JITEL 2017
XIII Jornadas de Ingeniería Telemática
27-29 Septiembre 2017, Valencia (España)

Actas de las XIII Jornadas de Ingeniería Telemática (JITEL 2017), Valencia (España), 27-29 de Septiembre de 2017.

ISBN: 978-84-9048-595-8
DOI: http://dx.doi.org/10.4995/JITEL2017.2017.6585

Implementación de mecanismos de mitigación de tormentas de broadcast en redes de área local mediante Redes Definidas por Software

Bárbara Valera Muros, Jonathan Prados Garzón, Juan José Ramos Muñoz, Jorge Navarro Ortiz
Departamento de Teoría de la Señal, Telemática y Comunicaciones,
Universidad de Granada
Calle Periodista Daniel Saucedo Aranda s/n, E-18071 (Granada)
bvaleramu@gmail.com, jpg@ugr.es, jjramos@ugr.es, jorgenavarro@ugr.es

***Resumen*—El uso de Ethernet como tecnología de red para redes corporativas se justifica por su bajo coste y facilidad de configuración y mantenimiento. Sin embargo, estas redes no son muy escalables, debido en parte a las inundaciones o tormentas de broadcasts, que afectan al rendimiento tanto de los dispositivos de red como finales. Para mitigar el impacto de las inundaciones por broadcast, se ha previsto utilizar técnicas de filtrado y caché en distintos nodos de la red. Sin embargo, el paradigma de Redes Definidas por Software permite definir nuevas aproximaciones, gracias a la capacidad de reprogramar la red de forma centralizada y flexible que proporciona. En este trabajo se aborda la implementación de una red de área local con soporte para filtrar algunos paquetes broadcast mediante la utilización de Redes Definidas por Software. Esta solución permitiría desplegar redes de área local más amplias, adecuadas para los requisitos de redes corporativas. Para ello, se describe el desarrollo de filtros para varios protocolos de red, su implementación en el controlador OpendayLight, y la evaluación del rendimiento obtenido.**

***Palabras Clave*—Address Resolution Protocol, ARP, Broadcast, Controlador, Filtrado, Internet Control Message Protocol version 6, OpenDayLight, Redes Defindas por Software, SDN**

I. INTRODUCCIÓN

Los servicios de datos móviles se han convertido poco a poco en imprescindibles para la mayoría de usuarios. Esta tendencia supone un incremento del tráfico en las redes inalámbricas. Dicho incremento representa uno de los mayores retos a los que cualquier red de comunicación tendrá que enfrentarse en el futuro [1]. A esto se le sumaría la reducción de la latencia y los costes asociados, de forma que se plantean nuevos diseños y arquitecturas de red con el fin de satisfacer las crecientes exigencias de futuras aplicaciones. Este nuevo paradigma surge como una de las posibilidades para suplir esa creciente demanda, considerándose una opción dinámica, gestionable, económica y adaptable. Además, reduce los costes asociados a las redes, conocidos como CAPital EXpenditures (CAPEX) y OPerating EXpense (OPEX), lo que se suma a las ventajas de utilizar esta arquitectura a la hora de renovar las redes de comunicación [2]. El aprovechamiento de las redes es un sector en el que se ha invertido gran cantidad de recursos; pero los requerimientos de las telecomunicaciones son cada vez mayores, por lo que un salto de generación móvil implicaría un cambio completo de la red que suponga una solución definitiva y no temporal, como se ha hecho hasta ahora.

Este artículo presenta el desarrollo de procedimientos para redes SDN (Software Defined Networks) que permiten reducir el problema de los broadcasts para mejorar la escalabilidad de las redes. La sistemática seguida en el proyecto se inicia con el estudio bibliográfico de las tecnologías implicadas, principalmente SDN, y una familiarización con las herramientas necesarias para trabajar. Entre ellas destacan: el protocolo OpenFlow [3], el controlador OpenDayLight (ODL) [4] y el emulador de redes Mininet [5]. Una vez definidos los escenarios sobre los que aplicar la solución, se diseña un algoritmo para la detección e identificación, filtrado y reenvío de paquetes en el controlador SDN. Posteriormente, se realiza la programación del código a partir de un conmutador inteligente. Por último, se evalúa la solución implementada teniendo en cuenta las mejoras en el rendimiento del sistema, así como las posibles vías de aplicación futuras. El artículo se estructura en seis secciones. La Sección II describe el problema abordado y la visión general de la solución. La Sección III presenta la revisión del estado del arte de las tecnologías implicadas. El grueso del artículo se incluye en la Sección IV, que contiene las fases de diseño e implementación de la solución. La sección V plantea diferentes entornos experimentales en los que comprobar el funcionamiento de la solución, describiendo las pruebas realizadas y los resultados obtenidos. Por último, la sección VI presenta

las conclusiones y la proyección de futuro.

II. MOTIVACIÓN

A la hora de definir la arquitectura de las redes 5G, existen propuestas en las que se presentan propuestas de arquitecturas basadas en Ethernet, para aprovechar la capacidad de autoconfiguración, la existencia de hardware que lo implementa, y la sencillez en la gestión de este protocolo [6][7]. Concretamente, la arquitectura [6] está optimizada para el protocolo IPv6, y donde SDN es la clave para resolver los retos previstos. Con esta visión se pretende eliminar parte de la complejidad del Evolved Packet Core (EPC) de la red 5G, de forma que sea más escalable y eficiente. Sin embargo, la escalabilidad de las redes Ethernet está limitada por las inundaciones de red (o tormentas de broadcast) [8], causadas por los protocolos de arranque que utilizan los usuarios finales, como Address Resolution Protocol (ARP) [9] o Dynamic Host Configuration Protocol (DHCP) [10]. Para superar estas limitaciones, las nuevas arquitecturas basadas en Ethernet tratan de reducir las inundaciones de las tramas broadcast a la vez que ofrecen los servicios Ethernet esperados. Las tormentas de broadcasts o difusión se producen cuando varios dispositivos envían paquetes a la dirección de difusión de la red. Este fenómeno no sólo consume recursos de red, sino que que afecta al rendimiento de los dispositivos. Por otra parte, StateLess Address AutoConfiguration (SLAAC) [11] permite la autoconfiguración de los hosts IPv6, lo que a su vez supone multitud de solicitudes multicast.

Así, el principal problema tratado es la disminución de la eficiencia de estos sistemas al producirse una inundación. Como solución, se propone el filtrado de mensajes, de forma que el controlador disponga de unas tablas identificativas para cada nodo y sea el encargado de reenviar los mensajes, dirigiéndolos a un destinatario limitado directamente y evitando las inundaciones de red siempre que sea posible. El controlador actuará como un conmutador inteligente capaz de aprender no solo las direcciones de los nodos implicados en el intercambio de mensajes, sino también su papel en dicho intercambio, identificando los diferentes agentes que intervienen para posteriormente dirigir los mensajes del protocolo en cuestión y evitar la sobrecarga de la red, mejorando en definitiva la eficiencia del sistema.

III. ESTADO DEL ARTE

A. Tormentas de Broadcast

Como ya se ha mencionado, el principal problema en cuanto a escalabilidad de las redes viene dado por las tormentas de broadcast. En términos generales, esta difusión amplia es una forma de distribución de información en la que un nodo envía un mensaje a todos los nodos de la red de manera simultánea. Es utilizado principalmente por los protocolos de arranque y de configuración y disminuye la eficiencia de los sistemas aumentando el tráfico de la red. Además, ya que no solo aumenta el tráfico sino el número de paquetes recibidos por cada terminal, reduce también el rendimiento de los equipos. En [8] se estudia el efecto de una tormenta de broadcast en hosts de redes IP. Se trata de un experimento realizado por Cisco para medir el efecto de estas tormentas en una estación SPARC con una tarjeta estándar de Ethernet. Se demostró que una estación de trabajo puede dejar de funcionar debido a las inundaciones broadcast de la red. Además, se observaron puntualmente picos de miles de broadcast por segundo en las tormentas de broadcast, lo que supone una disminución del rendimiento del sistema de hasta el 25%. Se plantean nuevos diseños para mitigar estos problemas, entre los que cabe destacar Ethane [12] y SEATTLE [13]. En el caso de Ethane, se presenta la posibilidad de que las tormentas de broadcast sean gestionadas por un controlador, mientras que SEATTLE propone una transformación de los mensajes broadcast en unicast, aunque esto requeriría unos conmutadores específicos muy costosos. Se concluye que SDN es una opción viable frente a estos diseños para implementar una solución que supla la demanda de los usuarios.

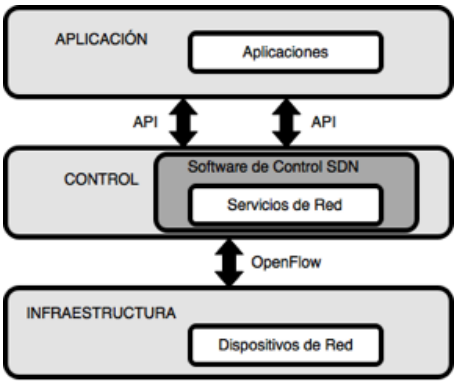

Fig. 1. Arquitectura simplificada de SDN.

Respecto a los protocolos que mayor cantidad de tráfico generan, destacan ARP e Internet Control Message Protocol version 6 (ICMPv6) [14], por lo que el diseño de la solución aborda los procedimientos de autoconfiguración de IPv6 y los procedimientos de resolución de direcciones IP y físicas en IPv4 [15] e IPv6 [16].

B. Software Defined Networking

Las redes SDN permiten atender las necesidades de las aplicaciones y servicios de la red de forma dinámica y escalable. En SDN, la red se programa de forma centralizada, con un controlador lógico que gobierna el funcinamiento de los distintos conmutadores SDN. Esto permite que la red se adapte al entorno con la posibilidad de utilizar el conocimiento de la red completa. En estas redes, el controlador actúa como "cerebro" encargado de comunicar a los conmutadores de la red qué deben hacer con cada flujo de paquetes nuevo. Se plantea el concepto de separación del plano de control de red (software) y del plano de datos (hardware que conmuta los paquetes de datos en la red), tal y como muestra Fig. 1 descrita en [17].

En las redes SDN, las aplicaciones de red usarán la

interfaz de programación (Application Programming Interface, API) NorthBound sobre el plano de control para reforzar sus principios en el plano de datos sin interactuar con el mismo directamente. La interfaz entre el plano de control y el de datos se apoya en SouthBound APIs, que permiten al controlador SDN comunicarse con los equipos de la red en el plano de datos [18]. Dichos equipos deberán soportar las APIs estandarizadas en este nivel. SDN posibilita así la administración de la red al completo a través de sistemas inteligentes que permitan la asignación de recursos según la demanda, redes virtualizadas o servicios cloud seguros. Por tanto, la red estática evoluciona en una plataforma de servicio independiente capaz de responder rápidamente a las necesidades de mercado de los usuarios finales, lo que simplifica en gran medida el diseño y las operaciones de red.

C. Protocolo OpenFlow

OpenFlow es un protocolo de comunicaciones diseñado para dirigir el manejo y enrutamiento del tráfico en una red conmutada, en términos de flujos. Un flujo es un grupo de paquetes definido. Dado que se trata de un estándar abierto, se ha convertido en el modelo estándar de implementación de SDN para la gestión de la red. Se ha traducido así como la primera interfaz de comunicaciones definida entre las capas de control y de transporte en esta arquitectura [19]. El diseño del protocolo se apoya sobre tres bases: los conmutadores con soporte para OpenFlow (que encaminan los paquetes), las tablas de flujos instaladas en dichos conmutadores para la gestión del tráfico y el controlador encargado de comunicar a los conmutadores la información necesaria para administrar el tráfico de la red (añadiendo y eliminando flujos) [20]. Así, aunque el conmutador OpenFlow es responsable del reenvío de paquetes, las decisiones de enrutamiento son tomadas por el controlador. Ambos se comunican a través de OpenFlow, que define los mensajes que hacen referencia a los paquetes enviados, recibidos, la identificación de estados y la modificación de las tablas de flujos para el encaminamiento. De esta forma, cuando el conmutador recibe un paquete para el que no tiene entradas en la tabla de flujo, se lo reenvía al controlador, que es el encargado de decidir si el paquete es descartado o si se agrega una entrada en las tablas. Una vez agregado, el conmutador podrá gestionarlo por sí mismo, en caso de volver a recibir un paquete similar. El proceso que determina qué hacer con cada paquete se denomina Pipeline. Básicamente, cada conmutador dispone de varias tablas, con multitud de flujos cada una. Cuando llega un paquete, se busca hacer el llamado emparejamiento o "matching" en el que se compara cada uno de los valores seleccionados como criterio de emparejamiento del paquete recibido, con los de los flujos de la tabla inicial. En caso de no encontrar ningún flujo coincidente, se pasa a la siguiente tabla. En otro caso, se ejecutaría la acción determinada para ese flujo, entre las que se encuentra la de enviar ese paquete a otra tabla de orden superior. En caso de no haber matching con ninguna tabla, se envía el paquete a una tabla "missing", que decide si debe inundar la red con el paquete, mandarlo al controlador o comenzar de nuevo con un matching más flexible.

D. Controlador OpenDayLight

Como controlador, existen diferentes opciones para implementar la solución con soporte para OpenFlow, a destacar NOX (basado en C++), POX y Ryu (basados en Python), que sin embargo suponen una lenta ejecución de la red. OpenDayLight (ODL) es una alternativa de código abierto y robusto que presenta buen rendimiento de ejecución y soporte de producción, ya que se encuentra respaldado, entre otros fabricantes de dispositivos de red, por Cisco [4]. Al desarrollarse sobre Java, su mayor limitación es la complejidad en la creación de aplicaciones, aunque esto lo hace compatible con la mayoría de sistemas operativos. ODL dispone de una capa de abstracción que separa el controlador de los elementos de red, y esta capa puede basarse en APIs o en modelos. Esta última unifica las APIs, proporcionando una mayor abstracción. Además, se utiliza el lenguaje de modelado YANG (Yet Another Next Generation) [21] para la descripción de las estructuras basadas en modelos, lo que simplifica el desarrollo de aplicaciones en el controlador [22][21].

IV. Propuesta

Para afrontar el problema planteado, se realiza un diseño que permite filtrar las inundaciones de paquetes en la red, aumentando su escalabilidad y el rendimiento del sistema. Para ello, se analizan qué procedimientos requieren de envíos broadcast para distintos protocolos de red. Se categorizan dos casos principales: IPv4 e IPv6. En IPv4 se analiza el protocolo ARP, identificando los mensajes "Request" y "Reply" para resolver la asociación de las direcciones IP y físicas de los dispositivos de la red cada vez que inician una conexión. Sin embargo, en IPv6 se tiene en cuenta el procedimento de autoconfiguación de direcciones (SLAAC) de los hosts, lo que supone multitud de peticiones multicast, ya que en esta versión del protocolo no existe broadcast propiamente dicho. Al conectarse un nodo a una red IPv6, se inicia el procedimiento de descubrimiento de vecino (NDP, Neighbor Discovery Protocol) para descubrir la presencia de otros nodos en el mismo enlace [16]. Se envía una solicitud de router (Router Solicitation) de enlace local mediante multicast, para conocer los parámetros de configuración de red. El router responde con un anuncio de router (Router Advertisement). Se realiza también la solicitud (Neighbor Solicitation) y anuncio (Neighbor Advertisement) de nodos correspondiente para comunicarse con el resto de elementos de la red.

El algoritmo de filtrado se detalla en las siguientes subsecciones.

A. Funcionamiento general

El diseño consta de tres fases. En primer lugar se realiza la clasificación e identificación de paquetes para su posterior filtrado, de acuerdo al tipo de protocolo al que pertenece. Posteriormente, según el tipo de mensaje

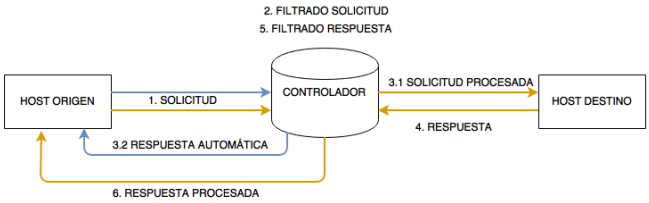

Fig. 2. Esquema básico de la comunicación establecida entre los hosts y el controlador.

que contenga el paquete, se realiza un filtrado en el que el controlador almacena información de los nodos y decide si reenviar una solicitud al resto de la red, o si es capaz de devolver una respuesta apropiada automáticamente. En la fase de reenvío, el controlador realiza el envío, bien de la respuesta directamente, o del paquete a los nodos de la red que sean necesarios.

Se consideran por tanto dos posibles casos: uno, cuando el controlador filtra la solicitud y responde automáticamente. Y otro, cuando se pone en contacto con el nodo destino y se encarga de procesar la respuesta del mismo posteriormente, tal y como muestra la Fig. 2.

B. *Clasificación de paquetes de broadcast*

En la fase de clasificación, se estudia el tipo de trama Ethernet recibida, para determinar qué pasos de filtrado seguir. Además, se extraen las direcciones físicas e IP de origen y destino para almacenarlas en la tabla del controlador en la que se encuentra la información para generar respuestas automáticas para resoluciones de direcciones. La fase de clasificación sigue el diagrama de flujo de Fig. 3.

Posteriormente, si es la trama es un paquete ARP, se comprueba si la entrada está en la tabla y está actualizada. Si se trata de una respuesta, se almacena la carga del paquete para utilizarla posteriormente en las respuestas automáticas. Si se trata de una solicitud, se estudia si el controlador puede o no responder con la información de la que dispone. El diagrama de flujo de esta fase se muestra en Fig. 4.

El caso de datagramas IPv6, el proceso es más complejo. En primer lugar, se comprueba si de trata de un mensaje ICMP y, en ese caso, de qué tipo. Si es una *solicitud de router*, se comprueba si se dispone de una respuesta almacenada para responder automáticamente al nodo solicitante. Si es un *anuncio de router*, se almacena el prefijo de la red con la información necesaria para la configuración del resto de nodos y, en caso de disponer de una entrada previa y actualizada del router, se bloquea el reenvío de este paquete al resto de la red. En la solicitud de vecino se estudia si el nodo origen está en la tabla y se crea una entrada si no dispone ya de una, y posteriormente realiza el mismo estudio para el nodo destino. De esta manera, se comprueba si se puede responder automáticamente dicha solicitud. Los anuncios de vecino crean o actualizan las entradas del controlador y, en caso de ser una entrada actualizada, son bloqueados por el controlador para que no se realice el reenvío al resto de la red. Este proceso se muestra en el diagrama de flujo de Fig. 5.

Una vez se ha filtrado el paquete, se comprueba si la variable de reenvío ha sido modificada por el controlador. De no ser así se realiza un reenvío por defecto al resto de la red, mientras que si se ha editado se realiza el bloqueo de este paquete y se envía la respuesta generada por el controlador directamente al nodo solicitante. La fase de reenvío aparece representada en Fig. 6.

V. Evaluación de la solución

Por último, se realiza la evaluación de los resultados obtenidos mediante una serie de experimentos sobre una plataforma de SDN emulada. Concretamente, se comprueba el funcionamiento de la solución en IPv4, en IPv6, el funcionamiento conjunto de ambos ejecutándose de forma simultánea y concurrente y la mejora de rendimiento que supone la implementación frente a un sistema que no realice el filtrado de mensajes.

Para ello, se utiliza el controlador OpenDayLight, montado en una red Mininet [23]. El emulador de redes virtuales Mininet es la herramienta básica para trabajar con SDN. Permite crear redes virtuales junto con todos sus elementos en una única máquina, facilitando la posterior interacción con dichas redes mediante líneas de comandos. Al ser emulador, en lugar de simular el funcionamiento de la red introduce errores aleatorios para que los resultados obtenidos sean los más parecidos a la realidad posible. Como ventajas, destacar que permite el desarrollo de redes diseñadas en hardware, además de la ejecución en tiempo real, lo que permite evaluar condiciones de errores en la red. Dispone de multitud de topologías para la emulación de diferentes escenarios y permite crear nuevas topologías mediante la programación de entornos con Python, lo que facilita el estudio de las redes SDN.

1) Descripción del experimento con protocolo IPv4: Para evaluar el funcionamiento del algoritmo de filtrado para el caso de tráfico IPv4, se diseña esta prueba donde se evalúa el funcionamiento del controlador cuando se produce broadcast de tramas ARP. Así, tras iniciar la red, disponiendo de 3 hosts sin entradas en la tabla del controlador, se realiza un “ping” del host h1 al host h2. Básicamente, al realizar un “ping” se envía un mensaje ICMP desde el nodo origen de tipo *echo request*, que el nodo destino debe responder con un *echo reply*. En ese momento, se produce una petición ARP *request* con origen en h1 y destino en h2. Como respuesta, h2 genera un mensaje ARP *reply* con destino h1. De esta forma, ambos hosts han quedado reconocidos y registrados en el controlador junto con la información relevante para que éste sea capaz de generar respuestas automáticas para atender futuras peticiones. De igual forma, se realiza otro “ping” de h1 hacia h3. Llegados a este punto, si h2 realizase un ping con destino h3, habría tres posibilidades:

a. De no funcionar correctamente el filtrado, no sería posible completar la petición, ya que el controlador podría reconocer las entradas pero no generar una respuesta apropiada.

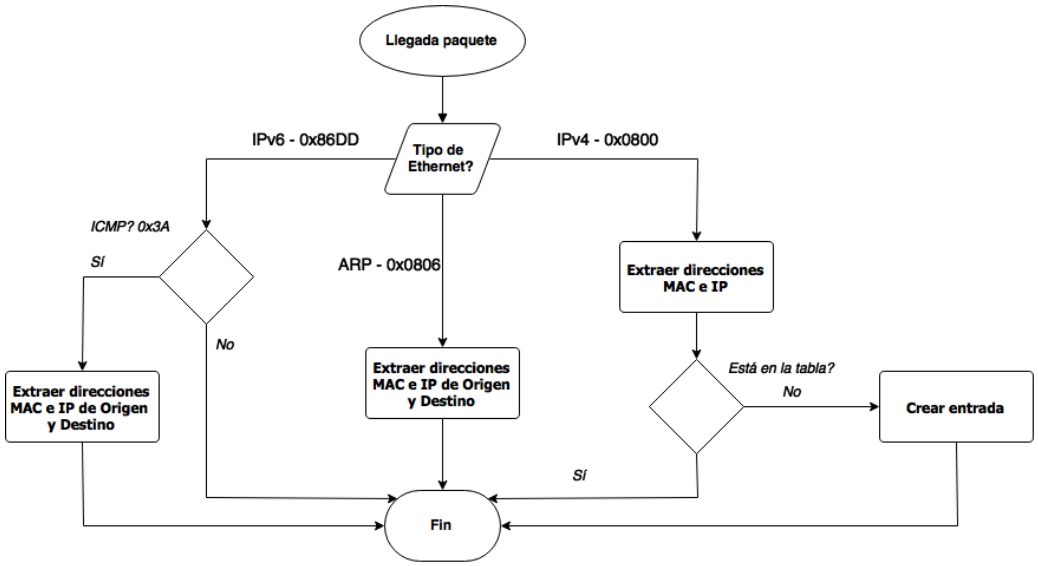

Fig. 3. Diagrama de flujo de la fase de clasificación.

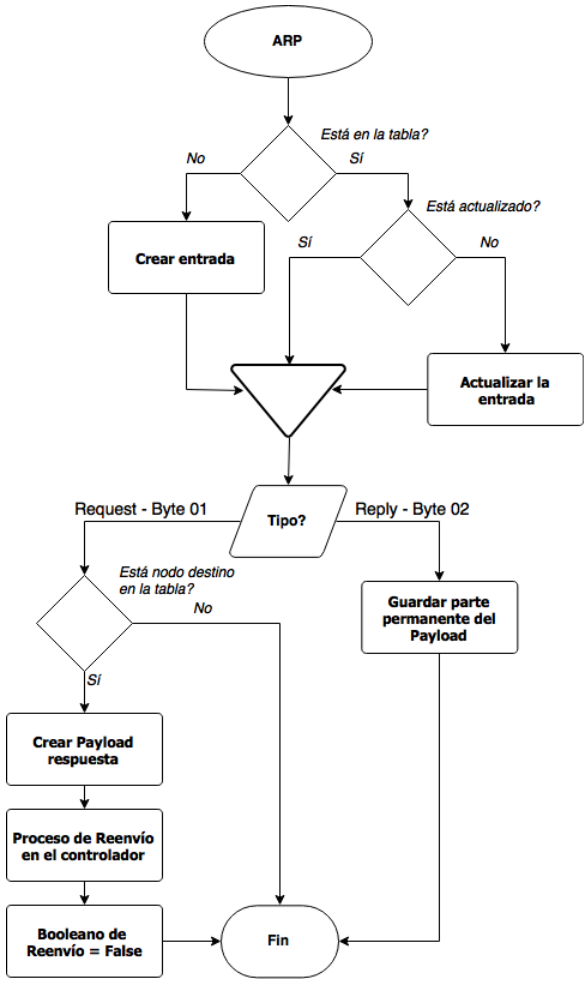

Fig. 4. Diagrama de flujo de la fase de filtrado para ARP.

b. De no realizar el filtrado, h3 respondería con un ARP *reply* a h2. El ping original *request* se habría enviado como broadcast al resto de la red, generando dicha respuesta por parte de h3.
c. De haber filtrado correctamente los mensajes, el controlador bloquea el broadcast de h2 y responde automáticamente con la dirección física asociada a h3. Por tanto, la comunicación entre nodos sería posible sin necesidad tampoco un mensaje ARP *reply* desde h3. Este es el resultado obtenido en este experimento.

2) Descripción del experimento con protocolo IPv6: Para evaluar el funcionamiento del algoritmo de filtrado para el caso de tráfico IPv6, se filtran los mensajes ICMPv6. Estos mensajes se envían al iniciarse una red o añadirse un nuevo nodo, no solo cuando los nodos se van a comunicar, por lo que en este caso se realiza un "ping" para que ambos nodos tengan la dirección del vecino y posteriormente se deshabilita una interfaz, simulando que un nodo desaparece de la red. Al volver a activarse, son necesarios mensajes de solicitud y anuncio de router para realizar la configuración del nodo, y solicitud y anuncio de vecino para ponerse en contacto con el resto de la red. Sin embargo, en caso de realizarse un filtrado apropiado, ambos nodos podrían comunicarse sin necesidad de estos mensajes, tal y como ocurre en el experimento. Si se realiza el envío de mensajes "ping" y "ping6" simultáneamente, se comprueba que no solo es posible sino que se realiza el filtrado de los mensajes de manera conjunta.

3) Descripción del experimento con protocolos IPv4 e IPv6: En este escenario se desea evaluar si la implementación realizada soporta la coexistencia de paquetes de IPv4 e IPv6.

A. Entorno experimental

Para la realización de los experimentos se utilizó un equipo personal con procesador Intel Core i7 a 2.7GHz, memoria RAM de 4GB y unidad de estado sólido, Solid-State Drive (SSD) de 500GB de capacidad. Sobre este equipo se ejecutaba una máquina virtual Oracle VirtualBox, Sistema Operativo Ubuntu 14.04 (64 bits), con la

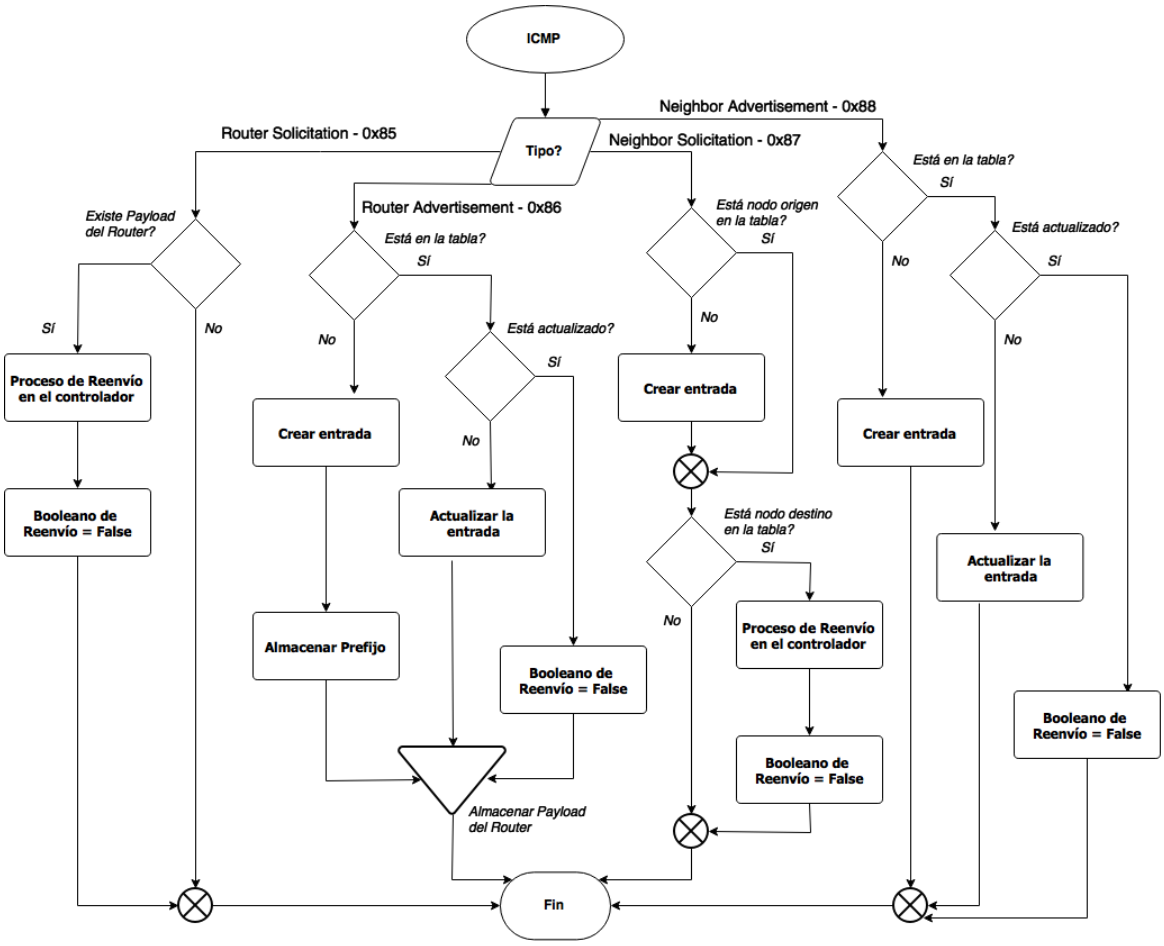

Fig. 5. Diagrama de flujo de la fase de filtrado para ICMP.

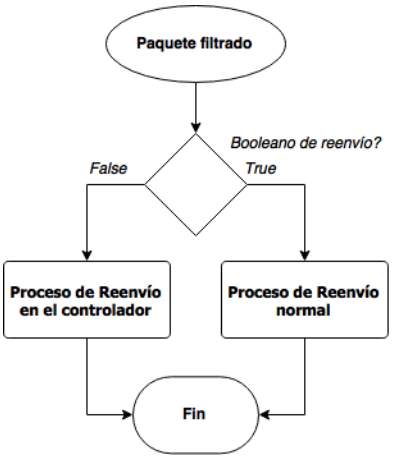

Fig. 6. Diagrama de flujo de la fase de reenvío.

herramienta Mininet, controlador OpenDayLighty el analizador de protocolos Wireshark.

Básicamente, se estudió el rendimiento del sistema para una red con 5, 10 y 20 nodos que generan tráfico de los paquetes IPv4 ó IPv6 identificados en las secciones previas, unas veces con, y otras sin el algoritmo de filtrado implementado. De esta forma se pretende obtener resultados comparables para ambos casos, y analizar si realmente el filtrado supone o no una mejora para la red. Para que la diferencia del tráfico generado no sea excesiva, la opción sin filtrado tiene un controlador funcionando como conmutador con aprendizaje de direcciones de enlace, en lugar de como concentrador. De esta manera, aprende las direcciones de los nodos y las añade a una tabla para su posterior enrutamiento, en lugar de únicamente dejar pasar todo el tráfico. Se utiliza la topología básica de IPv6 para realizar este experimento, mostrada en Fig. 8, variando en este caso el número de nodos.

1) Escenario con protocolo IPv4: Se plantea un escenario con ODL como controlador remoto, un conmutador OpenFlow y tres hosts. Para ello, se lanza el controlador, se instala la aplicación, se crea la topología en Mininet y se añade una regla para que el flujo del conmutador se dirija al controlador.

2) Escenario con protocolo IPv6: En este caso se dispone de dos hosts y un router, además del conmutador OpenFlow y del controlador, tal y como muestra Fig. 8.

Para realizar la configuración de la red en IPv6 es necesario instalar y configurar el demonio de anuncios de router (Router ADVertisement Daemon, RADVD) [24]. Con el archivo de configuración, se establece el prefijo de red que utilizan los dispositivos en la autoconfiguración de direcciones, así como la interfaz del router destinada al envío de dicho prefijo, que es la interfaz del nodo que

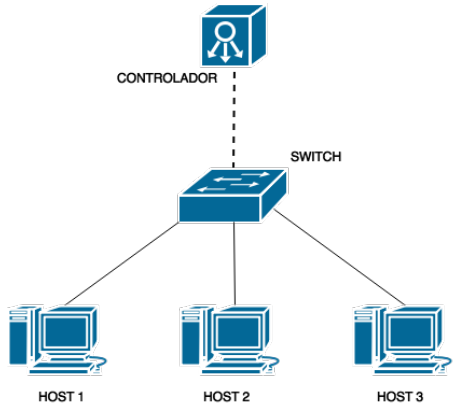

Fig. 7. Entorno experimental con IPv4 y conmutador OpenFlow.

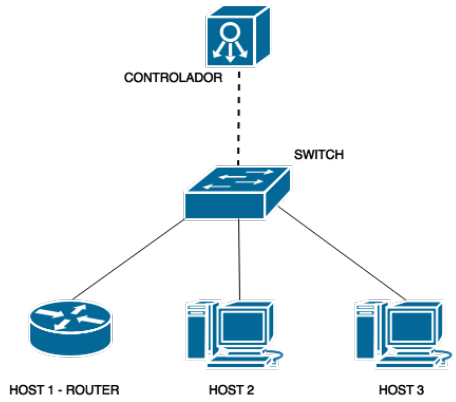

Fig. 8. Entorno experimental con IPv6 y conmutador OpenFlow.

actúa como router en la red. Tras lanzar la topología en Mininet, se configura el router y el resto de nodos, se habilita IPv6 en Mininet y se inicia el servicio RADVD. Así, cada nodo dispone de una dirección de enlace IPv6 y, en el caso de los nodos que no actúan como router, de una dirección global asignada por el router a partir del prefijo creado.

Con este entorno no sólo se evalúa la implementación correcta del algoritmo en el controlador, sino también la configuración de IPv6, ya que en este escenario hay un host encargado de funcionar como router, asignando al resto de elementos de la red sus direcciones globales.

3) Escenario con protocolos IPv4 e IPv6: Se crean archivos para la configuración automática de IPv6 y la generación de paquetes "ping" y "ping6". Se realiza la transmisión de 20 paquetes de cada protocolo por nodo, con un intervalo de 5 segundos entre paquetes. Posteriormente, se deshabilitan las interfaces, se vuelven a habilitar y se vuelve a lanzar el script de envío de paquetes.

B. Resultados experimentales

En el caso de las pruebas de funcionamiento correcto del filtrado en el controlador SDN, los resultados muestran que para ambos protocolos, tanto independiente como simultáneamente, la solución propuesta es viable.

Respecto al rendimiento, se muestra en tabla I el resumen de las estadísticas para cada caso estudiado,

Tabla I
NÚMERO DE PAQUETES TRANSMITIDOS PARA CADA CASO EN LA PRUEBA DE RENDIMIENTO.

Escenario	ARP	ARP filtrado	ICMPv6	ICMPv6 filtrado
5 Nodos	94	30	1064	654
10 Nodos	1585	811	9348	7947
20 Nodos	–	63458	–	583506

presentando el número de paquetes transmitidos por protocolo en cada caso, para una ejecución de cada escenario. Debido a las limitaciones en cuanto a equipamiento para la realización del proyecto, los experimentos se realizan para redes con un máximo de 20 nodos. En el caso del sistema con un controlador sin filtrado, la red Mininet con 20 nodos deja de funcionar debido a la sobrecarga de la misma, por lo que no se puede comparar este resultado con el que se obtiene para la red con la solución implementada. Destaca sin embargo la mejora de alrededor del 70% en cuanto a disminución de paquetes ARP para el caso con 5 nodos y del 50% para el caso de 10 nodos; y del 40% y 15% para 5 y 10 nodos en el caso de IPv6, respectivamente.

Por tanto, los resultados muestran que la utilización de un controlador centralizado con la solución implementada supone una notable disminución del tráfico de la red.

VI. CONCLUSIONES

En este trabajo se aborda la implementación de un algoritmo de filtrado basado en SDN para mitigar la degradación de rendimiento de los sistemas por las inundaciones de red por broadcasts debidas a protocolos de arranque. El objetivo es filtrar dichos mensajes para aumentar la escalabilidad de estas redes Ethernet. Se diseña para varios procedimientos de de ARP e ICMPv6, con un diseño escalable a futuros protocolos. Posteriormente, se realizan pruebas de rendimiento en un entorno emulado y se obtienen unos resultados satisfactorios en cuanto a la disminución del tráfico enviado en estas redes para ambos protocolos.

Como vías de investigación futuras, se propone la integración de la solución implementada en una red real de telecomunicaciones; la escalabilidad del código implementado a otros protocolos que generan tráfico broadcast o multicast como DHCP, Bonjour, etc; y el diseño de la arquitectura de la quinta generación móvil como una red Ethernet implementada sobre SDN, en la que la nube de acceso dispondría de un controlador SDN encargado de gestionar el tráfico de la red.

VII. AGRADECIMIENTOS

Este trabajo está parcialmente financiado por el Ministerio de Economía, Industria y Competitividad y el Fondo Europeo de Desarrollo Regional FEDER (proyectos TEC2016-76795-C6-4-R y TIN2013-46223-P).

REFERENCIAS

[1] Cisco, "Visual networking index: Global mobile data traffic forecast update, 2015–2020," *Tech. Rep., Cisco*, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/

visual-networking-index-vni/mobile-white-paper-c11-520862.html

[2] O. N. Foundation, "Software defined networking: The new norm for network," Tech. Rep., April 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[3] ——, "Openflow switch specification," Open Networking Foundation, Tech. Rep., 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

[4] L. Foundation, *OpenDaylight Developer Guide*, 2015. [Online]. Available: http://go.linuxfoundation.org/l/6342/2015-06-28/2l76qr/6342/128124/bk_developers_guide_20150629.pdf

[5] B. Lantz, N. Handigol, B. Heller, and V. Jeyakumar, *Introduction to Mininet*, December 2015. [Online]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

[6] A. F. Cattoni, P. E. Mogensen, S. Vesterinen, M. Laitila, L. Schumacher, P. Ameigeiras, and J. J. Ramos-Munoz, "Ethernet-based mobility architecture for 5g," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, Oct 2014.

[7] P. Ameigeiras, J. J. Ramos-Muñoz, L. Schumacher, J. Prados-Garzon, J. Navarro-Ortiz, and J. M. Lopez-Soler, "Link-level access cloud architecture design based on sdn for 5g networks," *IEEE Network*, vol. 29, no. 2, March 2015.

[8] Cisco, "Internetwork design guide - broadcasts in switched lan internetworks." [Online]. Available: http://docwiki.cisco.com/wiki/Internetwork_Design_Guide_--_Broadcasts_in_Switched_LAN_Internetworks#Table:_Average_Number_of_Broadcasts_and_Multicasts_for_Novell_Networks

[9] D. C. Plummer, *RFC 826 An Ethernet Address Resolution Protocol*, Std., November 1982.

[10] R. Droms, *RFC 2131 Dynamic Host Configuration Protocol*, Std., March 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2131.txt

[11] S. Thomson, T. Narten, and T. Jinmei, *RFC 4862 IPv6 Stateless Address Autoconfiguration*, Std., September 2007. [Online]. Available: https://tools.ietf.org/pdf/rfc4862.pdf

[12] M. Casado, Ed., *Ethane: Taking Control of the Enterprise*, vol. 37, no. 4. NY, USA: Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, October 2007.

[13] C. Kim, M. Caesar, and J. Rexford, "Floodless in seattle: A scalable ethernet architecture for large enterprises," vol. 29, no. 1. NY, USA: ACM Trans. Computer Systems (TOCS), February 2011.

[14] A. Conta and S. Deering, *RFC 2463 ICMP for the Internet Protocol Version 6 (IPv6)*, Std., December 1998. [Online]. Available: http://tools.ietf.org/pdf/rfc2463.pdf

[15] P. L. Weiguo, *Address Resolution Protocol*, Std., September 2015. [Online]. Available: http://icourse.cuc.edu.cn/networkprogramming/lectures/Unit2_ARP.pdf

[16] S. Deering and R. Hinden, *RFC 2460 Internet Protocol, Version 6 (IPv6) Specification*, Std., December 1998. [Online]. Available: http://tools.ietf.org/html/rfc2460

[17] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74 – 98, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614002588

[18] F. Longo, S. Distefano, D. Bruneo, and M. Scarpa, "Dependability modeling of software defined networking," *Computer Networks*, vol. 83, pp. 280 – 296, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128615001139

[19] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1 – 30, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614002254

[20] T. N. D. and K. Gray, *SDN: Software Defined Networks*, 1st ed. O'Reilly Media, Inc., 2013.

[21] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, June 2014, pp. 1–6.

[22] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," in *Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2015 12th International Conference on*, Oct 2015, pp. 223–226.

[23] "Mininet: An instant virtual network on your laptop (or other pc)." [Online]. Available: http://mininet.org

[24] L. S. Design. (2016) Linux ipv6 router advertisement daemon (radvd). (último acceso el 2/5/2017). [Online]. Available: http://www.litech.org/radvd/