Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

DISEÑO E IMPLEMENTACIÓN DE ESCENARIOS VIRTUALES DE RED PARA EVALUAR EL FUNCIONAMIENTO DE MPTCP

TRABAJO FIN DE MÁSTER

Ricardo Flores Moyano

2013

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE ESCENARIOS VIRTUALES DE RED PARA EVALUAR EL FUNCIONAMIENTO DE MPTCP

Autor
Ricardo Flores Moyano

Director
David Fernández Cambronero

Departamento de Ingeniería de Sistemas Telemáticos

2013

Resumen

Los protocolos que sustentan Internet deben actualizarse para adaptarse a los requerimientos de los nuevos servicios multimedia que se desarrollan. IPv6 es uno de los primeros pasos en la actualización, ahora se debe pensar en que TCP debe permitir que el multihoming y la agregación de recursos sea una realidad, permitiendo optimizar las conexiones, incrementando el Goodput y mejorando la robustez de las comunicaciones frente a la pérdida de determinados enlaces. MPTCP, al ser una versión modificada de TCP, implementa un transporte Multiruta haciendo uso de todas las interfaces de red presentes en el host, creando una conexión única a nivel de transporte compuesta por varias conexiones (subflujos) a nivel de red.

El objetivo principal del presente trabajo es evaluar el funcionamiento de MPTCP. Para cumplir con este objetivo se pretende utilizar escenarios virtuales de red porque es una forma económica de verificar el funcionamiento de los protocolos ya que no es necesario utilizar infraestructura de red real, además nos entrega una mayor libertad al momento de modificar determinados parámetros como ancho de banda, delay, pérdida de paquetes, interfaces, topología de red, etc. Se utiliza la herramienta VNX desarrollada en el DIT de la ETSIT-UPM, que permite la definición y la implementación automática de escenarios de red formados por máquinas virtuales de diferentes tipos.

Para poder diseñar escenarios de prueba es importante tener claro el funcionamiento general del protocolo y cuáles son sus objetivos así como considerar sus requerimientos tanto a nivel de host como a nivel de red. De las pruebas realizadas se pudo evaluar el comportamiento de MPTCP en distintas configuraciones de red y bajo diferentes parámetros como BW, RTT y Congestión.

De la experimentación se llega a la conclusión de que MPTCP, dentro del entorno de prueba virtual, funciona bastante bien cumpliendo en un gran porcentaje sus objetivos de diseño, pero en determinados casos como por ejemplo excesivos valores de RTT, tráfico TCP presente e interfaz de host única, el protocolo no tuvo un desempeño del todo satisfactorio. De las conclusiones obtenidas se plantea continuar con la experimentación del protocolo dentro de un entorno de red real que permitan posteriormente recomendar su utilización dentro de un entorno de producción.

Abstract

The protocols that support Internet must be updated to adapt to the requirements of new multimedia services develop. IPv6 is one of the first steps in the upgrade, now is time to think that TCP should allow multihoming and resource pooling become a reality, thus optimizing connections, increasing the Goodput and improving the robustness of communications against loss of certain links. MPTCP, being a modified version of TCP, implements a Multipath transport using all network interfaces present on the host, creating a single connection at the transport level composed of multiple connections (subflows) at the network level.

The main objective of this work is to evaluate the performance of MPTCP. To accomplish this goal it going to use virtual network scenarios because it is an economical way to verify the performance of protocols since it is not necessary to use real network infrastructure also gives us more freedom when modifying certain parameters such as bandwidth, delay, packet loss, interfaces, network topology, etc. It uses VNX tool developed at the DIT of ETSIT-UPM, which allows the definition and automatic deployment of network scenarios consisting of virtual machines of different types.

In order to design test scenarios is important to understand the general operation of the protocol and what your goals are, also consider the requirements both at the host and network level. Of the tests performed it could be assessed MPTCP behavior in various network configurations and under different parameters like BW, RTT and Congestion.

From experimentation we reach the conclusion that MPTCP, in the virtual test environment, it works pretty well in a large percentage meeting its design goals, but in certain cases such as excessive RTTs, TCP traffic present and interface single host, protocol performance was not entirely satisfactory. From the conclusions obtained it is important to continue the experimentation inside a real network environment to recommend the use of MPTCP within a production environment.

Índice general

Índice de figuras

Siglas

MPTCP	MultiPath Transmission Control Protocol
RFC	Request For Comments
WLAN	Wireless LAN
3G	Tercera generación de telefonía móvil
IP	Internet Protocol
ISP	Internet Service Provider
RTT	Round Trip Time
VNX	Virtual Networks over linuX
BW	Bandwidth
Nonce	Number used Once
ID	Identification
MAC	Message Authentication Code
SHA	Secure Hash Algorithm
BDP	Bandwidth Delay Product
XML	eXtensible Markup Language
KVM	Kernel-based Virtual Machine
Rootfs	Root File System
DIT	Departamento de Ingeniería de Sistemas Telemáticos

1 Introducción

Las redes actuales tienen como objetivo fundamental mantener la disponibilidad de su infraestructura casi al 100%. Dejar sin servicio a los usuarios o desconectar momentáneamente un servidor de Internet es una situación crítica e impensable. Frente a estas posibles situaciones los operadores de red tratan de garantizar la disponibilidad de los servicios a través del concepto de redundancia. Para asegurar la rentabilidad de sus inversiones, los operadores de red utilizan varias técnicas de ingeniería de tráfico que permita balancear la carga de tráfico a través de los enlaces disponibles. Una de estas técnicas es el Multihoming que conjuntamente con BGP permiten mejorar la fiabilidad de las conexiones. El inconveniente del Multihoming es que funciona bien para determinados entornos con IPv4 y para entornos con IPv6 todavía no existe ningún estándar y algunos problemas asociados siguen sin resolverse.

Otro escenario que muestra la subutilización de los recursos está en el ambiente de los smartphones, laptops, tablets, etc. Los dispositivos en cuestión poseen varias interfaces como la WLAN, 3G, Ethernet que no las utilizan de forma eficiente. Los dispositivos solamente pueden utilizar una interfaz a la vez para el envío de tráfico. A todo esto se suma la creciente demanda de Ancho de Banda, no solamente a nivel de empresarial sino también a nivel de usuarios domésticos debido a la nueva clase de servicios multimedia que se van desarrollando. El tráfico de Internet funciona por ráfagas y desplegar infraestructura para soportar la creciente demanda de ancho de banda no es una situación sencilla para los proveedores debido a que realizan una fuerte inversión económica. Alcanzar una buena QoS no sólo depende de la ampliación de la capacidad de los enlaces, más bien se debe mejorar los protocolos que sustentan Internet.

El concepto de Multiruta es inherente a la Internet actual pero nos damos cuenta que TCP es un protocolo de ruta única, es decir, cuando una conexión TCP se establece, la conexión se enlaza a las direcciones IP de los dos hosts que están comunicándose. Esta situación crea inconsistencias tangibles. MPTCP es la respuesta a la necesidad de actualización que requiere TCP para permitir la evolución de los servicios que se pueden ofrecer en Internet utilizando de forma óptima todas las interfaces de red presentes en servidores y dispositivos finales.

1.1 Motivación

El estudio de MPTCP permite comprender los beneficios a obtener de la posibilidad de utilizar todas las interfaces físicas presentes en un host para establecer una única sesión a nivel de la capa de transporte. Las RFCs desarrolladas hasta el momento plantean objetivos a MPTCP indicando que mejoraría el Goodput, la robustez de las conexiones y aliviaría de cierta forma los inconvenientes derivados de la congestión.

En vista de esto es importante trasladar el funcionamiento teórico a escenarios de red que nos permitan evaluar el protocolo desarrollado para comprobar su funcionamiento así como modificar parámetros de red para analizar el comportamiento resultante.

Estar en contacto directo con el protocolo nos permitirá comprenderlo mejor, adquirir habilidades nuevas dentro del ambiente Linux, proponer situaciones específicas de análisis en base a experiencias previas y lo más importante tener la capacidad de recomendar o no su implementación en un ambiente de producción como Internet.

1.2 Objetivos

El objetivo principal del presente trabajo es comprobar si MPTCP puede crear una sesión única a nivel de la capa de transporte que pueda manejar más de una conexión a nivel de la capa de red, incrementando el BW total del host. Existen cuestiones adicionales a verificar y se desprenden de los objetivos propios de MPTCP planteados en la RFC 6356 (Coupled Congestion Control):

- Equidad frente a sesiones TCP establecidas sobre los enlaces utilizados por MPTCP.
- El desempeño de todos los subflujos de MPTCP debería ser al menos igual al de TCP sobre uno de los trayectos que utiliza MPTCP.
- MPTCP debería preferir trayectos que estén experimentando menos congestión.

Adicionalmente se pretende observar el comportamiento de MPTCP bajo las siguientes circunstancias:

- Host con interfaces de distinto BW.
- Redes con un alto RTT y búfer.
- Robustez del protocolo frente a pérdidas de conectividad.

- Interacción con dispositivos de capa 3.
- Hosts con número de interfaces de red distintas.

1.3 Estructura del documento

El presente trabajo se compone de los fundamentos teóricos de MPTCP y conceptos complementarios, las pruebas realizadas sobre los escenarios virtuales así como sus resultados y las conclusiones y recomendaciones obtenidas de la comprensión oportuna del funcionamiento observado. El presente trabajo de estructura de la siguiente manera:

- *Capítulo 1*, Introducción: Se reflexiona sobre la situación actual de Internet y lo que se podría hacer para mejorar su desempeño. Además se establecen los objetivos de investigación.
- *Capítulo 2*, Fundamentos Teóricos: Se realiza la explicación teórica para comprender el funcionamiento de MPTCP así como conceptos complementarios que permitieron desarrollar las pruebas con el protocolo.
- *Capítulo 3*, Diseño de escenarios de Prueba e Implementación: Considerando los objetivos de la investigación y la funcionalidad de VNX se procedió a diseñar escenarios de red que permitan probar las distintas características de MPTCP dentro de un entorno virtual.
- *Capítulo 4*, Conclusiones: Se plantean un conjunto de conclusiones y recomendaciones obtenidas de la comprensión oportuna del funcionamiento observado. Además se plantean trabajos futuros a realizarse.
- Referencias.
- Anexos.

2 Fundamentos Teóricos

En esta parte se pretende colocar el material teórico necesario para comprender que es MPTCP y cuál sería una buena estrategia para desplegar un entorno de prueba que se ajuste a nuestros objetivos de investigación.

2.1 Multihoming

El multihoming es una técnica utilizada para incrementar la fiabilidad de la conexión de un host dentro de una red. Está definido como la conexión de un host a más de una red.

El diagrama mostrado en la figura 1 ilustra un escenario típico donde se puede utilizar MPTCP. Dos hosts A y B se comunican entre sí. Estos hosts poseen más de una interfaz y más de una dirección IP, proporcionando más de una conexión independiente a Internet. Es importante mencionar que pueden existir hasta cuatro formas distintas de establecer conectividad entre A y B: A1-B1, A1-B2, A2-B1 y A2-B2. MPTCP aprovecha esta posibilidad de establecer distintas conexiones a través de todas las interfaces para crear una malla completa de conexiones entre los hots. Este comportamiento lo veremos posteriormente en la sección de experimentos.

Figura 1. Escenario Multihomed

2.2 Agrupación de recursos

La agrupación de recursos permite que los recursos de red se comporten como un solo recurso lógico. El objetivo es aumentar la fiabilidad, la flexibilidad y la eficiencia de los enlaces a través de mecanismos que desplazan la carga entre las distintas partes de la red.

En la figura 2 se observa que la Agrupación de Recursos permite optimizar el consumo de Ancho de Banda haciendo uso de todos los recursos disponibles mediante un balanceo de carga. Como veremos más adelante MPTCP utiliza este principio junto al de Multihoming para establecer un nuevo concepto dentro de la capa de transporte.

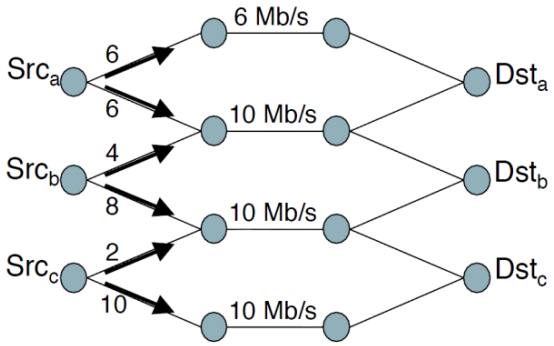


Figura 2. Agrupación de Recursos

2.3 MPTCP

TCP Multiruta (MPTCP) es una versión modificada de TCP que implementa un transporte Multiruta. Consigue este objetivo mediante la agrupación de múltiples rutas dentro de una única conexión en la capa de transporte, de forma transparente para la aplicación [1].

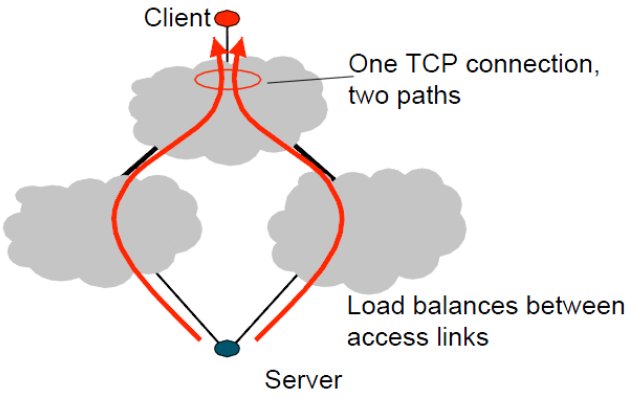


Figura 3. Visión de MPTCP

De la posibilidad de utilizar múltiples rutas se desprenden las siguientes ventajas:

- Aumenta la eficiencia en el uso de recursos, y por lo tanto aumenta la capacidad de red disponible para los dispositivos finales (Goodput).
- Aumenta la capacidad de recuperación de la conectividad al proporcionar múltiples rutas, protegiendo a los dispositivos finales de la falla de un enlace.

Los objetivos principales para MPTCP son: ser implementable y utilizable sin cambios significativos en la infraestructura existente de Internet, ser utilizable para las aplicaciones sin modificarlas, ser estable y seguro en la congestión sobre la amplia gama de rutas existentes en Internet, incluyendo las interacciones con NAT. MPTCP debe funcionar sobre cualquier ruta de Internet donde TCP funcione.

2.3.1 Modelo de Referencia

La arquitectura de MPTCP está basada en las ideas propuestas por "Transport next-generation" (Tng). Tng divide la capa de transporte en dos subcapas (Semántica y Flow+endpoint). MPTCP, que proporciona compatibilidad de aplicaciones a través de la conservación de la semántica de TCP (ordenación global de datos y fiabilidad), es una instancia de la capa semántica "orientada a la aplicación". Y el componente de subflujo TCP, que proporciona compatibilidad con la red al aparecer y comportarse como un flujo TCP en la red, es una instancia de la capa Flow+endpoint "orientada a la red" [1].

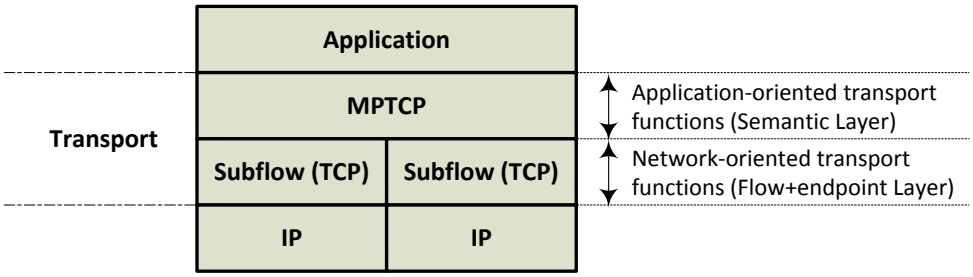


Figura 4. Descomposición de la capa de transporte en MPTCP

Como se puede observar en la figura 4 la instancia MPTCP situada bajo la capa de aplicación debe gestionar múltiples subflujos TCP, para ello implementa las siguientes funciones:

- *Gestión de rutas*: Es la función para detectar y utilizar varias rutas entre dos hosts. Permite publicar direcciones IP alternativas y establecer nuevos subflujos para que se unan a la conexión MPTCP existente.

- *Planificación de paquetes*: Esta función separa el flujo de bytes recibidos desde la aplicación en segmentos para que sean transmitidos sobre uno de los subflujos disponibles. MPTCP hace uso de un mapeo de secuencia de datos, asociando los segmentos enviados en diferentes subflujos a un número de secuencia de nivel de conexión, permitiendo así que los segmentos enviados en diferentes subflujos sean reordenados correctamente en el receptor.
- *Interfaz de subflujo (TCP de ruta única)*: El componente de subflujo toma segmentos desde el componente de Planificación de paquetes y los transmite sobre la ruta especificada. MPTCP utiliza TCP por debajo para la compatibilidad con la red. TCP añade sus propios números de secuencia a los segmentos, los cuales se utilizan para detectar y retransmitir paquetes perdidos en la capa de subflujo. En el lado del receptor el segmento tiene que subir a través de la pila para obtener el flujo de bytes original para entregárselo a la capa de aplicación.

2.3.2 Control de Congestión

Uno de los componentes más importantes de TCP es su control de congestión, lo que le permite adaptar el BW de forma dinámica en respuesta a las cambiantes condiciones de la red. Para ello cada emisor TCP mantiene una ventana de congestión (*cwnd*), que regula la cantidad de paquetes que el emisor puede enviar sin esperar un asentimiento, si todo va bien la ventana de congestión crece linealmente y se reduce a la mitad cuando se produce una pérdida de paquetes (posible congestión). Por último, el control de congestión asegura la *equidad*: cuando múltiples conexiones utilizan el mismo enlace congestionado, cada uno de ellos de forma independiente converge al mismo valor promedio de la ventana de congestión.

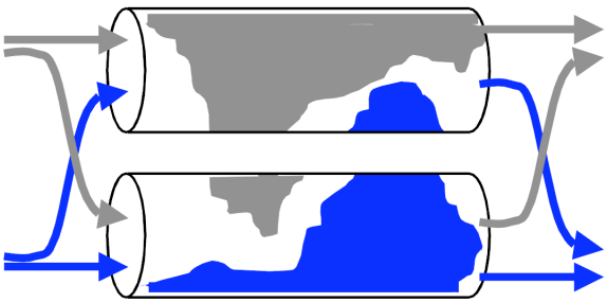


Figura 5. Congestión sobre Múltiples rutas

De acuerdo con el control de congestión de TCP analizado, MPTCP tendría que solventar el inconveniente de la congestión en Múltiples rutas (figura 5) y por lo tanto define tres objetivos que se deben cumplir [2]:

1. Garantizar la equidad para TCP. Si varios subflujos de la misma conexión MPTCP comparten un cuello de botella con otras conexiones TCP, MPTCP no debería tomar más BW que TCP.
2. El BW de todos los subflujos juntos debería ser al menos igual al de una conexión TCP sobre cualquiera de las rutas utilizadas por la conexión MPTCP, garantizando que existe un incentivo para el despliegue de MPTCP.
3. MPTCP debería preferir rutas eficientes, lo que significa que debería enviar un mayor porcentaje de tráfico por las rutas que experimentan menor congestión.

Luego de haber analizado el modelo de referencia con sus componentes funcionales y los objetivos del Control de Congestión sobre múltiples rutas es importante mencionar un ejemplo que permita comprender la interacción del conjunto. Todo comienza en la capa de aplicación cuando existen datos para ser transmitidos. La *Gestión de rutas* se ocupa del descubrimiento de múltiples rutas entre los hosts. El *Planificador de paquetes* a continuación recibe un flujo de datos desde la aplicación con destino hacia la red, y lleva a cabo las operaciones necesarias sobre él (por ejemplo, la separación de datos en segmentos a nivel de conexión, y la adición de números de secuencia a nivel de conexión) antes de enviarlo a un subflujo. El *subflujo* (nivel de subflujo) añade sus propios números de secuencia, ACKs y los pasa a la red. El *subflujo* receptor reordena los datos (si es necesario) y los pasa al componente de *Planificación de paquetes*, que realiza un reordenamiento a nivel de conexión y envía el flujo de datos a la aplicación.

Por último, el componente de *Control de congestión* existe como parte de la *Planificación de paquetes,* para planificar qué segmentos se deben enviar, a qué *velocidad* y en que *subflujo*.

2.3.3 Señalización

MPTCP usa el campo *Opciones* de la cabecera TCP para la señalización adicional. Con este mecanismo, la señalización requerida para operar MPTCP se transporta por separado de los datos, permitiendo que su creación y procesamiento sea independiente, lo que permite retener la compatibilidad arquitectónica con las entidades de red.

Existen desventajas en el uso del campo *Opciones* como un *espacio limitado*, *interacción con middleboxes* y *problemas de confiabilidad* [1]. El diseño detallado de MPTCP alivia estos problemas en la medida de lo posible, considerando

cuidadosamente el tamaño de las opciones MPTCP para no tener pérdidas en los datos de control que obliguen a conmutar a un funcionamiento normal de TCP.

2.3.4 Establecimiento de una conexión MPTCP

Para explicar de forma general como se establece una conexión MPTCP, se utiliza el esquema de la figura 6, así como las recomendaciones planteadas en [3].

Primero el Host A envía un paquete SYN al Host B, igual al procedimiento realizado por una conexión TCP normal. Este paquete contiene sin embargo la opción MP_CAPABLE en el campo Opciones de TCP. Luego el Host B responde con un paquete SYN/ACK que también contiene la opción MP_CAPABLE (solo cuando se puede utilizar MPTCP). Los dos paquetes también contienen las llaves (key) de Autenticación e información adicional necesaria por MPTCP. Finalmente el Host A responde con un ACK al Host B, el cual contiene las dos llaves de los paquetes anteriores. En este punto el primer subflujo está configurado y al igual que TCP es una negociación en tres pasos. Posteriormente se pueden establecer subflujos adicionales a través de las demás interfaces. En el ejemplo tanto el Host A como el Host B podrían crear un nuevo subflujo, pero se recomienda que el emisor (Host A) realice esta operación.

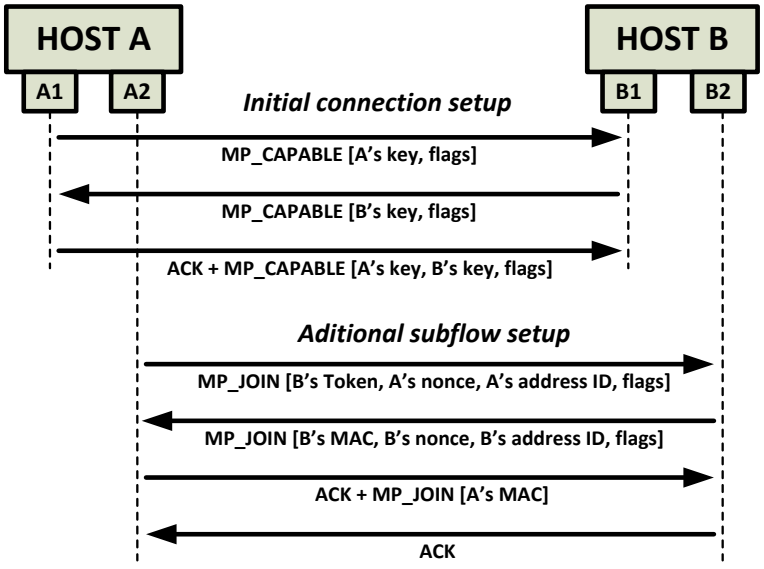


Figura 6. Negociación en MPTCP

La creación de un subflujo adicional es también una negociación con SYN, SYN/ACK y ACK pero ahora la opción MP_JOIN es incluida. El Host A también envía

con el primer SYN, un Token del Host B, generado con las llaves intercambiadas previamente. Esto es utilizado para identificar y autenticar el primer subflujo. Además un "nonce" es enviado para evitar ataques de reinyección. El ID de dirección enviado especifica una dirección en concreto, permitiendo que el subflujo se pueda distinguir de los demás. Debido a este ID de dirección la conexión es independiente de la dirección de origen. Los *flags* se pueden utilizar para opciones más avanzadas (por ejemplo utilizarlo solo como enlace de respaldo).

Cuando el paquete SYN con la opción MP_JOIN del subflujo es verificado por el Host B, este responde con un SYN/ACK. Este ACK contiene un Código de Autenticación de Mensaje (MAC), un nonce aleatorio y el ID de dirección para el Host B. El MAC es utilizado para la autenticación del subflujo. Luego el Host A responde con un paquete ACK y la opción MP_JOIN, también con los datos adicionales al igual que lo realizó el Host B. Luego de esto el Host B necesita asentir este paquete con otro ACK ya que el Host A debe estar seguro de que el Host B recibió el paquete. Para todas las técnicas de autenticación SHA-1 es utilizado.

2.3.5 Limitaciones en el Búfer de recepción

TCP y MPTCP proporcionan una entrega de datos confiable y en orden a la aplicación. La red puede reordenar paquetes o perderlos, por lo que el receptor debe almacenar los paquetes en desorden antes de enviar un ACK acumulativo y pasarlos a la aplicación. En consecuencia, el emisor también asigna un pool (depósito) de memoria de tamaño similar para almacenar los segmentos en tránsito hasta que sean asentidos [4]. Esta situación obliga a considerar un tamaño específico del búfer de recepción. En la ausencia de pérdidas, se necesita un *Producto Ancho de Banda – Retardo* (BDP) de almacenamiento en búfer para que el control de flujo de TCP funcione correctamente. Para optimizar el tamaño del búfer de recepción, el kernel de Linux incorpora una función que permite dinámicamente ajustar el búfer de recepción de TCP para garantizar que el doble del BDP de la ruta pueda ser almacenado en el búfer del receptor. Este valor permite soportar la reordenación que realiza la red (esto requiere un BDP), así como la retransmisión rápida (otro BDP) [5].

$$Búfer\ TCP = 2 * BDP$$

$$Búfer\ TCP = 2 * (BW * RTT) \qquad (1)$$

Para el caso de MPTCP la ecuación (1) debería considerar que el receptor debe manejar el BW de distintos subflujos por lo tanto se tiene:

$$Búfer\ MPTCP = 2 * \sum(x_i * RTT_{max}) \qquad (2)$$

Donde x representa el BW del subflujo i, RTT_{max} es el RTT máximo del conjunto de subflujos. Considerando la ecuación (2) y asumiendo que en las rutas no existen pérdidas ni planificación especial de paquetes, se obtienen los mismos resultados que TCP pero trasladados al dominio de múltiples rutas:

- Permite que se continúe enviando tráfico por todas las rutas mientras se espera por la entrega de un paquete enviado tempranamente por la ruta más lenta.
- Permite que todas las rutas continúen enviando tráfico mientras cualquier ruta está experimentando una retransmisión rápida.

Se observa que los requerimientos de memoria para MPTCP son mucho mayores en comparación a TCP, debido principalmente por el término RTT_{max} de (2). Por ejemplo se necesitaría un Búfer de 500KB para un *BW* de 5 Mbps por subflujo (dos subflujos) y un *RTT* máximo de 200mS.

MPTCP emplea una serie de mecanismos como la *Retransmisión oportunista*, *Penalización de subflujos lentos*, *Autoajuste de Búfer con Nivelación* para aprovechar al máximo la memoria disponible en el host para que el tamaño del búfer de recepción no sea una limitante en la utilización del protocolo [4].

2.4 Virtualización de redes

Una vez entendido el funcionamiento de MPTCP, es momento de encontrar una forma de verificar su funcionamiento dentro de una topología de red. Una opción de primera mano sería armar una pequeña red entre dos host y dotarles de la capacidad de hablar MPTCP entre ellos. Para esto sería necesario instalar el kernel MPTCP de Linux sobre dos PCs y posteriormente realizar todas las pruebas respectivas. Esta opción no es muy conveniente desde el punto de vista económico y funcional. Mejor se consideró la opción de trabajar con el concepto de Virtualización de redes utilizando la herramienta VNX.

El desarrollo de escenarios virtuales de red es una forma económica para evaluar el funcionamiento de los protocolos ya que no es necesario utilizar infraestructura de red real, además nos entrega una mayor libertad al momento de modificar determinados parámetros como ancho de banda, delay, pérdida de paquetes, interfaces, etc.

2.4.1 Virtual Networks over Linux (VNX)

VNX es una herramienta de virtualización de código abierto de propósito general diseñada para ayudar a la creación de bancos de prueba de redes virtuales de forma automática. Permite la definición y la implementación automática de escenarios de red formados por máquinas virtuales de diferentes tipos (Linux, Windows, FreeBSD, Olive o routers Dynamips, etc) conectadas entre sí siguiendo una topología definida por el usuario, posiblemente conectados a redes externas [6].

VNX utiliza libvirt para interactuar con las capacidades de virtualización del Host (máquina Linux nativa), lo que permite el uso de la mayoría de las plataformas de virtualización para Linux (KVM, Xen, etc).

VNX está constituido de dos partes principales:

- Un lenguaje XML que permite describir el escenario de red virtual (lenguaje de especificación VNX).
- El programa VNX, que analiza la descripción de escenarios para crear y administrar el escenario virtual sobre una máquina Linux.

La figura 7 nos permite comprender de forma general el funcionamiento de VNX. Para una mayor profundización en la comprensión de la arquitectura de VNX se puede observar [7].

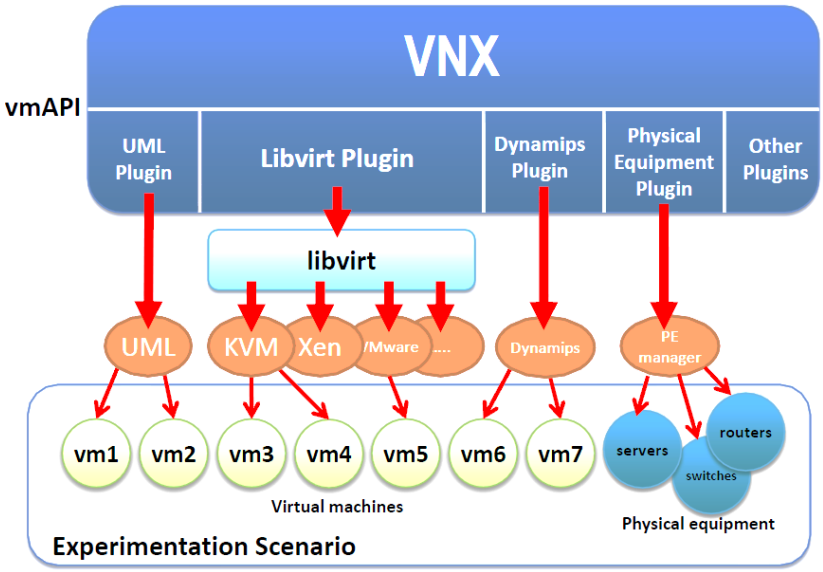


Figura 7. Arquitectura VNX

3 Diseño de escenarios de prueba e Implementación

Al reconocer el modo de trabajo y funcionalidades de VNX es momento de diseñar escenarios de red para probar las características fundamentales de MPTCP así como verificar el comportamiento cuando determinados eventos ajenos a una situación normal ocurren (fallos en las interfaces, congestión, latencia, etc.).

3.1 Requerimientos

Para poder diseñar los escenarios de prueba primero es importante mencionar los requerimientos tanto a nivel de host como a nivel de red por parte de MPTCP. En esta parte se especifican los requisitos principales que son fácilmente deducibles de la comprensión del protocolo. Conforme se fueron implementando los escenarios dentro de VNX se presentaron inconvenientes que fueron resueltos en su momento y se los explicará en el transcurso del presente capítulo.

3.1.1 Sistema Operativo

Lo primero es habilitar a nuestros hosts para que hablen MPTCP con sus pares. Ip networking lab (INL) ha desarrollado un Kernel MPTCP de Linux que incorpora el protocolo. En [8] se indican varios métodos para poder instalar MPTCP. De acuerdo a los recursos disponibles se ha optado por instalar el protocolo desde los repositorios existentes en INL. El kernel desarrollado es para arquitecturas de 32/64 bits y específicamente para distribuciones basadas en Debian (Ubuntu 12.10, Debian Squeeze, Debian Wheezy).

Considerando estos requisitos se ha creado un rootfs que implementa un kernel MPTCP con la versión estable 0.86 #4 desde el cual se crearán todas las máquinas virtuales dentro de VNX. Para propósitos de prueba se ha implementado un rootfs con interfaz gráfica (lubuntu) y un rootfs sin interfaz gráfica (Ubuntu). Es recomendable actualizar el sistema operativo periódicamente una vez instalado el kernel MPTCP.

Adicionalmente se utiliza un rootfs de Ubuntu 12.10 con un kernel genérico que solamente tiene implementado TCP. El objetivo es utilizarlo para realizar pruebas conjuntas de MPTCP con TCP y observar el comportamiento resultante.

3.1.2 Interfaces de red

Los hosts deben ser multihomed, es decir poseer dos o más conexiones de red independientes. En los escenarios de prueba propuestos, un host siempre tendrá disponible dos interfaces de red mientras que su par MPTCP podrá tener entre una o dos interfaces de red, dependiendo de la característica a ser observada. Esta situación en la práctica no representa ninguna complicación ya que en el archivo XML, que VNX utiliza, se definirán las interfaces de red necesarias para un determinado escenario de prueba.

3.1.3 Configuración del encaminamiento

Con múltiples direcciones definidas en varias interfaces es importante indicarle al kernel que seleccione una interfaz y Gateway en función de la dirección de origen en vez de utilizar las predeterminadas [8]. Esto se consigue mediante la configuración de una tabla de encaminamiento por interfaz de salida, la misma que es identificada por un número. Para solventar esta situación se ha creado un script que contiene la configuración del encaminamiento propuesta por INL (anexo 1). Cada vez que se inicialice un escenario de prueba, dentro de cada host MPTCP se debe ejecutar el script de encaminamiento previamente mencionado.

3.1.4 Monitoreo del Ancho de Banda

Se ha creado un script (anexo 2) que nos entrega el consumo de ancho de banda de cada host, mostrando también el consumo por cada interfaz. Este script nos permite verificar si MPTCP hace uso de todas las interfaces presentes así como observar el balanceo de carga realizado. Existen herramientas específicas, como iftop o iptraf e inclusive el propio Iperf, que se lo utilizará en las pruebas, que permiten observar el consumo de BW de una determinada interfaz y obtener las respectivas estadísticas. Sin embargo se ha decido crear un script a medida para que se adapte a nuestros requerimientos, modificando su funcionamiento según las características del escenario.

Este script toma muestras del BW por interfaz cada segundo durante el tiempo que dure la prueba, calcula el BW total del host y los resultados los almacena en un archivo txt para construir posteriormente las gráficas y realizar los análisis respectivos.

3.1.5 Generación de tráfico

Para crear tráfico se ha utilizado la herramienta *Iperf* que crea flujos de datos TCP entre los dos extremos finales de la red (hosts). Cada experimento cuenta con un script (anexo 3) en el cual se detallan las actividades a realizar en el host local y remoto. Todas las pruebas se ejecutan desde el cliente Iperf localizado en el host 1 hacia el servidor Iperf localizado en el host 2 (son unidireccionales). Se desea observar un comportamiento de MPTCP estable por lo que la duración de cada prueba es de 5 minutos. Es importante mencionar que todos los scripts creados son modificables y permiten especificar características para un determinado experimento según el comportamiento a ser observado.

3.1.6 Descripción del escenario de prueba

Como se había indicado previamente, VNX necesita una descripción del escenario de prueba donde se especifiquen parámetros como: número de redes existentes, sistema operativo a cargar en la máquina virtual (rootfs), interfaces de red por host, esquema de direccionamiento IP, conexiones de red, etc. Esto se lo realiza mediante la creación de un archivo XML que contiene toda la información que VNX necesita para desplegar el escenario de red virtual. Para el desarrollo del trabajo de investigación se han creado 4 escenarios de prueba que son:

- simple_mptcp.xml
- simple_mptcp2.xml
- simple_mptcp3.xml
- simple_mptcp4.xml

El contenido del archivo XML es muy sencillo de entender, permitiendo realizar modificaciones en pocos instantes. A continuación se muestra un ejemplo de ejecución de un escenario de prueba con VNX:

```
#sudo vnx -f simple_mptcp.xml -v --create
```

Es importante considerar que una modificación en el rootfs base, de donde se crearán las distintas máquinas virtuales, obliga a la destrucción del escenario de prueba y posterior creación para que dichas modificaciones sean actualizadas. Por tal motivo es altamente recomendable que cualquier archivo creado en una máquina virtual sea respaldado para evitar inconvenientes.

3.2 Escenario básico de prueba

En una primera instancia se ha diseñado un escenario básico de prueba. La figura 8 muestra el diagrama de red para analizar las siguientes cuestiones:

- Si MPTCP utiliza todas las interfaces de red presentes en el host.
- La señalización que realiza MPTCP mediante el campo de opciones de la cabecera TCP para establecer las sesiones y añadir los subflujos.
- BW presente en cada interfaz de red.
- Parámetros de funcionamiento adecuados para realizar la simulación actual y simulaciones posteriores.

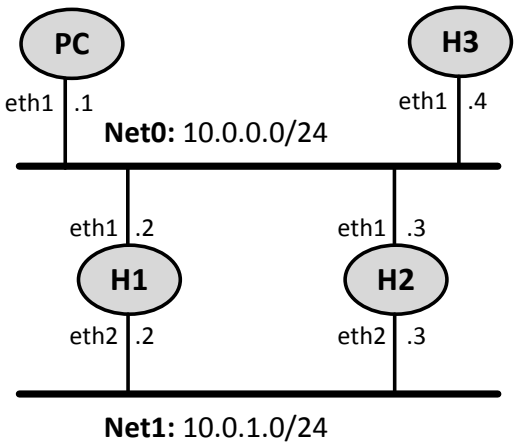


Figura 8. Diagrama de red del escenario básico de prueba

El escenario consta de dos hosts H1 y H2, cada host dispone de dos interfaces Ethernet (eth1 y eth2) para representar un dispositivo multihomed. Se han definido dos redes Net0 y Net1 a las cuales están conectadas H1 y H2. Adicionalmente se incluye la entidad PC para tener acceso remoto en la topología y al host H3 que nos permitirá realizar pruebas conjuntas de MPTCP y TCP.

3.2.1 Configuraciones previas a la ejecución del experimento

Una vez que VNX logra desplegar el escenario de prueba es importante considerar las siguientes tareas preparatorias antes de ejecutar la prueba con exp-1.sh:

1. Verificar que las interfaces de los hosts están activas y funcionales.

2. Verificar que el script que configura el encaminamiento (routing-h1.sh) contiene las direcciones IP correspondientes al Host donde está ubicado.
3. Ejecutar el script routing-h1.sh en cada host que contiene el kernel MPTCP de Linux.
4. Editar el script monitor2int.sh para que tome muestras durante el tiempo que durará el experimento.
5. Editar el script exp-1.sh para especificar el tiempo que durará el experimento así como la dirección del host remoto contra el que se realizarán las pruebas con Iperf. Adicionalmente especificar las tareas locales y remotas a realizar.
6. Ejecutar exp-1.sh
7. Observar las capturas de BW realizadas por monitor2int.sh y almacenadas en el archivo resultados.txt.
8. Finalmente Construir las gráficas con una hoja de cálculo.

Esta guía es susceptible de incluir tareas adicionales en vista de las nuevas herramientas que debieron desarrollarse para solventar inconvenientes detectados. De igual manera en escenarios de prueba posteriores considerar que se van a incluir hosts adicionales para realizar el encaminamiento y la inclusión de tráfico TCP, por lo tanto sobre ellos también realizar las tareas descritas previamente.

3.2.2 Captura de paquetes con Wireshark

En una primera configuración del escenario básico de prueba (simple_mptcp.xml), H1 estaba cargado con un rootfs con interfaz gráfica (lubuntu) para realizar las capturas del tráfico con Wireshark. Posteriormente se cambió por un rootfs sin interfaz gráfica para preservar los recursos de procesamiento del host. La captura de paquetes mediante Wireshark nos permitió observar los diferentes atributos de MPTCP en el campo de opciones de la cabecera TCP. Como nos indica la RFC 6182 una negociación en tres pasos debe realizarse entre los dos host tanto en el establecimiento del primer subflujo como en los subflujos posteriores.

Es importante mencionar que para Hosts que no poseen interfaz gráfica, así como para preservar recursos de procesamiento, es recomendable utilizar tcpdump. No tiene un conjunto de herramientas tan sofisticadas como Wireshark pero se adapta a nuestras necesidades.

A continuación se presentan las capturas realizadas que nos permiten observar la señalización que MTPCP realiza mediante la inclusión de subtipos de mensajes dentro

del campo de opciones de TCP. Se ejecutó la prueba desde H1 hacia H2 durante 60 segundos.

a. Negociación en tres pasos para MPTCP (SYN):

Al inicio del establecimiento de la sesión MPTCP, el origen envía el subtipo "*Multipath Capable*" al destino, incluye su key. Este paquete verifica que ambas partes de la conexión pueden utilizar MPTCP a nivel de transporte.

```
⊟ Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, Multipath TCP
  ⊞ Maximum segment size: 1460 bytes
  ⊞ TCP SACK Permitted Option: True
  ⊞ Timestamps: TSval 649950, TSecr 0
  ⊞ No-Operation (NOP)
  ⊞ Window scale: 6 (multiply by 64)
  ⊟ Multipath TCP: Multipath Capable
      Kind: Multipath TCP (30)
      Length: 12
      0000 .... = Multipath TCP subtype: Multipath Capable (0)
      .... 0000 = Multipath TCP version: 0
    ⊟ Multipath TCP flags: 0x81
        1... .... = Checksum required: 1
        .... ...1 = Use HMAC-SHA1: 1
      Multipath TCP Sender's Key: 14501903020638344962
```

Figura 9. Subtipo *Multipath Capable*

b. Negociación en tres pasos para MPTCP (ACK) y envío de dirección IP adicional:

Dentro del establecimiento de la sesión MPTCP se puede incluir el subtipo "*Add Address*" el cual permite que tanto el origen como el destino de la conexión publiquen las direcciones IP que tienen disponibles para crear subflujos posteriormente.

```
⊟ Options: (40 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, Multipath TCP, Multipath TCP
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊞ Timestamps: TSval 649950, TSecr 646378
  ⊟ Multipath TCP: Multipath Capable
      Kind: Multipath TCP (30)
      Length: 20
      0000 .... = Multipath TCP subtype: Multipath Capable (0)
      .... 0000 = Multipath TCP version: 0
    ⊟ Multipath TCP flags: 0x81
        1... .... = Checksum required: 1
        .... ...1 = Use HMAC-SHA1: 1
      Multipath TCP Sender's Key: 14501903020638344962
      Multipath TCP Receiver's Key: 2627584060664155438
  ⊟ Multipath TCP: Add Address
      Kind: Multipath TCP (30)
      Length: 8
      0011 .... = Multipath TCP subtype: Add Address (3)
      .... 0100 = Multipath TCP IPVer: 4
      Multipath TCP Address ID: 1
      Multipath TCP Address: 10.0.0.2 (10.0.0.2)
```

Figura 10. Subtipo *Add Address*

c. Establecimiento de un segundo subflujo MPTCP (JOIN):

Para verificar si se puede crear un nuevo subflujo, MPTCP envía un subtipo "*Join Connection*". Solamente el origen de la conexión puede enviar este subtipo para establecer subflujos adicionales entre los dos hosts. Por defecto MPTCP trata de crear una malla completa de conexiones a través de todas las direcciones IP existentes en los hosts.

```
⊟ Options: (36 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, Multipath TCP
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊞ Timestamps: TSval 649951, TSecr 646379
  ⊟ Multipath TCP: Join Connection
      Kind: Multipath TCP (30)
      Length: 24
      0001 .... = Multipath TCP subtype: Join Connection (1)
      Multipath TCP Sender's MAC: 495974068
      Multipath TCP Sender's MAC: 3927234630
      Multipath TCP Sender's MAC: 2202062155
      Multipath TCP Sender's MAC: 4293021371
      Multipath TCP Sender's MAC: 394304775
```

Figura 11. Subtipo *Join Connection*

d. Intercambio normal de MPTCP (DATA SEQUENCE SIGNALING):

Este subtipo permite a MPTCP rastrear la conexión entre los hosts y contiene los datos actuales. En este paquete se incluye el asentimiento (ACK), el número de secuencia a nivel de conexión, el número de secuencia del subflujo, entre otros.

```
⊟ Options: (32 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps, Multipath TCP
  ⊞ No-Operation (NOP)
  ⊞ No-Operation (NOP)
  ⊞ Timestamps: TSval 649950, TSecr 646378
  ⊟ Multipath TCP: Data Sequence Signal
      Kind: Multipath TCP (30)
      Length: 20
      0010 .... = Multipath TCP subtype: Data Sequence Signal (2)
    ⊟ Multipath TCP flags: 0x05
        ...0 .... = DATA_FIN: 0
        .... 0... = Data Sequence Number is 8 octets: 0
        .... .1.. = Data Sequence Number, Subflow Sequence Number, Data-level Length, Checksum present: 1
        .... ..0. = Data ACK is 8 octets: 0
        .... ...1 = Data ACK is present: 1
      Multipath TCP Data ACK: 2069737397
      Multipath TCP Data Sequence Number: 4278141086
      Multipath TCP Subflow Sequence Number: 1
      Multipath TCP Data-level Length: 24
      Multipath TCP Checksum: 26238
```

Figura 12. Subtipo *Data Sequence Signaling*

e. Número de subflujos creados:

Como se menciona en el apartado *c*, MPTCP trata de crear una malla completa de conexiones utilizando todas las direcciones IP existentes (sección 2.1). La captura de Wireshark de la figura 13, nos permite observar que se establecen 4 conexiones (subflujos) entre H1 y H2. La distribución de BW entre los subflujos depende del Control de Congestión y será analizado en una sección posterior cuando se implemente el escenario de prueba avanzado 1.

	Address A	Port A	Address B	Port B
1	10.0.0.2	60493	10.0.0.3	5001
2	10.0.0.2	51599	10.0.1.3	5001

	Address A	Port A	Address B	Port B
3	10.0.1.2	40039	10.0.0.3	5001
4	10.0.1.2	41103	10.0.1.3	5001

Figura 13. Subflujos creados por MPTCP

3.2.3 Pruebas básicas realizadas

En la siguiente sección se detallan las pruebas realizadas sobre el escenario básico de prueba (simple_mptcp2.xml) así como las conclusiones parciales obtenidas de las herramientas utilizadas y de MPTCP. Los resultados obtenidos en esta primera parte son fundamentales porque permiten guiar la investigación.

- Prueba 0:

A continuación se muestra la gráfica obtenida de la prueba inicial realizada en la topología. Se ejecutó la prueba desde H1 hacia H2 durante 5 minutos.

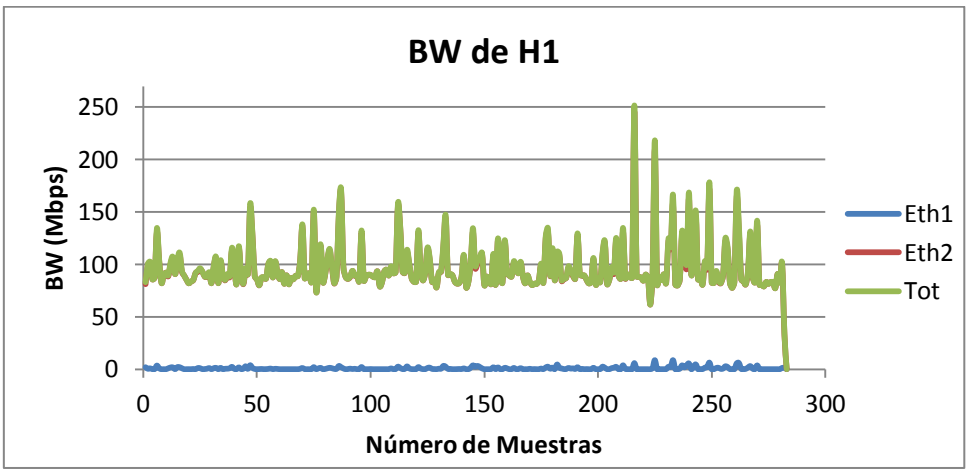


Figura 14. Resultados de la prueba 0

En la siguiente tabla se resumen los resultados obtenidos de la prueba. Es importante contrastar los resultados entregados por el script y los resultados entregados por Iperf.

Resultados				
Host	Eth1	Eth2	Total	Iperf
H1	0,814 Mbps	96,53 Mbps	97,34 Mbps	89,2 Mbps
H2	0,76 Mbps	95,84 Mbps	96,60 Mbps	88,7 Mbps

- **Análisis de resultados:**

La figura 14 muestra un ancho de banda inestable en las interfaces de H1 que también se corresponden a las de H2. En este punto es importante reconocer que no existe una referencia dentro de la simulación que nos indique cual debería ser el BW máximo admisible por cada interfaz, si bien el hardware simulado de Eth1 y eEth2 nos indica que estamos trabajando con interfaces a 1000 Mbps esto en la realidad no se está cumpliendo.

El objetivo entonces es tratar de limitar el BW a un valor específico que nos permita observar de mejor manera como MPTCP hace uso de los recursos disponibles. Para solventar este inconveniente se han utilizado las herramientas de TC (traffic controller) que vienen incluidas en el kernel de Linux [9]. La tarea consistió en comprender el funcionamiento de las herramientas de TC, así como encontrar un script que cumpla con la funcionalidad requerida para el escenario de prueba. Finalmente se encontró un script [10] que permitió limitar el ancho de banda a un valor específico utilizando la disciplina htb (anexo 4). El script permite especificar el BW tanto de transmisión como de recepción. Si se especifica un BW de recepción (DNLD) de 5 Mbps y un BW de transmisión (UPLD) de 5 Mbps, en total el host puede manejar un BW de 10 Mbps.

Considerando nuevamente las tareas previas a la ejecución del experimento, mencionadas en la sección 3.2.1, es necesario realizar lo siguiente antes del paso 5:

- Especificar el BW que tendrán las interfaces de los hosts mediante la edición del script control-bw1.sh para Eth1 y control-bw2.sh para Eth2.
- Ejecutar el script de control de BW.

Se ejecutaron nuevamente experimentos en el escenario básico de prueba para observar el funcionamiento del control de ancho de banda y otras características relevantes como: efectividad, estabilidad, adaptabilidad, escalabilidad e impacto dentro del escenario de red. Para este conjunto de pruebas de utilizó el script de control de BW en cada interfaz de H1 y H2. Los resultados obtenidos fueron satisfactorios.

- Prueba 1:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	15 Mbps	15 Mbps	15,10 Mbps	15,12 Mbps	30,22 Mbps	28,1 Mbps
H2	15 Mbps	15 Mbps	14,93 Mbps	14,97 Mbps	29,91 Mbps	27,9 Mbps

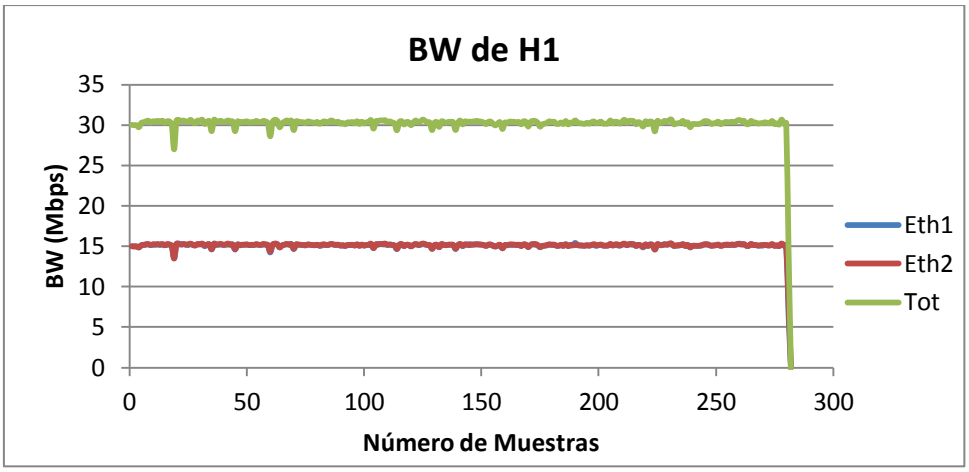


Figura 15. Resultados de la prueba 1

En la figura 14 se observa que las dos interfaces alcanzan un BW de 15 Mbps. La medición se aproxima bastante bien a lo que Iperf entrega que son 28,1 Mbps. Cabe recordar que el monitor de BW utilizado mide todo el tráfico presente en la interfaz mientras que Iperf considera solamente el tráfico útil a nivel de aplicación, por esta situación existe la pequeña diferencia entre las dos mediciones. La gráfica obtenida sobre H2 es igual, por lo tanto se va a prescindir de incluirla dentro de esta sección.

- Prueba 2:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	30 Mbps	30 Mbps	31,06 Mbps	31,08 Mbps	62,15 Mbps	56,1 Mbps
H2	30 Mbps	30 Mbps	29,94 Mbps	29,88 Mbps	59,83 Mbps	55,3 Mbps

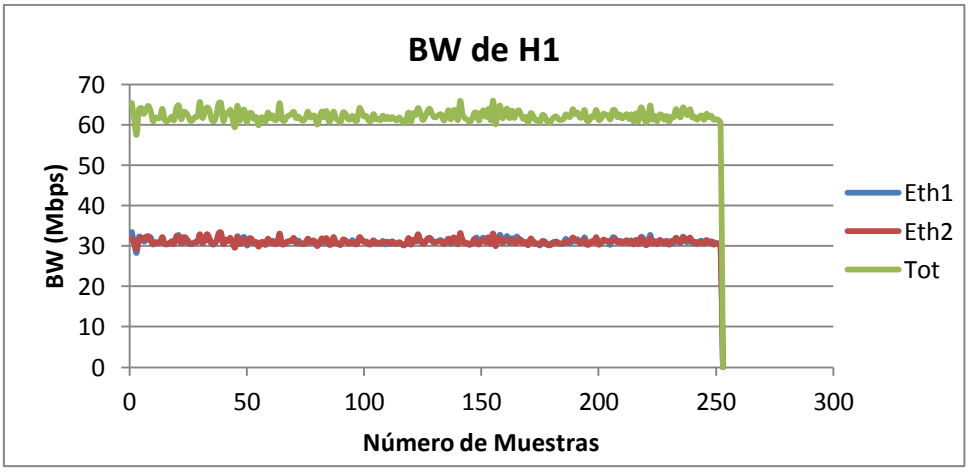


Figura 16. Resultados de la prueba 2

- Prueba 3:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	100 Mbps	100 Mbps	84,10 Mbps	0,13 Mbps	84,24 Mbps	77,4 Mbps
H2	100 Mbps	100 Mbps	83,00 Mbps	0,05 Mbps	83,05 Mbps	76,9 Mbps

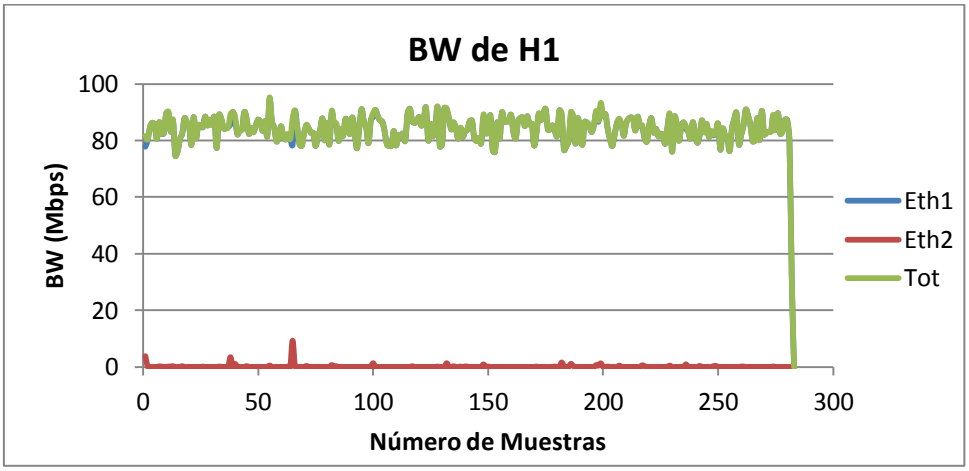


Figura 17. Resultados de la prueba 3

- **Análisis de resultados:**

Se realizaron varias pruebas con distintos valores de BW hasta alcanzar los 100 Mbps por interfaz. Se observa que al incrementar el BW la gráfica del consumo muestra una inestabilidad en el comportamiento de MPTCP, volviéndose crítica para valores cercanos a los 50 Mbps y superiores. Hasta un máximo de 20 Mbps configurado en cada interfaz, los resultados mostraron un comportamiento normal, evidenciando que MPTCP se comporta de acuerdo a la teoría.

Este conjunto de pruebas también permitió observar que al aumentar el BW disponible por cada interfaz, la captura de muestras del BW se alteró, haciendo que se obtengan menos muestras de las que se deberían obtener. Esto hace que la diferencia entre los valores que entrega Iperf y el script de monitoreo sea más notoria al aumentar el BW, al igual que la ligera diferencia entre las mediciones de H1 y H2.

La relación entre la inestabilidad y el BW se debe a la potencia limitada de procesamiento de las máquinas virtuales, ya que al incrementar el BW el procesador se exige más. Esto nos permite establecer un límite en el BW para probar MPTCP en los próximos escenarios a 20 Mbps en total por host, considerando también la complejidad de dicho escenario y teniendo en cuenta la posibilidad de preservar recursos.

Es importante mencionar que Iperf tiene varios parámetros adicionales que permiten realizar pruebas más completas. Para el caso de este escenario se utilizó una prueba unidireccional desde H1 hacia H2, la opción de prueba bidireccional (-d) no se la utilizó. Adicionalmente se probó la opción de ejecutar pruebas en paralelo (-P) para tratar de saturar el enlace a su máxima capacidad pero el resultado obtenido mostró una mayor inestabilidad.

3.2.4 Probando MPTCP en interfaces con distinto BW

Luego de haber probado el funcionamiento básico de MPTCP se decidió ampliar los experimentos sobre el escenario básico de prueba. El objetivo es observar el comportamiento de MPTCP cuando el host tiene interfaces de red con distinto BW.

- Prueba 4:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	1 Mbps	6 Mbps	1,00 Mbps	5,99 Mbps	6,99 Mbps	6,56 Mbps
H2	1 Mbps	6 Mbps	0,99 Mbps	5,97 Mbps	6,97 Mbps	6,45 Mbps

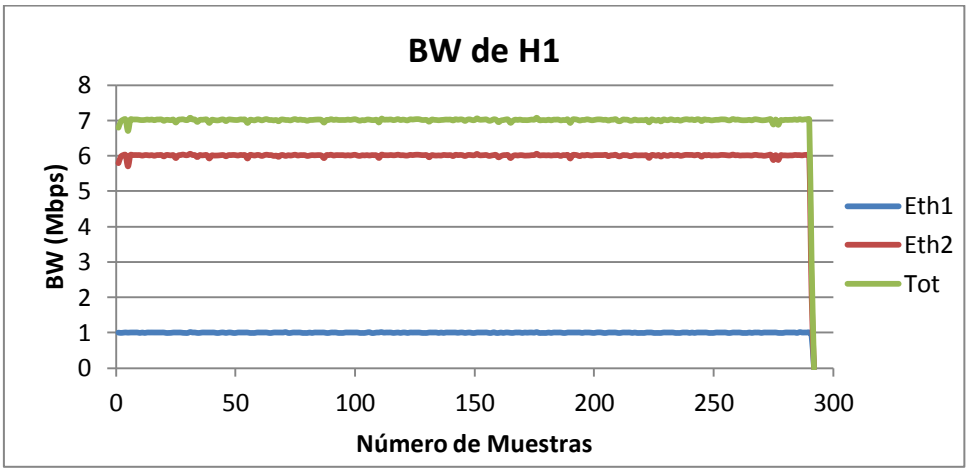


Figura 18. Resultados de la prueba 4

- **Análisis de resultados:**

Los resultados nos indican que MPTCP utiliza todo BW disponible en cada interfaz y que la diferencia de valores no influye en el comportamiento normal del protocolo.

Se realizaron pruebas con otros valores de BW en cada interfaz de H1 y H2 y los resultados fueron similares a los obtenidos en la figura 18.

3.2.5 Probando la Robustez en el Escenario básico de prueba

A continuación se procedió a probar el comportamiento de MPTCP cuando la conectividad por alguna de las interfaces del host se interrumpe. Para simular esta situación se procedió a desactivar la interfaz Eth1 de H1 con el comando ifdown y a activarla nuevamente con el comando ifup. Se utilizó el siguiente cronograma:

- En 1'30" se desactiva Eth1 de H1.
- En 3'00" se activa Eth1 de H1.

- Prueba 5:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	10 Mbps	5 Mbps	6,98 Mbps	5,00 Mbps	11,98 Mbps	11,2 Mbps
H2	10 Mbps	5 Mbps	6,93 Mbps	4,99 Mbps	11,92 Mbps	10,9 Mbps

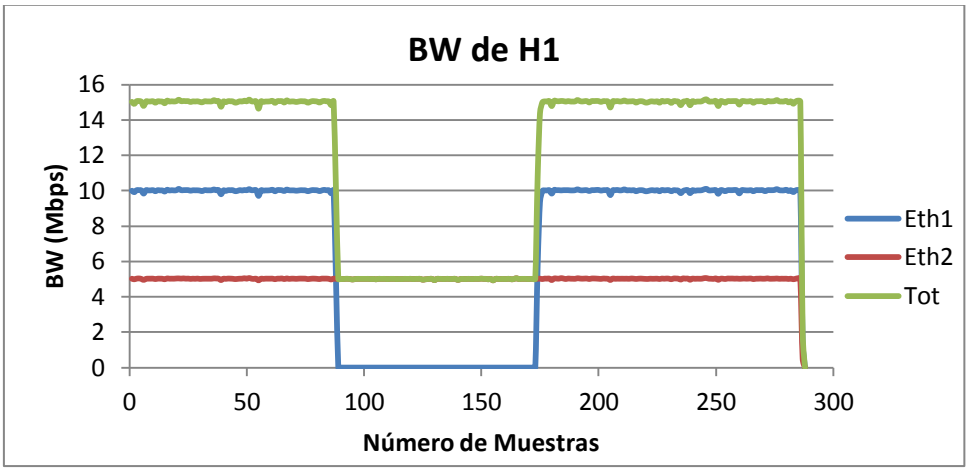


Figura 19. Resultados de la prueba 5

- **Análisis de resultados:**

La figura 19 nos muestra que la recuperación del BW total, al momento de activar nuevamente Eth1 de H1, es casi inmediata, no necesita de un tiempo largo para que un

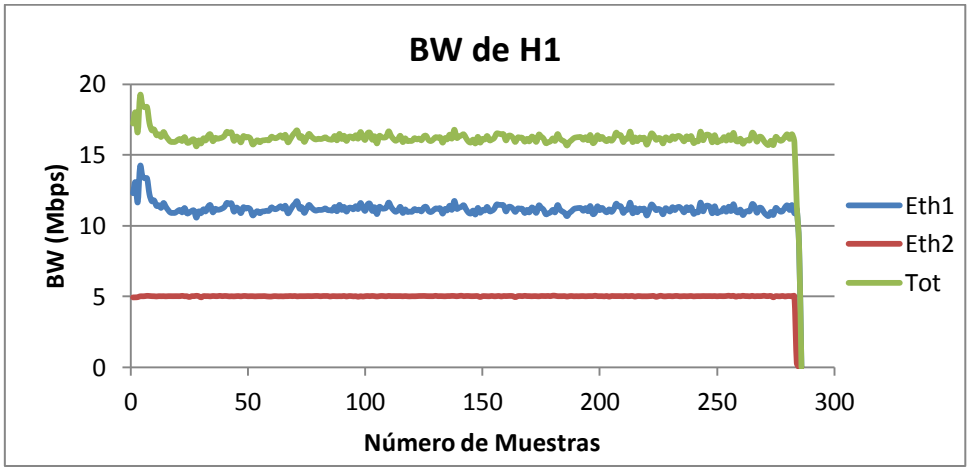
nuevo subflujo se una a la sesión MPTCP existente. De igual manera se observa que MPTCP mantiene la conexión a nivel de transporte a pesar de la pérdida del enlace, no necesita realizar ningún proceso adicional, demostrando la robustez del protocolo en estas situaciones.

3.2.6 Incluyendo tráfico TCP en la Topología

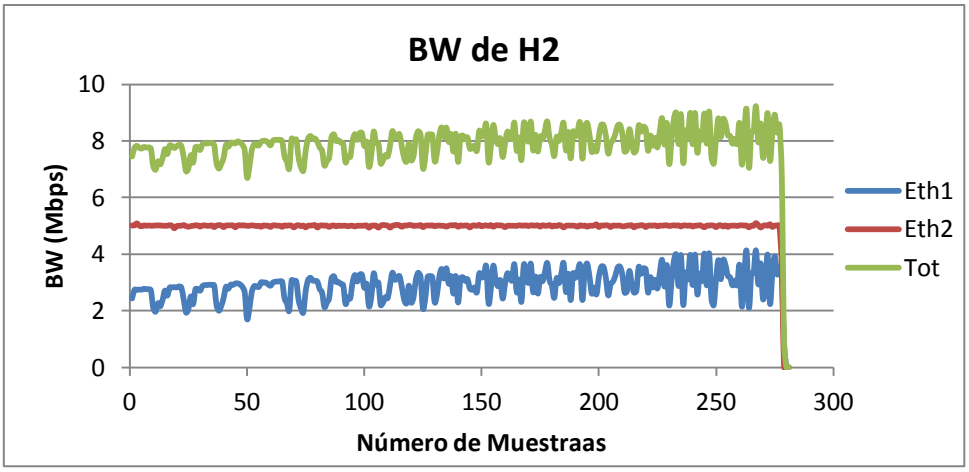
Como se había indicado previamente, la inclusión de H3 dentro de la topología nos permitirá generar tráfico TCP para observar si MPTCP es justo con TCP y no trata de ocupar el 100% del BW disponible en la red (NET0).

- Prueba 6:

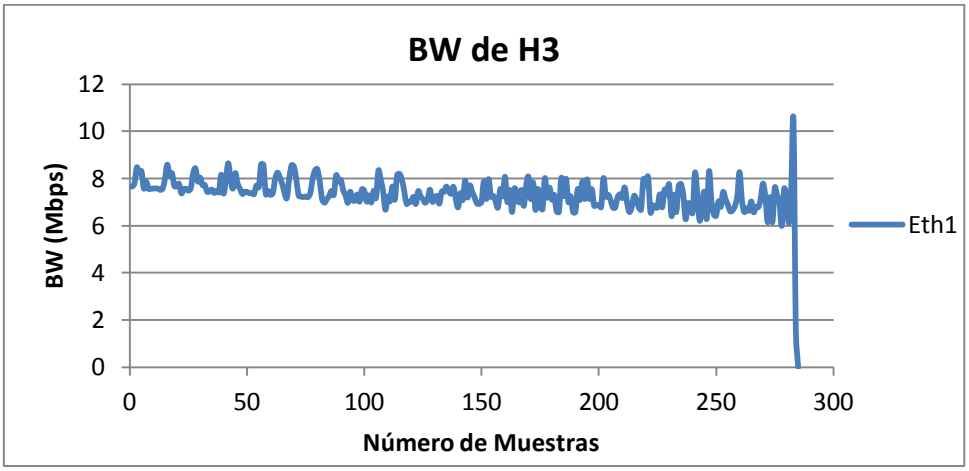
Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	10 Mbps	5 Mbps	11,19 Mbps	4,98 Mbps	16,17 Mbps	7,45 Mbps en H1-H2 7,03 Mbps en H1-H3
H2	10 Mbps	5 Mbps	2,93 Mbps	4,97 Mbps	7,90 Mbps	7,42 Mbps H1-H2
H3	10 Mbps	---	7,34 Mbps	---	7,34 Mbps	6,98 Mbps en H1-H3



(a)



(b)



(c)

Figura 20. Resultados de la prueba 6: (a) en H1, (b) en H2 y (c) en H3

- **Análisis de resultados:**

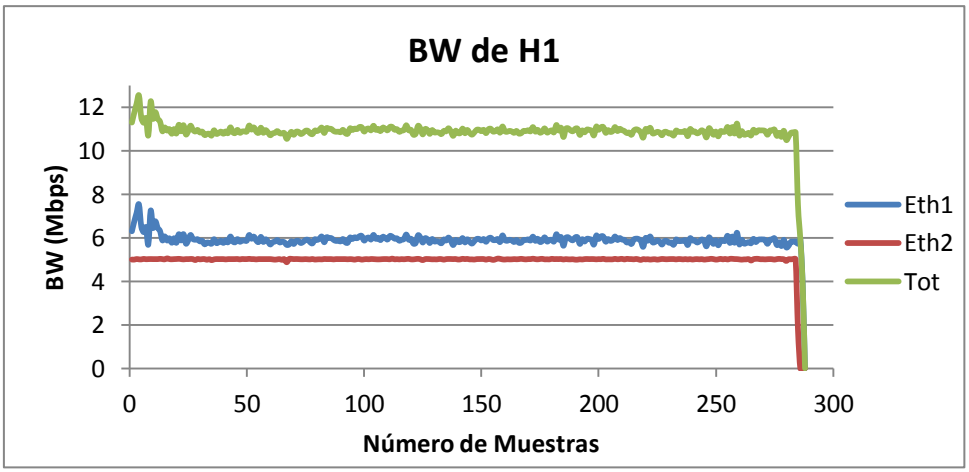
Los resultados obtenidos nos permiten observar una alteración en el control de BW sobre la interfaz Eth1 de H1, se configuró un máximo de 10 Mbps pero las mediciones nos entrega un valor de 11,19 Mbps. Al comparar este valor con los obtenidos en Eth1 de H2 más el valor de Eth1 de H3 observamos que nos entregan un valor de 10,27 Mbps (2,93 Mbps + 7,34 Mbps), que es más cercano al valor configurado en la realidad. Esta diferencia se debe tal vez a la separación temporal entre la ejecución de la prueba H1-H2 y H1-H3 y la carga presente en NET0 ya que los valores obtenidos sobre NET1 no presentan diferencias.

MPTCP no ocupa todo el BW disponible de NET0 (*primer* objetivo), según los resultados consume aproximadamente el 28,5% (2,93 Mbps), dejando el restante 71,4%

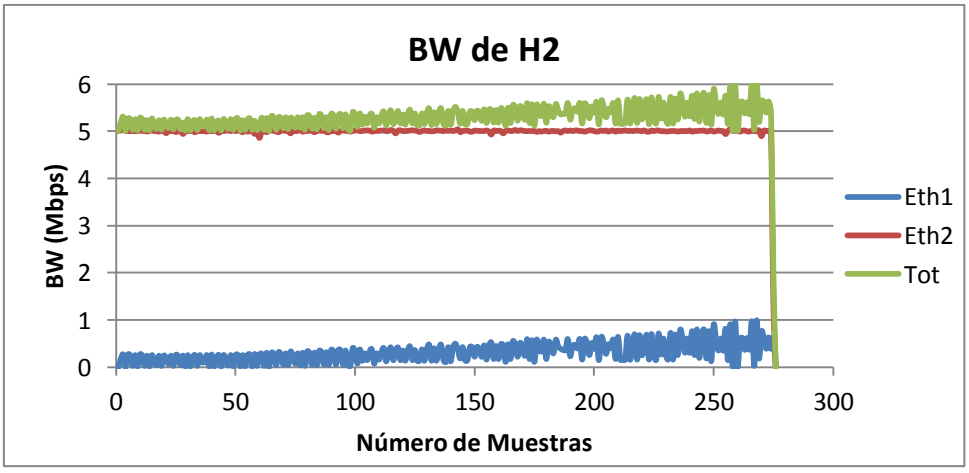
(7,34 Mbps) para la conexión TCP. La figura 20c nos indica que el BW tiende a disminuir conforme el experimento transcurre, no presenta una disminución considerable pero si notoria. Se observa que el comportamiento del BW en NET0 es irregular si lo comparamos con el BW en NET1. El comportamiento obtenido representa el ajuste que MPTCP realiza sobre el tamaño de la ventana de la conexión (control de flujo, control de congestión), debido a que H1 está saturado al mantener conexiones con H2 y H3.

- Prueba 7:

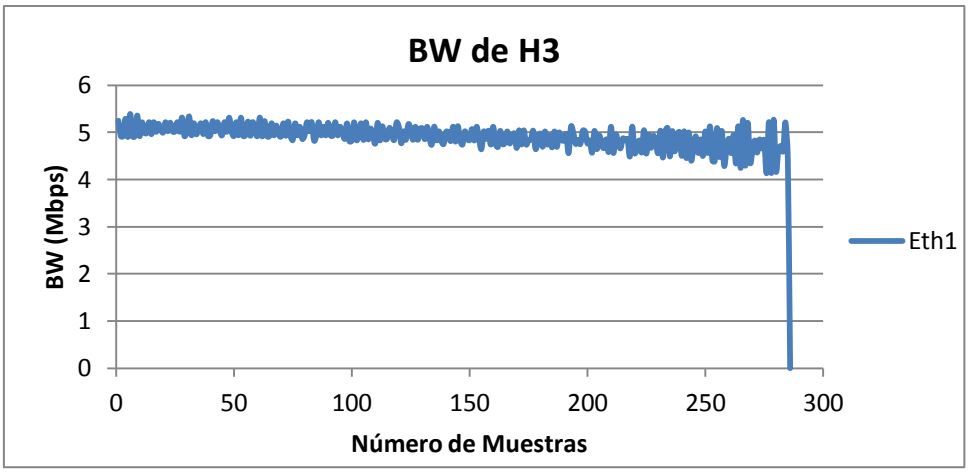
Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	5 Mbps	5 Mbps	5,9 Mbps	4,96 Mbps	10,86 Mbps	5,02 Mbps en H1-H2 4,71 Mbps en H1-H3
H2	5 Mbps	5 Mbps	0,31 Mbps	4,98 Mbps	5,29 Mbps	4,99 Mbps en H1-H2
H3	5 Mbps	---	4,91 Mbps	---	4,91 Mbps	4,65 Mbps en H1-H3



(a)



(b)



(c)

Figura 21. Resultados de la prueba 7: (a) en H1, (b) en H2 y (c) en H3

- **Análisis de resultados:**

El comportamiento de la prueba anterior se ve reflejado en esta nueva prueba. Es importante notar que el control de congestión de MPTCP cumple con su *segundo* objetivo al permitir que el BW resultante de la sumatoria de todos los subflujos es al menos igual al de TCP sobre cualquier trayecto usado por una conexión MPTCP (sección 2.3.2).

Para la prueba 6 observamos que TCP sobre H3 obtiene un BW de 7,34 Mbps y por lo tanto MPTCP sobre H2 debería al menos obtener ese valor al sumar el BW de cada subflujo. Sobrepasa el BW de H2, no por mucho (7,90 Mbps), siendo equitativo con TCP. Esta característica se observa nuevamente en la prueba 7 donde H2 tiene un BW de 5,29 Mbps ligeramente superior a los 4,91 Mbps de la conexión TCP sobre H3.

De igual manera es importante mencionar que MPTCP cumple con el *tercer* objetivo del control de congestión al colocar el mayor porcentaje de tráfico en la ruta menos congestionada. Tanto en la prueba 6 como la prueba 7, se coloca el mayor porcentaje de tráfico en NET1, a través de la interfaz Eth2 de H1 y H2, ya que sobre ésta red no existe otro tráfico más que el de MPTCP.

De la inclusión de tráfico TCP se observa que por el momento se cumplen los objetivos del control de congestión.

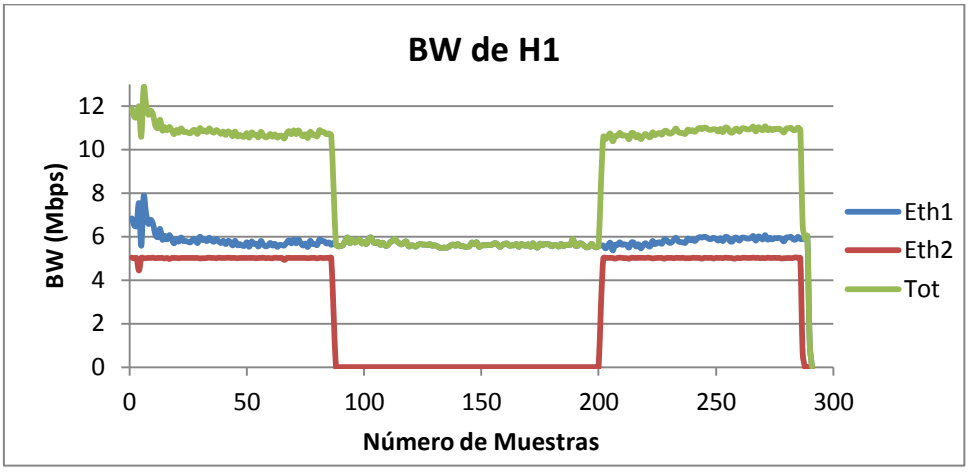
3.2.7 Analizando la robustez de MPTCP con la presencia de tráfico TCP

Se incluye una conexión TCP entre H1 y H3 para observar el comportamiento de MPTCP frente a la pérdida de conectividad de H2 y H1 en NET1. Se utilizó el siguiente cronograma:

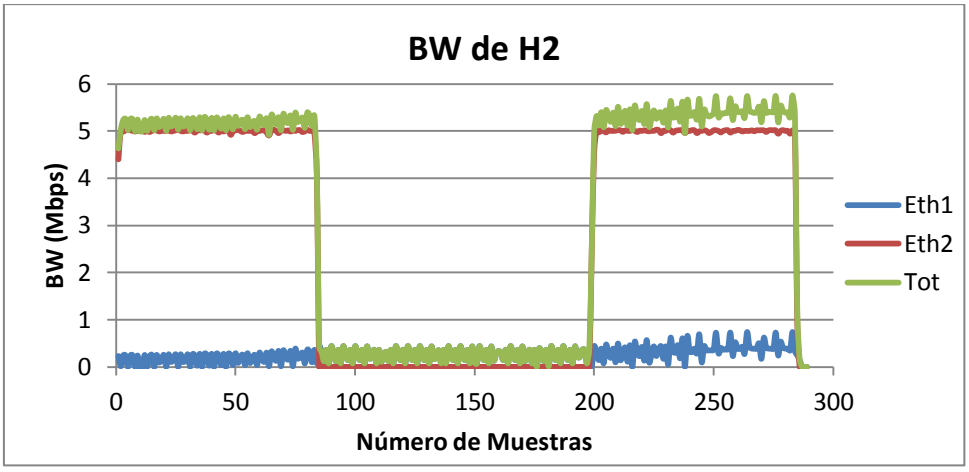
- En 1'30" se desactiva Eth2 de H2.
- En 3'30" se activa Eth2 de H2.

- Prueba 8:

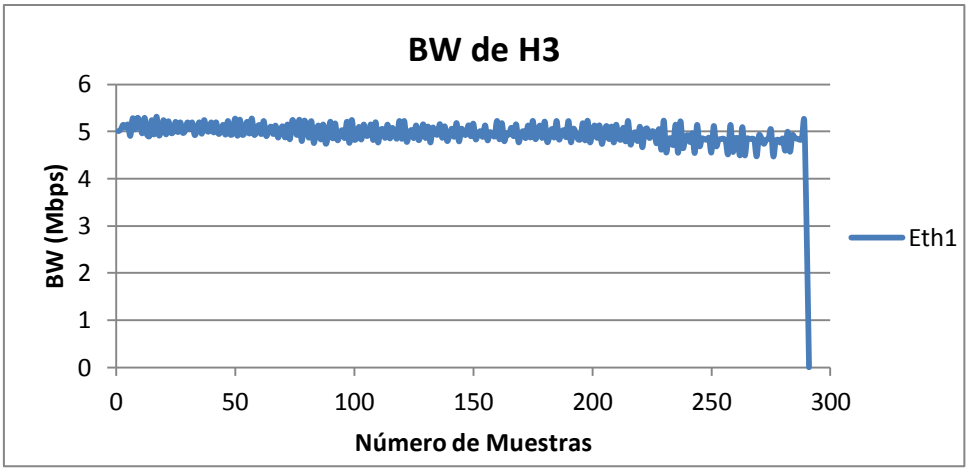
Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	5 Mbps	5 Mbps	5,74 Mbps	2,97 Mbps	8,71 Mbps	3,06 Mbps en H1-H2 4,75 Mbps en H1-H3
H2	5 Mbps	5 Mbps	0,26 Mbps	2,95 Mbps	3,22 Mbps	3,04 Mbps en H1-H2
H3	5 Mbps	---	4,95 Mbps	---	4,95 Mbps	4,69 Mbps en H1-H3



(a)



(b)



(c)

Figura 22. Resultados de la prueba 8: (a) en H1, (b) en H2 y (c) en H3

- **Análisis de resultados:**

Al eliminarse un subflujo y quedarse solamente uno, obtendríamos el equivalente a un escenario TCP con un enlace a 5 Mbps y dos conexiones establecidas. Se esperaría que existiera un balanceo de carga entre la conexión TCP y la MPTCP (2,5 Mbps por conexión) ya que el control de congestión asegura la *equidad*. El resultado muestra que la sesión TCP entre H1 y H3 ocupa casi el 100% del BW disponible, dejando a H2 solamente con 0,26 Mbps aproximadamente.

En [4] se indica que TCP tiene dos formas de indicar el cierre de la conexión: Con FIN para el cierre normal y RST para errores como cuando un extremo de la conexión deja de estar activo. En el caso de RST, MPTCP debería cerrar solamente la conexión del subflujo para que no cause el fallo de toda la conexión. Este objetivo se está cumpliendo ya que se mantiene un subflujo, pero el control de congestión no cumple con la equidad planteada tanto en el diseño de MPTCP como de TCP.

3.2.8 Comportamiento de MPTCP al incluir Delay

Como se observó en la sección 2.3.5, MPTCP tiene limitaciones en el búfer de recepción. Con el siguiente conjunto de pruebas se pretende observar el comportamiento de MPTCP conforme el Delay (RTT) presente en los enlaces se incrementa.

La herramienta Iperf permite modificar los búferes de envío y recepción mediante la opción –w (tamaño de la ventana TCP), sino se la utiliza, cada vez que se ejecuta una prueba asigna valores en función de resultados obtenidos de pruebas anteriores con el objetivo de obtener el mejor rendimiento (conclusión obtenida experimentalmente). Es importante mencionar que existen valores máximos para dichos búferes configurados en el kernel de Linux, para nuestro caso este valor es de 416KB. En una prueba posterior se va a ampliar la capacidad para que MPTCP pueda mejorar su desempeño.

Para incluir un Delay se utilizó la herramienta Netem [11] que permite simular las características de un enlace WAN. Se utilizaron los valores de la siguiente tabla para observar el comportamiento del BW con diferentes valores de Delay. Se ejecutó el experimento durante 40 segundos con cada valor de Delay. Luego los valores obtenidos se promediaron para obtener un único valor que posteriormente permitiera construir la gráfica.

- Prueba 9:

Configuraciones			Resultados (Mbps)
Delay por interfaz	Eth1 (Mbps)	Eth2 (Mbps)	
50 ms 5ms distribution normal	5	5	9,81
100 ms 10ms distribution normal	5	5	9,84
200 ms 10ms distribution normal	5	5	6,33
400 ms 20ms distribution normal	5	5	3,99
800 ms 20ms distribution normal	5	5	2,05
1600 ms 40ms distribution normal	5	5	0,91

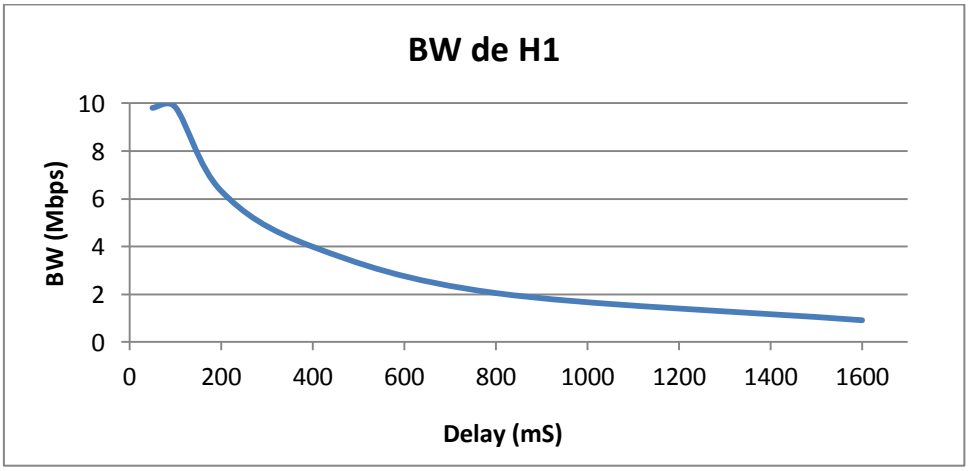


Figura 23. Resultados obtenidos de la prueba 9

- **Análisis de resultados:**

Los resultados claramente muestran que el desempeño de MPTCP decrece notablemente cuando los búferes no cuentan con la memoria suficiente para manejar el BW de los subflujos presentes en enlaces con un alto Delay.

Se realizó una prueba entre H1 y H3 para observar el comportamiento de TCP en enlaces con un alto Delay y los resultados fueron positivos, obteniéndose un BW de 5,95 Mbps para un Delay de 800 mS.

3.2.9 Simulando un dispositivo móvil

En esta sección se pretende observar el comportamiento de MPTCP dentro de un dispositivo móvil simulado que incorpore interfaces WLAN y 3G [4]. En este punto fue necesario incrementar el tamaño de los búferes de transmisión y recepción desde el kernel de Linux para que MPTCP pueda manejar más paquetes de red. Para esto se utilizaron los siguientes comandos estableciendo el límite a 12MB [12].

Los valores *net.core.rmem_max* y *net.core.wmem_max* corresponden al tamaño máximo de las ventanas TCP.

```
# echo 'net.core.wmem_max=12582912' >> /etc/sysctl.conf
# echo 'net.core.rmem_max=12582912' >> /etc/sysctl.conf
```

Los valores *net.ipv4.tcp_rmem* y *net.ipv4.tcp_wmem* corresponden a los valores mínimo, por defecto y máximo de las ventanas TCP que se irán ajustando según los algoritmos de control de congestión utilizado. Se desarrolló un script que permite automatizar el proceso (anexo 5):

```
# echo 'net.ipv4.tcp_rmem= 10240 87380 12582912' >> /etc/sysctl.conf
# echo 'net.ipv4.tcp_wmem= 10240 87380 12582912' >> /etc/sysctl.conf
```

Para simular las interfaces móviles de H1 se disponen de las siguientes características:

- **WLAN:** 8 Mbps, 20 mS de RTT (± 5 mS y Distribución Normal), 80 mS de Búfer.
- **3G:** 1 Mbps, 150 mS de RTT (± 10 mS y Distribución Normal), 2 s de Búfer.

Para tal objetivo se utilizó Netem y Traffic Controller con la disciplina tbf (anexo 6) ya que permite especificar un determinado tamaño de búfer para el enlace, situación que no es posible de realizar con la disciplina htb. En este escenario se utiliza la opción –w de Iperf para limitar el tamaño de la ventana (búfer de envío/recepción) a

un valor específico y observar el comportamiento del BW conforme se aumenta dicho valor.

Se ejecutó el experimento durante 40 segundos con cada valor de tamaño de ventana mostrado en la siguiente tabla. Luego los valores obtenidos se promediaron para obtener un único valor que posteriormente permitiera construir la gráfica.

- Prueba 10:

Configuraciones			Resultados		
Tamaño de Ventana (-w)	WLAN Eth1 (Mbps)	3G Eth2 (Mbps)	WLAN Eth1 (Mbps)	3G Eth2 (Mbps)	Total (Mbps)
50 KB	8	1	4,21	0,27	4,48
100 KB	8	1	5,2	0,29	5,49
150 KB	8	1	6,47	0,28	6,75
200 KB	8	1	6,79	0,28	7,07
300 KB	8	1	6,54	0,32	6,86
400 KB	8	1	7,04	0,33	7,37
500 KB	8	1	7,3	0,74	8,04
1 MB	8	1	7,7	0,98	8,68
2 MB	8	1	7,72	0,97	8,69
4 MB	8	1	7,97	0,98	8,95

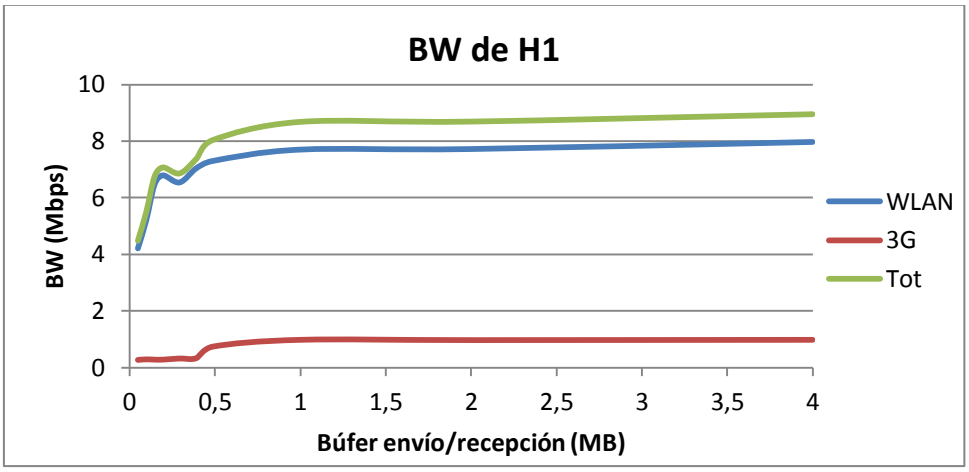


Figura 24. Resultados de la prueba 10

- **Análisis de resultados:**

Los resultados muestran que el BW esperado se obtiene solamente cuando se supera un determinado umbral en el tamaño de los búferes de envío/recepción (-w). Para el

caso de la interfaz WLAN el umbral se estima en aproximadamente 3,5 MB y para la interfaz 3G en aproximadamente 1 MB. Utilizando la fórmula (2) con un BW total de 9 Mbps y un RTT máximo de 160 mS se necesita un búfer de 360 KB, pero en la realidad se necesita mucho más debido a que la interfaz 3G al tener un búfer grande obliga a que el RTT máximo se incremente y trate de alcanzar los 2 segundos, alterando por completo el tamaño del búfer. La gráfica obtenida es similar a la de [4].

3.3 Escenario Avanzado de prueba 1

Una vez encontrado los parámetros óptimos de ajuste del escenario de red virtual para que las pruebas entreguen resultados coherentes, es momento de diseñar un nuevo escenario para probar las características de MPTCP en nuevas configuraciones de red (simple_mptcp4.xml). La figura 25 muestra el diagrama de red para analizar las siguientes cuestiones:

- Interacción de MPTCP con un dispositivo de capa 3 (router) que a la vez simularía una especie de Middlebox ya que utiliza una disciplina de colas para realizar el ajuste del BW deseado.
- Comprobar el comportamiento de MPTCP observado previamente en este nuevo escenario de prueba (distintos BW, combinación con tráfico TCP y robustez).

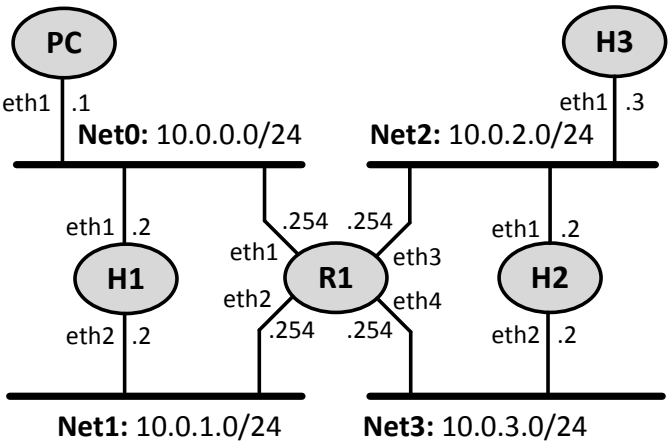


Figura 25. Diagrama de red del escenario avanzado de prueba 1

El punto de partida fue incluir en H1, H2 y R1 un rootfs con el kernel MPTCP de Linux. H3 tiene un rootfs con el kernel genérico de Linux. Se configuró un BW distinto para cada host y las pruebas mostraron que el control no estaba funcionando,

se obtenía siempre el BW configurado en H1, a pesar de que H2 tenga configurado un BW inferior. Se implementó los controles de BW en las interfaces de R1 pero el resultado fue el mismo. Para solventar esta situación se implementó un escenario adicional en VNX (figura 26) con un rootfs basado en un kernel genérico de Linux para verificar que el problema no sea causado por el kernel MPTCP así como se desarrolló otro tipo de control de BW con las herramientas de TC utilizando la disciplina htb pero con una configuración distinta al control de BW desarrollado inicialmente.

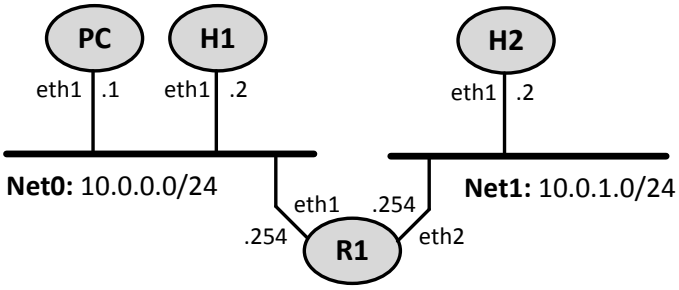


Figura 26. Diagrama de red del escenario alterno

El control de BW creado (anexo 7) se caracteriza por limitar el tráfico de salida en una determinada interfaz. Se implementó en las dos interfaces de R1 y las pruebas se realizaron durante 5 minutos. En una primera instancia desde H1 hacia H2 y luego en sentido contrario. Los resultados mostraron que el control limitaba efectivamente el BW.

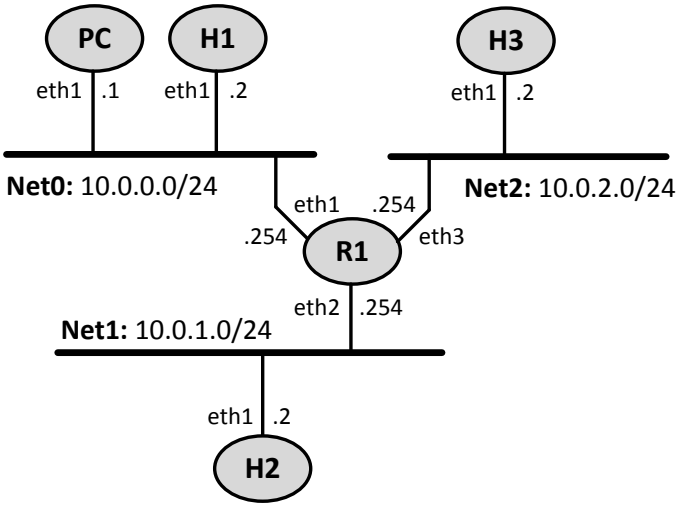


Figura 27. Diagrama de red de la ampliación del escenario alterno

Se amplió del diagrama de red de la figura 26 para incorporar otro host (figura 27) y para verificar que el control de BW desarrollado funcione para un número mayor de

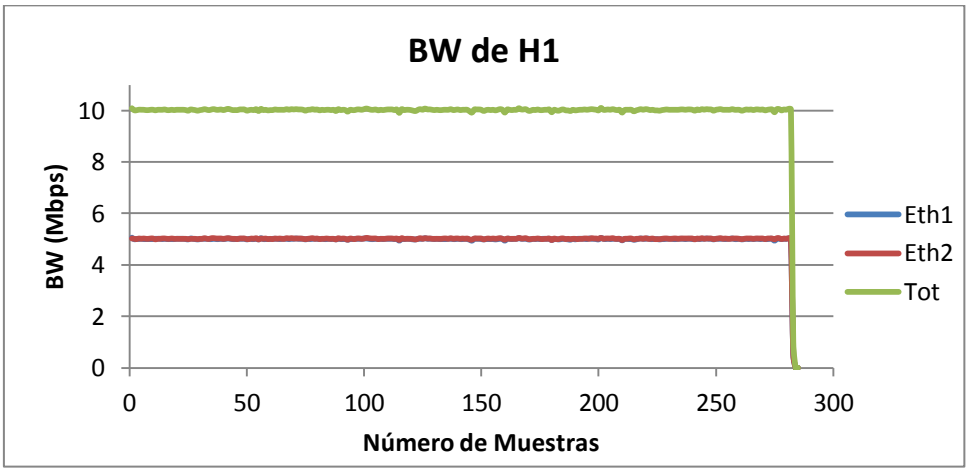
interfaces. Se decidió utilizar el control de BW desarrollado en el escenario básico de prueba que utiliza la disciplina htb en las interfaces de H1, H2 y H3. El control de BW desarrollado en este parte que utiliza la disciplina htb en las interfaces de R1. Los resultados obtenidos mostraron un control eficaz del BW en las distintas pruebas realizadas (H1-H2, H1-H3 y H2-H3).

3.3.1 Pruebas básicas realizadas

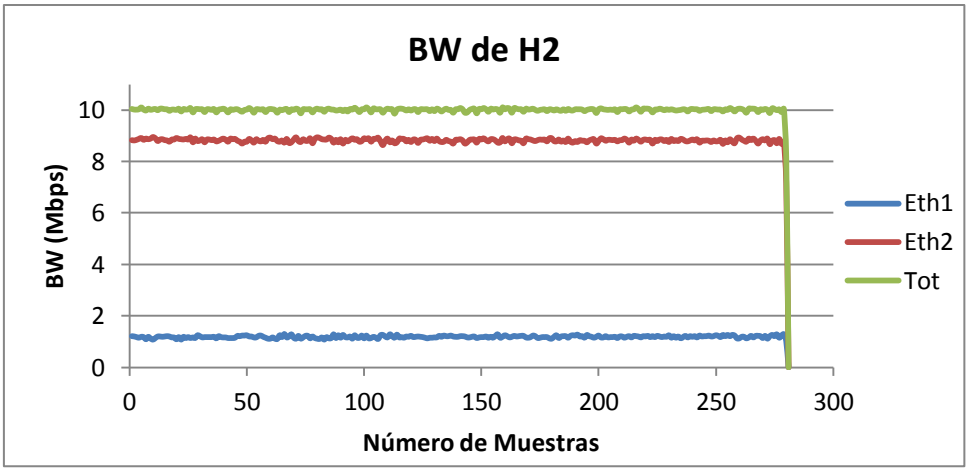
Con los resultados obtenidos se decidió configurar nuevamente el escenario de la figura 25, incluyendo un rootfs con un kernel genérico de Linux en R1 y H3, y el kernel MPTCP de Linux en H1 y H2. Se realizaron varias pruebas tratando de verificar el funcionamiento de MPTCP con distintas configuraciones de BW. A continuación se detallan las pruebas más relevantes ya que proporcionaron resultados interesantes para analizar.

- Prueba 11:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	5 Mbps	5 Mbps	4,99 Mbps	5,00 Mbps	9,99 Mbps	9,40 Mbps
H2	10 Mbps	10 Mbps	1,18 Mbps	8,81 Mbps	9,99 Mbps	9,38 Mbps



(a)



(b)

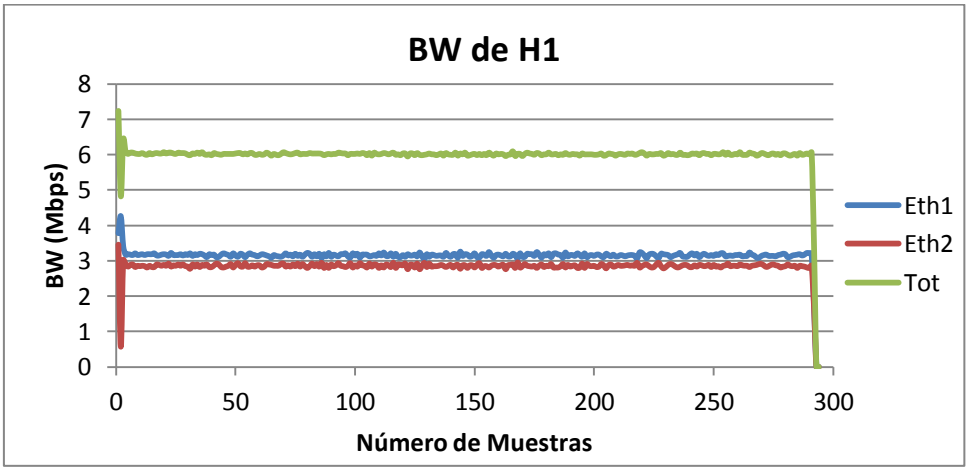
Figura 28. Resultados obtenidos de la prueba 11: (a) en H1 y (b) en H2

- **Análisis de resultados:**

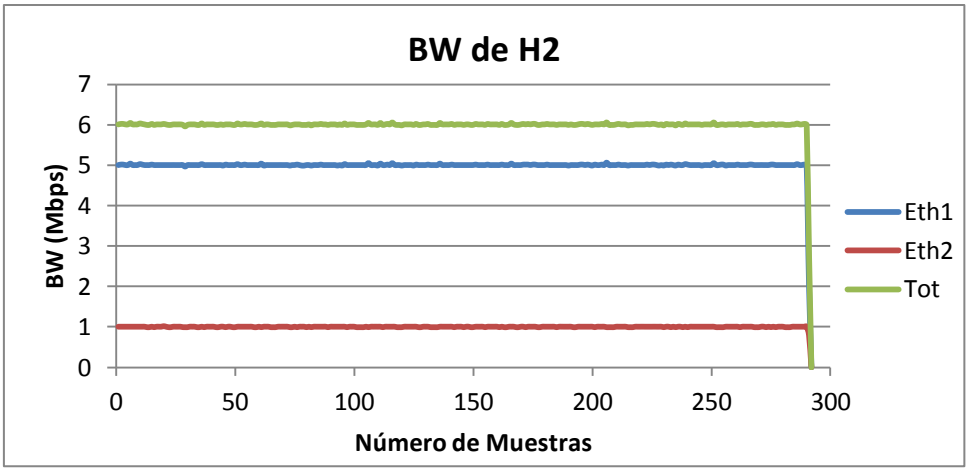
En H2 se debería obtener 5 Mbps por interfaz. Se obtiene un BW total de 10 Mbps pero no repartido equitativamente entre las dos interfaces de R1 (H2). Para encontrar una respuesta a esta situación se realizaron más pruebas pero con distintas configuraciones de BW.

- Prueba 12:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	10 Mbps	4 Mbps	3,16 Mbps	2,84 Mbps	6,00 Mbps	5,66 Mbps
H2	5 Mbps	1 Mbps	4,99 Mbps	1,00 Mbps	5,99 Mbps	5,63 Mbps



(a)



(b)

Figura 29. Resultados obtenidos de la prueba 12: (a) en H1 y (b) en H2

- **Análisis de resultados:**

En H2 se obtienen los valores configurados en cada interfaz y en H1 se trata de balancear los 4 Mbps que puede manejar H2, entre sus dos interfaces, que se corresponden con las interfaces de R1. Configurando H1-Eth1 con 2 Mbps, H1-Eth2 con 4 Mbps, H2-Eth1 con 3 Mbps y H2-Eth2 con 1 Mbps se obtiene un resultado similar. Se realizaron pruebas adicionales configurando una interfaz de H1 con un BW similar a una interfaz de H2 y se obtuvieron dos tipos de resultados: el primero donde se observó un comportamiento del BW similar en H1 y H2 y otro donde nuevamente se repartió el BW de forma no equitativa entre las interfaces de H2. Considerando los resultados obtenidos, se observa que existen dos posibles resultados en el lado del Host que tiene un BW total superior:

- Realizar una repartición de BW no por igual (figura 28b).
- Realizar una repartición de BW casi por igual (figura 29a).

Implementar interfaces con un BW igual en ambos hosts no condiciona a que se obtenga un comportamiento similar en ambos extremos. En este punto es importante recordar que MPTCP trata de crear una malla completa de conexiones a través de todas las direcciones IP existentes en los hosts (sección 3.2.2, apartado *e*), por lo tanto teóricamente se podrían crear los siguientes subflujos:

1. H1-Eth1 y H2-Eth1 = 10.0.0.2 y 10.0.2.2
2. H1-Eth1 y H2-Eth2 = 10.0.0.2 y 10.0.3.2
3. H1-Eth2 y H2-Eth1 = 10.0.1.2 y 10.0.2.2
4. H1-Eth2 y H2-Eth2 = 10.0.1.2 y 10.0.3.2

El número exacto de subflujos creados depende del componente Gestión de Rutas y la repartición de carga realizada entre todos ellos depende principalmente del Control de Congestión. La repartición no equitativa observada en las pruebas indica que existen subflujos con un mayor BW que otros y que posiblemente no se han establecido todos los subflujos que formarían la malla de conexiones.

El tercer objetivo del control de congestión garantiza que MPTCP colocará el mayor porcentaje de tráfico en los enlaces menos congestionados. Al establecer el primer subflujo el protocolo trata de obtener la mayor parte del BW disponible en ausencia de otro tráfico. Como el experimento (*prueba* 11) se realiza contra la interfaz Eth1 de H2, se pensaría que los flujos principales se establecerían entre ésta interfaz y las interfaces de H1 ya que puede manejar hasta 10 Mbps. Para los subflujos posteriores, el control de congestión simplemente trata de completar la cuota de BW que el host puede manejar, sin sobrepasar el límite establecido por los controles de BW. Esto explicaría la repartición de carga no equitativa de BW entre los distintos enlaces.

La gestión de subflujos y la repartición de carga dependen exclusivamente del análisis que MPTCP realiza sobre los enlaces para adaptarse a la situación actual, pudiendo obtenerse distintos resultados. Para comprobar esta explicación se procedió a repetir la prueba 11 con un tiempo de duración de 40 segundos para capturar los paquetes MPTCP sobre Net0 y Net1 con la ayuda de TCPDUMP ejecutándose sobre el PC (host con Ubuntu nativo). Se utilizó tcpdump porque Wireshark no funcionaba correctamente debido a una limitación en los recursos de procesamiento. De la captura se observan los siguientes subflujos creados:

- **NET0:**
  - **Subflujo 1 (10.0.0.2 y 10.0.2.2)**

```
IP 10.0.0.2.33851 > 10.0.2.2.5001: Flags [S], seq 868609192,
win 14600, options [mss 1460,sackOK,TS val 727424 ecr
0,nop,wscale 6,Unknown Option 30008166f23d1dd5d60188], length 0

IP 10.0.2.2.5001 > 10.0.0.2.33851: Flags [S.], seq 1501058123,
ack 868609193, win 14280, options [mss 1460,sackOK,TS val
726982 ecr 727424,nop,wscale 6,Unknown Option
300081cd89a96c8c3aeb78], length 0
```

  - **Subflujo 2 (10.0.1.2 y 10.0.2.2)**

```
IP 10.0.1.2.57138 > 10.0.2.2.5001: Flags [S], seq 648810967,
win 14600, options [mss 1460,sackOK,TS val 828476 ecr
0,nop,wscale 6,Unknown Option 30100103bce83992bf70a0], length 0

IP 10.0.2.2.5001 > 10.0.1.2.57138: Flags [S.], seq 1279565215,
ack 648810968, win 14280, options [mss 1460,sackOK,TS val
```

```
828032 ecr 828476,nop,wscale 6,Unknown Option
30100061848ddab2d2118e8ede9254], length 0
```

- **NET1:**

  - **Subflujo 3 (10.0.0.2 y 10.0.3.2)**

```
IP 10.0.0.2.53089 > 10.0.3.2.5001: Flags [S], seq 111903939,
win 14600, options [mss 1460,sackOK,TS val 727431 ecr
0,nop,wscale 6,Unknown Option 30100012970374f70954c6], length 0

IP 10.0.3.2.5001 > 10.0.0.2.53089: Flags [S.], seq 3517520559,
ack 111903940, win 14280, options [mss 1460,sackOK,TS val
726994 ecr 727431,nop,wscale 6,Unknown Option
3010010c739eb7ac0a3ecd220df06a], length 0
```

  - **Subflujo 4 (10.0.1.2 y 10.0.2.2)**

```
IP 10.0.1.2.39799 > 10.0.3.2.5001: Flags [S], seq 1548079234,
win 14600, options [mss 1460,sackOK,TS val 828476 ecr
0,nop,wscale 6,Unknown Option 30100103bce839b8411199], length 0

IP 10.0.3.2.5001 > 10.0.1.2.39799: Flags [S.], seq 3283986632,
ack 1548079235, win 14280, options [mss 1460,sackOK,TS val
828032 ecr 828476,nop,wscale 6,Unknown Option
3010013c0d126dd75f2139379f1d91], length 0
```

El resultado de la captura nos muestra que MPTCP por defecto ha creado una malla de conexiones con todas las combinaciones posibles de direcciones presentes en los hosts, dando como resultado 4 subflujos. A continuación se abrió el archivo generado por tcpdump con Wireshark para observar la proporción de paquetes que tiene cada subflujo y determinar que ruta es la que obtiene un mayor BW (Opción Conversaciones dentro del menú de estadísticas).

Address A	Port A	Address B	Port B	Packets A→B	bps A→B
10.0.0.2	33857	10.0.2.2	5001	10 317	4061267.17
10.0.0.2	52292	10.0.3.2	5001	2 323	916929.70

(a)

Address A	Port A	Address B	Port B	Packets A→B	bps A→B
10.0.1.2	55012	10.0.2.2	5001	10 355	4035934.02
10.0.1.2	56540	10.0.3.2	5001	2 382	927380.08

(b)

Figura 30. Conversaciones TCP: (a) Subflujos en Net0 y (b) Subflujos en Net1

A continuación se resume la repartición de carga en las interfaces de H2:

Interfaz	Conexión	BW	Subtotal	TOTAL
Eth1	10.0.0.2 y 10.0.2.2	4,06 Mbps	8,09 Mbps	9,92 Mbps
	10.0.1.2 y 10.0.2.2	4,03 Mbps		
Eth2	10.0.0.2 y 10.0.3.2	0,91 Mbps	1,83 Mbps	
	10.0.1.2 y 10.0.3.2	0,92 Mbps		

Estos valores son similares a los entregados por el Script de Monitoreo:

- H2-Eth1 = 8,21 Mbps
- H2-Eth2 = 1,78 Mbps

Se comprueba que el Control de congestión de MPTCP no realiza una repartición equitativa del BW a través de todos los subflujos creados. Al inicio trata de obtener el mayor BW posible del enlace y luego simplemente completa la cuota disponible utilizando otros subflujos.

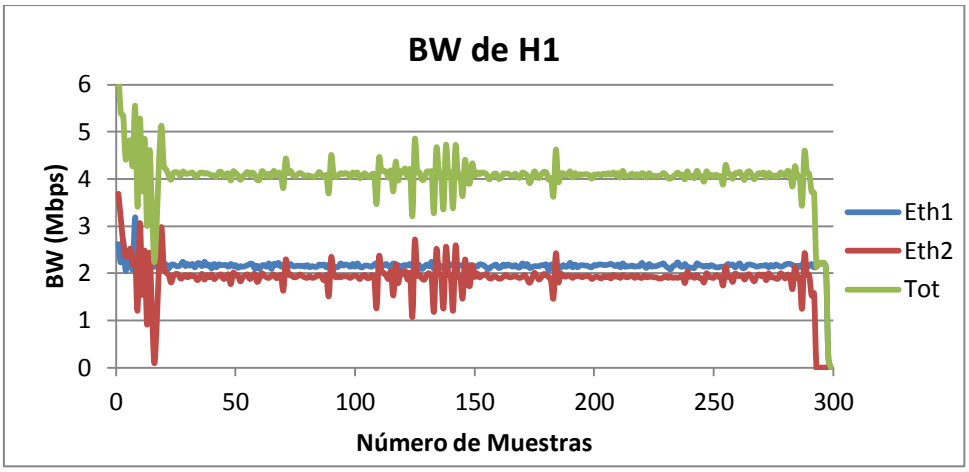
Se trató de limitar el número de subflujos a dos utilizando la opción `sysctl -w net.mptcp.mptcp_ndiffports=2` para observar si se podía balancear la carga sobre los enlaces pero el resultado mostró que solamente se estaba utilizando un enlace y el BW total fue de aproximadamente 5 Mbps. Se repitió el proceso para 3 y 4 subflujos y el resultado obtenido fue el mismo. En este parte se podría decir que la opción que implementa el kernel MPTCP de Linux para especificar el número de subflujos que el protocolo va a crear, no funciona.

3.3.2 Incluyendo tráfico TCP en el escenario

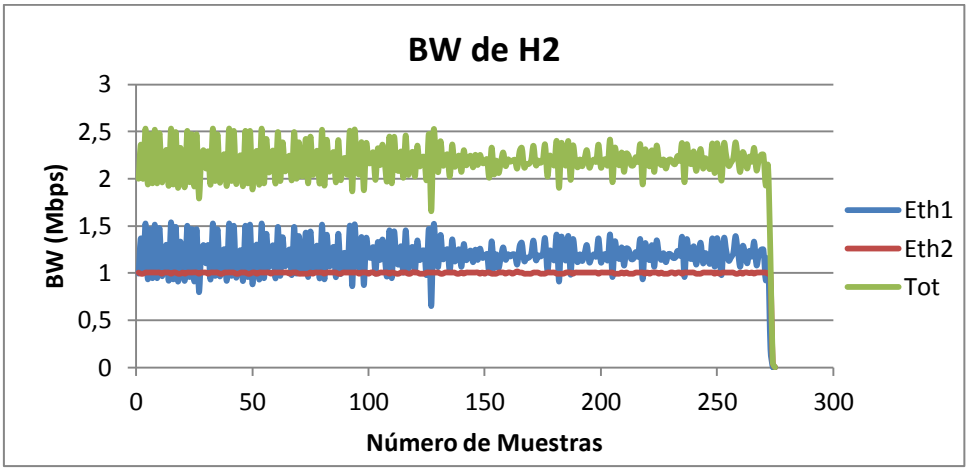
Al igual que en el escenario previo, se incluye tráfico TCP para observar el comportamiento de MPTCP. Al inicio se configuraron todas las interfaces con 5 Mbps y los resultados obtenidos fueron similares a los de la prueba 7. Para un mejor análisis se decidió configurar interfaces con distinto BW.

- Prueba 13:

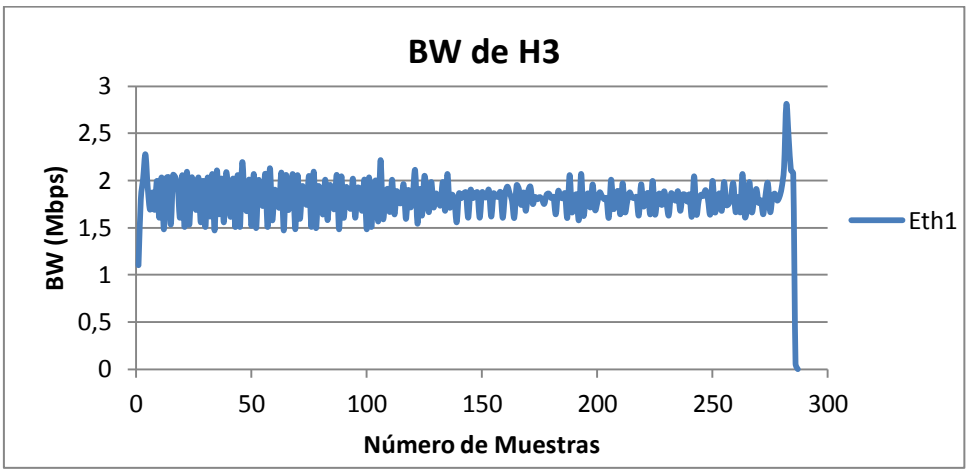
Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	2 Mbps	4 Mbps	2,16 Mbps	1,93 Mbps	4,09 Mbps	2,06 Mbps en H1-H2 1,73 Mbps en H1-H3
H2	3 Mbps	1 Mbps	1,18 Mbps	0,99 Mbps	2,17 Mbps	2,04 Mbps en H1-H2
H3	3 Mbps	---	1,81 Mbps	---	1,81 Mbps	1,71 Mbps en H1-H3



(a)



(b)



(c)

Figura 31. Resultados obtenidos de la prueba 13: (a) en H1, (b) en H2 y (c) en H3

- **Análisis de resultados:**

Los resultados obtenidos son similares a los de la prueba 6 y 7, por lo tanto se vuelve a verificar los objetivos de MPTCP al ser equitativo con TCP y no ocupar el 100% del BW disponible en la interfaz Eth1 de R1 (H2 con 1,18 Mbps y H3 con 1,81 Mbps). También se obtiene un desempeño al menos igual al de una conexión TCP sobre cualquiera de los trayectos que MPTCP utiliza (H2 con un total de 2,17 Mbps ligeramente superior a los 1,81 Mbps de H3). La irregularidad presente en las gráficas es similar a la observada en pruebas anteriores y se debe al ajuste del tamaño de ventana que realiza MPTCP.

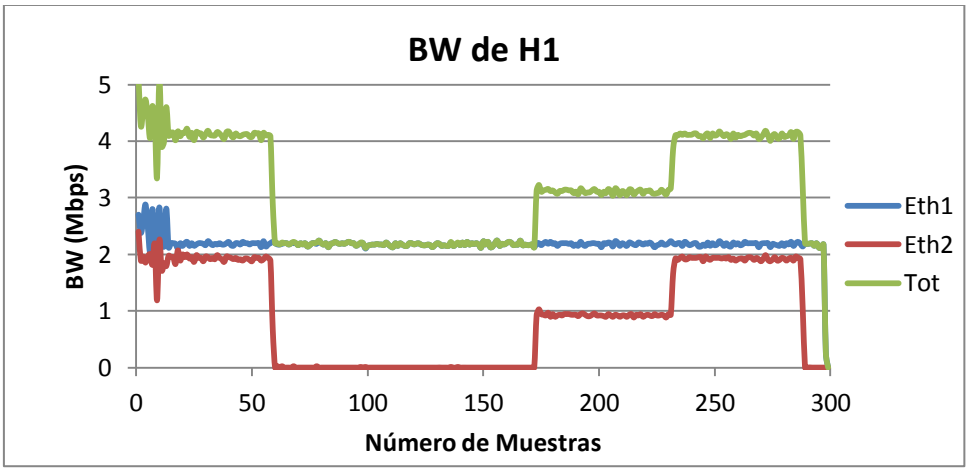
3.3.3 Probando la robustez en el escenario avanzado de prueba 1

Al igual que en la prueba 8, se pretende observar la robustez de MPTCP en este escenario de prueba. Se utilizó el siguiente cronograma:

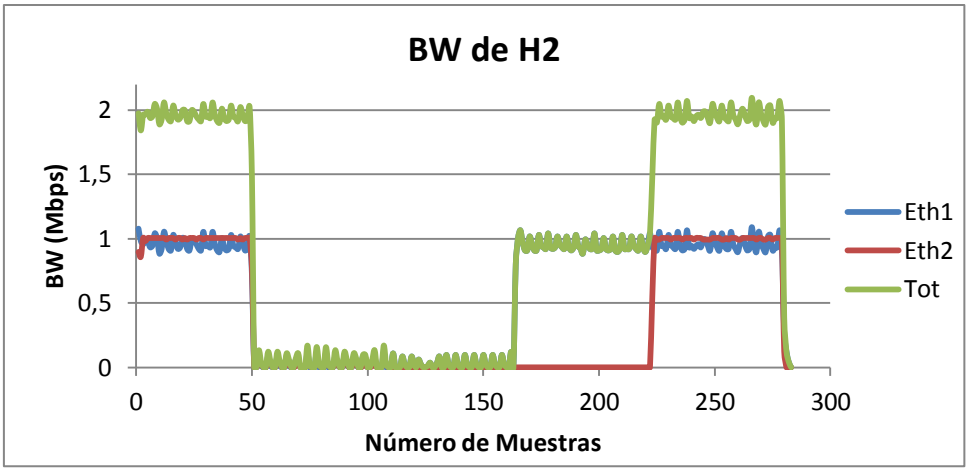
- En 1'00" se desactiva Eth2 de R1, en 2'00" se desactiva Eth4 de R1.
- En 3'00" se activa Eth2 de R1, en 4'00" se activa Eth4 de R1.

- Prueba 14:

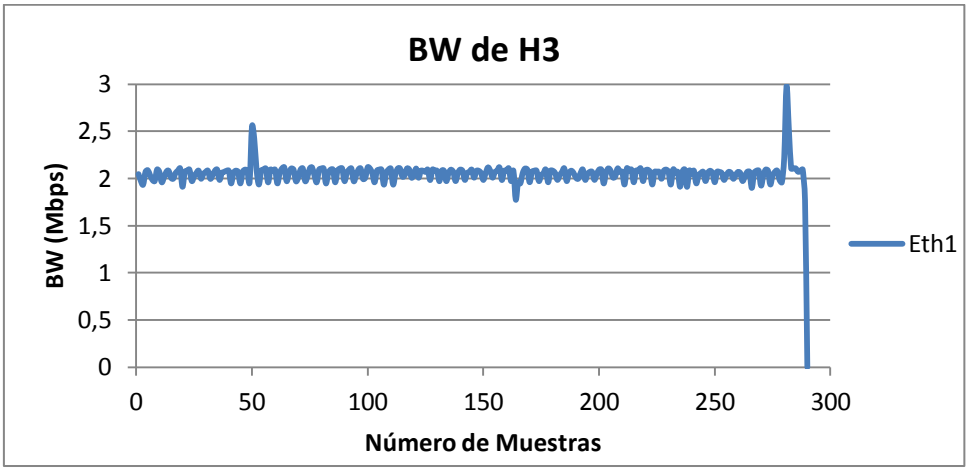
Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	2 Mbps	4 Mbps	2,18 Mbps	0,92 Mbps	3,10 Mbps	0,94 Mbps en H1-H2 1,98 Mbps en H1-H3
H2	3 Mbps	1 Mbps	0,57 Mbps	0,38 Mbps	0,95 Mbps	0,93 Mbps en H1-H2
H3	3 Mbps	---	2,05 Mbps	---	2,05 Mbps	1,93 Mbps en H1-H3



(a)



(b)



(c)

Figura 32. Resultados obtenidos de la prueba 14: (a) en H1, (b) en H2 y (c) en H3

- **Análisis de resultados:**

Los resultados nos indican un comportamiento coherente de MPTCP hasta los 60 segundos, donde el BW total de 4 Mbps se distribuye por igual entre H2 y H3.

A los 60 segundos cuando se desactiva la interfaz Eth2 de R1, se corta por completo el tráfico a H2 a pesar de que su interfaz de 1 Mbps está activa y H1 a través de su interfaz Eth1 puede manejar un total de 2 Mbps de los cuales H3 ocupa casi el 100% del BW. Esta situación muestra que no se aplica la equidad entre la conexión MPTCP y TCP. A los 120 segundos, cuando se desactiva la interfaz Eth4 de R1 el protocolo nuevamente debería realizar una distribución del BW de 2 Mbps entre H2 y H3, pero tal situación no ocurre, H3 mantiene su consumo de 2 Mbps y H2 sigue con un consumo de cero.

A los 180 segundos al activar Eth2 de R1 el protocolo funciona correctamente y se obtienen los resultados esperados, en donde los 3 Mbps se comparten entre H2 (1 Mbps) y H3 (2 Mbps). En la etapa final al activar Eth4 de R1 el consumo de BW de cada Host vuelve a ser similar a la etapa inicial donde se comparte por igual los 4 Mbps.

Los resultados obtenidos en esta prueba son similares a los obtenidos en la prueba 8. Al desactivar una interfaz se debería repartir la carga entre las conexiones de MPTCP y TCP pero en la realidad solamente la conexión TCP permanece con tráfico mientras que la conexión MPTCP tiene un BW de 0,036 Mbps aproximadamente. La variante observada indica que al activar nuevamente la interfaz Eth2 de R1, MPTCP entrega a H2 un BW de 1 Mbps de los 1,5 Mbps que en teoría debería entregar. Mejora la situación pero igual no se aplica la *equidad* entre las conexiones.

Esta situación observada en los dos experimentos nos permite plantear la siguiente hipótesis: en vista de la presencia de tráfico TCP, MPTCP coloca el mayor porcentaje de tráfico en los subflujos que no comparten el tráfico con TCP. Para el caso del escenario básico de prueba sería el subflujo principal H1-Eth2 y H2-Eth2 (5 Mbps); para el caso del escenario avanzado de prueba 1 serían los subflujos principales H1-Eth2 y H2-Eth1 (1 Mbps), H1-Eth2 y H2-Eth2 (1 Mbps). Cuando se desactivan los enlaces que forman parte de los subflujos principales el BW total del host es igual al valor de BW que tienen los subflujos secundarios, el mismo que es prácticamente despreciable. Cuando se activan nuevamente los enlaces que forman parte de los subflujos principales la situación vuelve a ser normal.

Parecería ser que MPTCP establece una configuración al inicio de la conexión en base a la situación actual de la red, tratando de cumplir con los objetivos del control de congestión. Tal configuración es estática ya que al momento de desactivar un enlace el control de congestión no realiza un balanceo de carga para adaptarse a las nuevas condiciones de la red. Cuando la conectividad se restablece, MPTCP recupera su funcionamiento normal.

3.4 Escenario avanzado de prueba 2

Como parte final se pretende utilizar el escenario de red de la figura 33 para analizar las siguientes cuestiones:

- H2 al tener solamente una interfaz de red podría suponer la creación de un cuello de botella si dicha interfaz tuviese un BW menor a la sumatoria del BW de cada interfaz presente en H1.
- Comprobar el comportamiento de MPTCP observado previamente en este nuevo escenario de prueba (distintos BW, combinación con tráfico TCP y robustez).

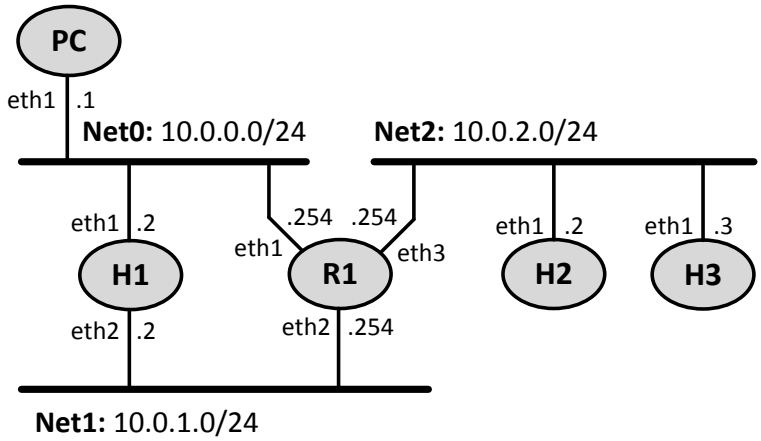


Figura 33. Diagrama de red del escenario avanzado de prueba 1

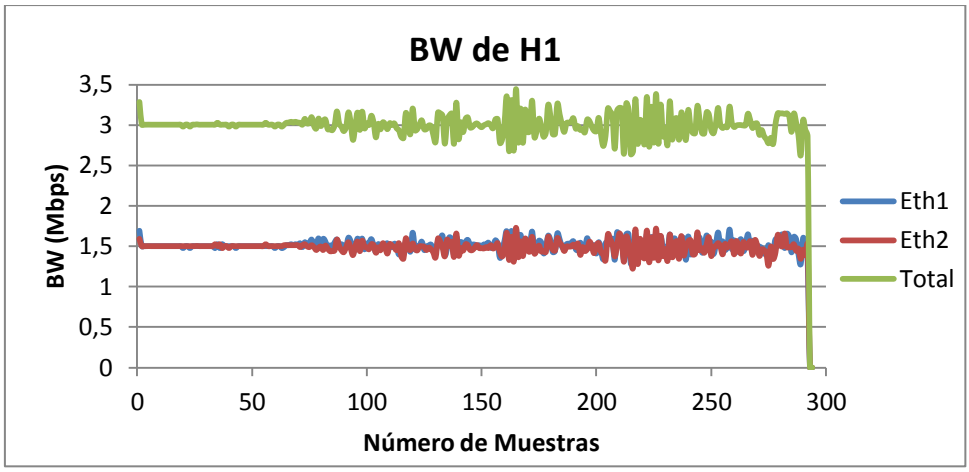
El rootfs utilizado en H1 y H2 es el kernel MPTCP de Linux y el rootfs utilizado en H3 y R1 es el kernel genérico de Linux. El escenario se denomina simple_mptcp3.xml.

3.4.1 Verificando los cuellos de botella

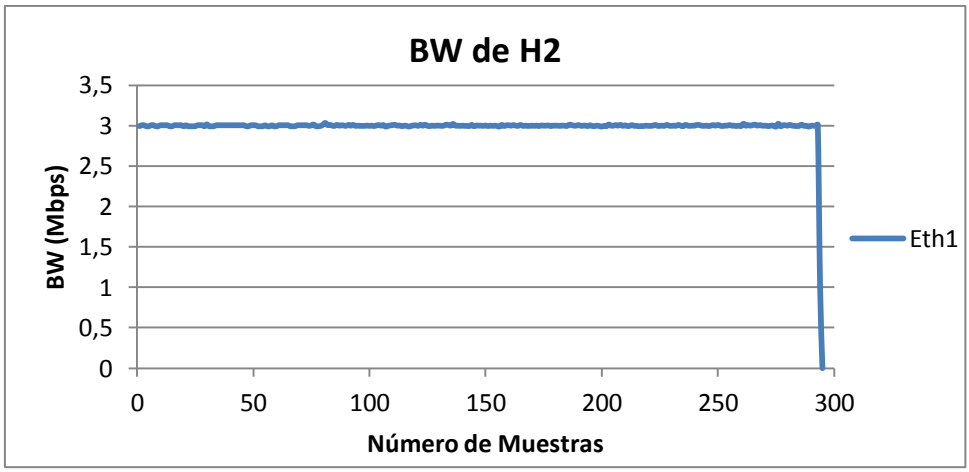
En una primera instancia se configuró H1 con 5 Mbps por interfaz y H2 con 10 Mbps, los resultados fueron satisfactorios, mostrando que las interfaces de H1 utilizaban toda su capacidad y que H2 pudo manejar todo el BW disponible. Posteriormente se procedió a realizar la siguiente prueba donde se simula un cuello de botella en H2 ya que el BW disponible en su interfaz no está en la capacidad de manejar todo el BW disponible en H1.

- Prueba 15:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	5 Mbps	5 Mbps	1,51 Mbps	1,48 Mbps	2,99 Mbps	2,82 Mbps
H2	3 Mbps	---	2,99 Mbps	---	2,99 Mbps	2,82 Mbps



(a)



(b)

Figura 34. Resultados obtenidos de la prueba 15: (a) en H1 y (b) en H2

- **Análisis de resultados:**

Se observa una pequeña inestabilidad en el BW de H1 desde los 65 segundos aproximadamente. Esto posiblemente se deba al ajuste del tamaño de ventana que se realiza ya que las interfaces no están ocupando todo el BW disponible. Por otro lado se observa que MPTCP realiza un balanceo de carga entre las dos interfaces de H1 para ocupar el total de BW que puede manejar H2. El consumo de BW en H2 no muestra inestabilidad alguna ya que se utiliza todo el recurso disponible.

3.4.2 Experimentando con distintos BWs

Considerando la prueba anterior se configuraron BWs distintos en las interfaces de H1, manteniendo el cuello de botella en H2.

- Prueba 16:

Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	4 Mbps	1 Mbps	2,02 Mbps	0,98 Mbps	3,00 Mbps	2,83 Mbps
H2	3 Mbps	---	2,99 Mbps	---	2,99 Mbps	2,82 Mbps

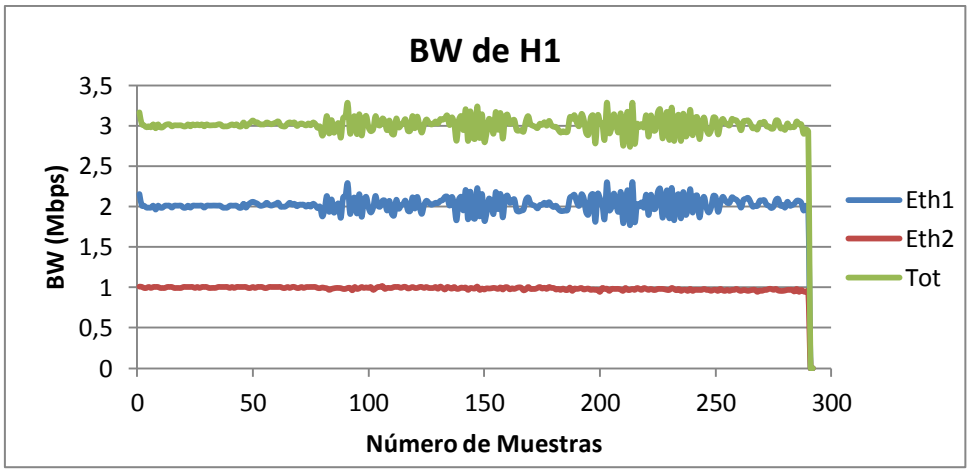


Figura 35. Resultados obtenidos de la prueba 16

- **Análisis de resultados:**

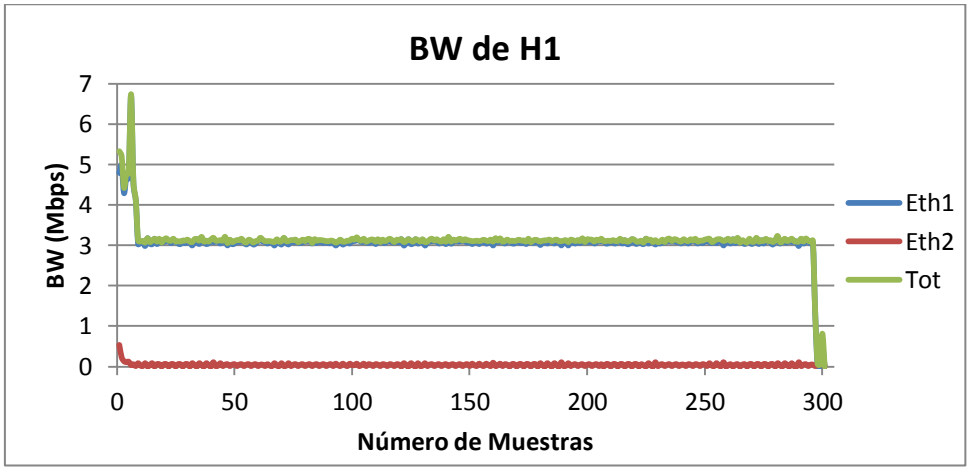
Similar a los resultados de la prueba anterior, se trata de adaptar el consumo de BW en H1 a lo que H2 puede manejar. Se obtiene el máximo BW de Eth2 (1 Mbps) y la mitad del BW de Eth1 (2 Mbps). De igual forma se presenta una inestabilidad en Eth1 de H1 debido al ajuste realizado por el exceso de recursos existentes contrario a lo que se observa en Eth2 de H1 ya que se ocupa el máximo posible. La gráfica obtenida en H2 es similar a la figura 34b. A esta prueba se le cambió el BW de 1 Mbps por 2 Mbps en la interfaz Eth2 de H1 para asegurar que la inestabilidad presente en la gráfica se debe al exceso de recursos presente. Los resultados mostraron que efectivamente se obtenía un comportamiento similar al del BW en Eth1, comprobando la explicación previamente expuesta.

3.4.3 Incluyendo tráfico TCP al experimento

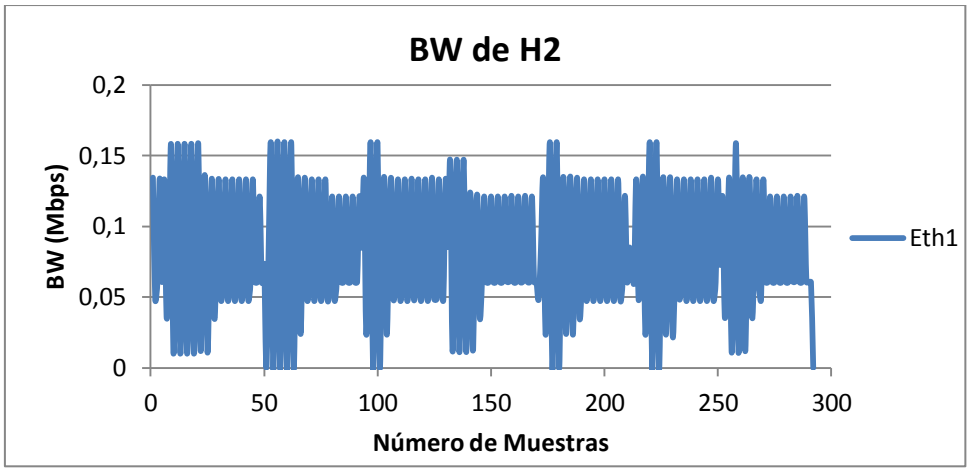
Al igual que en escenarios de prueba anteriores, se pretende incluir tráfico TCP para observar el comportamiento de MPTCP frente a conexiones TCP. Se utilizaron las configuraciones de la prueba anterior y se incluye H3 con un BW de 3 Mbps.

- Prueba 17:

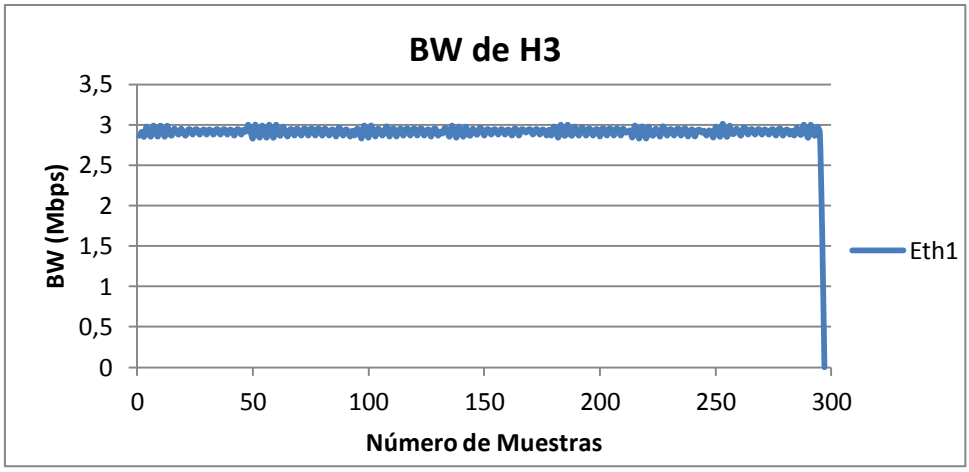
Configuraciones			Resultados			
Host	Eth1	Eth2	Eth1	Eth2	Total	Iperf
H1	4 Mbps	1 Mbps	3,09 Mbps	0,04 Mbps	3,13 Mbps	0,085 Mbps en H1-H2 2,80 Mbps en H1-H3
H2	3 Mbps	---	0,08 Mbps	---	0,08 Mbps	0,085 Mbps en H1-H2
H3	3 Mbps	---	2,91 Mbps	---	2,91 Mbps	2,74 Mbps en H1-H3



(a)



(b)



(c)

Figura 36. Resultados obtenidos de la prueba 17: (a) en H1, (b) en H2 y (c) en H3

- **Análisis de resultados:**

Se obtienen resultados similares a los obtenidos en escenarios previos. La conexión TCP ocupa casi el 100% del BW que tiene configurado en su interfaz pero la conexión MPTCP apenas alcanza un consumo de 0,085 Mbps. Este resultado es desastroso y muestra que el Control de Congestión no está funcionando de forma correcta. Según la teoría se debería establecer dos subflujos entre H1 y H2 (10.0.0.2 – 10.0.2.2 y 10.0.1.2 – 10.0.2.2) y con las experiencias previas se debería obtener un subflujo principal (mayor porcentaje de BW) y un secundario (menor porcentaje de BW). La repartición de carga debería otorgar 1,5 Mbps a cada conexión (TCP y MPTCP), además el control de congestión de MPTCP debería repartir los 1,5 Mbps a través de los dos subflujos creados, dando el mayor porcentaje de BW al subflujo que no comparte tráfico con TCP (subflujo principal). Como H2 solamente tiene una interfaz, MPTCP le otorga casi el 100% del BW disponible en Eth3 de R1 a H3 (2,91 Mbps a TCP). Este comportamiento ha sido común en todos los escenarios de prueba, pero ahora la situación es más crítica ya que H2 cuenta solamente con una interfaz (Eth1).

Posteriormente se realizó una prueba de robustez con la misma configuración de la prueba 17 utilizando el siguiente cronograma:

- En 1’30” se desactiva Eth2 de R1.
- En 3’30” se activa Eth2 de R1.

Los resultados obtenidos son similares a los de la prueba 17. Finalmente se observa que H1 al contar con dos interfaces y MPTCP no representa una mayor ventaja cuando se inyecta tráfico TCP, más bien empeora la situación.

4 Conclusiones

La experimentación con el kernel MPTCP de Linux y la configuración de escenarios virtuales dentro de VNX permiten obtener las siguientes conclusiones generales del trabajo de investigación:

4.1 Con respecto a MPTCP

En todos los escenarios se comprueba que MPTCP hace uso de todas las interfaces de red presentes en el host, creando una conexión única a nivel de transporte compuesta por varias conexiones (subflujos) a nivel de red. En condiciones ideales de funcionamiento, es decir, sin RTT excesivos, host con múltiples interfaces y BWs iguales o distintos, MPTCP funciona de acuerdo a la teoría, logrando optimizar la conexión al obtener un BW mayor. De igual manera el protocolo es robusto frente a la pérdida de un enlace, mantiene la conectividad y elimina la necesidad de ejecutar mecanismos adicionales para restablecer la conectividad.

Uno de los factores determinantes para el funcionamiento óptimo del protocolo es el RTT presente en la comunicación extremo a extremo. Los desarrolladores del protocolo establecen una fórmula que muestra una relación directamente proporcional entre el tamaño del búfer de recepción, el BW y el RTT. A más RTT y BW, más consumo de memoria por parte de MPTCP. Si el sistema operativo está en la capacidad de otorgar la memoria que MPTCP necesita se obtendrá un buen desempeño, caso contrario se obtendrá una degradación del BW conforme el RTT aumenta.

El control de congestión no siempre realiza una repartición de BW por igual entre todos los subflujos presentes, depende de la capacidad para determinar qué enlaces están experimentando una menor congestión y colocar el tráfico allí, es por eso que existen subflujos principales (mayor porcentaje de BW) y subflujos secundarios (menor porcentaje de BW). Algo muy importante a reconocer es que si bien no se realiza una repartición equitativa del BW si se logra utilizar toda la cuota de BW que un extremo de la conexión puede manejar. Hasta aquí se podría decir que MPTCP cumple todo lo que plantea en sus RFCs.

De la inclusión de tráfico TCP en los experimentos se reconocen dos aspectos importantes: *primero*, el desempeño de MPTCP depende principalmente del número de interfaces presentes en el host. Si el host tiene por lo menos dos interfaces el

desempeño va a ser igual o superior al de TCP ya que una interfaz comparte el BW con la conexión TCP y la otra consume el BW total con MPTCP. Si el host MPTCP dispone de una sola interfaz el resultado va a ser catastrófico ya que la conexión TCP va a estar consumiendo casi el 100% del BW disponible, situación que es una violación directa de los principios de diseño de MPTCP y TCP. Como *segundo* aspecto se observa que la pérdida de un enlace no obliga a que el control de congestión realice un balanceo de carga entre las conexiones TCP y MPTCP, existe una preferencia para la conexión TCP que se observó en todos los escenarios de prueba. Esta situación indicaría que el control de congestión no es adaptativo, más bien establece una configuración estática al inicio y es la que mantiene durante el intercambio de tráfico.

En todos los escenarios de prueba, la robustez es una característica que no fue afectada por la inclusión de tráfico TCP.

De forma general MPTCP, dentro del entorno de prueba virtual, utilizó de forma correcta los conceptos de Multihoming y Agrupación de recursos para cumplir con sus objetivos generales de aumentar la robustez de las conexiones y obtener mayores BWs en los dispositivos finales. Los objetivos específicos del Control de congestión se cumplieron en un gran porcentaje en los dos primeros escenarios de prueba pero para el caso del escenario final no.

Finalmente se podría decir que MPTCP debería solucionar los conflictos ocasionados con la interacción de tráfico TCP para poderlo utilizar dentro de un entorno de producción.

4.2 Con respecto al Entorno de pruebas

Para cumplir con nuestro objetivo de verificar el funcionamiento de MPTCP, VNX fue una buena opción que se ajustó a nuestras necesidades, permitiendo configurar topologías interesantes y observar el desempeño del protocolo.

Uno de los problemas más críticos dentro de la investigación fue desarrollar las herramientas necesarias que permitan configurar un escenario estable que garantice que los resultados obtenidos son coherentes y representan un comportamiento similar al de una situación real. Considerando esta situación, los controladores de BW son de suma importancia ya que permiten especificar una referencia y en base a ella observar si los resultados obtenidos son válidos. Fue importante entender el funcionamiento de TC y sus diferentes disciplinas de colas existentes para buscar un script que limite

efectivamente el BW a un valor dado. Se desarrollaron distintos controladores que se adaptaron a las características de los escenarios de prueba creados.

Las herramientas complementarias desarrolladas como el monitor de interfaces, la configuración del encaminamiento, etc., permitieron crear un entorno de pruebas a medida, adaptándose a los requerimientos de la investigación.

Desde que se comenzó a realizar la investigación se tuvieron inconvenientes con las versiones del kernel MPTCP de Linux. Con una periodicidad de un mes se estaban lanzando nuevas versiones del kernel, retrasando la realización de las pruebas. Finalmente, en el mes de marzo se lanzó la primera versión estable (v0.86). Actualmente se cuenta con la misma versión pero la #4 actualizada al 5 de mayo. Dentro de la página web de INL se menciona que próximamente se liberará la versión 0.87.

4.3 Trabajos futuros

La experiencia obtenida a través de la investigación realizada nos permite pensar en ampliar la investigación a un entorno de red real donde se pueda evaluar el comportamiento de MPTCP bajo la influencia de:

- Pérdidas en los enlaces.
- Tres interfaces de red o más.
- Tráfico específico como web, multimedia, etc.
- Middleboxes.
- Enlaces con gran BW (10 Gbps o superior).

De los resultados que se obtengan en esta nueva experimentación se podría estar en una mejor situación para recomendar su uso dentro de un entorno de producción como Internet.

4.4 Recomendaciones

Para trabajar dentro de un entorno Linux es importante tener los conocimientos básicos respectivos que permitan desarrollar paulatinamente la investigación.

Entender el funcionamiento de TCP y principalmente de los mecanismos de control de congestión son fundamentales para entender lo que MPTCP debería hacer para gestionar eficazmente los recursos disponibles en la red.

Una lectura comprensiva de los diferentes artículos y RFCs nos permitirán entender el funcionamiento de MPTCP y posteriormente diseñar escenarios de red para verificar sus características.

Referencias

[1] C. Raiciu, M. Handley, D. Wischik. "Architectural Guidelines for Multipath TCP Development". IETF RFC (6182), March 2011.

[2] A. Ford, C. Raiciu, M. Handley, S. Barré, J. Iyengar. "Coupled Congestion Control for Multipath Transport Protocols". IETF RFC (6356), October 2011.

[3] A. Ford, C. Raiciu, M. Handley, O. Bonaventure. "TCP Extensions for Multipath Operation with Multiple Addresses". IETF RFC (6824), January 2013.

[4] C. Raiciu, C. Paasch, S. Barré, A. Ford, M. Honda, F. Duchene, O. Bonaventure and M. Handley. "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP". *USENIX Symposium of Networked Systems Design and Implementation (NSDI'12), San Jose (CA), 2012*.

[5] S. Barré, C. Paasch and O. Bonaventure. "MultiPath TCP: From Theory to Practice", *NETWORKING 2011. Proceedings of the 10th International IFIP TC 6 Networking Conference*, pp. 444-457, 2011.

[6] http://web.dit.upm.es/vnxwiki/index.php/Main_Page

[7] Virtual networks over Linux (Poster), *TERENA NETWORKING CONFERENCE 2012*. https://tnc2012.terena.org/core/poster/24

[8] http://mptcp.info.ucl.ac.be/

[9] Departamento de Tecnología Electrónica, Universidad de Sevilla, "Laboratorio 5 Qos y Control de Tráfico. Documento PDF (rev. 1.171)

[10] http://www.iplocation.net/tools/traffic-control.php

[11] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[12] http://www.cyberciti.biz/faq/linux-tcp-tuning/

Anexos

1. Encaminamiento de MPTCP (routing-h1.sh)

```
#!/bin/bash
# Esto crea dos tablas de enrutamiento diferentes, las cuales
# seran utilizadas en base a la direccion de origen.

ip rule add from 10.0.0.2 table 1
ip rule add from 10.0.1.2 table 2

# Configuracion de las dos tablas de enrutamiento diferentes

ip route add 10.0.0.0/24 dev eth1 scope link table 1
ip route add default via 10.0.0.254 dev eth1 table 1

ip route add 10.0.1.0/24 dev eth2 scope link table 2
ip route add default via 10.0.1.254 dev eth2 table 2

# Ruta por defecto para el proceso de seleccion del trafico de
# internet normal

#ip route add default scope global nexthop via 10.0.0.254 dev eth1
```

2. Monitoreo de Ancho de Banda (monitor2int.sh)

```
#!/bin/bash

speed=1
muestras=0

>/root/resultados.txt

while [ $muestras -le 180 ]
do
      #WLAN1R1=`cat /sys/class/net/eth1/statistics/rx_bytes`
      WLAN1T1=`cat /sys/class/net/eth1/statistics/tx_bytes`

      #WLAN2R1=`cat /sys/class/net/eth2/statistics/rx_bytes`
      WLAN2T1=`cat /sys/class/net/eth2/statistics/tx_bytes`

      sleep $speed

      #WLAN1R2=`cat /sys/class/net/eth1/statistics/rx_bytes`
      WLAN1T2=`cat /sys/class/net/eth1/statistics/tx_bytes`

      #WLAN2R2=`cat /sys/class/net/eth2/statistics/rx_bytes`
      WLAN2T2=`cat /sys/class/net/eth2/statistics/tx_bytes`

      WLAN1TBPS=`expr $WLAN1T2 - $WLAN1T1`
      #WLAN1RBPS=`expr $WLAN1R2 - $WLAN1R1`
      WLAN1TMbps=`echo "$WLAN1TBPS / 125000" | bc -l`
      #WLAN1RMbps=`echo "$WLAN1RBPS / 125000" | bc -l`
      #WLAN1ToMbps=`echo "$WLAN1TMbps + $WLAN1RMbps" | bc`

      WLAN2TBPS=`expr $WLAN2T2 - $WLAN2T1`
```

```
      #WLAN2RBPS=`expr $WLAN2R2 - $WLAN2R1`
      WLAN2TMbps=`echo "$WLAN2TBPS / 125000" | bc -l`
      #WLAN2RMbps=`echo "$WLAN2RBPS / 125000" | bc -l`
      #WLAN2ToMbps=`echo "$WLAN2TMbps + $WLAN2RMbps" | bc`

      #WLANToMbps=`echo "$WLAN1ToMbps + $WLAN2ToMbps" | bc`
      WLANTMbps=`echo "$WLAN1TMbps + $WLAN2TMbps" | bc`

      let muestras=muestras+1
      echo "$muestras" "$WLAN1TMbps" "$WLAN2TMbps" "$WLANTMbps" >>
resultados.txt

done
```

3. Experimento (exp-1.sh)

```
#!/bin/bash
#date
echo "PRUEBA DE ANCHO DE BANDA AL SERVIDOR MPTCP"
./monitor2int.sh &
ssh root@10.0.2.2 ./monitor2int.sh &
#ssh root@10.0.2.3 ./monitor1int.sh &
iperf -c 10.0.2.2 -t 300
#iperf -c 10.0.2.3 -t 300
#iperf -c 10.0.1.2 -t 30 -w 500KB & ./monitor1int.sh
echo "FIN"
```

4. Control de BW (control-bw1.sh)

```
#!/bin/bash
#
#  tc uses the following units when passed as a parameter.
#  kbps: Kilobytes per second
#  mbps: Megabytes per second
#  kbit: Kilobits per second
#  mbit: Megabits per second
#  bps: Bytes per second
#       Amounts of data can be specified in:
#       kb or k: Kilobytes
#       mb or m: Megabytes
#       mbit: Megabits
#       kbit: Kilobits
#  To get the byte figure from bits, divide the number by 8 bit
#
TC=/sbin/tc
IF=eth1            # Interface
DNLD=5mbit          # DOWNLOAD Limit
UPLD=5mbit          # UPLOAD Limit
IP=10.0.0.2        # Host IP
U32="$TC filter add dev $IF protocol ip parent 1:0 prio 1 u32"

start() {

    $TC qdisc add dev $IF root handle 1: htb default 30
    $TC class add dev $IF parent 1: classid 1:1 htb rate $DNLD
    $TC class add dev $IF parent 1: classid 1:2 htb rate $UPLD
```

```
    $U32 match ip dst $IP/32 flowid 1:1
    $U32 match ip src $IP/32 flowid 1:2

}

stop() {

    $TC qdisc del dev $IF root

}

restart() {

    stop
    sleep 1
    start

}

show() {

    $TC -s qdisc ls dev $IF

}

case "$1" in

  start)

    echo -n "Starting bandwidth shaping: "
    start
    echo "done"
    ;;

  stop)

    echo -n "Stopping bandwidth shaping: "
    stop
    echo "done"
    ;;

  restart)

    echo -n "Restarting bandwidth shaping: "
    restart
    echo "done"
    ;;

  show)

    echo "Bandwidth shaping status for $IF:\n"
    show
    echo ""
    ;;

  *)

    pwd=$(pwd)
    echo "Usage: $(/usr/bin/dirname $pwd)/tc.bash
{start|stop|restart|show}"
    ;;
```

```
esac

exit 0
```

5. Ampliación del búfer de envío y recepción (buffer.sh)

```
#!/bin/bash

#Configurar el tamano maximo posible del bufer de recepcion y
transmision
#El valor maximo es de 12 MB

echo 'net.core.wmem_max=12582912' >> /etc/sysctl.conf
echo 'net.core.rmem_max=12582912' >> /etc/sysctl.conf

#Tambien se necesita configurar el tamano minimo, inicial y maximo en
bytes

echo 'net.ipv4.tcp_rmem= 10240 87380 12582912' >> /etc/sysctl.conf
echo 'net.ipv4.tcp_wmem= 10240 87380 12582912' >> /etc/sysctl.conf

#No guardar metricas anteriores para que se consideren como
condiciones
#iniciales

echo 'net.ipv4.tcp_no_metrics_save = 1' >> /etc/sysctl.conf

#A continuacion recargar los cambios

sysctl -p
```

6. Simulación interfaces WLAN y 3G (tbf.sh)

```
#!/bin/bash
#
#  tc uses the following units when passed as a parameter.
#  kbps: Kilobytes per second
#  mbps: Megabytes per second
#  kbit: Kilobits per second
#  mbit: Megabits per second
#  bps: Bytes per second
#       Amounts of data can be specified in:
#       kb or k: Kilobytes
#       mb or m: Megabytes
#       mbit: Megabits
#       kbit: Kilobits
#  To get the byte figure from bits, divide the number by 8 bit
#
#TC=/sbin/tc

#----------------PARAMETROS DE ETH1----------------
IF1=eth1            # Interface
UPLD1=8mbit         # Outgoing traffic Limit
LAT1=80ms            # Latency attached
#--------------------------------------------------
```

```
#----------------PARAMETROS DE ETH2----------------
IF2=eth2            # Interface
UPLD2=1mbit         # Outgoing traffic Limit
LAT2=2000ms            # Latency attached
#--------------------------------------------------
#----------------PARAMETROS DE ETH3----------------
IF3=eth3            # Interface
UPLD3=1mbit         # Outgoing traffic Limit
LAT3=2000ms            # Lantency attached
#--------------------------------------------------
#------------------ACTIONS TO DO-------------------
start() {

    tc qdisc add dev $IF1 root handle 1:0 netem delay 20ms 5ms
distribution normal
    tc qdisc add dev $IF1 parent 1:1 handle 10: tbf rate $UPLD1 burst
100KB latency $LAT1
    tc qdisc add dev $IF2 root handle 1:0 netem delay 150ms 10ms
distribution normal
    tc qdisc add dev $IF2 parent 1:1 handle 10: tbf rate $UPLD2 burst
100KB latency $LAT2
    #tc qdisc add dev $IF3 root handle 3:0 tbf rate $UPLD3 burst 2048
latency $LAT3

}

stop() {

    tc qdisc del dev $IF1 root
    tc qdisc del dev $IF2 root
    #tc qdisc del dev $IF3 root

}

restart() {

    stop
    sleep 1
    start

}

show() {

    tc -s qdisc ls dev $IF1
    tc -s qdisc ls dev $IF2
    #tc -s qdisc ls dev $IF3

}

case "$1" in

  start)

    echo -n "Starting bandwidth shaping: "
    start
    echo "done"
    ;;

  stop)
```

```
    echo -n "Stopping bandwidth shaping: "
    stop
    echo "done"
    ;;

  restart)

    echo -n "Restarting bandwidth shaping: "
    restart
    echo "done"
    ;;

  show)

    echo "Bandwidth shaping status for $IF1:\n"
    show
    echo ""
    ;;

  *)

    pwd=$(pwd)
    echo "Usage: $(/usr/bin/dirname $pwd)/tc.bash
{start|stop|restart|show}"
    ;;

esac

exit 0
```

7. Control de BW en R1 (htb.sh)

```
#!/bin/bash
#
#  tc uses the following units when passed as a parameter.
#  kbps: Kilobytes per second
#  mbps: Megabytes per second
#  kbit: Kilobits per second
#  mbit: Megabits per second
#  bps: Bytes per second
#       Amounts of data can be specified in:
#       kb or k: Kilobytes
#       mb or m: Megabytes
#       mbit: Megabits
#       kbit: Kilobits
#  To get the byte figure from bits, divide the number by 8 bit
#
#TC=/sbin/tc

#----------------PARAMETROS DE ETH1----------------
IF1=eth1            # Interface
UPLD1=5mbit         # Outgoing traffic Limit
#--------------------------------------------------
#----------------PARAMETROS DE ETH2----------------
IF2=eth2            # Interface
UPLD2=5mbit         # Outgoing traffic Limit
#--------------------------------------------------
#----------------PARAMETROS DE ETH3----------------
IF3=eth3            # Interface
```

```
UPLD3=10mbit         # Outgoing traffic Limit
#------------------------------------------------
#----------------PARAMETROS DE ETH4---------------
IF4=eth4            # Interface
UPLD4=10mbit         # Outgoing traffic Limit
#------------------------------------------------
#------------------ACTIONS TO DO-------------------
start() {

    tc qdisc add dev $IF1 root handle 1: htb default 1
    tc class add dev $IF1 parent 1: classid 1:1 htb rate $UPLD1

    tc qdisc add dev $IF2 root handle 1: htb default 1
    tc class add dev $IF2 parent 1: classid 1:1 htb rate $UPLD2

    tc qdisc add dev $IF3 root handle 1: htb default 1
    tc class add dev $IF3 parent 1: classid 1:1 htb rate $UPLD3

    tc qdisc add dev $IF4 root handle 1: htb default 1
    tc class add dev $IF4 parent 1: classid 1:1 htb rate $UPLD4
}

stop() {

    tc qdisc del dev $IF1 root
    tc qdisc del dev $IF2 root
    tc qdisc del dev $IF3 root
    tc qdisc del dev $IF4 root

}

restart() {

    stop
    sleep 1
    start

}

show() {

    tc -s qdisc ls dev $IF1
    tc -s qdisc ls dev $IF2
    tc -s qdisc ls dev $IF3
    tc -s qdisc ls dev $IF4

}

case "$1" in

  start)

    echo -n "Starting bandwidth shaping: "
    start
    echo "done"
    ;;

  stop)

    echo -n "Stopping bandwidth shaping: "
    stop
```

```
    echo "done"
    ;;

  restart)

    echo -n "Restarting bandwidth shaping: "
    restart
    echo "done"
    ;;

  show)

    echo "Bandwidth shaping status for $IF1:\n"
    show
    echo ""
    ;;

  *)

    pwd=$(pwd)
    echo "Usage: $(/usr/bin/dirname $pwd)/tc.bash
{start|stop|restart|show}"
    ;;

esac

exit 0
```