figura 37 se muestra gráficamente el encaminamiento de un paquete IP a través de un nodo intermedio. Los campos IP origen y destino de la cabecera IP no se ven alterados, pero sí que se modifican los campos MAC origen y destino de la cabecera de la trama MAC.

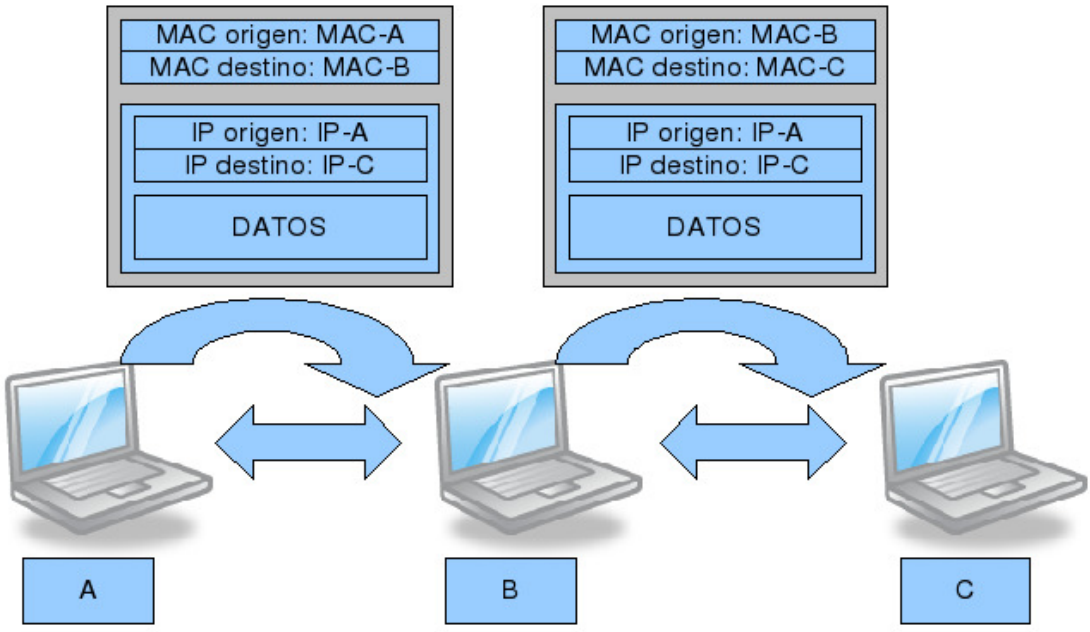


Figura 37. Encaminamiento de un paquete IP en una red ad-hoc

Si se desea que los nodos A y C no sean visibles entre ellos, deberán ejecutarse las siguientes órdenes (la primera en el nodo A y la segunda en el nodo C). El texto "eth0" debe sustituirse por el nombre de la interfaz de red en cada una de las máquinas:

```
iptables –A INPUT –m mac --mac-source MAC_C –i eth0 –j DROP
iptables –A INPUT –m mac --mac-source MAC_A –i eth0 –j DROP
```

También es necesario deshabilitar la recepción y el envío de los mensajes ICMP de tipo *redirect*. Estos mensajes son enviados por los nodos intermedios cuando descubren que están encaminando tráfico entre dos máquinas que se encuentran en la misma red. En este caso, el nodo intermedio envía un mensaje ICMP *redirect* para notificar a la máquina emisora de la existencia de una ruta alternativa (en este caso, una ruta directa entre emisor y receptor). Es necesario ejecutar estas dos órdenes en cada nodo de la red (sustituyendo "eth0" por el nombre de la interfaz utilizada para la red ad-hoc):

```
echo 0 > /proc/sys/net/ipv4/conf/eth0/accept_redirects
echo 0 > /proc/sys/net/ipv4/conf/eth0/send_redirects
```

La red ad-hoc se ha construido con las interfaces WiFi, funcionando con la especificación 802.11g (54Mbps). Se ha elegido para la red un canal de frecuencia no utilizado por ninguna otra red WiFi cercana.

La simulación de redes ad-hoc mediante el filtrado de paquetes explicado anteriormente aumenta en complejidad a medida que aumenta el número de nodos de la red. Generalmente, cuanto mayor es el número de nodos, mayor será el número de filtros que se deben aplicar. Para agilizar el proceso de configuración de la red ad-hoc, así como para evitar posibles errores, se ha desarrollado un generador online de scripts para simular redes ad-hoc[4].

Este generador online solicita al usuario el número total de nodos de la red, sus respectivas direcciones IP y MAC, y una lista con los enlaces existentes en la red. Como resultado, se generará un único script que debe ser ejecutado en todos los nodos de la red. Este script recibe como parámetros el número del nodo en el que se está ejecutando el código y el nombre de la interfaz de red:

```
./script_autogenerado.sh <numero_nodo> <interfaz_red>
```

El script borrará las reglas de IPTABLES, realizará el filtrado necesario a nivel de MAC para simular la topología de la red (filtrará diferentes direcciones según el parámetro `<numero_nodo>`), borrará las posibles rutas de encaminamiento, y deshabilitará los mensajes ICMP *redirect*. En la Figura 38 se muestra el aspecto de este generador online.

[4] El generador online automático de scripts está disponible en la dirección http://personales.alumno.upv.es/fracambe/adhocgenerator

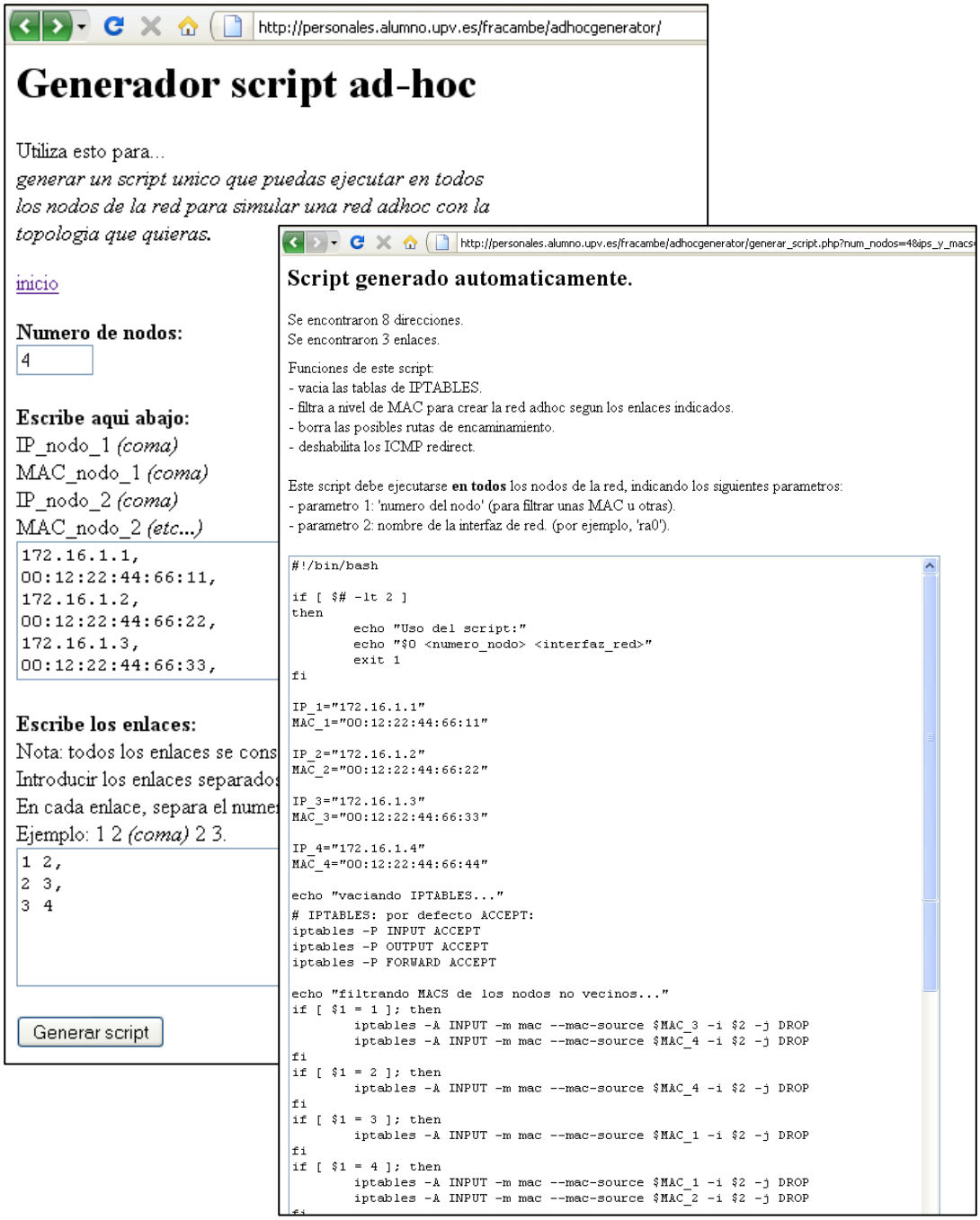


Figura 38. Aspecto del generador online de scripts para simular redes ad-hoc

7.2. Enrutamiento estático y dinámico

En el enrutamiento estático, las tablas de encaminamiento de cada uno de los nodos de la red son inicialmente rellenadas manualmente y no se ven modificadas a lo largo del tiempo. Esta solución sólo es válida en escenarios en los que los nodos no vayan a cambiar de posición, de lo contrario sería necesario actualizar las tablas de enrutamiento. Es la solución más sencilla, ya que no se requiere un protocolo de encaminamiento.

El enrutamiento dinámico es aquél que se logra utilizando un protocolo de encaminamiento. Este protocolo se encarga de mantener actualizadas las tablas de

enrutamiento, aunque la topología de la red varíe. En los protocolos de encaminamiento proactivos, como OLSR, cada nodo de la red envía periódicamente mensajes de control para mantener las tablas actualizadas.

En este apartado se estudia si el envío de los mensajes de control del protocolo OLSR afecta al resto de tráfico de la red ad-hoc. Se utilizará una topología de red como la de la Figura 36 formada únicamente por 4 nodos.

Según lo establecido en la especificación del protocolo OLSR [6], que coincide con los tiempos predeterminados en la implementación *olsrd*, cada nodo debe enviar sus mensajes HELLO cada 2 segundos y los TC cada 5 segundos, por defecto. Es decir, cada nodo envía mensajes propios a una frecuencia de 0.7 mensajes/segundo. Adicionalmente, algunos nodos deberán propagar mensajes TC recibidos desde vecinos que los seleccionaron como MPRs. Cuando se utiliza *olsrdP* (olsrd con soporte a seudónimos), cada nodo deberá reenviar además un mensaje TC por cada uno de los seudónimos que posea.

A continuación se compara, en diferentes escenarios, el número total de paquetes OLSR enviados por la red. Todas las mediciones se han realizado en periodos de 60 segundos, y una vez que los 4 nodos de la red eran conocedores de la topología completa de la red. En el primer escenario (Tabla 2) se utiliza la versión original de OLSR. En el escenario 2 (Tabla 3) cada nodo hace uso de dos seudónimos, mientras que en el escenario 3 (Tabla 4) hay 3 seudónimos por cada nodo.

Tabla 2. Escenario1: Protocolo OLSR original

Nodo de la red	***Nº paquetes enviados***	***Bytes enviados***
1 (extremo izda)	43	3294
2	45	5038
3	46	5044
4 (extremo dcha)	40	3100
Total	**174**	**16476 (94.7 Bytes/paquete)**

Tabla 3. Escenario 2: Protocolo OLSR con 2 seudónimos en cada nodo

Nodo de la red	***Nº paquetes enviados***	***Bytes enviados***
1 (extremo izda)	52	8368
2	51	8930
3	53	8878
4 (extremo dcha)	58	8972
Total	**214**	**35148 (164.2 Bytes/paquete)**

Tabla 4. Escenario 3: Protocolo OLSR con 3 seudónimos en cada nodo

Nodo de la red	***Nº paquetes enviados***	***Bytes enviados***
1 (extremo izda)	49	9958
2	57	10782
3	56	10472
4 (extremo dcha)	51	10138
Total	**213**	**41350 (192.1 Bytes/paquete)**

En el escenario 1, la topología está formada por 4 nodos. En el escenario 2 está compuesta por 12 (4 reales y 8 ficticios). Por último, en el escenario 3, la red es de 16 nodos (4 reales y 12 ficticios). Los mensajes HELLO simulados en los escenarios con seudónimos no se transmiten a la red, ni son reenviados por los nodos reales. Por lo tanto, en los 3 escenarios el número de mensajes HELLO es el mismo:

Nº Total mensajes HELLO = Nº nodos reales x Frecuencia envío HELLO x Tiempo
= 4 x 0.5 x 60 = 120 mensajes HELLO

En cuanto a los mensajes TC, estos son reenviados por los nodos seleccionados como MPRs. Por ejemplo, un nodo con seudónimos reenviará los mensajes TC de sus seudónimos (previamente se habrá simulado la recepción de estos), además de enviar su propio TC y su propio HELLO.

Teóricamente, el número de mensajes enviados por cada nodo en el escenario 1 es el siguiente:

Nodo 1: 12 mensajes TC propios
Nodo 4: 12 mensajes TC propios
Nodo 2: 12 TC propios + 12 TC (1) + 12 TC (3) + 12 TC (4) = 48 mensajes TC
Nodo 3: 12 TC propios + 12 TC (1) + 12 TC (2) + 12 TC (4) = 48 mensajes TC

En total, en el escenario 1 se envían 120 mensajes TC y 120 mensajes HELLO. El número total de mensajes (240) es muy superior al numero real de paquetes OLSR enviados en el escenario 1 (174). Esto se debe a que los mensajes OLSR, antes de ser enviados, son encolados. Los mensajes encolados en un mismo intervalo de tiempo t, son encapsulados dentro de un único paquete OLSR. Según los datos del escenario 1, se han encapsulado un promedio de 1.38 mensajes OLSR en cada paquete OLSR.

En el escenario 2 el número de mensajes TC enviados es mucho mayor. Teóricamente, en este escenario se envían aproximadamente 360 mensajes OLSR de tipo TC. El número teórico de mensajes OLSR enviados es de 480, una cifra muy superior al número real de paquetes OLSR enviados. La causa es la misma que en el escenario 1. En este caso, en cada paquete OLSR se han encapsulado un promedio de 2.24 mensajes OLSR. En la Tabla 3 se puede comprobar cómo ha aumentado el tamaño medio de los paquetes en este escenario respecto al escenario 1.

En el escenario 3 el número total de paquetes OLSR se mantiene igual que en el escenario 2, mientras que el número de mensajes TC enviados ha aumentado. Teóricamente, el número de mensajes TC en este escenario es de 480. El número total de paquetes no ha variado respecto al escenario 2 debido a que los mensajes TC de los seudónimos se simulan todos al mismo tiempo, permitiendo así que sean encapsulados en un mismo paquete OLSR.

A continuación, en la Figura 39, se representa gráficamente el número de paquetes y mensajes OLSR en los 3 escenarios. El número total de paquetes OLSR enviados es menor que la suma de mensajes HELLO y mensajes TC enviados, ya que varios mensajes OLSR pueden ser encapsulados dentro de un único paquete OLSR.

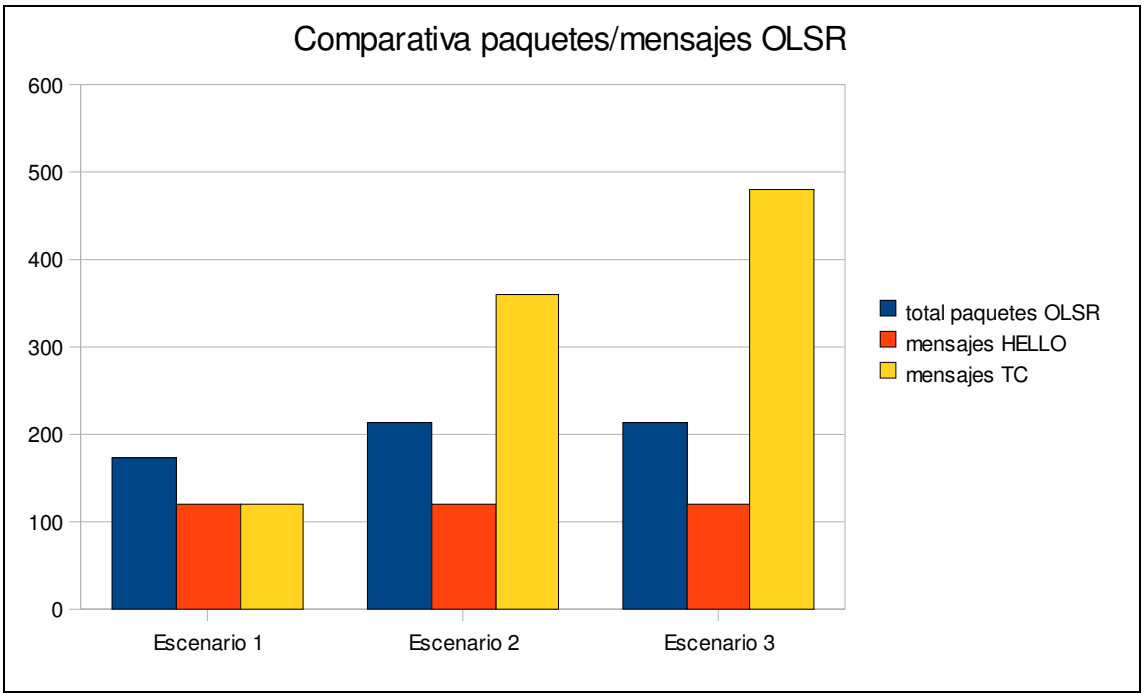


Figura 39. Comparativa entre nº de paquetes y mensajes OLSR

La cabecera de los mensajes TC y HELLO es de 12 bytes. El tamaño (en bytes) del cuerpo del mensaje TC varía con el número de vecinos (N) del nodo que lo origina, siguiendo la fórmula $4 + 8\ N$. En la Figura 40 se observa el crecimiento lineal del tamaño de los mensajes TC.

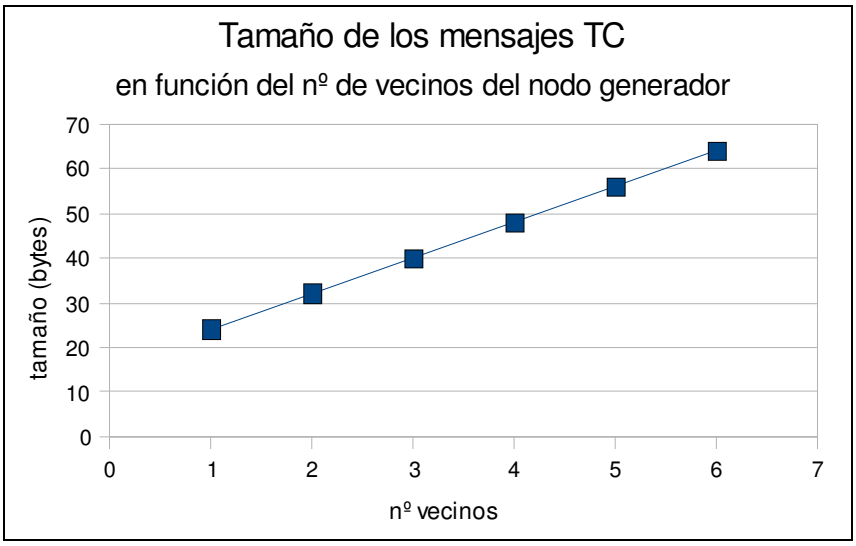


Figura 40. Tamaño de los mensajes TC

El tamaño de los mensajes HELLO también depende del número de vecinos (N) del nodo que origina el mensaje, aunque para estos mensajes no se puede establecer un tamaño exacto. Éste será el menor posible si los enlaces con todos los vecinos son del mismo tipo (por ejemplo, simétricos y MPR), mientras que el peor caso es aquel en el que todos los enlaces son diferentes. El tamaño del mensaje viene acotado por las siguientes ecuaciones, y se representa en la Figura 41:

Caso mejor: Tamaño = 20 +8 N
Caso peor: Tamaño = 16 + 12 N

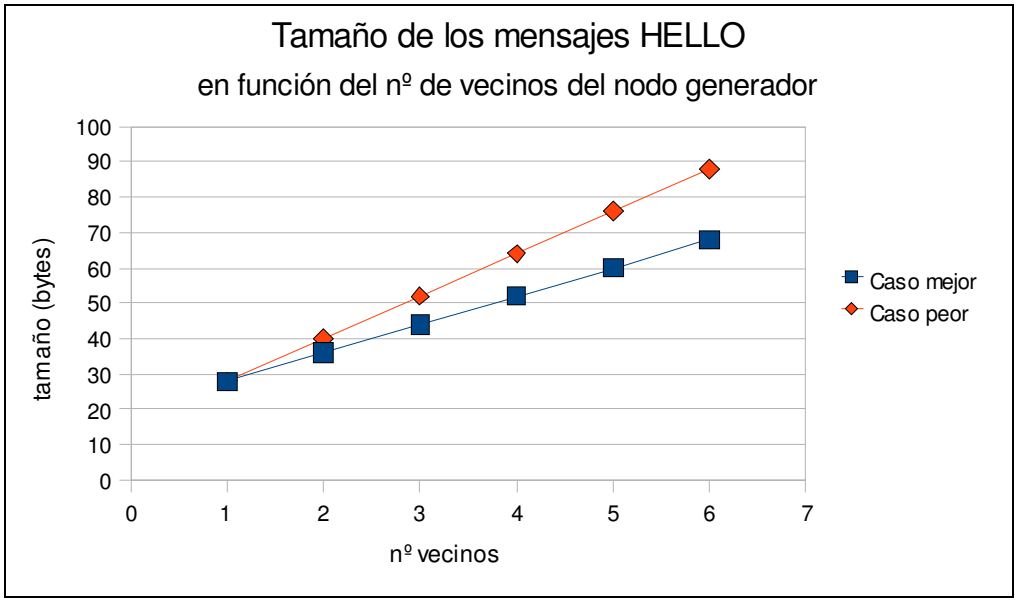


Figura 41. Tamaño de los mensajes HELLO

Se puede concluir que si se aumenta el número de vecinos de los nodos, aumentará el número total de mensajes TC y HELLO, y aumentará también el tamaño de dichos mensajes. Además, se incrementará el número de paquetes OLSR, pero lo hará en menor medida, ya que los mensajes OLSR tienden a agruparse en paquetes OLSR. En consecuencia, el tamaño promedio de los paquetes crecerá más significativamente que el tamaño de los mensajes.

A continuación, con los datos obtenidos de los tres escenarios, se ha calculado la utilización de la red cuando sólo hay tráfico OLSR. En primer lugar, es necesario conocer la tasa de transferencia de los paquetes OLSR, en cada uno de los 3 escenarios. Este valor se obtiene dividiendo el número total de bytes de paquetes OLSR enviados entre la duración de la prueba (60 segundos). Los resultados se resumen en la Tabla 5:

Tabla 5. Tasa de transferencia de paquetes OLSR

# Escenario	***Tasa de transferencia (bit/s)***
Escenario 1 (OLSR original)	2196 bits/s
Escenario 2 (2 seudónimos por nodo)	4606 bits/s
Escenario 3 (3 seudónimos por nodo)	5513 bits/s

Además, se debe calcular la tasa de transferencia máxima. Es decir, la tasa de transferencia cuando la utilización de la red es del 100%. Para ello, se ha realizado la transferencia de un fichero desde un extremo al otro de la red. Se ha utilizado encaminamiento estático para realizar esta medición, y así obtener la tasa de transferencia máxima de un fichero sin que ésta se vea afectada por la posible sobrecarga del protocolo de encaminamiento. El resultado de esta prueba ha sido una tasa de 6.88 Mbps. Al tratarse de una topología de red en cadena, y como la distancia entre los dos extremos es de 3 saltos, la tasa máxima en la

red será de 20.6 Mbps aproximadamente (6.88 x 3). Este experimento se detalla con más profundidad en la sección 7.5. Throughput.

A partir de los resultados obtenidos, se confirma que la utilización de la red es insignificante cuando sólo está presente el tráfico del protocolo de encaminamiento OLSR. Tampoco afectará al rendimiento de la red la utilización de seudónimos. En la Tabla 6 se listan los valores de utilización de la red en cada uno de los escenarios.

Tabla 6. Utilización de la red por el tráfico de control OLSR

# Escenario	***Utilización de la red (%)***
Escenario 1 (OLSR original)	0.01 %
Escenario 2 (2 seudónimos por nodo)	0.021 %
Escenario 3 (3 seudónimos por nodo)	0.026 %

A continuación se estudia el tiempo de inicio de sesión HIP, el RTT (Round Trip Delay Time) y el rendimiento en la red ad-hoc con la topología simulada (Figura 36). Los siguientes resultados se han obtenido utilizando encaminamiento estático.

7.3. Session Startup

El tiempo de inicio de sesión, o Session Startup es el tiempo transcurrido desde que se desea iniciar una asociación HIP hasta que se recibe y se termina de procesar el paquete R2.

El establecimiento de una asociación HIP implica el intercambio de 4 paquetes: I1, R1, I2 y R2. Esto requiere un tiempo de cómputo y de transferencia a través de la red. Con la modificación del protocolo HIP, los paquetes HIP se envían cifrados, por lo que el tiempo necesario para cifrar y descifrar estos paquetes será sumado al tiempo de inicio de sesión de la versión original.

En la Tabla 7 se resume el tiempo empleado para cifrar y descifrar los distintos tipos de paquetes. La variación en el tiempo empleado en cada tipo de paquete se debe a la diferencia en el tamaño de estos. Así, los paquetes HIP más pequeños son los de tipo I1, y los más grandes los de tipo I2.

Tabla 7. Tamaño de los paquetes HIP y tiempos de cifrado

Tipo paquete HIP	***Tiempo cifrar (ms)***	***Tiempo descifrar (ms)***
I1 (40 bytes)	0.56	1.31
R1 (608 bytes)	3.7	74.8
I2 (659 bytes)	4.6	96
R2 (216 bytes)	1.2	47.8
Close / Close ack (208 bytes)	1.2-1.9	22

En la Figura 42 se muestra la comparación entre los tiempos de inicio de sesión utilizando la versión HIP original y la versión HIP modificada (HIP MANET), en función del número de saltos.

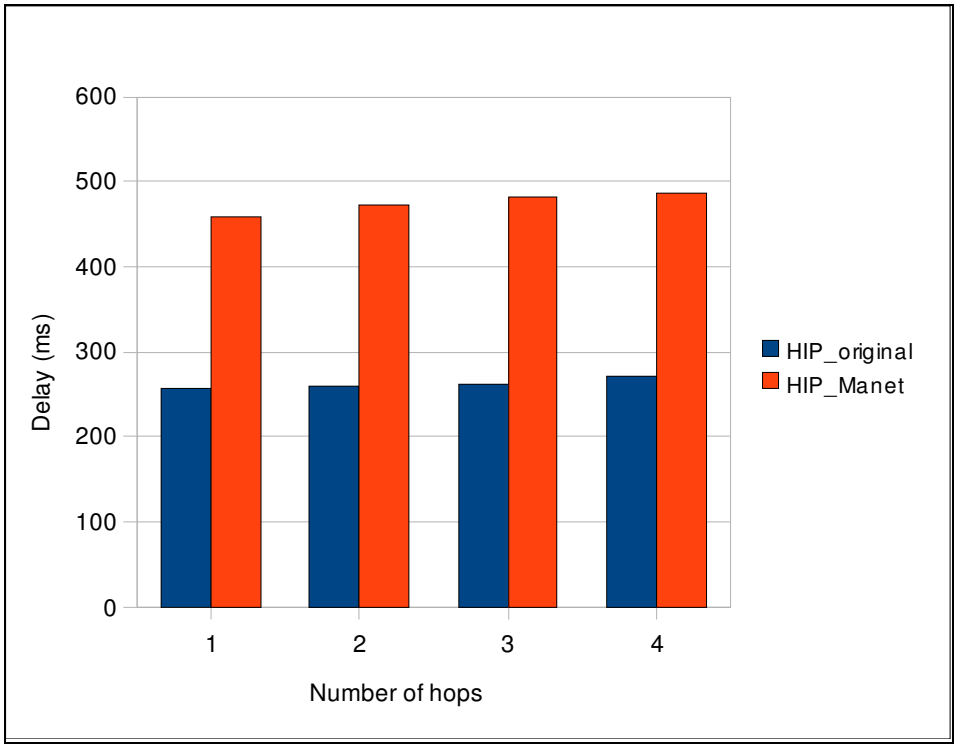


Figura 42. Session Startup

Lo primero que se observa en la imagen es el considerable aumento del tiempo de inicio de sesión al utilizar la versión HIP MANET. Este aumento es consecuencia del cifrado de los paquetes HIP. De hecho, si se suma el tiempo empleado para cifrar y descifrar los paquetes I1, R1, I2 y R2 se obtiene un retardo de 230 ms. Este tiempo se ajusta aproximadamente a la diferencia existente entre ambas versiones cuando el número de saltos es 1.

Además, en la versión original de HIP (en color azul) el retardo aumenta mínimamente al aumentar el número de saltos. Esto se debe a que el encaminamiento de los paquetes es una tarea muy rápida, en comparación con el resto (creación de los paquetes HIP, resolución del puzzle, comprobación de los campos firma y *HMAC*, etc). En cambio, en la versión HIP MANET (en color rojo), el retardo aumenta de forma ligeramente mayor. La causa de este aumento es el envío del paquete I1 cifrado y por broadcast. Es decir, en la versión modificada, cada nodo intermedio debe descifrar el paquete I1 y comprobar si es el destinatario final, mientras que en la versión original los paquetes son encaminados directamente.

En el caso concreto de asociaciones HIP con un salto, la versión modificada de HIP supone un aumento del 78 % en el tiempo de establecimiento de la asociación respecto a la versión original de HIP.

7.4. Round Trip Delay Time

Round Trip Delay Time (RTT) es el tiempo que tarda un paquete enviado desde un emisor en llegar a su destino y ser devuelto al emisor inicial. Esta medida es importante ya que

muchas aplicaciones y protocolos requieren el envío de pequeños paquetes que son contestados por los receptores para confirmar su recepción. El RTT estima la eficiencia de este mecanismo en una red real.

Para el cálculo del RTT se ha utilizado el comando ping. Esta herramienta envía al destino indicado un mensaje ICMP de tipo *ECHO_REQUEST*. El receptor del mensaje responderá con un mensaje ICMP del mismo tamaño, pero de tipo *ECHO_REPLY*.

En la Figura 43 se muestran los resultados obtenidos en función del tamaño de los mensajes y del número de saltos entre el nodo origen y destino. Se han enviado los mensajes ICMP a través de una conexión sin cifrar. En cambio, en la Figura 44 se muestran los resultados de realizar el mismo experimento a través de una conexión cifrada por HIP (ESP). Las mediciones en el caso de HIP se han realizado tras establecer previamente la asociación HIP. Es decir, los resultados no se han visto afectados por el envío de los paquetes I1, R1, I2 y R2.

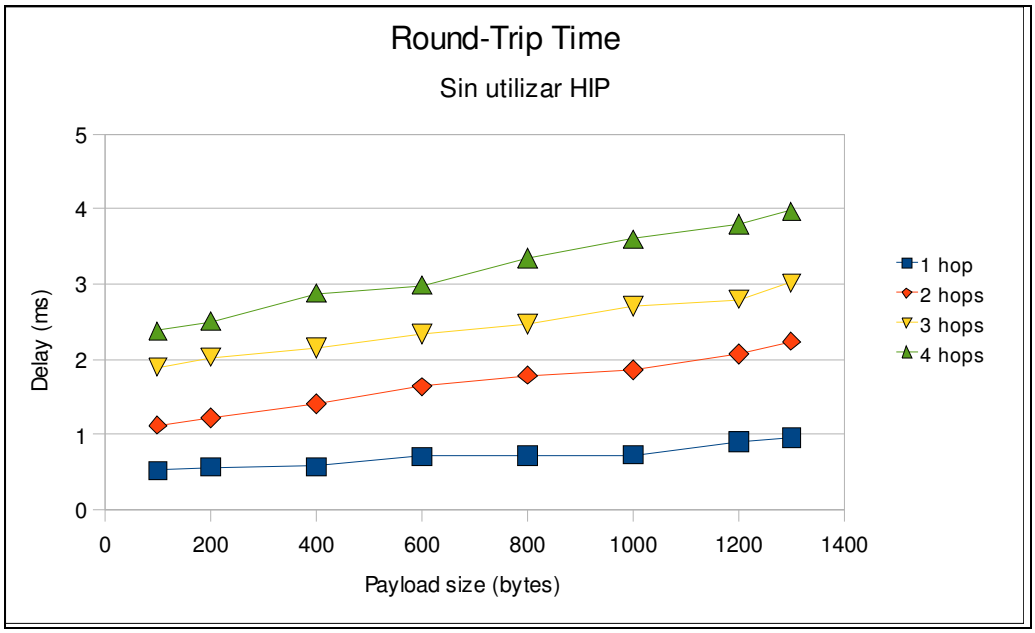


Figura 43. RTT sin utilizar el protocolo HIP

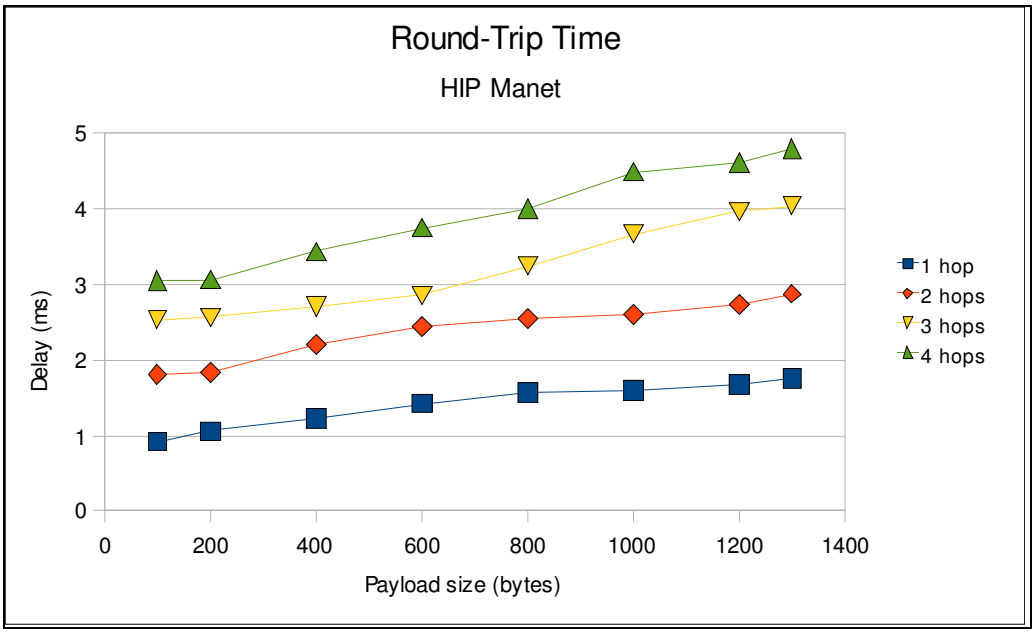


Figura 44. RTT utilizando el protocolo HIP

El tamaño de los mensajes ICMP varía entre los 100 bytes y 1300 bytes, permitiendo así que sean encapsulados dentro de un único datagrama IP. Tamaños mayores requerirían fraccionar el datagrama IP, ya que las tramas de la capa de enlace sólo pueden contener hasta 1500 bytes de datos. De este modo, en las conexiones con múltiples saltos, los nodos intermedios deberán recibir todo el mensaje ICMP antes de reenviarlo. El retardo que esto supone se observa fácilmente en los resultados de la Figura 43. Concretamente para los mensajes de 1300 bytes, donde el retardo es mayor y está menos expuesto a posibles errores de precisión, se observa cómo el tiempo aumenta linealmente con el número de saltos, en el caso sin HIP: un salto requiere aproximadamente 1ms, cuatro saltos consumen 4ms. Este crecimiento no es tan lineal en la Figura 44 debido al coste añadido de cifrar y descifrar los paquetes en los nodos origen y destino.

Además de aumentar el retardo cuando el número de saltos crece, éste también se incrementa cuando los paquetes enviados son de mayor tamaño. En la Figura 43 se comprueba que la variación de 100 bytes a 1300 bytes supone un incremento de 0.4 ms en el RTT para conexiones de un salto no cifradas. Si la conexión es de 4 saltos, el incremento es de 1.6 ms, que coincide con lo que cabía esperar: 0.4 x 4 saltos.

Si en lugar de enviar un único paquete IP, se enviasen múltiples paquetes, ocurriría lo siguiente. En algún instante, el emisor estaría enviando el paquete X y simultáneamente el siguiente nodo intermedio estaría intentando enviar el paquete X-k. Se produciría entonces una competición por el acceso al medio, lo que supondría un retardo en la entrega de los paquetes.

Comparando ambas gráficas se observa que la tendencia es similar. La diferencia radica en que al utilizar la conexión cifrada (Figura 44) se requiere un tiempo adicional para cifrar los paquetes en el emisor y descifrarlos en el receptor. Este tiempo extra sólo depende del tamaño de los paquetes y no del número de saltos, ya que los nodos intermedios se limitan a encaminar el tráfico, sin descifrarlo. En las gráficas se comprueba que el sobrecoste de la conexión cifrada es de 0.8 ms para paquetes de 1300 bytes (independientemente del

número de saltos). Para una conexión de 4 saltos, el cifrado supone un aumento del 20 % en el RTT. Este sobrecoste es asumible, incluso en aplicaciones con requisitos de tiempo real.

7.5. Throughput

Se llama *throughput* al volumen de información que logra alcanzar con éxito al nodo destino. A continuación se compara el *throughput* entre una conexión segura utilizando HIP y una conexión no cifrada. Concretamente, se ha estudiado la tasa de transferencia de datos utilizando el protocolo HTTP.

Se ha utilizado la aplicación *woof*[5] como servidor web y la herramienta *wget*[6] como cliente. En ambos casos se trata de aplicaciones gratuitas, de código abierto, pequeñas y muy ligeras. Desde el principio se pensó en utilizar aplicaciones pequeñas y específicas para reducir así la probabilidad de fluctuaciones en el rendimiento, causadas por motivos ajenos a los estudiados. Además, las características técnicas del banco de pruebas recomiendan utilizar software que no requiera muchos recursos. Los ficheros que se han transferido durante las pruebas tenían un tamaño aproximado de 100 MB.

En la Figura 45 se compara el *throughput* entre una conexión sin cifrado (en color azul) y una conexión segura utilizando HIP. Las mediciones persiguen comparar el rendimiento en el envío de datos, por lo que se han hecho, para el caso de HIP, tras establecer previamente la asociación HIP. Es decir, el rendimiento que se muestra en la gráfica no se ha visto afectado por el envío de los paquetes I1, R1, I2 y R2.

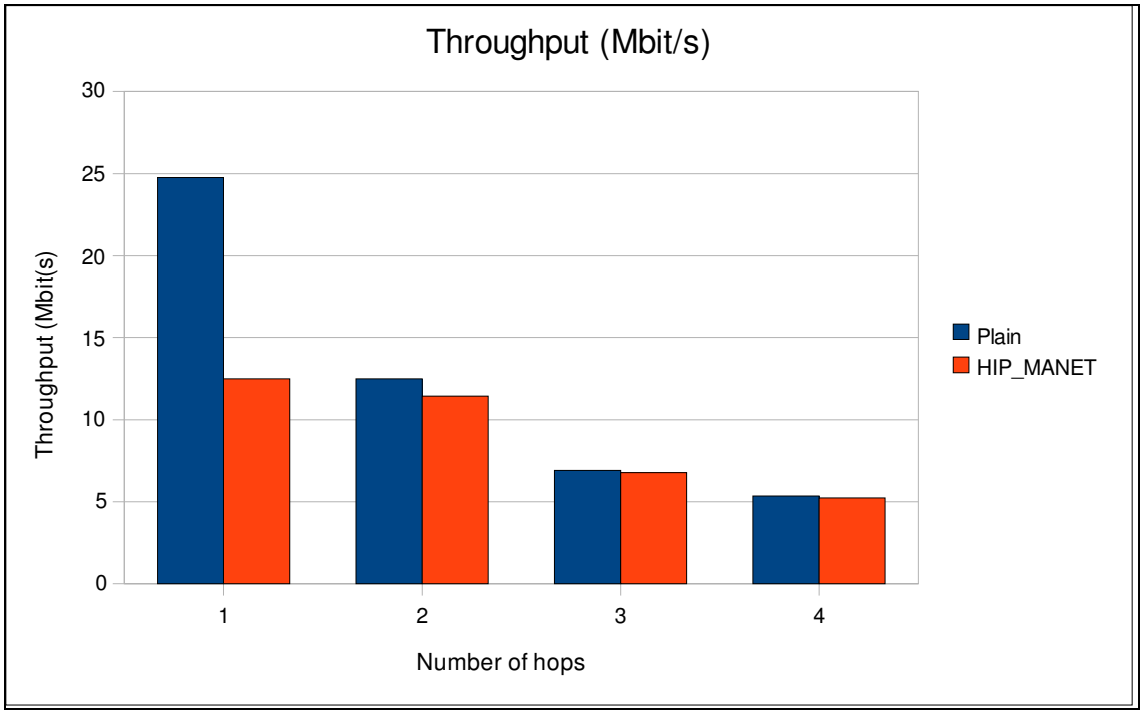


Figura 45. Throughput

[5] woof es un trabajo de Simon Budig. Está disponible en http://www.home.unix-ag.org/simon/woof.html.
[6] Toda la información sobre wget puede encontrarse en el sitio web http://www.gnu.org/software/wget.

Tal y como era de esperar, el *throughput* es menor al utilizar una conexión cifrada. Hay que recordar que en el protocolo HIP, tras establecer la asociación, los datos se envían cifrados con el protocolo ESP, garantizando así la confidencialidad, la integridad y la autenticidad de los paquetes. El protocolo ESP utiliza criptografía simétrica por ser más eficiente computacionalmente que la criptografía asimétrica, siendo esta última utilizada únicamente durante el establecimiento de la asociación HIP.

Los resultados muestran que la conexión sin cifrar (en color azul) con un solo salto alcanza el *throughput* promedio de la especificación 802.11g (en torno a los 22Mbps). También con un salto, el protocolo HIP (color rojo) obtiene un *throughput* que no sobrepasa los 12Mbps. Es decir, en conexiones de un salto, el *throughput* desciende al 50% al utilizar el cifrado de datos del protocolo HIP (ESP).

A medida que aumenta el número de saltos, el *throughput* de ambas conexiones se reduce. Concretamente, el *throughput* de la conexión sin cifrar se reduce aproximadamente a la mitad cuando el número de saltos es dos, disminuye hasta un tercio cuando hay tres saltos, y alcanza un cuarto de su valor inicial cuando el número de saltos es cuatro. Esto ocurre porque en la conexión sin cifrar el cuello de botella viene dado por el medio de transmisión. Es decir, en dicha conexión las máquinas transmitirán la información tan rápido como sea posible por el medio. Pero a medida que aumenta el número de saltos, aumenta también el número de máquinas intentando acceder al medio al mismo tiempo. Cuando más de una máquina desea enviar en un mismo instante, se produce un mecanismo de competición entre ellas para evitar colisiones y conseguir que finalmente sólo envíe una y las demás esperen. Este mecanismo de competición supone un retardo extra respecto al escenario en el que sólo hay una máquina que quiere enviar, y por este motivo disminuye el *throughput* al aumentar el número de saltos.

En el caso de la conexión cifrada, el cuello de botella está localizado en el cifrado de los datos, pero sólo en las conexiones de un salto. La máquina emisora utilizada en el banco de pruebas no es capaz de cifrar tan rápido como permite transmitir el medio (22Mbps aproximadamente) y cifra a una velocidad cercana a los 12Mbps, con una utilización del procesador del 100%. Para las comunicaciones en dos saltos, la velocidad máxima de transmisión se equipara a la velocidad de cifrado. A partir de dos saltos, el cuello de botella en las conexiones cifradas será también el medio de transmisión. Para conexiones de tres saltos, el *throughput* es de 6.8Mbps aproximadamente, independientemente de si se utiliza el cifrado o no. Para conexiones de cuatro saltos, la velocidad de transferencia se sitúa aproximadamente en 5.3Mbps.

Hay que recordar que la implementación que se ha utilizado de HIP está aún en fase de desarrollo. Es de esperar que en el futuro el cifrado de los datos se realice de forma mucho más eficiente, y el cuello de botella, incluso en este banco de pruebas, sea siempre el medio de transmisión y no el cifrado.

8. CONCLUSIONES

En este documento se ha presentado una modificación de los protocolos HIP y OLSR que soluciona el problema del anonimato en las redes ad-hoc: los nodos podrán intercambiar información sin que el resto de la red sepa qué dos máquinas participan en cada comunicación. Además, la modificación del protocolo HIP ha permitido su uso en las redes ad-hoc, garantizando así la confidencialidad y la integridad de los datos, además de resistencia a ataques de Denegación de Servicio (DoS), hombre-en-medio (MitM) y reenvío de paquetes.

HIP también proporciona una identidad a cada nodo de la red, independiente de su dirección IP, por lo que los nodos han podido cambiar de dirección IP sin perder su identidad. Las modificaciones realizadas han permitido que los nodos de la red puedan comunicarse entre ellos sin necesidad de conocer la dirección IP del destino y sin consultar un servidor centralizado de resolución de nombres. El único requisito para establecer una conexión segura es conocer la identidad del destino. Gracias a estas identidades se autentifica a las partes implicadas en una comunicación, y se limita la función de las direcciones IP únicamente al enrutado de paquetes.

Los resultados obtenidos en el banco de pruebas reflejan que el uso de seudónimos supone un aumento del tráfico de control OLSR. Esto se debe a que los seudónimos se representan como nuevos nodos de la red. El número de bytes de control se incrementa, pero el número de paquetes OLSR no crece tanto como el número de mensajes OLSR, ya que los segundos tienden a agruparse dentro de los primeros. Pese al aumento del tráfico de control, éste no llega a alcanzar el 0.03% de la utilización total de la red, por lo que no afecta al *throughput*.

En las pruebas también se ha evaluado que el tiempo de establecimiento de una asociación HIP no es despreciable. Este tiempo se sitúa en torno a los 260ms en la versión original de HIP y cerca de los 460ms cuando los paquetes HIP son cifrados para lograr anonimato. Una vez establecida la asociación HIP, ésta puede permanecer activada y ser reutilizada en comunicaciones posteriores entre los mismos dos nodos. Por tanto, es posible que en algunas aplicaciones no sea viable establecer una asociación HIP justo en el momento de enviar información, pero sí sea interesante establecer la asociación previamente. De este modo, cuando se necesitase enviar información, no se sufriría la penalización por el inicio de la asociación HIP.

En cuanto al *throughput*, éste disminuye significativamente en las conexiones cifradas de un salto, debido al cuello de botella que supone el cifrado y descifrado de los datos. Sin embargo, en conexiones que requieren de más de un salto, el *throughput* obtenido aplicando cifrado es similar al obtenido en conexiones sin cifrar. Además, no hay que olvidar que se ha partido de una implementación de HIP que se encuentra en fase de desarrollo. Se espera que en el futuro el cifrado sea más eficiente.

El resultado final ha sido una aplicación eficaz, fácil de utilizar, probada con éxito en diferentes escenarios, documentada, lista para ser distribuida y que ha alcanzado satisfactoriamente los objetivos definidos inicialmente.

9. Bibliografía

[1] Moskowitz, R., "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

[2] Moskowitz, R., "Host Identity Protocol", RFC 5201, April 2008.

[3] Jokela, P., "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, April 2008.

[4] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory vol. IT-22, number 6, pages 644-654, November 1976.

[5] Krawczyk, H., "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols", in Proceedings of CRYPTO 2003, pages 400-425, August 2003.

[6] Clausen, T. and P. Jacquet, Eds., "Optimized Link State Routing Protocol (OLSR)", RFC 3626, October 2003.

[7] Ziegler, R., "Linux Firewalls" Second Edition, Ed. New Riders.

[8] Gurtov, A., "Host Identity Protocol (HIP): Towards the Secure Mobile Internet", John Wiley & Sons.

[9] IANA IPv4 Address Space Registry, http://www.iana.org/assignments/ipv4-address-space/, June 2009.

[10] Hafslund, A., "Secure Extension to the OLSR protocol", OLSR Interop and Workshop, 2004.

[11] Kong, J., Hong, X., "ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks", MOBIHOC, 2003.

[12] Jung Ha Paik, "A3RP: Anonymous and Authenticated Ad Hoc Routing Protocol", International Conference on Information Security and Assurance, 2008.

[13] Zhang, Y., "MASK: anonymous on-demand routing in mobile ad hoc Networks", Transactions on Wireless Communications, vol. 21, pages 2376-2385, IEEE, September 2006.

[14] Lin, X., "ASRPAKE: An Anonymous Secure Routing Protocol with Authenticated Key Exchange for Wireless Ad Hoc Networks", in Proceedings of International Conference on Communications (ICC), IEEE, 2007.

[15] Rivest, R., Shamir, A., Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM, Vol. 21 (2), pages.120–126, 1978.

[16] Internet Engineering Task Force, “Manet working group charter”, http://www.ietf.org/html.charters/manet-charter.html.

[17] OpenHip Project. http://www.openhip.org.

[18] Orlrd: an adhoc wireless mesh routing daemon. http://www.olsr.org.

[19] Wireshark. http://www.wireshark.org.

[20] Perkins, C., “IP Mobility Support for IPv4”, RFC 3344, August 2002.

[21] The Boeing Company, “Secure Mobile Architecture (SMA) for Automation Security”,

[22] Tarkoma, S., “Report on HIP Applications”, Technical Report, InfraHIP Project, October 2005.

[23] Lacuesta, R., Peñalver, L., “Spontaneous networks: Trust in a world of equals”, ICNS, 2006.

[24] Kivinen, T., “More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)”, RFC 3526, May 2003.

10. ANEXOS

ANEXO I: Manual de usuario de HIP MANET

Índice del Manual de Usuario HIP MANET.

I. INSTALACIÓN (sólo LINUX):

./bootstrap.sh
./configure
make install

II. CONFIGURACIÓN.

1. Generar fichero de configuración por defecto:
   - Ejecutar /usr/local/sbin/hitgen –conf
   (se generará el fichero /usr/local/etc/hip/hip.conf)
2. Generar criptografía asimétrica (clave pública y privada):
   - Ejecutar /usr/local/sbin/hitgen
   (se generará el fichero /usr/local/etc/hip/my_host_identities.xml)

(Los pasos 1 y 2 también son necesarios en la versión HIP Original).

3. Añadir las NUEVAS OPCIONES al fichero de configuración (/usr/local/etc/hip/hip.conf).

<adhoc_mode>
no/yes: indica si el paquete I1 se enviará por unicast (no) o broadcast (yes).
<anonymous_mode> no/yes:
indica si los paquetes HIP se cifran con la Clave Pública destino.
<forwarding_broadcast>
no/yes: HIP se encarga de retransmitir los I1 Broadcast (yes) o no.
<ip_broadcast>
172.16.2.255: dirección de broadcast para los paquetes I1 (si adhoc_mode es 'yes').

Si no se indican estas entradas en el fichero de configuración, se tomarán los siguientes valores por defecto:

<adhoc_mode>no</adhoc_mode>
<anonymous_mode>no</anonymous_mode>
<forwarding_broadcast>yes</forwarding_broadcast>
<ip_broadcast>255.255.255.255</ip_broadcast>

III. EJEMPLOS DE CONFIGURACIÓN (en fichero hip.conf)

**** OPCION 1. Funcionamiento como HIP Original:***
No añadir <adhoc_mode>, ni <anonymous_mode>, ni <forwarding_broadcast>, ni <ip_broadcast>.

Si no funciona, añadir:
<preferred_interface>ra0</preferred_interface>
(necesario para indicar la interfaz de red de salida de los paquetes HIP).

**** OPCIÓN 2. Funcionamiento como HIP Original, pero añadiendo las nuevas opciones con estos valores:***
<adhoc_mode>no</adhoc_mode>
<anonymous_mode>no</anonymous_mode>
<forwarding_broadcast>no</forwarding_broadcast>
<ip_broadcast>172.16.2.255</ip_broadcast>

<preferred_interface>ra0</preferred_interface>
(necesario para indicar la interfaz de red de salida de los paquetes HIP).

**** OPCIÓN 3. Funcionamiento adhoc pero utilizando agente externo para el reenvío de los I1 broadcast.***
<adhoc_mode>yes</adhoc_mode>
<forwarding_broadcast>no</forwarding_broadcast>

Nota: se necesitará un agente externo en cada nodo para el reenvío de los broadcast (ie: OLSR+BMF).

Nota: en funcionamiento adhoc, los campos <addr> del fichero known_host_identities.xml no se utilizan (I1 se envía por Broadcast, y el resto de paquetes a la dirección desde la que responde el host destino), pero debe aparecer el campo <addr> en el fichero known_host_identities.xml (con un valor ficticio, por ejemplo). Esto es así para garantizar que el fichero XML está bien formado.

**** OPCIÓN 4. Funcionamiento 'anónimo'. Los paquetes se envían cifrados con la clave pública del host destino.***

Así los nodos intermedios no leen en las cabeceras HIP qué 2 identidades (HIT) se están comunicando.
<adhoc_mode>no</adhoc_mode>
<anonymous_mode>yes</anonymous_mode>

Posible error: Error. No puede enviarse el mensaje encriptado. No conozco la Clave pública del receptor.

Solución: añadir al fichero known_host_identities.xml la clave pública del destino (campos N y E).

- Con HIP_MANET, esto se añade automáticamente al establecer una asociación HIP. Lo más cómodo es establecer una asociación HIP con la configuración OPCIÓN 1 ó 2, y a continuación volver a la OPCIÓN 4, pero ahora ya no obtendremos el error.

**** OPCIÓN 5. Funcionamiento adhoc y utilizando el propio HIP_MANET para el reenvío de los I1 broadcast.***

<adhoc_mode>yes</adhoc_mode>
<forwarding_broadcast>yes</forwarding_broadcast>
<ip_broadcast>172.16.2.255</ip_broadcast>

IV. EJECUCIÓN.

cd /usr/local/sbin/
./hip -v -V

// '-v': verbose (implementado en la versión Original de OpenHIP.
// '-V': verbose añadido en la version HIP_MANET.

V. ERRORES COMUNES:

1. Error: (Cannot determine HIT for this LSI).
   - Causa: intentas establecer una asociación HIP con un nodo cuya información (HIT, IP, etc...) no aparece en el fichero known_host_identities.xml.
   - Solución: añade dicha información al fichero known_host_identities.xml.

2. Error. No puede enviarse el mensaje encriptado. No conozco la Clave pública del receptor.
   - Causa: intentas establecer una asociación HIP con un nodo cuya Clave pública no aparece en el fichero known_host_identities.xml. Además en la configuración (fichero hip.conf) el campo 'anonymous_mode' está a 'yes', por lo que se tienen que cifrar todos los paquetes HIP con la Clave Pública del destino.
   - Solución: añadir al fichero known_host_identities.xml la clave pública del destino (campos N y E). Con HIP_MANET, esto se añade automáticamente al establecer una asociación HIP. Lo más cómodo es establecer una asociación HIP con la configuración OPCIÓN 1 ó 2, y a continuación volver a la OPCIÓN 4, pero ahora ya no obtendremos el error.

3. Error en demonio HIP: Received ACQUIRE for LSI 1.210.105.199 (Peer address not found for HOST_DESTINO-1024).
   - Causa: en el fichero known_host_identities.xml no aparece el campo '<addr>' para la máquina 'HOST_DESTINO'.
   - Solución: Añadir el campo <addr> ...</addr> para dicha máquina. Si se está ejecutando HIP con <adhoc_mode>yes</adhoc_mode>, el campo <addr> no se

utilizará, ya que el paquete I1 se enviará por difusión, pero <addr> es necesario igualmente en el fichero.

4. Error: Utilizo OLSR con el plugin BMF (para reenviar los broadcasts por toda la red), pero HIP_MANET parece que no funciona (el paquete I1 enviado por broadcast no llega al destino).

- Posible causa: comprueba que en el fichero "hip.conf" aparece "<adhoc_mode>yes</adhoc_mode>". Si es así, comprueba que BMF está funcionando bien en todos los nodos. Para camprobarlo, sigue estos pasos:
  1. Habilita la recepción de paquetes ICMP Broadcast en todos los nodos.
     Ejecuta
     echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
     en cada nodo.
  2. Ejecuta en un nodo de la red
     ping -b 172.16.2.255
     (asumiendo que las IP de la red están en 172.16.2.1- 172.16.2.254; máscara igual a 255.255.255.0).
  3. Si BMF está funcionando, deberían responder al ping anterior TODOS los nodos de la red, incluido el emisor.

5. Error. Al ejecutar HIP (/usr/local/sbin/hip) aparece el siguiente error:

Error setting TAP parameters.
Error initializing TAP device.

- Posible causa: el demonio HIP se encuentra ya en ejecución (o no ha terminado correctamente).
- Solución: ejecuta "top" para comprobar si está en ejecución. Si es así, sal de 'top' ('q'), y termina el proceso 'hip': ejecuta

kill -9 <PID de HIP>

6. Error. Al ejecutar HIP aparece el siguiente error:

*** hipd already running (PID 10547)
*** (may need to remove /var/run/hip.pid)

- Causa: el demonio HIP ha terminado de forma incorrecta.
- Solución: ejecuta

rm /var/run/hip.pid

<u>VI. EJEMPLO FICHERO</u> known_host_identities.xml con la información 'Clave Pública' de un nodo destino (campos N y E):

```
<?xml version="1.0" encoding="UTF-8"?>
<known_host_identities>
    <host_identity alg="RSA" alg_id="5" length="128" anon="no" incoming="yes">
        <name>eeepc5-1024</name>
         <addr>172.16.2.14</addr>
        <N>E5D9A53552A7BFC44F1EF0642A8... DEFAE3242EB</N>
        <E>010001</E>
        <HIT>2001:1e:62b7: ...</HIT>
```

<LSI>1.76.234.66</LSI>
</host_identity>
</known_host_identities>

Nota: si se utiliza HIP MANET y se establece alguna asociación HIP con otro host, la clave pública de dicho host destino se almacena en 'known_host_identities.xml'. Esto no sucedía en la versión Original de OpenHip. Esta clave es la que se necesitará en futuras asociaciones HIP 'anónimas' (<anonymous_mode>yes</anonymous_mode>) con dicho host destino.

ANEXO II: Algoritmo Diffie-Hellman

El algoritmo de Diffie-Hellman (en honor a sus creadores, Whitfield Diffie y Martin Hellman) [4] permite acordar una clave secreta entre dos máquinas, a través de un canal inseguro y enviando únicamente dos mensajes. La clave secreta resultante no puede ser descubierta por un atacante, aunque éste obtenga los dos mensajes enviados por el protocolo. La principal aplicación de este protocolo es acordar una clave simétrica con la que posteriormente cifrar las comunicaciones entre dos máquinas.

El protocolo de Diffie-Hellman fue publicado en 1976. Actualmente se conoce que es vulnerable a ataques de hombre en medio (MitM): un atacante podría situarse entre ambas máquinas y acordar una clave simétrica con cada una de las partes, haciéndose pasar por el host A de cara al host B y viceversa. Una vez establecidas las 2 claves simétricas, el atacante haría de puente entre los 2 hosts, descifrando toda la comunicación y volviéndola a cifrar para enviársela al otro host.

Para corregir la vulnerabilidad del protocolo, éste debe ser utilizado conjuntamente con algún sistema que autentique los mensajes. Esto ocurre, por ejemplo, durante el establecimiento de la asociación HIP, donde los paquetes R1 e I2, además de contener los mensajes de Diffie-Hellman, están firmados digitalmente.

En la Figura 46 se muestra un ejemplo de funcionamiento del protocolo Diffie-Hellman.

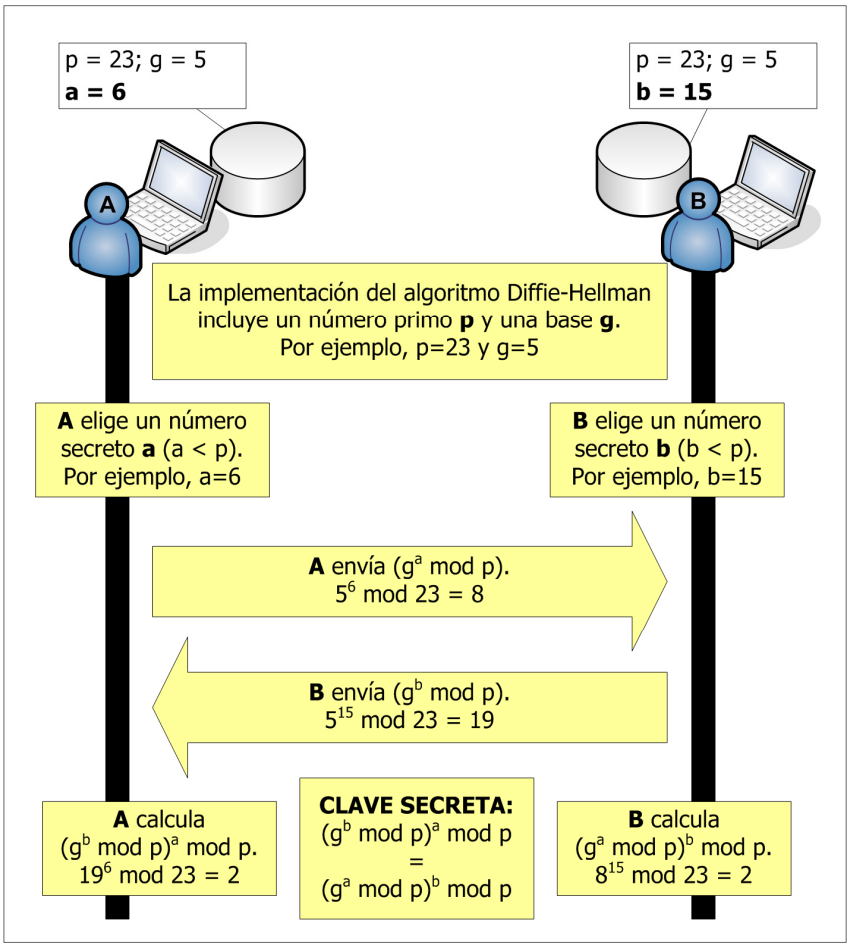


Figura 46. Ejemplo de funcionamiento del protocolo Diffie-Hellman

Los valores de “p” y “g” son públicos y cualquier atacante puede conocerlos, pero esto no supone una vulnerabilidad. Aunque un atacante conociese dichos valores y capturara los dos mensajes enviados entre las máquinas A y B, no sería capaz de averiguar la clave secreta. A continuación se muestra la información capturada por un atacante en el escenario de la Figura 46:

$(g^a \bmod p) = 8 \rightarrow (5^a \bmod 23) = 8$
$(g^b \bmod p) = 19 \rightarrow (5^b \bmod 23) = 19$

A partir de las ecuaciones anteriores, intentar calcular los valores de “a” y “b” es lo que se conoce como el problema del algoritmo discreto, un problema que se cree computacionalmente intratable y cuya notación es la siguiente:

$a = \log \text{disc}_g (g^a \bmod p) = \log \text{disc}_5 (8)$
$b = \log \text{disc}_g (g^b \bmod p) = \log \text{disc}_5 (19)$

Con los valores del ejemplo sí que es posible encontrar la solución, ya que se ha escogido un número primo “p” muy pequeño (p = 23), y se sabe que “a” y “b” son menores que “p”. Por lo tanto, para obtener los valores secretos en este ejemplo, un atacante tendría que probar sólo 22 posibles valores.

Por suerte, las implementaciones actuales del protocolo Diffie-Hellman utilizan números primos muy grandes, lo que impide a un atacante calcular los valores de “a” y “b”. El valor “g” no necesita ser grande, y en la práctica su valor es 2 ó 5. En el RFC 3526 [24] aparecen publicados los números primos que deben utilizarse. A modo de ejemplo, se facilita aquí el número primo de 1024 bytes propuesto. El valor “g” utilizado es 2:

$p = 2^{8192} - 2^{8128} - 1 + 2^{64} \times ((2^{8062} \text{ pi}) + 4743158)$

ANEXO III: Herramientas utilizadas

Además de las implementaciones de los protocolos HIP y OLSR que se han tomado como punto de partida, se han utilizado una serie de herramientas durante las fases de implementación y testeo que cabe mencionar.

Doxygen

Doxygen es un generador de documentación para proyectos software escritos en C, C++, Java o Python entre otros. Una vez configurado, Doxygen recorre todo el código fuente del proyecto y genera documentación en formato HTML o LaTeX. Esta documentación facilita enormemente el mantenimiento de una aplicación. En el contexto de nuestro proyecto, doxygen ha sido de gran ayuda para entender el funcionamiento de las implementaciones de HIP y OLSR: realización de trazas, acceso rápido a la definición de funciones, representación gráfica de las estructuras, etc (Figura 47).

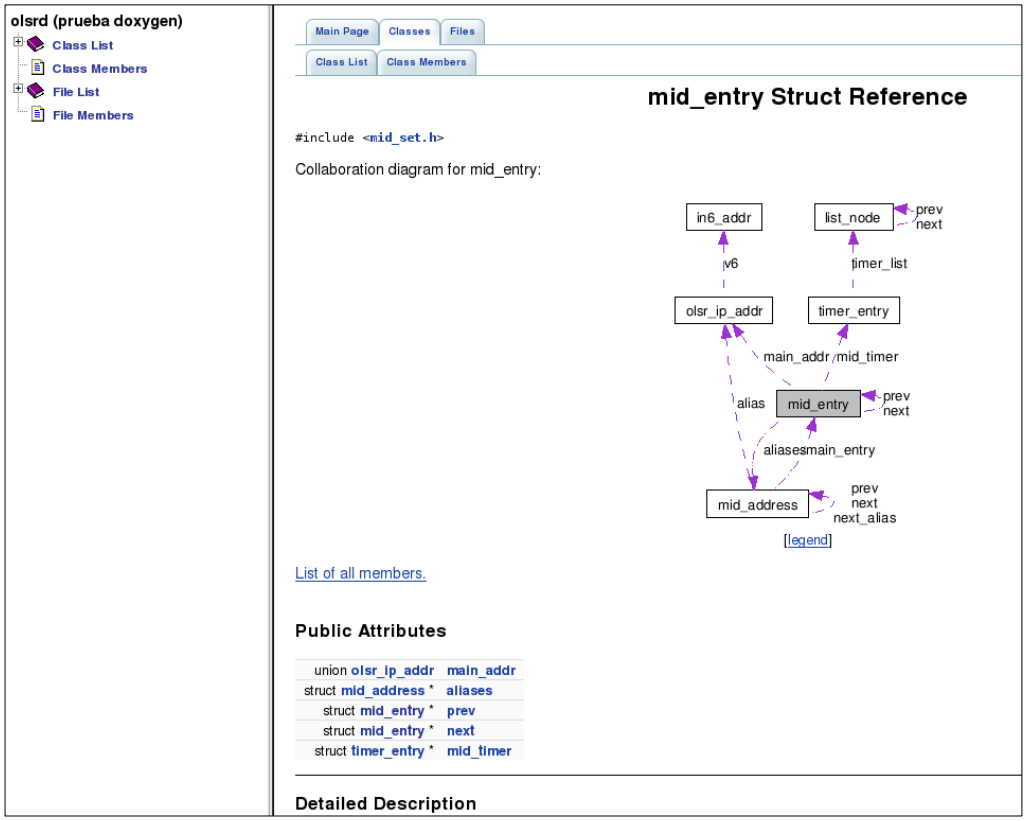


Figura 47. Ejemplo de documentación generada con Doxygen

Wireshark

Wireshark [19] es un analizador de protocolos de red. Esta herramienta permite capturar tramas de red durante un periodo de tiempo, con el fin de analizarlas, ya sea en tiempo real o con posterioridad (Figura 48). Esta aplicación es especialmente útil durante el desarrollo y validación de nuevos protocolos de red. Wireshark se ha utilizado en este trabajo para estudiar los paquetes HIP y OLSR enviados y recibidos por cada máquina, y validar el funcionamiento de las modificaciones realizadas sobre ambos protocolos.

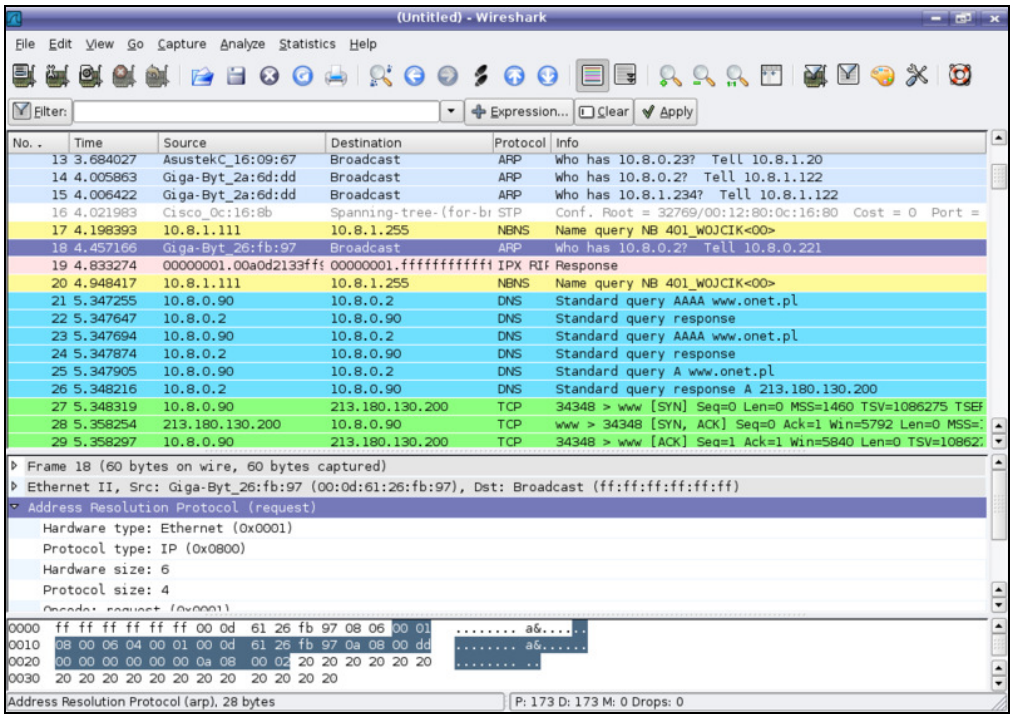


Figura 48. Captura de pantalla de la herramienta wireshark

Actualmente HIP es un protocolo de red experimental, por lo que Wireshark no reconoce los paquetes HIP como tales, y los cataloga todos como paquetes de protocolo desconocido. Para que Wireshark sea realmente útil y reconozca los paquetes HIP I1, R1, I2, R2, etc, así como todos sus campos, es necesario aplicar un parche durante la instalación del analizador de protocolos. A continuación se explican los pasos a seguir durante la instalación.

0. Descargar estos 3 archivos:
    libcap-0.8.1
    wireshark-0.99.5-hip-base05.patch
        (http://sourceforge.net/project/showfiles.php?group_id=132288&package_id=152401&release_id=417797)
    wireshark-0.99.5.tar.bz2

1. Instalar libcap-0.8.1
    (Descomprimir)
    ./configure
    make; make install

2. Aplicar el parche e instalar wireshark:
    cd /usr/local/src
    tar xjf ~/wireshark-0.99.5.tar.bz2
    cd wireshark-0.99.5
    patch -p1 < ~/wireshark-0.99.5-hip-base05.patch
    aclocal
    autonf
    ./configure
    make install

Valgrind y KCachegrind

Valgrind es un conjunto de herramientas para la depuración de problemas de memoria y de rendimiento. *Kcachegrind*[7] es un visor para los resultados devueltos por *Valgrind*. Dada una ejecución del software que queremos depurar, *Valgrind* encuentra posibles errores de memoria. *Valgrind* también guarda una traza de la ejecución.

Posteriormente, en la visualización de resultados con *Kcachegrind,* se pueden detectar fácilmente los cuellos de botella de la aplicación. *Kcachegrind* también muestra un grafo con las llamadas a funciones. Gracias a esto se puede saber, dada una función, qué otras funciones son invocadas desde la primera.

En la implementación de OLSR se hace un uso intensivo de llamadas a funciones por medio de punteros a funciones. Esto hace casi imposible la realización de trazas a partir del código fuente, ya que en éste aparecen punteros y no nombres de funciones. Es por este motivo que los grafos generados con *Kcachegrind* han sido de gran ayuda para comprender este código. En la Figura 49 se observa el grafo generado para una ejecución de OLSR.

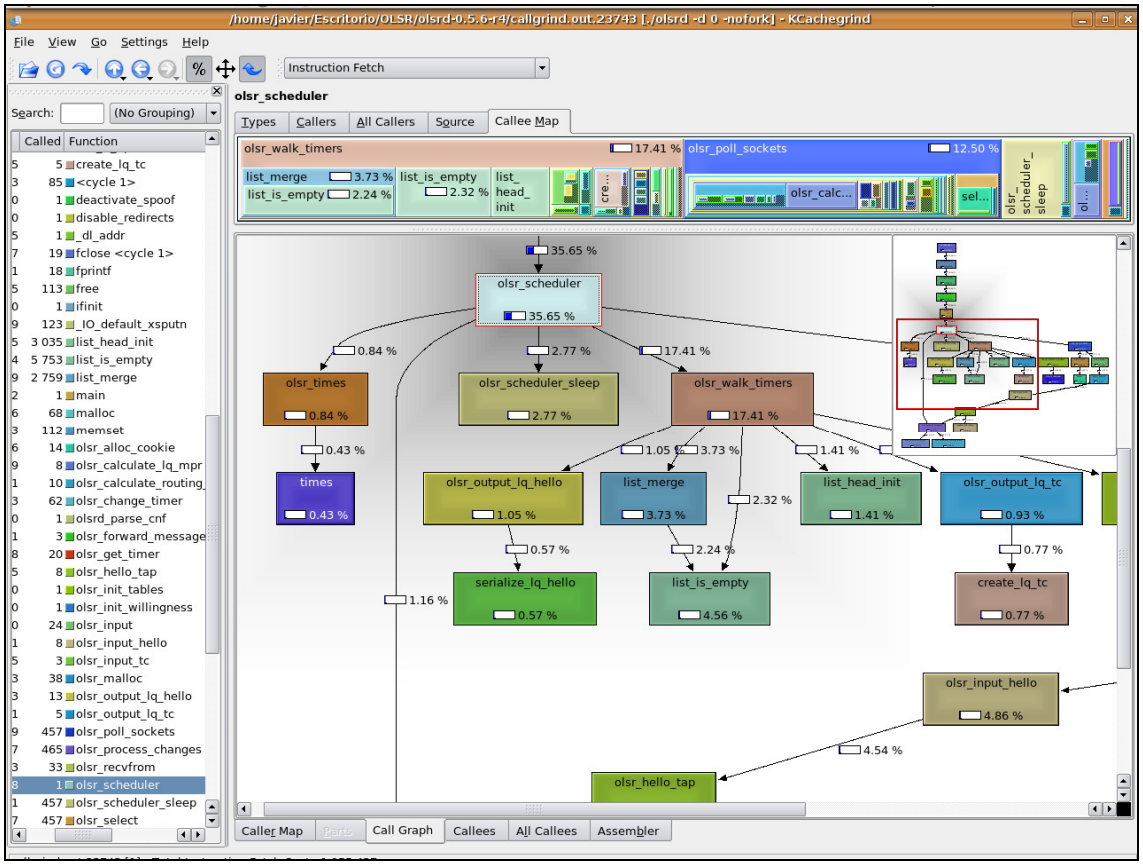


Figura 49. Aplicación KCachegrind mostrando el grafo de ejecución de OLSR

OpenSSL

OpenSSL[8] es un paquete de herramientas y librerías de programación relacionadas con la criptografía. Es un proyecto software de código abierto y se utiliza en la implementación

[7]El proyecto Kcachegrind se encuentra alojado en la dirección http://kcachegrind.sourceforge.net
[8] OpenSSL puede descargarse desde la página web oficial: http://www.openssl.org

de HIP para generar las claves asimétricas y las firmas digitales, entre otras cosas. En la versión modificada de HIP se ha utilizado esta librería para cifrar y descifrar los paquetes HIP con criptografía asimétrica RSA.

Vinagre

Vinagre[9] es un potente cliente de VNC (Virtual Network Computing). Permite acceder de forma simultánea y remota al escritorio de otros ordenadores. Se ha utilizado durante el desarrollo de este proyecto para agilizar la configuración de los ordenadores eeePC que formaban la red ad-hoc, permitiendo controlar todos estos ordenadores desde una única máquina. Las comunicaciones entre los eeePC y el ordenador principal se han realizado a través de la conexión Ethernet. En la Figura 50 aparecen cuatro de los cinco eeePC, cuyos escritorios se visualizan en el monitor central gracias a Vinagre.

Figura 50. Banco de pruebas formado por ordenadores eeePC

[9] Página oficial del proyecto Vinagre: http://projects.gnome.org/vinagre

ANEXO IV: Script para la configuración de la interfaz de red con seudónimos.

El funcionamiento de este script se detalla en el apartado 6.3.

```
#!/bin/bash
# PSEUDONYM MANAGER v.1.0 - Autor. F. Javier Campos (Junio 2009)

clear
echo -E "#       PSEUDONYM MANAGER  v.1.0       #\n\n"

if [ $# -lt 3 ]
then
     echo "USO DEL SCRIPT:"
     echo "    $0 < interfaz > del numero_pseudonimos"
     echo "  ó bien"
     echo -e "    $0 < interfaz > add <fichero_pseudonimos>\n"
     exit 1
fi

if [ $UID != 0 ] # comprobar si Root
then
     echo -e "ERROR.\tDebes ejecutar este script como ROOT."
     echo -e "\tPrueba a escribir:"
     echo -e "\t\tsudo $0 $1 $2 $3\n"
     exit 1
fi

if [ "$2" ==  "del" ]
then
     echo -e "BORRANDO interfaces virtuales de '$1'...\n"
     cont=$3
     while [ $cont -ge 0 ]
     do
          echo "  $1:$cont"
          ifconfig $1:$cont down
          cont=`expr $cont - 1`
     done

     echo -e "\nHECHO!\n\nEjecutando 'ifconfig'...\n"
     ifconfig
else
     if [ "$2" ==  "add" ]
     then
          if [ -f $3 ] # si el fichero existe...
          then
             echo -e "CREANDO pseudonimos en '$1'...\n"
             cont=0
             for linea in `cat $3` ; do
                   ifconfig $1:$cont ${linea}
                   echo "  $1:$cont = ${linea}"
                   cont=`expr $cont + 1`
             done
             echo -e "\nHECHO!\n\nEjecutando 'ifconfig'...\n"
             ifconfig
          else
             echo "Error. El fichero '$3' no existe."
          fi
     fi
fi
```

ANEXO V: Script de la aplicación HOP (HIP + OLSR + Pseudonyms)

En el apartado 6.4 se explica el uso de este script.

```
#!/bin/bash

# HOP v.1.0
# Autor. F. Javier Campos
# Junio 2009

clear
echo "########################################"
echo "#                                      #"
echo "#              HOP  v.1.0              #"
echo "#                                      #"
echo -e "########################################\n"

if [ $# -lt 3 ]
then
      echo "USO DEL SCRIPT:"
      echo "    $0 < interfaz > -manual nombre_fichero_pseudonimos"
      echo "  ó bien"
      echo -e "    $0 < interfaz > -auto numero_pseudos\n\n"
      echo "Nota: <interfaz> se debe especificar también en
            /etc/olsrd.conf y /usr/local/etc/hip/hip.conf"
      echo "Nota 2: en modo '-auto' se genera el fichero 'pseudos.temp'
            con los pseudonimos elegidos."
      exit 1
fi

if [ $UID != 0 ]
then
      echo -e "ERROR.\tDebes ejecutar este script como ROOT."
      echo -e "\tPrueba a escribir:"
      echo -e "\t\tsudo $0 $1 $2 $3\n"
      exit 1
fi

# si no existen los scripts... error
if [ ! -f ./scripts/pseudonym-manager.sh ]
then
      echo "Error. El fichero 'pseudonym-manager.sh' no se encuentra en
      el directorio ./scripts."
      exit 1
fi
if [ ! -f ./scripts/filtrar_arp.sh ]
then
      echo "Error. El fichero 'filtrar_arp.sh' no se encuentra en el
      directorio ./scripts."
      exit 1
fi


# prerrequisito, tener instalado arptables
if [ ! -f /sbin/arptables ]
then
```

```
      echo "Error. Se necesita 'arptables'"
      echo "Prueba a ejecutar: 'apt-get install arptables'"
      exit 1
fi

if [ ! -f /usr/sbin/olsrd ] # existe olrd ?
then
      echo "Error. Se necesita tener instalado 'olsrd'"
      exit 1
fi

if [ ! -f /usr/local/sbin/hip ] # existe hip ?
then
      echo "Error. Se necesita tener instalado 'hip'"
      exit 1
fi

if [ "$2" ==  "-manual" ]
then
      # crear nuevas direcciones ip
      ./scripts/pseudonym-manager.sh $1 add $3

      if [ -f $3 ] # si el fichero existe...
      then
            # filtrar ARPs
            ./scripts/filtrar_arp.sh $3

            # ejecutar OLSR en modo -pseudo
            /usr/sbin/olsrd -pseudo $3 > salida_olsr.txt &

            # ejecutar HIP-MANET en modo -pseudo
            /usr/local/sbin/hip -v -V -pseudo $3
      else
            echo "Error. El fichero '$3' no existe."
            exit 1
      fi
elif [ "$2" ==  "-auto" ]
then
      # borrar direcciones ip antiguas
      ./scripts/pseudonym-manager.sh $1 del $3

      echo -e "\nEjecutando OLSR con generacion automática de pseudonimos
                (background)\n"

      # OLSR con auto-pseudonimos
      /usr/sbin/olsrd -pseudo pseudos.temp -auto $3 > salida_olsr.txt &

      echo -e "\n\nEsperando la generación automática de pseudonimos (35
               segundos)...\n"
      sleep 35 # espero a que OLSR escuche la red (+30 segundos)

      # añadir direcciones IP en la interfaz
      ./scripts/pseudonym-manager.sh $1 add pseudos.temp

      # filtrar ARPs
      ./scripts/filtrar_arp.sh pseudos.temp

      # ejecutar HIP-MANET en modo -pseudo
      /usr/local/sbin/hip -v -V -pseudo pseudos.temp
fi
```