UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

FACULTAD DE
INGENIERÍA

Experimentando con Multi-Path TCP

Informe de Proyecto de Grado presentado por

Mauricio Braica, Nicolás Bruzzese

como requisito de graduación de la carrera de Ingeniería en Computación de
Facultad de Ingeniería de la Universidad de la República

Supervisores

Eduardo Grampín
Leonardo Alberro

Montevideo, 1 de marzo de 2023

Agradecimientos

En lo personal, con este informe se aproxima el fin de lo que ha sido una larga etapa que ha estado repleta de emociones. Por ello, me gustaría tener un momento para agradecer a todas aquellas personas que me fueron importantes en este tiempo.

En primera instancia me gustaría agradecer a mis padres que siempre me apoyaron, alentándome a sacar siempre lo mejor y me enseñaron la importancia de finalizar los estudios. Junto con ellos, le quiero dar las gracias a mi hermana cuyo apoyo emocional fue invaluable durante este último tramo.

Más aún, quiero reconocer a amigos como Martín Ferreira, Nicolás Guggeri y Juan Lago que me acompañaron desde el inicio. Con un agradecimiento especial a mi compañero de tesis Nicolás Bruzzese por su solidez y las charlas que siempre lograban alegrar el día.

Finalmente, le agradezco a mis tutores por sus guías y consejos que fueron de gran ayuda para la realización de la tesis.

Mauricio Braica Alcalde

A través de este trabajo concluye una de las etapas más importantes de mi vida, y es por eso que quisiera aprovechar este instante para agradecerle a todas aquellas personas que me acompañaron durante este proceso.

Quisiera comenzar dándole las gracias a mis padres y a mi hermano por el apoyo que me brindaron todos estos años, por impulsarme seguir adelante con mi carrera universitaria y mis metas en general. También agradecerles a mis abuelos y tíos por siempre estar pendientes de mis estudios, por alegrarse cada vez que logro algún avance, y por desearme el mejor de todos los éxitos.

Por otro lado, quiero agradecerle a mi amigo y compañero de proyecto de grado por haberme acompañado con este y otros tantos trabajos durante todos estos años de la carrera.

Les agradezco a mis tutores de proyecto de grado por la dedicación y la guía que nos brindaron para el desarrollo de este trabajo. Gracias por confiar en nosotros para la realización de este proyecto final.

Nicolás Germán Bruzzese De León

Resumen

Multi-Path TCP (MPTCP) consiste en un conjunto de extensiones definidas para el protocolo de transporte TCP, cuya finalidad radica en brindarle a una conexión de transporte la posibilidad de operar a través de múltiples rutas de forma simultánea. Específicamente, MPTCP ofrece los mismos servicios que TCP y, además, provee los mecanismos necesarios para establecer y utilizar múltiples flujos TCP a través de rutas potencialmente disjuntas.

Una topología de red fat-tree garantiza la existencia de múltiples rutas físicas entre cada par de equipos conectados en su frontera. Una correcta utilización del protocolo MPTCP en una topología con estas características puede optimizar el uso de recursos físicos disponibles, implicando beneficios para las conexiones de transporte. En particular, estos aspectos son trasladables a los centros de datos masivos que disponen de una infraestructura de red fat-tree.

Este proyecto aborda el estudio del protocolo de transporte Multi-Path TCP en el marco de una topología de red fat-tree.

El estudio del protocolo se realiza por medio de un proceso de emulación en máquinas virtuales, utilizando distribuciones Linux que disponen de una implementación nativa y en desarrollo del protocolo Multi-Path TCP. Partiendo de una máquina virtual, se virtualiza la topología de red fat-tree sobre la cual se ejecutan diversos casos de prueba que involucran el protocolo estudiado.

Durante el desarrollo del proyecto se emplean herramientas de carácter esencial para lograr una correcta trayectoria. Entre éstas se destacan Wireshark para la captura y análisis de tráfico de red, Mininet para la virtualización de la topología fat-tree, y Multi-Path TCP Deamon para la gestión de rutas establecidas por el protocolo. La codificación de programas se realiza utilizando el lenguaje de programación C, exceptuando la implementación de la topología fat-tree que se codifica en Python.

Finalmente, se evidencia el funcionamiento del protocolo Multi-Path TCP y se determina su grado de maleabilidad aplicable a distintos escenarios. Se concluye su relevancia en infraestructuras de red fat-tree cuando se propone optimizar el uso de recursos disponibles.

Palabras clave: Multi-Path TCP, Fat-Tree, Ruta, Linux, Emulación, Centro de Datos.

Índice general

1. Introducción

En la actualidad, el consumo de servicios provistos a través de Internet se a tornado en una necesidad habitual por parte de organizaciones e individuos particulares. Debido a la amplia variedad de servicios, en un contexto donde su demanda se mantiene en aumento, los proveedores de servicios suelen recurrir a los centros de datos de gran escala.

De esta forma, un centro de datos consiste en una instalación física que dispone de una infraestructura de red diseñada con el fin de proveer servicios de forma masiva, a distintas clases de consumidores. Servicios que involucran almacenamiento de datos, *web hosting*, o incluso la provisión de recursos computacionales virtuales, son algunos de los servicios que un centro de datos permite ofrecer (Medhi y Ramasamy, 2017).

A nivel de infraestructura, los centros de datos basan su diseño de red en la redundancia para asegurar su resiliencia frente imprevistos. Parte de esta redundancia radica en la disposición de múltiples conexiones físicas entre los distintos componentes de red que conforman la infraestructura. En consecuencia, los centros de datos presentan múltiples rutas potencialmente disjuntas que interconectan sus sistemas terminales (Medhi y Ramasamy, 2017). A pesar de que los centros de datos presentan redundancia en su topología, los protocolos de transporte que suelen ser empleados en Internet no son capaces de aprovecharla.

En lo referente al transporte de información en Internet, comúnmente se emplea el protocolo Transmission Control Protocol (TCP) (Eddy, 2022). Éste permite establecer una conexión lógica entre sus extremos, dando origen a un flujo de datos confiables. Adicionalmente, TCP implementa mecanismos para el control de congestión y para el control de flujo, entre otros. Sin embargo, este protocolo se encuentra limitado a operar por medio de un único flujo de datos que se restringe a una de las rutas existentes entre los extremos implicados.

Por el contrario, el protocolo Multi-Path TCP (MPTCP) (Ford y cols., 2020) surge como una extensión del protocolo TCP tradicional, heredando todos sus mecanismos y añadiendo la capacidad de operar a través de múltiples flujos en forma simultánea. La asignación de cada uno de estos flujos con respecto a las distintas rutas de un centro de datos, prevé un mejor aprovechamiento de los recursos disponibles (Raiciu y cols., 2011).

Considerando el potencial que implica el protocolo Multi-Path TCP, se establece como objetivo principal su estudio en relación a una topología de red fat-tree, siendo ésta un diseño de infraestructura característico de algunos centros de datos. El alcance de este objetivo involucra la investigación de la operativa del protocolo Multi-Path TCP, junto con una evaluación de su desempeño. Particularmente, interesa contrastar su rendimiento con respecto al de su predecesor, el protocolo TCP.

Para el desarrollo del objetivo propuesto, se comienza con un proceso de adquisición de conocimientos referentes al protocolo Multi-Path TCP y a la topología de red fat-tree. Posteriormente, se establece una fase de experimentación que consiste en el diseño y ejecución de casos de prueba y de validación. Un caso de validación posee como finalidad corroborar el correcto funcionamiento del las distintas implementaciones utilizadas, destacando entre éstas la propia implementación del protocolo MPTCP. Por otro lado, los casos de prueba se definen con el propósito de evaluar el rendimiento del protocolo MPTCP en el contexto de una infraestructura de red fat-tree.

La ejecución de los distintos casos de prueba y de validación se realiza por medio de una máquina virtual con sistema operativo Linux que dispone de su propia implementación nativa de Multi-Path TCP. En adición, dentro de esta misma máquina virtual se establece una emulación que genera una topología de red fat-tree.

Por otro lado, se incursiona en el estudio e implementación de un manejador de rutas. Éste corresponde a un programa encargado de gestionar arbitrariamente el funcionamiento de las conexiones asociadas al protocolo MPTCP.

En función de los objetivos establecidos, se espera comprobar un rendimiento superior por parte del protocolo Multi-Path TCP en comparación con el protocolo TCP. De igual manera, se prevén ventajas significativas con respecto a la robustez provista por MPTCP.

Este documento respeta la siguiente estructura:

1. **Introducción**
   Brinda un panorama general del proyecto abordado, estableciendo el contexto sobre el cual se desarrolla. Además de considerar la motivación implicada, también detalla los objetivos principales.

2. **Estado del arte**
   Introduce conceptos esenciales para la comprensión del proyecto. Comienza exponiendo el protocolo TCP para luego profundizar con las características del protocolo Multi-Path TCP. Adicionalmente, presenta la topología fat-tree y el programa manejador de rutas Multi-Path TCP Deamon.

3. **Enfoque de estudio**
   Detalla el ambiente de trabajo definido para la experimentación con el protocolo Multi-Path TCP, exponiendo decisiones de diseño e implementaciones realizadas.

4. **Experimentación**
   Describe los distintos casos de prueba realizados y exhibe los resultados obtenidos a partir de su ejecución.

5. **Conclusiones y trabajo futuro**
   Presenta las conclusiones del estudio presentado en este documento. En adición, sugiere trabajo a futuro con el fin de continuar y complementar la investigación expuesta en este proyecto.

6. **Referencias**
   Enumera el conjunto de fuentes consultadas para la realización de este proyecto.

7. **Anexos**
   Incluye información adicional que complementa el trabajo presentado.

2. Estado del arte

En este capítulo se introducen conceptos de carácter esencial para la comprensión del proyecto. Inicialmente, se expone el concepto de protocolos de Internet para luego adentrarse con los protocolos de trasporte TCP y MPTCP. Una vez detallados los distintos aspectos referentes al protocolo MPTCP, se procede con la exposición del programa Multi-Path TCP Deamon, el cual permite la definición de un manejador de rutas que opera en función de pautas especificadas por el usuario. Finalmente, el capítulo concluye con la exposición de las redes de centros de datos donde se enfatiza en la topología de red fat-tree.

2.1. Protocolos de Internet

Internet es una infraestructura que proporciona servicios a aplicaciones distribuidas, siendo que sus componentes se comunican a través de protocolos. Un protocolo denota un conjunto de reglas o convenciones que dos o más entidades deben seguir para lograr comunicarse correctamente. A su vez, un protocolo también estipula qué acciones deben ser realizadas en cada instante del proceso de comunicación (Kurose y Ross, 2010).

En Internet, los protocolos se organizan en una arquitectura en capas. Esto permite definir un modelo de servicio de capa, donde cada una utiliza aquellas que posee por debajo para ofrecer servicios a la capa que yace inmediatamente por encima suyo.

2.1.1. Capa de protocolos de Internet

Corresponde destacar que existen variantes en cuanto a modelos de capas de protocolos de Internet. En esta ocasión, se opta por un modelo que consta de cinco capas como se muestra en la Figura 2.1.

Figura 2.1: Modelo de Capas de Internet.

- Capa de aplicación: en esta capa residen las aplicaciones de red y sus protocolos asociados. El objetivo de un protocolo de la capa de aplicación consiste en intercambiar paquetes de información entre aplicaciones ejecutadas en sistemas terminales (hosts). En esta capa se encuentran los protocolos HTTP, SMTP y DNS, entre otros.
- Capa de transporte: los protocolos que residen en esta capa tienen como objetivo intercambiar información entre procesos de aplicaciones. En esta capa se destacan los protocolos TCP y UDP.
- Capa de red: esta capa es responsable de intercambiar información entre sistemas terminales. En esta capa reside el protocolo IP y los protocolos de enrutamiento que se encargan de determinar qué trayecto recorren los paquetes de información entre su origen y destino.
- Capa de enlace: esta capa se caracteriza por transmitir información entre equipos directamente conectados o adyacentes. Algunos de los protocolos que se encuentran en esta capa son Ethernet, WiFi y PPP.

- Capa física: su objetivo consiste en transportar los bits individuales de un nodo a otro. Los protocolos que residen en esta capa determinan la forma en que se transportan los bits.

El protocolo Multi-Path TCP surge como una extensión del protocolo TCP, siendo que este último reside en la capa de transporte. Por lo tanto, el estudio que se aborda en este proyecto está centrado en la capa de transporte.

2.2. Protocolo TCP

Transmission Control Protocol (TCP) es un protocolo de transporte fiable y orientado a la conexión que se emplea ampliamente en Internet para el intercambio de datos (Eddy, 2022). Entre los servicios que este protocolo es capaz de ofrecer, se destacan los siguientes:

- Servicio orientado a la conexión: cuando dos procesos desean intercambiar información utilizando TCP, primero proceden a establecer una conexión lógica para así luego comenzar con la transmisión de datos.

  En este contexto, una conexión lógica implica que solo los hosts involucrados son conscientes de que ésta existe, mientras que se exenta de tal conocimiento a todo elemento de red que se encuentra entre estos hosts.

- Servicio full-duplex: en términos generales, una conexión TCP se caracteriza por ser una conexión punto a punto. Un servicio full-duplex permite que ambos puntos puedan enviar y recibir información durante la misma sesión. En otras palabras, este servicio brinda la posibilidad de que se transmitan flujos de datos en ambos sentidos de forma simultánea.

- Servicio de transferencia de datos fiable: a partir de este servicio, TCP garantiza que el flujo de bytes que recibe un sistema terminal en un extremo de la conexión es exactamente el mismo flujo de bytes que fue enviado por el sistema terminal en el otro extremo. Esto implica que un flujo de datos transmitido de un extremo al otro mantiene su integridad, siendo recibido por el extremo receptor en el mismo orden en que fue enviado por el extremo emisor.

  Este servicio ofrece un flujo de datos fiable de extremo a extremo, por encima de un servicio de mejor esfuerzo pero no fiable que es brindado por el protocolo IP en la capa de red.

- Servicio de control de flujo: en ocasiones, la velocidad con la que el extremo emisor transmite datos no coincide con la velocidad con la cual el extremo receptor logra obtenerlos y procesarlos. Cuando el receptor es quien opera a una menor velocidad se puede incurrir en un desbordamiento de datos recibidos.

  El servicio de control de flujo permite que el emisor controle la velocidad con la cual transmite datos al receptor, fomentando que se eviten desbordamientos y que los recursos de red involucrados sean aprovechados de mejor manera.

- Servicio de control de congestión: cuando se detecta una congestión de red, este servicio se propone limitar la velocidad con la cual un emisor transmite tráfico a través de su conexión TCP.

  Se trata de un servicio que apunta a optimizar la utilización de recursos de red de manera general, sin estar enfocado específicamente en un una conexión TCP en particular. Es decir, este servicio busca proporcionar una equidad entre las conexiones TCP, en lo referente a la utilización de recursos de red disponibles.

2.2.1. Estructura de segmento TCP

La unidad de envío del protocolo TCP se denomina como segmento. Un segmento es un paquete de información compuesto por campos de cabecera TCP y un campo de carga útil de datos. En la Figura 2.2 se aprecia la estructura de un segmento TCP.

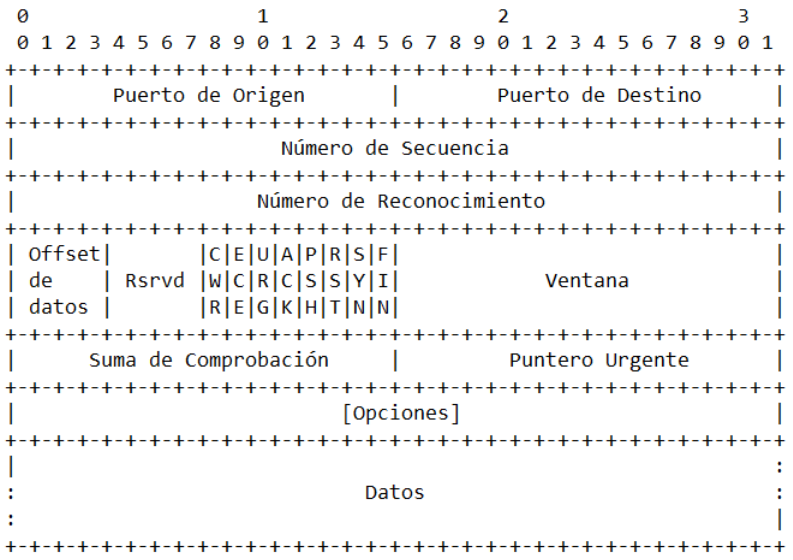

Figura 2.2: Estructura de segmento TCP. Adaptado de (Eddy, 2022).

A continuación, se detalla la semántica de los campos presentes en la estructura del segmento TCP:

- Puertos de origen y destino: tienen como finalidad permitir la multiplexación y demultiplexación[1] de datos respecto a las aplicaciones de la capa superior.
- Número de secuencia: expresa el número del primer byte del segmento dentro del flujo de bytes.
- Número de reconocimiento: se utiliza para indicar el número de secuencia del siguiente byte que el proceso emisor del segmento espera recibir del otro proceso.
- Offset de datos: indica la longitud de la cabecera TCP y por lo tanto, determina dónde empieza el campo de datos.
- Reservado (Rsrvd): conjunto de bits reservados para uso futuro. Su valor debe ser cero.
- Bits de control: también conocidos como *flags*, le añaden semántica al segmento TCP:
  - CWR: ventana de congestión reducida.
  - ECE: indica si el par TCP es compatible con ECN.
  - URG: el campo puntero urgente es significativo.
  - ACK: el campo número de reconocimiento es significativo.

[1] La multiplexación es una tarea que consiste en reunir fragmentos de datos en el host de origen que proceden desde diferentes *sockets*, para luego añadirle a cada fragmento información de cabecera TCP y así conformar segmentos TCP que posteriormente son enviados hacia la capa de red. Por otro lado, la demultiplexación implica la entrega de datos contenidos en segmentos TCP de la capa de trasporte al *socket* correcto en el host de destino (Kurose y Ross, 2010).

- PSH: le indica al receptor que procese los paquetes tan pronto como los reciba.
- RST: reiniciar conexión.
- SYN: sincronizar números de secuencia.
- FIN: el emisor no enviará más datos.

- Ventana: determina el número de bytes que el emisor está dispuesto a aceptar. Es utilizado para el servicio de control de flujo.
- Suma de comprobación (*Checksum*): se incluye para comprobar la integridad del segmento.
- Puntero urgente: si hay datos urgentes, este campo indica la posición del último byte asociado a los datos urgentes.
- Opciones: se trata de un campo opcional y de longitud variable que tiene como finalidad introducir funcionalidades al protocolo TCP.
- Datos: contiene la carga útil de datos provenientes del proceso aplicativo.

Resulta importante destacar el campo *Opciones* del cabezal TCP, siendo que este campo permite introducir modificaciones y optimizaciones al protocolo TCP sin implicar cambios en su definición. En relación a esto, el protocolo Multi-Path TCP opera a través de TCP, donde la señalización de sus respectivas operaciones se incluye en el campo *Opciones* del cabezal TCP.

2.3. Protocolo Multi-Path TCP

El protocolo Multi-Path TCP (MPTCP) se define como un conjunto de extensiones para el protocolo TCP tradicional, cuya finalidad consiste en brindarle a una conexión de transporte la posibilidad de operar a través de múltiples rutas en forma simultánea (Ford y cols., 2020). En esencia, MPTCP ofrece el mismo tipo de servicio que TCP le brinda a las aplicaciones, siendo que además provee los componentes necesarios para establecer y utilizar múltiples flujos TCP (subflujos) a través de rutas potencialmente disjuntas.

De igual manera que TCP, el protocolo MPTCP opera a nivel de capa de transporte y pretende ser transparente para las capas superiores en inferiores. En la Figura 2.3 se puede visualizar como MPTCP consiste en un conjunto de funcionalidades adicionales que operan por encima del protocolo TCP estándar.

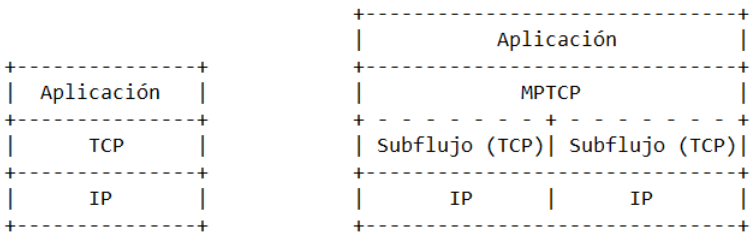

Figura 2.3: Comparación de stack de los protocolos TCP y MPTCP. Adaptado de (Ford y cols., 2020).

2.3.1. Decisiones de diseño

Desde el punto de vista del diseño del protocolo MPTCP, se consideran dos restricciones fundamentales:

- El protocolo debe ser compatible hacia atrás con TCP, con el fin de que su utilización pueda ser extendida en mayor medida.
- Puede ser asumido que uno o ambos hosts estén conectados en múltiples subredes distintas, disponiendo de múltiples direcciones.

En adición, la restricción de compatibilidad hacia atrás implica tres aspectos a considerar:

- Restricciones externas: el protocolo debe funcionar a través de la vasta cantidad de *middleboxes* existentes, tales como NATs, *firewalls* y *proxies*. Además, se espera que se asemeje al ya existente protocolo TCP, en el contexto de ser transmitido por cable.
- Restricciones aplicativas: el protocolo debe poder ser utilizado sin implicar cambios en las aplicaciones ya existentes que utilizan una API[2] TCP. A su vez, el protocolo debe brindar el mismo modelo de servicio que TCP podría ofrecerle a la aplicación en cuestión.
- Repliegue: el protocolo debe ser capaz de operar como el protocolo TCP estándar, para permitir la comunicación con hosts de tipo legado. Esta funcionalidad debe ser transparente para el usuario.

Finalmente, se establece que la presencia de múltiples direcciones en un host es suficiente para indicar la existencia de múltiples rutas. Estas rutas no necesariamente deben de ser completamente disjuntas. En otras palabras, las rutas pueden compartir uno o varios routers entre ellas.

2.3.2. Terminología

En esta subsección se expone la terminología específica del protocolo MPTCP:

- Ruta (*Path*): secuencia de enlaces entre emisor y receptor, definida en este contexto por la 4-tupla {dirección origen, puerto origen, dirección destino, puerto destino}.
- Subflujo (*Subflow*): corresponde a un flujo de segmentos TCP operando a través de una ruta, formando así parte de una conexión MPTCP más grande. Un subflujo se inicializa y se finaliza de la misma forma que una conexión TCP.
- Conexión MPTCP: es un conjunto de uno o más subflujos, a través de los cuales una aplicación puede comunicarse entre dos hosts. Cada conexión se mapea con un *socket* de aplicación.
- Nivel de conexión (*Connection-level*): referencia a las propiedades de la conexión MPTCP.
- Nivel de subflujo (*Subflow-level*): referencia a las propiedades de un subflujo individual que está asociado a una conexión MPTCP.
- Token: es un identificador localmente único, el cual es asignado a una conexión MPTCP por un host.
- Address ID: es un identificador localmente único, asignado por un host a una de sus direcciones locales disponibles.

[2] Una API o interfaz de programación de aplicaciones es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones (Red Hat, 2023)

2.3.3. Funcionamiento de Multi-Path TCP

En esta subsección se resume de forma general el comportamiento normal de una conexión MPTCP (Ford y cols., 2020):

- Para una aplicación que no utilice MPTCP, el protocolo se va a comportar como TCP estándar.
- Una conexión MPTCP se inicializa de forma similar a una conexión TCP regular.
- Si existen rutas extras que estén disponibles, se crearán sesiones TCP adicionales (también denominadas como "subflujos" MPTCP) en dichas rutas. Cada sesión adicional será combinada con la sesión ya existente, la cual continuará viéndose como una conexión individual para las aplicaciones en cada host.
- MPTCP identifica múltiples rutas a través de la presencia de múltiples direcciones en los hosts. La combinación de estas múltiples direcciones (tomando pares de direcciones, siendo una de cada host) dan lugar a las rutas adicionales.
- El descubrimiento de direcciones y establecimiento de subflujos adicionales se logra a través de un método encargado del manejo de rutas. En esencia, un host puede inicializar un nuevo subflujo utilizando su propia dirección adicional o, por otro lado, puede proceder a anunciar su dirección disponible al otro host esperando que éste sea el que inicialice el nuevo subflujo.
- MPTCP agrega números de secuencia de nivel de conexión para lograr el reensamblado de segmentos que provienen de múltiples subflujos con diferentes retardos de red.
- Los subflujos se finalizan como una conexión TCP normal (llevando a cabo el *four-way FIN handshake*). La conexión MPTCP se finaliza con un FIN a nivel de conexión.

En la Figura 2.4 se presenta un ejemplo de una conexión MPTCP. En este ejemplo se considera el *Host A* que inicialmente establece una conexión MPTCP con el *Host B*, utilizando las direcciones IP A1 y B1. Este establecimiento de conexión crea la conexión MPTCP junto con su primer subflujo asociado.

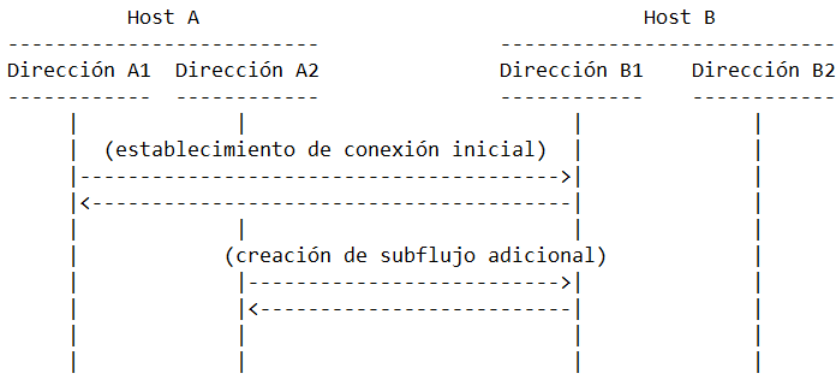

Figura 2.4: Ejemplo de utilización de MPTCP. Adaptado de (Ford y cols., 2020).

Una vez que la conexión se establece, el *Host A* procede a crear un subflujo adicional con el *Host B*, utilizando las direcciones A2 y B1.

Corresponde destacar que potencialmente otros dos subflujos podrían llegar a ser establecidos utilizando los pares de direcciones A1⟷B2 y A2⟷B2.

2.3.4. Estructura de la opción TCP usada para Multi-Path TCP

El cabezal del protocolo TCP dispone de un campo de largo variable denominado *Opciones*. Este campo brinda la posibilidad de añadir funcionalidades u optimizaciones al protocolo TCP estándar, sin la necesidad de redefinir el propio protocolo. Específicamente, el campo *Opciones* del cabezal TCP almacena desde ninguna a varias opciones TCP, siendo que cada una de éstas debe cumplir con cierta estructura predefinida (Eddy, 2022).

Multi-Path TCP opera a través del protocolo TCP utilizando el campo *Opciones* para señalizar sus operaciones. En la Figura 2.5 se expone el formato de la opción TCP utilizada exclusivamente para el señalamiento de operaciones MPTCP.

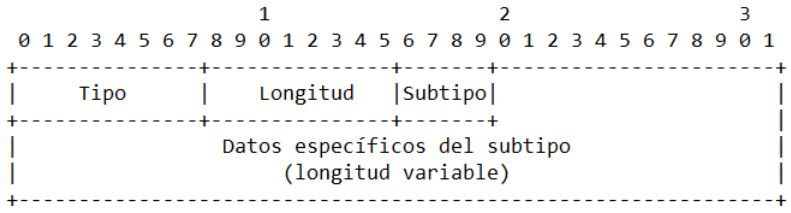

Figura 2.5: Formato de opción TCP empleada para el señalamiento de operaciones MPTCP. Adaptado de (Ford y cols., 2020).

La semántica de los campos expuestos en la Figura 2.5 corresponde a la siguiente:

- Tipo (*Kind*): representa el tipo de opción TCP, siendo el valor 30 (decimal treinta) aquel que representa específicamente a la opción MPTCP.
- Longitud (*Length*): representa el largo de la opción TCP, incluyendo el byte de *Tipo* y el byte de *Longitud*.
- Subtipo (*Subtype*): representa el tipo de operación MPTCP señalizada.

El valor asignado al campo *Subtipo* determina el formato de *Datos específicos del subtipo*. Esto se debe a que el valor de *Subtipo* referencia a una operación específica de MPTCP donde cada una de éstas implica intercambiar distintos tipos de datos. En la Figura 2.6 se visualizan los subtipos definidos para MPTCP junto con su significado asociado.

Valor	Símbolo	Nombre
0x0	MP_CAPABLE	Multipath Capable
0x1	MP_JOIN	Join Connection
0x2	DSS	Data Sequence Signal (Data ACK and Data Sequence Mapping)
0x3	ADD_ADDR	Add Address
0x4	REMOVE_ADDR	Remove Address
0x5	MP_PRIO	Change Subflow Priority
0x6	MP_FAIL	Fallback
0x7	MP_FASTCLOSE	Fast Close
0x8	MP_TCPRST	Subflow Reset
0xf	MP_EXPERIMENTAL	Reserved for Private Use

Figura 2.6: Subtipos de MPTCP. Adaptado de (Ford y cols., 2020).

2.3.5. Operaciones Multi-Path TCP

En esta subsección se exponen las principales operaciones del protocolo MPTCP, enfatizando en los aspectos claves de cada una.

Inicio de conexión MPTCP

El inicio de conexión comienza con un intercambio de señales SYN, SYN/ACK y ACK a través de una ruta individual, donde cada paquete contiene la opción MPTCP denominada como *Multipath Capable* (MP_CAPABLE). Esta opción declara la capacidad y deseo del emisor de emplear el protocolo MPTCP. La Figura 2.7 expone el comportamiento descrito a través del *Host A* que inicia una conexión MPTCP con el *Host B*.

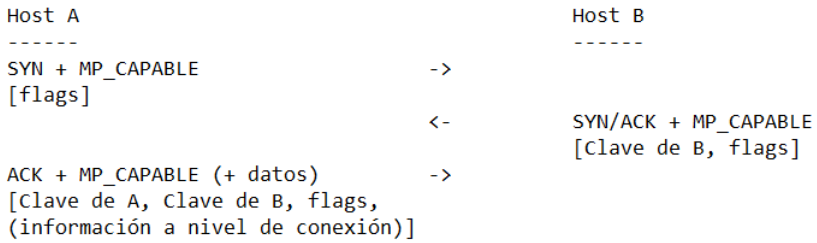

Figura 2.7: Establecimiento de conexión MPTCP. Adaptado de (Ford y cols., 2020).

Cada extremo debe generar una clave (*key*) aleatoria que será transmitida durante el proceso de establecimiento de conexión. Posteriormente, las claves serán utilizadas para la generación de parámetros de conexión.

Cuando un extremo recibe un segmento con el SYN inicial, éste procede a generar su clave aleatoria para luego enviársela al remitente a través de un segmento con SYN/ACK. En respuesta a un segmento con SYN/ACK, un extremo debe enviar un segmento con ACK que contenga su propia clave generada, incluyendo también la clave que acaba de recibir del otro extremo.

En la Figura 2.7, el *Host B* es el primero en generar y transmitir su clave aleatoria al *Host A*. En consecuencia, el *Host A* genera su propia clave aleatoria para luego transmitirla al *Host B*. En este último paso, el *Host A* también transmite la clave del *Host B* con el fin de realizar eco.

Cuando un extremo recibe la clave generada por su contraparte, éste utiliza la clave junto con una función de hash para generar un *token*. A partir del valor *token*, un host es capaz de identificar localmente a una conexión MPTCP.

El establecimiento de conexión también implica un intercambio de *flags* con el fin de negociar aspectos de la conexión MPTCP. Entre estos aspectos se destaca la determinación de si se desea emplear una suma de comprobación (*Checksum*). Otro aspecto negociable a través de *flags* corresponde al algoritmo criptográfico que será utilizado a largo de la conexión.

En caso de incumplir con las convenciones definidas para el establecimiento de conexión MPTCP, se procede entonces a establecer una conexión TCP estándar.

Establecimiento de un nuevo subflujo

Una vez que una conexión MPTCP se ha inicializado a través del intercambio de opciones *Multipath Capable* (MP_CAPABLE), nuevos subflujos adicionales pueden ser añadidos a la conexión.

Cada host tiene conocimiento de sus propias direcciones y puede llegar a tener conocimiento de las direcciones del otro host (en una conexión MPTCP, un host puede anunciarle sus direcciones IP al otro host implicado). Es entonces que a través de esta información, un host puede inicializar un nuevo subflujo sobre un par de direcciones que aún no esté siendo utilizado. Si bien cualquiera de los hosts puede inicializar nuevos subflujos, se espera que el host que inicializó la conexión en primer lugar sea aquel que lleve a cabo el proceso de inicializar nuevos subflujos.

El establecimiento de un nuevo subflujo se realiza por medio del intercambio de señales SYN, SYN/ACK y ACK, incluyendo en cada caso la opción MPTCP conocida como *Join Connection* (MP_JOIN). La Figura 2.8 muestra el intercambio descrito. Cuando un extremo procede a establecer un nuevo subflujo, éste comienza por enviar un segmento con señal SYN. En este primer segmento incluye su *token* generado durante el proceso de establecimiento de conexión, un número aleatorio (*nonce*), un valor de *Address ID* asociado a la dirección que está utilizando para establecer el nuevo subflujo, y su señalamiento de *flags*.

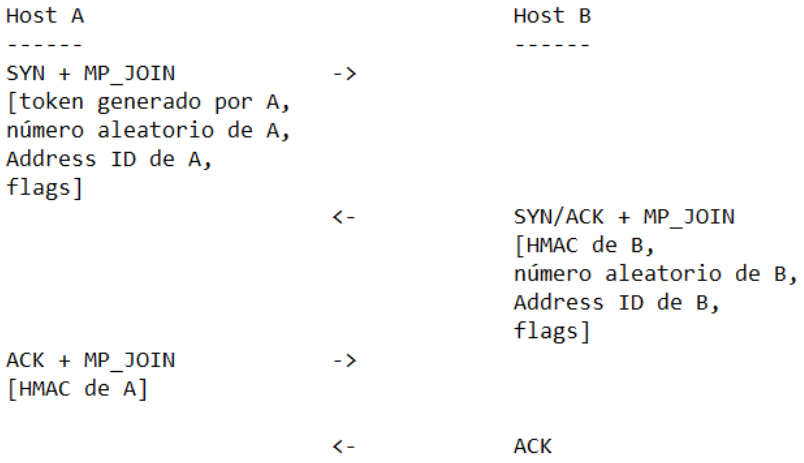

Figura 2.8: Establecimiento de un nuevo subflujo. Adaptado de (Ford y cols., 2020).

La generación y envío de números aleatorios (*nonce*) se utiliza por motivos de seguridad. Posteriormente cada extremo deberá utilizar el número aleatorio recibido, y las claves intercambiadas durante el proceso de inicio de conexión, para generar y enviar un valor de HMAC.

Un *Address ID* es un identificador que un host le asigna a una de sus direcciones locales disponibles. El identificador luego se transmite al host remoto, con el fin de que éste establezca una asociación entre dicho identificador y la dirección que representa en el host local. En otras palabras, el *Address ID* permite que un host identifique direcciones que están disponibles en el otro host.

En respuesta a un segmento SYN, se procede a enviar un segmento SYN/ACK con un valor de HMAC, un número aleatorio (*nonce*), un valor de *Address ID*, y un señalamiento de *flags*.

El intercambio de *flags* entre el primer y segundo segmento se utiliza para negociar la prioridad del subflujo que está por ser establecido. De esta forma, los extremos pueden acordar que el nuevo subflujo sea de tipo *backup* o no. Un subflujo definido como *backup* solo será utilizado para transmitir datos en caso de que todos los demás subflujos que no sean de tipo *backup* se encuentren como no disponibles. Por el contrario, un subflujo no definido como *backup* comenzará a transmitir datos en cuanto esté completamente establecido.

Finalmente, al recibir un segmento SYN/ACK se debe responder con un segmento ACK que contenga un valor de HMAC.

Una vez que culmina el intercambio de segmentos con señales SYN, SYN/ACK y ACK con la opción *Join Connection* (MP_JOIN), el nuevo subflujo permanece en estado "pre-establecido" hasta que el último de los tres segmentos mencionados sea reconocido por el extremo correspondiente. Una vez que es reconocido, el subflujo pasa a estado "establecido" y puede comenzar a trasmitir datos con normalidad.

La Figura 2.9 ejemplifica las operaciones de inicio de conexión MPTCP y de establecimiento de un nuevo subflujo, entre un *Host A* y un *Host B*.

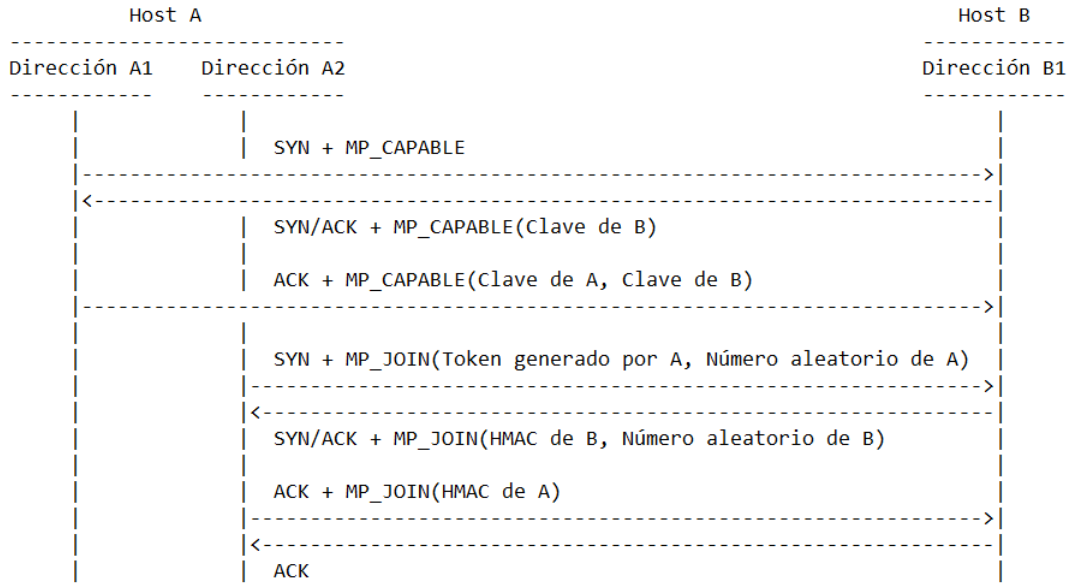

Figura 2.9: Inicio de conexión MPTCP y establecimiento de un nuevo subflujo. Adaptado de (Ford y cols., 2020).

Transferencia de datos

Durante la operación de transferencia de datos, MPTCP obtiene el flujo de datos de entrada de una aplicación y lo reparte entre uno o más subflujos. A continuación, le añade la suficiente información de control para permitir el reensamblado, junto con la entrega confiable y ordenada, en la aplicación receptora.

El señalamiento de esta operación MPTCP se realiza a través de la opción *Data Sequence Signal* (DSS). La Figura 2.10 expone esta opción junto con sus parámetros involucrados.

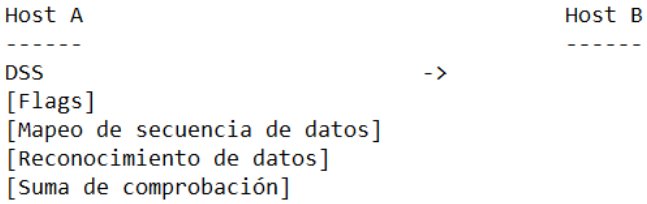

Figura 2.10: Transferencia de datos. Adaptado de (Ford y cols., 2020).

El segmento con opción *Data Sequence Signal* (DSS) incluye un señalamiento de *flags* que determina qué tipo de información se incluye. En particular, dependiendo de la confi-

guración de *flags*, el segmento puede incluir información de mapeo de secuencia de datos y/o información de reconocimiento de datos. La suma de comprobación solo se incluye si fue negociada durante el proceso de inicio de conexión MPTCP.

El mapeo de secuencia de datos corresponde a un conjunto de datos que mapea el número de secuencia a nivel de subflujo con el número de secuencia a nivel de conexión. El número de secuencia a nivel de subflujo es relativo a éste y, por lo tanto, independiente respecto al de los demás subflujos existentes. Por otro lado, el número de secuencia a nivel de conexión es absoluto y aplica para toda la conexión.

El número de secuencia a nivel de conexión se emplea para el reensamblado en el extremo receptor. Dado que MPTCP permite la retransmisión de información a través de distintos subflujos, un mismo número de secuencia a nivel de conexión puede estar mapeado a varios números de secuencia a nivel de subflujo cuando ocurren retransmisiones.

El primer número de secuencia a nivel de conexión con el cual un host comienza a transmitir datos se calcula aplicando una función de hash a la clave que fue generada por el mismo host en el inicio de conexión MPTCP.

Por otro lado, el reconocimiento de datos corresponde a información que MPTCP utiliza para proveer resiliencia punto a punto. Utilizando esta información, el protocolo provee reconocimientos acumulativos a nivel de conexión.

Una característica importante a destacar consiste en que cada subflujo se modela como una conexión TCP estándar que luego, a través de la inspección de su campo de cabecera *Opciones*, se puede determinar que lógicamente pertenece a una conexión MPTCP de dimensiones mayores. Por lo tanto, cada subflujo individual adicionalmente dispone de las funcionalidades que la propia conexión TCP le provee. Entre estas funcionalidad se encuentra el reconocimiento de datos a nivel de conexión TCP (cada subflujo dispone de números de secuencia y de reconocimiento que son propios de la conexión TCP que lo modelan).

Cuando un emisor no dispone de más datos por enviar, éste utiliza un segmento con opción *Data Sequence Signal* (DSS) para enviar una señal de *data_fin* a través del campo *flags*. Esta señal fuerza a que se verifiquen que todos los datos enviados hayan sido recibidos exitosamente para luego proceder con el cierre de conexión. Aquellos datos que no hayan sido reconocidos deberán ser retransmitidos antes de finalizar la conexión.

Una conexión MPTCP se finaliza cerrando cada uno de los subflujos individuales, para luego recién cerrar la conexión MPTCP por completo.

Anuncio de direcciones

El conjunto de direcciones asociadas a un host puede variar a lo largo del tiempo de vida de una conexión MPTCP. En consecuencia, el protocolo permite el anuncio y eliminación de direcciones mientras se realiza la conexión.

En lo referente al anuncio de direcciones, existen dos formas de realizar esta tarea:

- Forma implícita: el host emisor utiliza una dirección local disponible para establecer un nuevo subflujo con el host receptor. Al establecer el nuevo subflujo, el host receptor logra advertirse de la dirección local con la que dispone el host emisor.

- Forma explícita: en determinadas circunstancias un host puede desear anunciarle a otro sobre la disponibilidad de una de sus direcciones locales, sin verse obligado a establecer un nuevo subflujo durante el proceso. En este escenario, un host envía un segmento con opción *Add Address* (ADD_ADDR) en el cual incluye información sobre su dirección local disponible. Posteriormente, el host que recibida tal información podrá utilizarla para establecer nuevos subflujos utilizando la opción *Join Connection* (MP_JOIN).

En la Figura 2.11 se detalla el intercambio de segmentos que permite el anuncio explícito de direcciones locales disponibles.

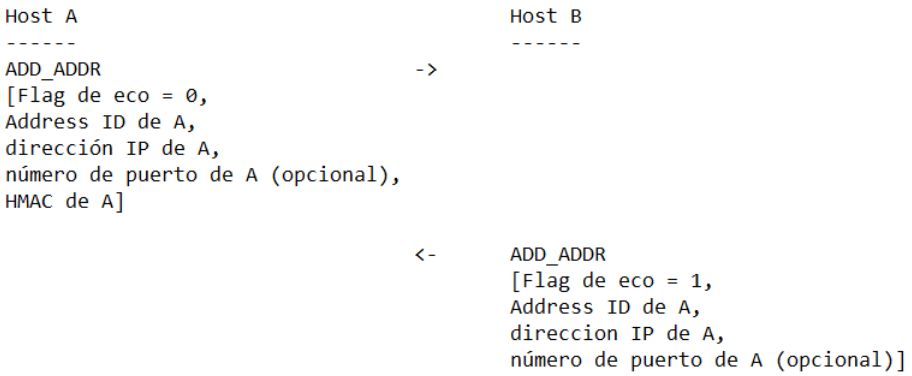

Figura 2.11: Anuncio explícito de dirección local disponible. Adaptado de (Ford y cols., 2020).

Cuando un extremo desea anunciar explícitamente una de sus direcciones locales disponibles al otro extremo, éste procede a enviar un segmento con opción *Add Address* (ADD_ADDR) incluyendo la *flag* de eco en valor cero, el valor de *Address ID* asociado a la dirección anunciar, la dirección IP que anuncia, y un valor de HMAC. Adicionalmente, puede incluir información referente al número de puerto que está asociado a la dirección anunciada.

La *flag* de eco se emplea para implementar un mecanismo de confirmación de recepción por medio de eco. De esta forma, el anunciante siempre configura la *flag* de eco en cero.

El valor de HMAC incluido se calcula a partir las claves intercambiadas en el inicio de conexión MPTCP, el *Address ID*, la dirección IP, y el número de puerto (en caso de que haya sido especificado).

Cuando un extremo recibe un anuncio de dirección, éste debe confirmar la recepción por medio de un eco. Esto implica enviarle al remitente el mismo anuncio de dirección recibido, pero sin incluir el valor de HMAC y configurado la *flag* de eco en valor uno.

Eliminación de direcciones

Mientras se mantiene una conexión MPTCP, una dirección local previamente anunciada puede dejar estar disponible. Esto puede deberse a motivos que involucren imperfectos en el funcionamiento de la red, o puede estar ligado directamente a una decisión arbitraria por parte de un host que desea inhabilitar una dirección suya.

Frente a esta situación, un host debe proceder a notificarle al otro sobre su dirección local que ha dejado de estar disponible. De esta forma, el host notificado comenzará a eliminar los subflujos existentes que estén asociados a dicha dirección que ya no se encuentra disponible.

Este tipo de anuncios se realizan a través de la opción MPTCP denominada como *Remove Address* (REMOVE_ADDR). A partir de esta opción es posible remover direcciones que previamente fueron añadidas a una conexión MPTCP, implicando que cada subflujo que las esté empleando proceda a ser cerrado. La Figura 2.12 muestra el envío de un segmento con la semántica de remover una dirección local previamente anunciada.

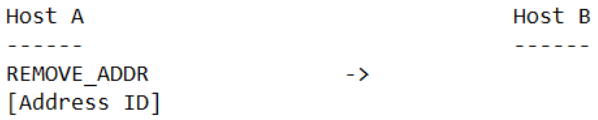

Figura 2.12: Eliminación de dirección. Adaptado de (Ford y cols., 2020).

Un segmento con opción *Remove Address* (REMOVE_ADDR) debe incluir el valor de *Address ID* que está asociado a la dirección que se desea eliminar. Específicamente, el segmento podrá incluir uno o varios valores de *Address ID* implicando una o varias eliminaciones de direcciones.

Cambio de prioridad de subflujo

Los subflujos establecidos a lo largo de una conexión MPTCP disponen de una prioridad asociada. En particular, MPTCP solo define dos tipos prioridades posibles: *backup* o *no backup*.

Un subflujo de tipo *backup* (o de tipo respaldo) solo transmite información cuando el resto de los subflujos existentes de tipo *no backup* se encuentran como no disponibles. Por el contrario, un subflujo de tipo *no backup* (o de tipo activo) comienza a transmitir información inmediatamente luego de ser establecido.

Si bien los subflujos se establecen con una prioridad inicial, en cada caso ésta puede ser modificada a lo largo de una conexión MPTCP. La opción *Change Subflow Priority* (MP_PRIO) permite modificar la prioridad de los subflujos. La Figura 2.13 expone el envío de un segmento con la finalidad realizar un cambio de prioridad.

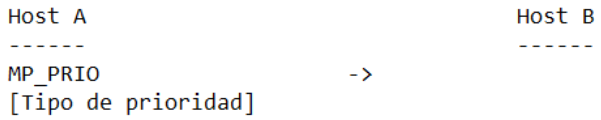

Figura 2.13: Cambio de prioridad de subflujo. Adaptado de (Ford y cols., 2020).

Cuando un extremo desea cambiar la prioridad de uno de sus subflujos, éste debe proceder a enviar un segmento con opción *Change Subflow Priority* (MP_PRIO) incluyendo la nueva prioridad a ser considerada. El segmento debe ser enviado a través del subflujo sobre el cual se desea aplicar el cambio de prioridad.

Resulta importante destacar que el cambio de prioridad corresponde a una solicitud que un extremo le realiza al otro. Al tratarse de una solicitud, el extremo que la recibe puede optar por ignorarla si lo cree conveniente.

Un último aspecto a considerar con respecto al cambio de prioridad es que consiste en una solicitud de carácter unidireccional. De esta forma, solo se limita a expresar la prioridad deseada en un solo sentido del subflujo implicado. A modo de ejemplo, esto permite la configuración de subflujos que solo deben transmitir información en un solo sentido, entre otras configuraciones posibles.

Cierre de conexión MPTCP

Cuando un host desea cerrar un subflujo asociado a una conexión MPTCP, éste puede recurrir a iniciar un intercambio de señales FIN/ACK al igual que en TCP estándar, sobre el subflujo en cuestión. Este intercambio da lugar al cierre del subflujo, sin finalizar la conexión MPTCP a la cual pertenece.

El cierre de una conexión MPTCP (cierre a nivel de conexión) implica finalizar cada uno de sus subflujos asociados (cierre a nivel de subflujo) para luego culminar con la conexión MPTCP en sí. Se diferencian dos escenarios posibles cuando se desea finalizar una conexión MPTCP:

- No hay más datos por enviar: ocurre cuando un extremo no dispone de más información para enviar y, por lo tanto, procede a notificarle al otro extremo que desea finalizar la conexión.
- Cierre de conexión abrupto: consiste en que un extremo le notifique al otro que ya no se aceptará más información entrante, y que procederá a finalizar la conexión de forma apresurada.

El cierre de conexión MPTCP cuando no hay más datos por enviar se realiza empleando la opción *Data Sequence Signal* (DSS), enviando una señal de *data_fin* a través de las *flags* propias de la opción. Cuando un extremo recibe un segmento que tiene la señalización recién descrita, se procede a verificar si todos los datos fueron trasmitidos correctamente o si se requiere realizar alguna retransmisión.

Una vez comprobada la correcta recepción de toda la información enviada, el extremo responde con un segmento con opción *Data Sequence Signal* (DSS) reconociendo a nivel de conexión la señal de *data_fin* que recibió anteriormente. A continuación, se procede a cerrar la conexión MPTCP.

La Figura 2.14 muestra el cierre de conexión MPTCP a partir de la opción *Data Sequence Signal* (DSS).

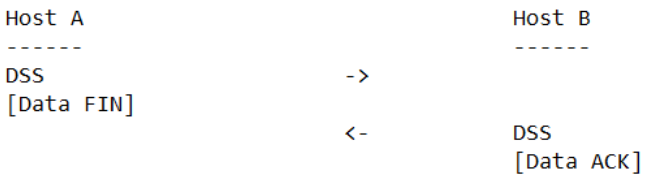

Figura 2.14: Cierre de conexión MPTCP utilizando opción DSS. Adaptado de (Ford y cols., 2020).

Por otro lado, el cierre de conexión MPTCP de forma abrupta emplea la opción *Fast Close* (MP_FASTCLOSE). Este tipo de cierre puede emplearse junto con una señal de ACK o de RST, implicando distintos comportamientos en cada caso:

- Cierre abrupto con señal ACK: en este escenario un extremo envía una señal ACK junto con la opción *Fast Close* (MP_FASTCLOSE) a través de un subflujo, incluyendo la clave que fue generada por el otro extremo durante el inicio de conexión. Además, procede a enviar señales de RST al resto de los subflujos existentes. Finalmente, el extremo queda a la espera de que el otro extremo envíe señales de RST a través los subflujos, para así finalizar los subflujos y la conexión MPTCP.
- Cierre abrupto con señal RST: en este escenario un extremo envía una señal RST junto con la opción *Fast Close* (MP_FASTCLOSE) a través de todos los subflujos, incluyendo la clave que fue generada por el otro extremo durante el inicio de conexión. Finalmente, el extremo finaliza inmediatamente los subflujos y la conexión MPTCP.

En ambos casos, el extremo que reciba una señalización de cierre de conexión abrupto procederá a enviar señales de RST a través de los subflujos. A continuación, finalizará la conexión MPTCP. La Figura 2.15 expone ambas variantes del cierre de conexión MPTCP utilizando la opción *Fast Close* (MP_FASTCLOSE).

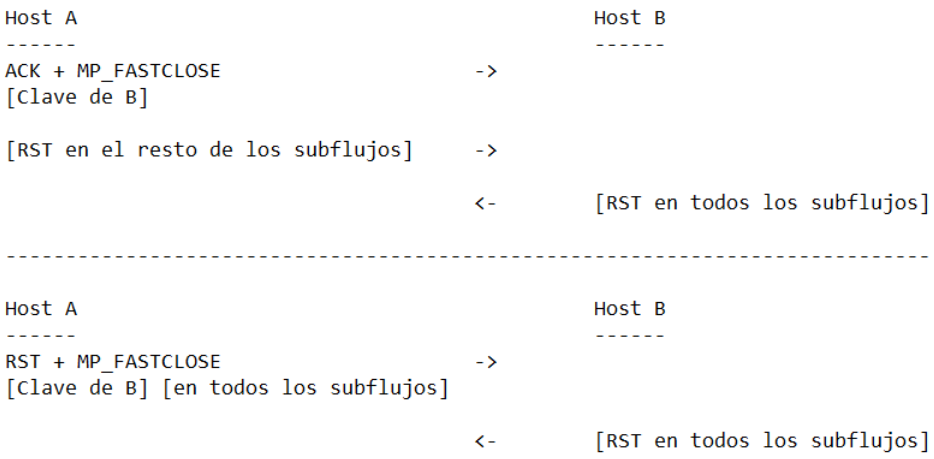

Figura 2.15: Cierre de conexión MPTCP utilizando opción MP_FASTCLOSE. Adaptado de (Ford y cols., 2020).

2.3.6. Implementación de Multi-Path TCP

Existen varias implementaciones del protocolo Multi-Path TCP para distintos sistemas operativos que abarcan desde computadoras de escritorio a dispositivos móviles. En particular, para el desarrollo del estudio que se aborda en este proyecto se emplea la implementación correspondiente al kernel de Linux.

La implementación de MPTCP para el kernel de Linux se origina partir de un grupo de trabajo perteneciente a la Universidad Católica de Lovaina de Bélgica, en el año 2009 (UCLouvain, 2022). Actualmente, esta implementación se encuentra en desarrollo por una comunidad de código libre (Baerts y cols., 2022). Para versiones recientes del kernel de Linux, la versión vigente de la implementación del protocolo MPTCP se denomina como MPTCPv1.

Corresponde destacar que la versión MPTCPv1 se basa en el RFC 8684 (Ford y cols., 2020), siendo éste el documento más reciente que especifica el protocolo MPTCP. A su vez, la versión MPTCPv1 representa varias liberaciones de la implementación del protocolo, donde cada una agrega nuevas funcionalidades y corrige errores.

Configuración del protocolo MPTCP

La implementación del protocolo MPTCP en su versión MPTCPv1 permite la configuración de varios parámetros esenciales para el empleo del protocolo.

Un tipo de parámetro a considerar corresponde a las variables del protocolo MPTCP que se encuentran en la ruta */proc/sys/net/mptcp/* (The kernel development community, 2023). Entre estas variables se destacan las siguientes:

- *enabled*: si su valor se configura en uno, entonces se permite la creación de *sockets* MPTCP. Configurar su valor en cero deshabilita la creación de *sockets* MPTCP.
- *checksum_enabled*: si su valor se configura en uno, entonces se establece que las opciones *Data Sequence Signal* (DSS) utilicen suma de comprobación (*checksum*). Configurar su valor en cero deshabilita la suma de comprobación.
- *allow_join_initial_addr_port*: si su valor se configura en uno, entonces se permite que los extremos puedan enviar solicitudes de establecimiento de nuevo subflujo, a través de la opción *Join Connection* (MP JOIN), a la misma dirección IP y mismo puerto que

fue utilizado para crear el primer subflujo de toda la conexión MPTCP. Configurar su valor en cero prohíbe la creación de subflujos utilizando las direcciones IP y los números de puertos del primer subflujo creado.

- *pm_type*: si su valor se configura en cero, entonces se establece que se utilizará el manejador de rutas provisto por el kernel para las conexiones MPTCP. En cambio, si su valor se configura en uno, entonces se establece que se utilizará un manejador de rutas provisto por el usuario.

Un manejador de rutas (*path manager*) consiste en un programa encargado de gestionar las conexiones MPTCP. Dada una conexión MPTCP, un manejador de rutas se encarga de invocar las distintas operaciones MPTCP en base a un criterio establecido.

Por defecto, el kernel de Linux dispone de un manejador de rutas perteneciente a la implementación de MPTCP (Baerts y cols., 2022). De esta forma, este manejador de rutas se encarga de gestionar las conexiones MPTCP en base a un comportamiento especificado por los implementadores del protocolo.

Sin embargo, si un usuario desea imponer sus propias convenciones en relación a cómo deben operar las conexiones MPTCP, entonces éste puede recurrir a definir un manejador de rutas a nivel de usuario (Martineau y Othman, 2020).

Finalmente, la configuración de la variable *pm_type* determina cuál de los dos tipos de manejador de rutas debe ser utilizado para gestionar las operaciones MPTCP.

El protocolo MPTCP también permite la configuración de parámetros que son utilizados por los manejadores de rutas para poder operar. Estos parámetros pueden ser especificados a través del comando *ip mptcp* (Canonical Ltd, 2019a).

En esencia, el comando *ip mptcp* permite realizar tres tipos de acciones:

- Definir endpoints: en este contexto, definir un *endpoint* consiste en especificar una dirección IP que será utilizada y/o anunciada para el establecimiento de nuevos subflujos. En cada caso el *endpoint* debe estar conformado por una dirección IP local al host, pudiendo también especificar un número de puerto asociado. En adición, cada *endpoint* debe tener al menos una *flag* asociada.

  Existen tres tipos de *flag* que son aplicables a un *endpoint*, implicando distintos comportamientos:

  - *signal*: el *endpoint* será anunciado al otro extremo a través de la opción *Add Address* (ADD_ADDR).
  - *subflow*: si la configuración de límites lo permite, el *endpoint* será utilizado como dirección de origen para establecer un nuevo subflujo, una vez que la conexión MPTCP esté establecida. Esta acción se realiza a través de la opción *Join Connection* (MP_JOIN).
  - *backup*: si el *endpoint* es de tipo *signal*, entonces éste será anunciado como una dirección de *backup*. Por otro lado, si el *endpoint* es de tipo *subflow*, entonces éste será utilizado para establecer un subflujo de *backup*.

- Definir límites: los límites corresponden a restricciones que son aplicables respecto al establecimiento de subflujos adicionales. Todo subflujo que se establezca luego del primer subflujo de la conexión MPTCP se considera como adicional.

  De esta forma, a través de los límites se puede definir la cantidad máxima de subflujos adicionales que pueden ser establecidos en una conexión MPTCP.

  A su vez, los límites también permiten especificar la cantidad máxima de opciones *Add Address* (ADD_ADDR) que un extremo puede aceptar del otro en una conexión MPTCP. Esto se traduce a especificar la cantidad máxima de direcciones disponibles que un extremo local puede llegar a conocer de un extremo remoto.

- Invocar un monitor: consiste en invocar un monitor en la consola de comandos de Linux que se encarga de dar seguimiento a las conexiones MPTCP existentes. Específicamente, el monitor registra la ejecución de las distintas operaciones MPTCP, para cada conexión MPTCP establecida.

En términos generales, se espera que un manejador de rutas a nivel de kernel respete la definición de los *endpoints* y de los límites especificados. Por otro lado, el comportamiento de un manejador de rutas a nivel de usuario queda completamente a cargo de su implementador, pudiendo optar por ignorar estos últimos parámetros según su criterio.

2.4. Multi-Path TCP Deamon

Multi-Path TCP Deamon (MPTCPD) (Othman y cols., 2023) es un programa diseñado para sistemas operativos basados en Linux, cuya finalidad consiste en permitir operaciones de manejo de rutas a nivel de espacio de usuario. Este programa se caracteriza por operar como demonio una vez que se inicializa, implicando que su ejecución permanezca activa en segundo plano sin requerir de la intervención de un usuario.

El desarrollo del programa MPTCPD está a cargo de una comunidad de código abierto que utiliza como punto de partida la implementación del protocolo Multi-Path TCP (Othman y cols., 2023). Si bien la primera liberación de MPTCPD en su versión alpha corresponde al año 2019, en la actualidad este programa aún continua en constante desarrollo.

En esencia, MPTCPD provee los medios para consumir funciones específicamente expuestas por el protocolo Multi-Path TCP, al mismo tiempo que permite monitorear eventos de operaciones asociadas a este protocolo. En particular, estos servicios ofrecidos por MPTCPD se emplean en la ejecución de *plugins* que finalmente dan lugar al manejo de rutas a nivel de usuario.

2.4.1. Plugins

Un *plugin* es un programa con una estructura predefinida que es desarrollado por un usuario para luego poder ser comprendido y ejecutado por MPTCPD.

A partir de un *plugin*, se definen funciones que se ejecutan como consecuencia de que se presente un evento asociado al protocolo Multi-Path TCP. En otras palabras, un *plugin* define qué acciones deben ser realizadas cuando ocurre un suceso relacionado a alguna conexión Multi-Path TCP existente.

Los eventos que un *plugin* es capaz de reconocer y de notificarle a un host local respecto a las conexiones Multi-Path TCP son:

- Nueva conexión (*New Connection*): indica que se está intentando establecer una conexión MPTCP.
- Conexión establecida (*Connection established*): indica que una conexión MPTCP fue establecida.
- Conexión cerrada (*Connection closed*): indica que una conexión MPTCP fue finalizada.
- Nueva dirección (*New address*): indica que se recibió un anuncio de una nueva dirección disponible del host remoto, en una conexión MPTCP.
- Dirección removida (*Address removed*): indica que se recibió una notificación por parte del host remoto para eliminar una de sus direcciones que previamente estaba disponible, en una conexión MPTCP.

- Nuevo subflujo (*New subflow*): indica se estableció un nuevo subflujo, en una conexión MPTCP.
- Subflujo cerrado (*Subflow closed*): indica que se finalizó un subflujo, en una conexión MPTCP.
- Prioridad de subflujo (*Subflow priority*): indica que el host fue notificado de un cambio de prioridad en un subflujo, en una conexión MPTCP.

Los eventos representan diferentes instantes de una conexión Multi-Path TCP. Cuando ocurren múltiples conexiones Multi-Path TCP, el *plugin* permite determinar a cuál conexión pertenece cada evento reconocido, utilizando los valores de *token* que identifican de forma única a cada conexión.

Cuando se detecta un evento de una conexión MPTCP, un *plugin* es capaz de invocar acciones como respuesta. En particular, estas acciones pueden involucrar la ejecución de determinadas operaciones que son expuestas por el propio protocolo Multi-Path TCP. La ejecución de estas operaciones puede impactar a nivel de las conexiones MPTCP, o a nivel de las variables del manejador de rutas.

Entre las operaciones disponibles a nivel de las conexiones MPTCP se encuentran: anunciar una dirección disponible, remover una dirección, establecer un nuevo subflujo, remover un subflujo, y cambiar la prioridad de un subflujo. Por otro lado, las operaciones a nivel de las variables del manejador de rutas corresponden a las siguientes: ingresar un nuevo *endpoint*, remover un *endpoint* existente, obtener los *endpoints* existentes, definir los valores de límites, y obtener los valores de límites.

La combinación de detectar eventos del protocolo Multi-Path TCP junto con la capacidad de definir qué acciones deben ser ejecutadas en respuesta a éstos, permite que un *plugin* funcione como un manejador de rutas a nivel de usuario.

Cuando se ejecuta MPTCPD en un host, éste se encarga de inicializar y ejecutar en segundo plano el *plugin* que le sea proporcionado. De esta forma, toda conexión Multi-Path TCP que ocurra en el host será detectada y gestionada por el *plugin*.

2.5. Redes de centros de datos

Un centro de datos (*data center*) es una instalación física que una organización utiliza, entre otros fines, para almacenar aplicaciones e información crítica. El diseño de un centro de datos se corresponde con una red de recursos informáticos y de almacenamiento que permiten ofrecer diversos servicios. Entre los componentes de un centro de datos que forman parte de su diseño se destacan los routers, los switches, los sistemas de almacenamiento y los servidores (Cisco, 2022).

En esta sección se aborda un diseño de centro de datos que corresponde a la topología de red fat-tree. En particular, esta topología se caracteriza por ofrecer múltiples rutas físicas que comunican cualquier par de hosts conectados en su frontera. Esta característica resulta interesante para ser estudiada en relación al protocolo Multi-Path TCP que se encarga de establecer múltiples rutas a nivel lógico entre hosts.

2.5.1. Topología de red fat-tree

Una topología fat-tree se define a partir de la cantidad de *points of delivery* (pods) que dispone. Sea k el parámetro que representa la cantidad de pods en una topología de estas características, entonces estos pods se numeran de izquierda a derecha comenzando por el *Pod-0* hasta el *Pod-(k-1)*. El parámetro k inicialmente determina la cantidad de pods para

luego condicionar el resto de las estructuras que componen la topología (Medhi y Ramasamy, 2017).

De esta forma, una topología fat-tree con k pods dispone de switches que se organizan en tres capas: *edge switches*, *aggregation switches* y *core switches*. Cada pod consiste en una agrupación de k switches (cada uno con k puertos) que se organizan en dos capas de $k/2$ switches, una capa corresponde a los *edge switches* y la otra a los *aggregation switches*. Cada *edge switch* se conecta con $k/2$ *aggregation switches*. La Figura 2.16 muestra la estructura interna de un pod para $k = 4$.

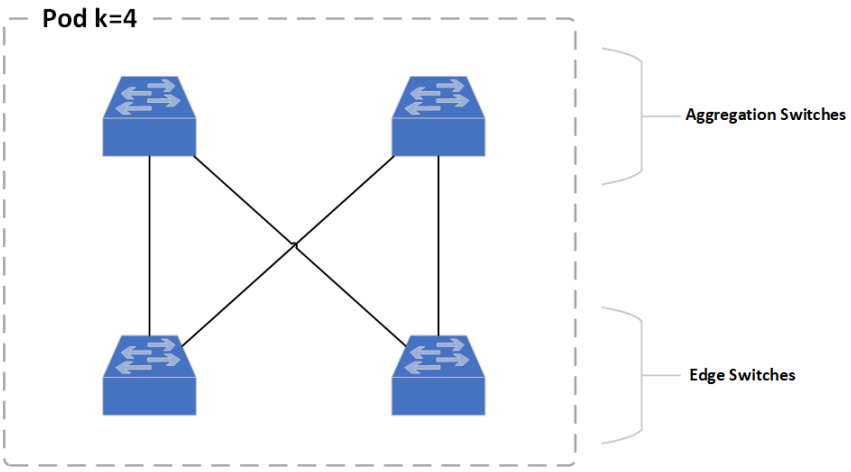

Figura 2.16: Estructura de un pod con $k = 4$.

Los puertos restantes de cada *edge switch*, es decir, aquellos que no se emplean para conectarse a un *aggregation switch*, se destinan para conectarse a hosts físicos o *racks* de servidores.

Finalmente, cada *aggregation switch* además de conectarse con $k/2$ *edge switches* dentro del pod, también se conectan con $k/2$ *core switches*. La Figura 2.17 expone una topología fat-tree completa para $k = 4$.

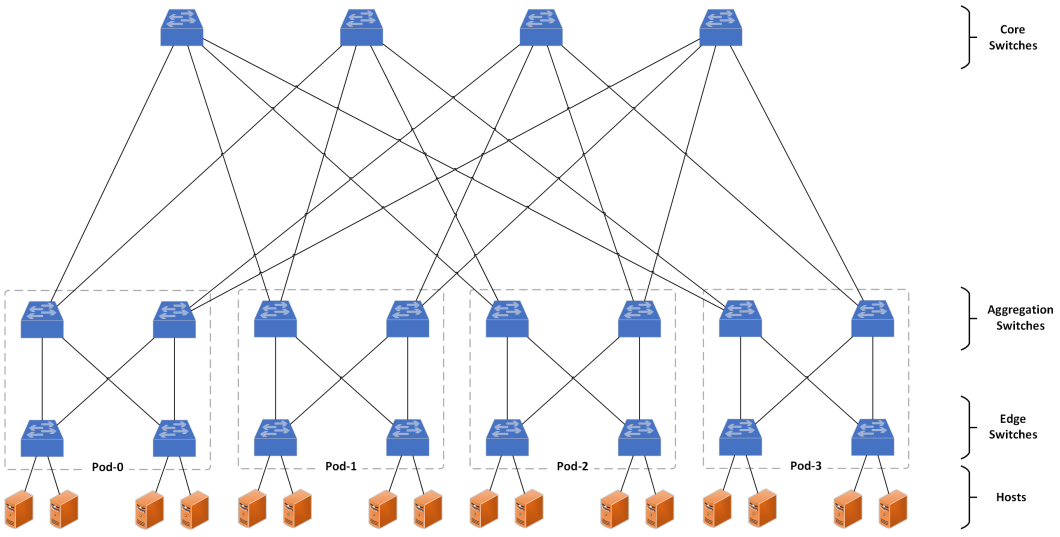

Figura 2.17: Topología fat-tree con $k = 4$.

A partir del parámetro k es posible determinar aspectos cuantitativos sobre la topología fat-tree que se esté considerando. La Figura 2.18 expone cómo influye el parámetro k sobre la estructura de la topología.

Cantidad de pods	**k**
Cantidad de core switches	$(k/2)^2$
Cantidad de aggregation switches	$k^2/2$
Cantidad de edge switches	$k^2/2$
Cantidad total de switches	$5k^2/4$
Cantidad de enlaces	$k^3/2$
Cantidad de hosts soportados	$k^3/4$

Figura 2.18: Aspectos cuantitativos determinados por el parámetro k (Medhi y Ramasamy, 2017).

Corresponde destacar que los switches que conforman la topología son similares entre sí. Esto implica que todos disponen de la misma cantidad de puertos que, a su vez, proveen típicamente la misma velocidad.

Otra característica importante a resaltar, consiste en la existencia de múltiples rutas entre cualquier par de hosts que no se encuentran conectados a un mismo *edge switch*. De forma general, para una topología fat-tree con k *pods* se cumple que:

- Existe una única ruta entre dos hosts conectados a un mismo pod que a su vez están conectados a un mismo *edge switch*.
- Existen $k/2$ rutas entre dos hosts conectados a un mismo pod que a su vez están conectados a distintos *edge switches*.
- Existen $k^2/4$ rutas entre dos hosts conectados a distintos pods.

3. Enfoque de estudio

Multi-Path TCP es un protocolo que opera nivel de capa de transporte en Internet, definido como una extensión del protocolo TCP. En estas condiciones, Multi-Path TCP provee los mismos servicios ofrecidos por TCP, añadiendo la posibilidad de emplear múltiples rutas de forma simultánea. En consecuencia, esta característica adicional supone mejoras para las conexiones de transporte tras aumentar su capacidad de resiliencia (*resilience*) y la utilización de anchos de banda (*throughput*).

Como se especifica en la sección 2.5, una infraestructura de red con topología de tipo fat-tree ofrece múltiples caminos que permiten comunicar cualquiera de sus hosts entre sí. Esta propiedad dispone del potencial de ser aprovechada por el protocolo MPTCP para permitir el establecimiento de múltiples subflujos, a través de caminos físicamente disjuntos. En adición, un centro de datos con topología de red fat-tree podría verse beneficiado a través de los servicios que obtenga de MPTCP.

El objetivo del estudio que se expone radica en experimentar con el protocolo Multi-Path TCP en el contexto de una topología de red fat-tree, con la finalidad de determinar su funcionamiento y eficacia frente a distintos escenarios propuestos (Raiciu y cols., 2011). Se establece comprobar si efectivamente Multi-Path TCP supone mejoras sustanciales en relación al protocolo TCP.

Adicionalmente, se desarrolla un programa manejador de rutas que se encargue de gestionar el comportamiento de las conexiones Multi-Path TCP. Este programa efectúa diversas políticas establecidas por el usuario sobre las conexiones asociadas a este protocolo.

En este capítulo se presenta el ambiente de trabajo definido para abarcar el estudio propuesto. Posteriormente, se detallan las implementaciones que fueron realizadas.

3.1. Ambiente de trabajo

Para el estudio del protocolo Multi-Path TCP se establece utilizar la implementación disponible para el kernel de Linux que se expone en la subsección 2.3.6. Se considera que esta implementación se corresponde con una de las más recientes, siendo que aún se encuentra en constante desarrollo. En particular, se emplea la implementación de Multi-Path TCP provista por el kernel de Linux en su versión 5.19.5, utilizando la distribución Ubuntu con versión 22.04.1 LTS (Canonical Ltd, 2023).

En relación a la definición del ambiente de trabajo, se opta por recurrir al empleo de emulaciones. De forma general, se entiende que una emulación corresponde a utilizar un programa de computadora que permite que un sistema anfitrión logre reproducir programas que fueron codificados con la intención de ser ejecutados en otro sistema de características diferentes (Wikipedia, 2022). Específicamente, al no disponer de una infraestructura de red real de tipo fat-tree se establece utilizar la herramienta Mininet (Mininet Project Contributors, 2022), en su versión 2.3.1, con el fin de emularla.

Mininet es un programa que tiene como objetivo la generación de redes virtuales de manera emulada dentro de un ordenador. Se caracteriza por utilizar una funcionalidad provista por el kernel de Linux denominada como *namespaces* (Canonical Ltd, 2019b) que le permite generar una topología de red virtual, donde cada uno de sus componentes se modela como una instancia del propio kernel de Linux del sistema anfitrión.

La definición de una topología a través de Mininet implica que un usuario establezca la cantidad de hosts, switches y routers que serán considerados. En adición, el usuario también deberá especificar cómo se interconectan estos componentes, detallando la configuración de sus interfaces y direcciones IP, entre otros aspectos relevantes. En la práctica, las características recién mencionadas deberán ser codificadas en un archivo Python, donde el usuario deberá utilizar la API provista por Mininet.

Teniendo en cuenta estos aspectos expuestos, el ambiente de trabajo se define como una máquina virtual encargada de ejecutar la distribución de Ubuntu previamente mencionada.

A su vez, dentro de esta máquina virtual se ejecuta la herramienta Mininet que emula la topología fat-tree deseada.

La configuración y gestión de la máquina virtual se realiza por medio de la herramienta VMware Workstation Pro (VMware, 2023), en su versión 16.2.3. Se destaca que la utilización de máquinas virtuales permite la definición de un ambiente controlado sobre el cual se pueden realizar copias de respaldo con facilidad. Estas copias podrán luego ser empleadas en caso de que ocurran imprevistos.

3.2. Implementaciones

3.2.1. Implementación de programas cliente y servidor MPTCP

Para evaluar el funcionamiento de la implementación del protocolo Multi-Path TCP que radica en el kernel de Linux, se desarrollaron dos programas sencillos que son capaces de comunicarse entre sí utilizando este protocolo. Consisten en un programa cliente y un programa servidor que se ejecutan en hosts diferentes para intercambiar datos de tipo texto.

La siguiente secuencia especifica el funcionamiento de los programas cliente y servidor MPTCP:

1. Sean dos hosts con conectividad entre sí, se comienza por ejecutar el programa servidor en uno de ellos. Luego, se ejecuta el programa cliente en el otro host restante. Respectivamente, ambas ejecuciones dan lugar a procesos de tipo servidor y cliente, donde el proceso cliente se conecta con el proceso servidor a través de MPTCP.

2. Una vez que ambos procesos establecen una conexión MPTCP, el proceso cliente procede a enviar el texto "Mensaje de prueba" al proceso servidor.

3. Cuando el proceso servidor recibe el texto "Mensaje de prueba", lo toma y lo convierte en "MENSAJE DE PRUEBA" (pasa cada una de sus letras a mayúsculas) para luego enviárselo al proceso cliente como respuesta.

4. Se repite el ciclo de intercambio de mensajes de texto descrito en los puntos 2 y 3 en un total de 1000 veces.

5. Concluidos todos los intercambios, el proceso cliente cierra la conexión con el proceso servidor. El proceso servidor entonces cierra conexión con el proceso cliente.

Corresponde destacar que el empleo de estos programas no queda restringido únicamente al ambiente de trabajo definido. En otras palabras, estos programas pueden ser ejecutados a partir de cualquier par de hosts que dispongan conectividad entre sí.

3.2.2. Implementación de topología fat-tree

Para la emulación de la topología fat-tree que forma parte del ambiente de trabajo especificado en el sección 3.1, se codifica un archivo Python que genera esta topología. A continuación, se describen los pasos seguidos para la construcción de este archivo:

1. Confeccionar la estructura de la topología fat-tree. Para ello, se utilizan 3 parámetros de entrada que especifican: el valor del parámetro k de la topología fat-tree, la cantidad de hosts por cada *edge switch*, y el número de interfaces por cada host.

2. Configurar los routers para permitir la conectividad entre los distintos hosts empleando el conjunto de protocolos de FRRouting (FRRouting Project Contributors, 2022). Específicamente, se utiliza el demonio BGPD que ejecuta el protocolo BGP, junto con el manejador de rutas Zebra. En este punto, se establecen las direcciones IP asociadas a las interfaces de cada componente.

3. Se establecen los parámetros correspondientes al protocolo Multi-Path TCP en cada host, según corresponda.

4. Se comienza a realizar una captura de paquetes enviados por las distintas interfaces de cada componente. La herramienta empleada para este cometido se denomina Tcpdump (The Tcpdump Group, 2023).

5. Se utiliza la herramienta Wondershaper (Mulhollon, 2004) con el objetivo de limitar la velocidad de carga y descarga de las interfaces de los componentes asociados a la topología.

Resulta importante mencionar que la implementación desarrollada se basa en otra ya existente (Alberro, Castro, y Grampin, 2022) que cumple con varias de estas características descritas.

3.2.3. Implementación de manejador de rutas

Un manejador de rutas consiste en un programa encargado de gestionar la operativa de las conexiones Multi-Path TCP. Su gestión involucra principalmente la creación, eliminación y cambio de prioridad de los subflujos, así como también el anuncio y remoción de direcciones disponibles.

Por defecto, la implementación de Multi-Path TCP dispone de su propio manejador de rutas que opera a nivel de kernel del sistema operativo. Sin embargo, el proceder de este manejador de rutas es de carácter general, siendo que no responde a los intereses específicos del usuario que utiliza este protocolo. Es por esto que existe la posibilidad de diseñar manejadores de rutas que obedezcan distintas convenciones impuestas por el usuario, es decir, un manejador de rutas que opera a nivel de usuario.

La implementación del manejador de rutas a nivel de usuario se construye sobre el programa Multi-Path TCP Deamon expuesto en la sección 2.4. Específicamente, las operaciones de manejo de rutas se definen en los *plugins* asociados a Multi-Path TCP Deamon.

De esta forma, la implementación del manejador de rutas a nivel de usuario involucra dos aspectos:

1. Implementar un *plugin* que disponga de todas las operaciones necesarias para el manejo de las conexiones MPTCP. Este *plugin* luego opera sobre Multi-Path TCP Deamon.

2. Implementar una interfaz que permita la interacción del usuario con el *plugin* que opera desde Multi-Path TCP Deamon, a modo de poder solicitar distintas operaciones disponibles. A este programa de tipo interfaz se le denomina como *consola de manejo*.

En términos generales, la utilización del manejador de rutas a nivel de usuario se describe a partir de los siguientes pasos:

1. Dado el host sobre el cual se desea utilizar el manejador de rutas a nivel de usuario, se debe configurar su variable MPTCP denotada como *pm_type* en valor uno. Como se especifica en la subsección 2.3.6, esta configuración indica que se utilizará un manejador de rutas que opere a nivel de usuario.

2. Se inicializa el programa Multi-Path TCP Deamon que permanece ejecutando en segundo plano. Este programa ejecuta su *plugin* asignado y, por lo tanto, se encarga de propagar las solicitudes que surgen del *plugin* hacia el kernel del sistema operativo.

3. Se inicializa la *consola de manejo*. Ésta se comunica con el *plugin* y expone sus funciones al usuario. La selección de una operación en la *consola de manejo* se propaga al *plugin*, luego Multi-Path TCP Deamon se encarga de propagarla hacia el kernel del sistema operativo. Finalmente, el kernel procede a efectuar la operación solicitada sobre la respectiva conexión MPTCP del host.

4. Toda conexión MPTCP que se inicialice a partir de este punto puede ser visualizada y gestionada desde la *consola de manejo*.

Operaciones del manejador de rutas

Cuando se asume utilizar un manejador de rutas a nivel de usuario, se acepta la responsabilidad de gestionar el funcionamiento completo de las conexiones MPTCP. Esto implica no solo modelar las operaciones que son de especial interés para el usuario, sino que también se requiere implementar todas las operaciones necesarias para que las conexiones operen correctamente.

En relación a estos aspectos, para la implementación del manejador de rutas se definen los siguientes tipos de operaciones:

- Operaciones automáticas: son operaciones que se ejecutan durante las conexiones MPTCP en base a la ocurrencia de eventos detectables por el *plugin*. Estos eventos corresponden a los mencionados en la subsección 2.4.1.
  En esencia, las operaciones automáticas son convenciones que se adoptan para permitir que las conexiones MPTCP funcionen correctamente.

- Operaciones manuales: son operaciones disponibles para ser solicitadas por el usuario a través de la *consola de manejo*. Permiten gestionar conexiones MPTCP en base al criterio del usuario.

Corresponde mencionar que para la operativa de un manejador de rutas a nivel de usuario se definen estructuras de datos que registran aspectos relevantes de las conexiones MPTCP existentes en el host. Las operaciones definidas utilizan y modifican los datos registrados en estas estructuras para poder operar.

Entre las operaciones de tipo automáticas se definen las siguientes:

- Anuncio de direcciones por parte del servidor: cuando se inicialice una conexión MPTCP, el host servidor procede a anunciar todas sus direcciones disponibles al host cliente.

- Establecimiento de nuevos subflujos por parte del cliente: cuando un host cliente recibe anuncios de direcciones disponibles durante el inicio de una conexión MPTCP, éste procede a establecer nuevos subflujos con el host servidor. Para esta tarea, el host cliente utiliza las direcciones que le hayan sido anunciadas, y procede a establecer tantos subflujos como los límites configurados le permitan.

- Evento de remoción de dirección: cuando ocurre un evento de remoción de direcciones, cada host debe proceder a cerrar todos los subflujos que tenga asociados a cada dirección removida.

- Cantidad de subflujos que excede el límite: frente a un evento de establecimiento de un nuevo subflujo, si un host detecta que se sobrepasan los límites de subflujos permitidos, debe proceder a cerrar el subflujo implicado.

Por otro lado, las operaciones manuales que se encuentran disponibles para ser solicitadas por el usuario corresponden a las siguientes:

- Visualizar información de conexión: esta operación imprime la información referente a la conexión MPTCP que se esté tratando.

- Creación de un nuevo subflujo: esta operación permite que el usuario establezca un nuevo subflujo asociado a la conexión, desde un host cliente.

- Cierre de un subflujo existente: esta operación permite que el usuario cierre un subflujo asociado a la conexión.

- Anuncio de una dirección disponible: esta operación permite que el usuario anuncie una dirección disponible, desde un host servidor.
- Remoción de una dirección no disponible: esta operación permite remover una dirección que haya sido previamente anunciada como disponible.
- Cambio de prioridad de un subflujo: esta operación permite cambiar la prioridad de los subflujos asociados a la conexión.
- Cambio de política de gestión de subflujos: esta operación permite alternar entre tres tipos de políticas de gestión de subflujos, definidas cada una a nivel de usuario. Estas políticas son manual, round-robin y vector de prioridades.

La implementación de un manejador de rutas a nivel de usuario permite definir funcionalidades complejas a partir de las operaciones disponibles en el protocolo MPTCP. La definición de distintas políticas de gestión de subflujos es un ejemplo de esto.

Políticas de gestión de subflujos

Como se detalla en la subsección 2.3.5, Multi-Path TCP dispone de una funcionalidad que permite redefinir la prioridad de los subflujos ya establecidos en las conexiones MPTCP. Se recuerda que el cambio de prioridad es de carácter unidireccional, donde la prioridad puede ser de tipo *backup* o *no backup*.

Utilizando la funcionalidad de cambio de prioridad es posible implementar políticas de gestión de subflujos que representen los intereses de un usuario. Se definen e implementan tres tipos de políticas de gestión de subflujos que se incluyen en el manejador de rutas a nivel de usuario:

- Manual: corresponde a la política más simple y definida por defecto al establecerse una conexión MPTCP. Cada subflujo que se establece comienza siendo de tipo *no backup*, y luego el usuario puede optar por redefinir la prioridad de cada subflujo existente en la conexión. Cada cambio de prioridad debe ser realizado de forma individual sobre cada subflujo.
- Round-robin (con turnos de *t segundos*): cuando se asigna una política de tipo round-robin, se definen turnos de *t segundos* donde cada subflujo pasa a estar en *no backup* mientras que el resto permanece en *backup*. La asignación de turnos es secuencial, desde el primero hasta el último subflujo en haber sido establecido. Al finalizar el turno del último subflujo, se repite la secuencia desde el comienzo.
- Vector de prioridades: en esta política cada subflujo dispone de una prioridad numérica asignada que corresponde a un número entero mayor que cero. Las prioridades con valores numéricos más bajos implican mayor prioridad.
  Los subflujos que posean la mayor prioridad de todas (el valor numérico más bajo) serán los que permanezcan en *no backup*, mientras que los restantes subflujos se encuentran en *backup*.

Los parámetros que especifican la duración de los turnos del round-robin y los valores del vector de prioridades se registran en un *archivo externo de prioridades* que es obtenido y leído por el *plugin* al momento de inicializarse. En la Figura 3.1 se presenta la estructura específica de este archivo.

La duración de los turnos del round-robin se obtiene directamente del archivo en cuestión, a través de interpretar la línea que comienza con la expresión "*round_robin*". Sin embargo, los valores del vector de prioridades se definen a partir de considerar una convención de mayor coincidencia:

1. Coincidencia por IPs local y remota: dado un subflujo de una conexión MPTCP, si su IP local y su IP remota se encuentran especificadas en alguna línea del *archivo externo de prioridades*, luego de la expresión "*subflow*", entonces se utiliza la prioridad denotada en dicha línea del archivo. En caso contrario, se procede al siguiente paso.

2. Coincidencia por IP local: si no hay coincidencia por IPs local y remota, se procede a evaluar solamente a partir de la IP local. Si en alguna línea del *archivo externo de prioridades* luego de la expresión "*interface*" se encuentra la IP local asociada al subflujo en cuestión, entonces se procede a utilizar la prioridad denotada en dicha línea. En caso contrario, se procede al siguiente y último paso.

3. Prioridad por defecto: al no haber coincidencias de ningún tipo al utilizar los valores de IP, se le asigna al subflujo la prioridad del *archivo externo de prioridades* que está denotada en la línea que comienza con la expresión "*default_vector*".

Este método que emplea la mayor coincidencia permite que un usuario utilice la política de vector de prioridades desde un enfoque general hasta uno más específico.

```
round_robin     <cantidad_de_segundos>

default_vector  <prioridad_numérica>

interface       <ip_local> <prioridad_numérica>
interface       <ip_local> <prioridad_numérica>
.               .
.               .
.               .
interface       <ip_local> <prioridad_numérica>


subflow         <ip_local> <ip_remota> <prioridad_numérica>
subflow         <ip_local> <ip_remota> <prioridad_numérica>
.               .
.               .
.               .
subflow         <ip_local> <ip_remota> <prioridad_numérica>
```

Figura 3.1: Estructura del archivo externo de prioridades.

Limitantes del manejador de rutas

A continuación, se detallan aspectos relacionados con los programas involucrados en la implementación y ejecución del manejador de rutas. Específicamente, se mencionan las limitaciones que éstos implican:

- La implementación de Multi-Path TCP Deamon aún se encuentra en desarrollo en su versión alpha 0.11, por lo que es común que presente errores de ejecución. En particular, la ejecución de operaciones del manejador de rutas a nivel de usuario con frecuencia falla y congestiona el programa. La solución práctica a este problema consiste en detener y reiniciar la ejecución de Multi-Path TCP Deamon.

- La implementación del protocolo Multi-Path TCP aún se encuentra en desarrollo. Esto puede suponer problemas en la ejecución del programa Multi-Path TCP Deamon, puesto que este último utiliza una API provista por la implementación de Multi-Path TCP.

- No es posible establecer subflujos de tipo *backup* de forma inicial, debido a una limitación de los programas empleados. Sin embargo, es posible cambiar la prioridad de los subflujos que ya se encuentren establecidos, pudiendo optar por prioridades de tipo *backup* o no *backup*.
- Durante el desarrollo del manejador de rutas a nivel de usuario se realizaron cambios menores en la implementación de Multi-Path TCP Deamon para permitir el pasaje de nuevos parámetros hacia los *plugins*. Consiste en una limitación que fue solucionada.

4. Experimentación

En este capítulo se expone la experimentación realizada con respecto a la implementación del protocolo Multi-Path TCP. Esta experimentación se compone a partir casos de prueba y de validación.

Un *caso de validación* se establece con el fin de determinar el correcto funcionamiento de una herramienta o implementación. En este contexto, interesa validar el protocolo MPTCP y el ambiente de trabajo definido. Dada su naturaleza, los casos de validación son los primeros en ser ejecutados.

Por otro lado, un *caso de prueba* se define con la finalidad de exponer el protocolo MPTCP ante distintos escenarios para evaluar su desempeño.

4.1. Herramientas utilizadas

Para la experimentación con el protocolo Multi-Path TCP se recurre a la utilización de herramientas específicas que facilitan la ejecución y análisis de las pruebas realizadas. Estas herramientas corresponden a las siguientes:

- *Mptcp-tools*: se trata de un conjunto de herramientas que provee una utilidad denominada como *use_mptcp*. Esta utilidad permite que un programa que solo opera con TCP de forma nativa pueda utilizar MPTCP, sin implicar cambios en su programación (Abeni, 2020).
- *Ifstat*: consiste en una herramienta que expone las estadísticas de las interfaces de red de un dispositivo. En particular, detalla el ancho de banda entrante y saliente de una interfaz (Kuznetsov, 2015).
- *Wireshark*: se trata de una herramienta que permite la captura y análisis de tráfico de red que circula a través de las distintas interfaces de un dispositivo (Combs y cols., 2022).
- *Iperf 2*: esta herramienta permite el envío de información entre dispositivos de red con la finalidad de que el usuario pueda realizar mediciones sobre el tráfico generado (McMahon, 2021).

4.2. Ambiente de ejecución

Sistema de cómputo utilizado

La Tabla 4.1 especifica las características del sistema de cómputo anfitrión utilizado.

Tarjeta madre	Gigabyte Technology Co., Ltd. B550M AORUS PRO-P
Procesador	AMD Ryzen 7 5800X
Tarjeta de video	NVIDIA GeForce GTX 1050 Ti
Memoria RAM	Kingston KF3200C16D4/16GX DDR4 (2 unidades conectadas en dual channel)
Memoria secundaria	Samsung SSD 980 PRO 500GB

Tabla 4.1: Especificaciones del sistema de cómputo anfitrión.

Por otro lado, la Tabla 4.2 detalla los recursos que le son asignados a la máquina virtual que conforma el ambiente de trabajo descrito en la sección 3.1.

Procesadores	2 núcleos
Memoria RAM	8 GB
Memoria secundaria	40 GB

Tabla 4.2: Recursos asignados a la máquina virtual del ambiente de trabajo.

Una característica a destacar sobre la emulación con una máquina virtual, corresponde a que ésta ejecuta como un programa en un sistema anfitrión. De esta forma, los recursos del sistema anfitrión se reparten entre la máquina virtual y el resto de los programas que también ejecutan en el sistema anfitrión. En consecuencia, esto puede impactar sobre el desempeño de la máquina virtual considerada, comprometiendo a los programas que ejecutan dentro de ésta.

En otras palabras, una emulación no permite realizar mediciones de desempeño con gran precisión debido a que los recursos de cómputo no son empleados exclusivamente por ésta.

Configuración general para la ejecución de pruebas

A continuación, se detalla la configuración general que se emplea durante la experimentación. Esta configuración aplica para la ejecución de todos los casos de prueba y de validación, a excepción de que explícitamente se afirme lo contrario en la descripción propia del caso de prueba o de validación en cuestión. La configuración general establecida corresponde a la siguiente:

- Topología fat-tree:
  - Las pruebas serán ejecutadas sobre una infraestructura de red virtual de tipo fat-tree con parámetro $k = 4$.
  - Cada *edge switch* se conecta con exactamente dos *hosts* (además de conectarse con otros dos *aggregation switches*).
  - Cada *host* dispone de una interfaz que lo conecta con su *edge switch*. Esta interfaz dispone de varias direcciones IPs asociadas que se emplean para los establecimientos de los distintos subflujos MPTCP.
  - Cada *core switch* se denota con el prefijo *R1* seguido de cuatro dígitos.
  - Cada *aggregation switch* se denota con el prefijo *R2* seguido de cuatro dígitos.
  - Cada *edge switch* se denota con el prefijo *R3* seguido de cuatro dígitos.
  - Cada *host* se denota con el prefijo *H* seguido del valor de un número natural.
  - Las *interfaces* de los componentes se denotan con el prefijo *eth* seguido de un número natural.

  En el apartado Anexo 1 se puede apreciar un esquema con las características de la topología fat-tree a ser utilizada.
- Limitador de velocidad para las interfaces: utilizando la herramienta Wondershaper, se limita la velocidad de todas las interfaces de la topología fat-tree empleada. Se limita su velocidad de *carga* y *descarga* a 1000 Kbits/seg.
- Manejador de rutas MPTCP: se define utilizar el manejador de rutas que opera a nivel de kernel del sistema operativo Ubuntu.
- Conexiones entre hosts: las conexiones TCP y MPTCP que se establezcan entre los distintos hosts de la topología, se realizan a través de la herramienta Iperf. En particular, para la creación de conexiones MPTCP entre hosts se emplea la utilidad use_mptcp en conjunto con la herramienta Iperf.

- Mediciones de tiempo y de volumen de datos: las mediciones realizadas que involucren el tiempo de duración de una conexión o el volumen de datos trasportados, se obtienen a través de la información provista por la herramienta Iperf. Estas mediciones se consideran con respecto al host que ocupa el rol de servidor en la conexión correspondiente.
- Captura y análisis de tráfico: las capturas de tráfico se realizan a través de la herramienta Tcpdump. Posteriormente, para el análisis de estas capturas se emplea la herramienta Wireshark.

4.3. Caso de validación 1: Protocolo MPTCP

Interesa definir un primer caso de validación que ponga a prueba la implementación del protocolo Multi-Path TCP provista por el kernel de Linux. Se prevé evaluar el funcionamiento del protocolo en el contexto de un ambiente emulado.

En adición, también se desea poner a prueba la utilidad use_mptcp para comprobar su eficacia.

Descripción del caso de validación

Para la ejecución de este caso de validación se utiliza una topología de red que consiste en dos máquinas virtuales directamente conectadas entre sí. Cada máquina virtual se emula a través de la herramienta VMware Workstation Pro. A continuación, se detalla la configuración de cada máquina virtual:

- Host servidor con IPs 192.168.1.140 y 192.168.2.140.
- Host cliente con IPs 192.168.1.141 y 192.168.2.141.

Los endpoints del host cliente se definen con la flag *subflow*, mientras que los endpoints del host servidor con la flag *signal*. En ambos hosts, se establece que la cantidad máxima de subflujos por conexión MPTCP sea 4.

Para el establecimiento de conexiones entre los hosts se emplean el *programa cliente MPTCP* y el *programa servidor MPTCP*. Ambos programas se exponen en la subsección 3.2.1. Adicionalmente, se desarrolla un tercer programa denominado como *programa servidor TCP*. Éste consiste en una versión modificada del *programa servidor MPTCP*, cuya única diferencia radica en que utiliza el protocolo TCP en lugar de MPTCP. El *programa servidor TCP* se define con el fin de probar la utilidad use_mptcp.

El procedimiento del caso de validación corresponde al siguiente:

- *Paso 1*: establecer una conexión MPTCP entre el host cliente y el host servidor, ejecutando respectivamente en éstos el *programa cliente MPTCP* y el *programa servidor MPTCP*. Realizar una captura de tráfico de toda la conexión.
- *Paso 2*: establecer una conexión entre el host cliente y el host servidor. Para ello, ejecutar en el host cliente el *programa cliente MPTCP*, mientras que en el host servidor se ejecuta el *programa servidor TCP*. Obtener captura de tráfico para su posterior análisis.
- *Paso 3*: idem *Paso 2*, con la diferencia de que el *programa servidor TCP* se ejecuta junto con la utilidad use_mptcp, en el host servidor. Nuevamente, registrar una captura de tráfico de la conexión.

Resultados obtenidos

En relación a la ejecución del *Paso 1*, se obtuvo una captura de tráfico que comprueba que efectivamente se estableció una conexión MPTCP entre los hosts implicados. Realizando una inspección de la captura de tráfico resultante, es posible describir el comportamiento que lleva a cabo la conexión MPTCP establecida:

1. Se establece la conexión Multi-Path TCP, dando lugar al primer subflujo de la conexión que utiliza las direcciones IP 192.168.1.141 (cliente) y 192.168.1.140 (servidor). Esta operación MPTCP corresponde a *Multipath Capable*.

2. Una vez establecida la conexión Multi-Path TCP, ambos hosts involucrados comienzan a intercambiar información (se recuerda que los dos programas que dan lugar a la conexión MPTCP se encargan de intercambiar datos de texto entre sí). El intercambio de información se realiza principalmente a través de la opción MPTCP denotada como *Data Sequence Signal*.

3. Se establece un nuevo subflujo que utiliza las direcciones IP 192.168.2.141 (cliente) y 192.168.1.140 (servidor). Esta operación MPTCP corresponde a *Join Connection*.

4. A través del subflujo con direcciones IP 192.168.1.140 (servidor) y 192.168.1.141 (cliente), el host servidor le anuncia su dirección IP 192.168.2.140 al host cliente. Esta operación MPTCP se denomina *Add Address*.

5. Se establece un nuevo subflujo que utiliza las direcciones IP 192.168.2.141 (cliente) y 192.168.2.140 (servidor). Nuevamente, esta operación MPTCP corresponde a *Join Connection*.

6. Cuando los programas terminan de intercambiar información entre sí, se envían señales de fin de conexión a través de los tres subflujos establecidos. Cada señalización de fin de conexión se envía a través de la operación MPTCP denominada *Data Sequence Signal*, incluyendo su flag *data_fin* en valor 1.

   Posteriormente, una vez que se efectúan todas las retransmisiones de paquetes requeridas, cada subflujo procede a cerrarse. Finalmente, se cierra la conexión Multi-Path TCP.

La sucesión de eventos previamente descrita se corresponde correctamente con las especificaciones del funcionamiento del protocolo MPTCP expuestas en la subsección 2.3.3.

Por otro lado, a partir de la ejecución del *Paso 2* se registró una conexión TCP en lugar de una conexión Multi-Path TCP. Este resultado se explica a través del siguiente comportamiento observado:

1. El host servidor ejecuta el *programa servidor TCP*. Esto provoca que se inicialice un proceso servidor que permanece escuchando a la espera de una solicitud de conexión TCP.

2. Por otro lado, el host cliente ejecuta el *programa cliente MPTCP*. En consecuencia, se inicializa un proceso cliente que intenta conectarse con el host servidor utilizando Multi-Path TCP.

3. Cuando el proceso servidor recibe una solicitud de conexión Multi-Path TCP, éste procede a responderle al proceso cliente utilizando TCP. Esto se debe a que el proceso servidor está configurado para solo utilizar TCP.

4. El proceso cliente recibe una respuesta TCP para su solicitud Multi-Path TCP. En este punto, el proceso cliente emplea la compatibilidad hacia atrás que el protocolo Multi-Path TCP provee. De esta forma, el proceso cliente se adapta y opta por continuar con el establecimiento de conexión, utilizando TCP en lugar de Multi-Path TCP.

El comportamiento registrado durante la ejecución del *Paso 2* coincide con el resultado esperado.

Finalmente, la ejecución del *Paso 3* genera resultados similares con respecto a los que fueron registrados durante la realización del *Paso 1*. Esto se debe a que cuando se combina la utilidad use_mptcp con el *programa servidor TCP*, la ejecución de éste último da lugar a un proceso servidor que opera con MPTCP. En otras palabras, ejecutar la utilidad use_mptcp junto con el *programa servidor TCP* es equivalente a ejecutar directamente el *programa servidor MPTCP*.

Análisis de resultados

Se comprueba el funcionamiento de la implementación del protocolo Multi-Path TCP que está presente en el kernel de Linux. Específicamente, se evidencia el funcionamiento de sus operaciones *Multipath Capable*, *Join Connection*, *Add Address* y *Data Sequence Signal*.

Si bien se limita la cantidad máxima de subflujos por conexión MPTCP a un total de 4, el kernel de Linux no necesariamente genera la cantidad máxima de subflujos permitida.

Por otro lado, es posible afirmar que la funcionalidad de compatibilidad hacia atrás del protocolo Multi-Path TCP se encuentra implementada y opera según lo esperado. En adición, se comprueba el funcionamiento de la utilidad use_mptcp.

4.4. Caso de validación 2: Ambiente de trabajo

Una vez evidenciado el funcionamiento de la implementación del protocolo Multi-Path TCP, corresponde asegurar la operativa del ambiente de trabajo definido.

Descripción del caso de validación

En la topología fat-tree, se define H1 como host servidor y H9 como host cliente. La configuración propia de cada host corresponde a la siguiente:

- Endpoints del host servidor H1: 200.0.0.100 *signal*, y 200.0.0.101 *signal*.
- Endpoints del host cliente H9: 200.0.8.100 *subflow*, y 200.0.8.101 *subflow* y *backup*.

El procedimiento del caso de validación corresponde al siguiente:

- *Paso 1*: establecer una conexión MPTCP entre el host H1 y el host H9. Realizar una captura de tráfico.
- *Paso 2*: idem *Paso 1*, pero eliminando la dirección IP 200.0.8.100 del host H9 mientras se lleva a cabo la conexión MPTCP.

Resultados obtenidos

La ejecución del *Paso 1* resulta en una conexión MPTCP que se establece de forma exitosa a través de la topología fat-tree. En la conexión MPTCP se inicializan tres subflujos:

- Subflujo 1: 200.0.0.100 $\longleftrightarrow$ 200.0.8.100 de tipo *no backup*.
- Subflujo 2: 200.0.0.101 $\longleftrightarrow$ 200.0.8.100 de tipo *no backup*.
- Subflujo 3: 200.0.0.100 $\longleftrightarrow$ 200.0.8.101 de tipo *backup*.

El flujo de envío de tráfico se concentra en los dos subflujos que son de tipo *no backup*, mientras que el subflujo de tipo *backup* permanece prácticamente inactivo. Específicamente, el subflujo de *backup* se muestra activo en situaciones que involucran principalmente la retransmisión de paquetes.

Por otro lado, la realización del *Paso 2* genera una conexión MPTCP análoga a la descrita en el *Paso 1*. Sin embargo, en esta ocasión se elimina la dirección IP 200.0.8.100 del host H9, lo cual provoca que los dos subflujos de tipo *no backup* dejen de operar. En consecuencia, el subflujo restante de tipo *backup* comienza a transmitir los paquetes de información.

En la Figura 4.1 se puede contemplar el momento en que el subflujo de tipo *backup* comienza a trasmitir información. Tal instante ocurre a los 30,732076 segundos, lo cual corresponde aproximadamente a 1,4 segundos transcurridos desde que los subflujos de tipo *no backup* dejan de operar.

Time	Source	Destination	Protocol	Length	Info
29.314412	200.0.8.100	200.0.0.100	MPTCP	24…	50452 → 9000
29.314426	200.0.0.100	200.0.8.100	MPTCP	78	9000 → 50452
29.314456	200.0.8.100	200.0.0.100	MPTCP	88…	50452 → 9000
29.314468	200.0.0.100	200.0.8.100	MPTCP	78	9000 → 50452
30.732076	200.0.8.101	200.0.0.100	MPTCP	71…	47937 → 9000
30.732090	200.0.0.100	200.0.8.101	MPTCP	78	9000 → 47937
30.732123	200.0.8.101	200.0.0.100	MPTCP	71…	47937 → 9000
30.732126	200.0.0.100	200.0.8.101	MPTCP	78	9000 → 47937
30.732150	200.0.8.101	200.0.0.100	MPTCP	71…	47937 → 9000

Figura 4.1: Instante en el que el subflujo de *backup* comienza a transmitir información.

Corresponde destacar que durante la ejecución de este caso de validación se observó que la definición de endpoints con flag *subflow* en el host servidor y de endpoints con flag *signal* en el host cliente genera problemas durante las conexiones MPTCP. En esencia, se desencadena un comportamiento en el que el cliente le anuncia sus direcciones al servidor, y el servidor entonces se dispone a inicializar subflujos con el cliente. Sin embargo, el cliente le responde con una señal RST de TCP a cada intento que realice el servidor de inicializar un nuevo subflujo.

Al realizar una inspección con respecto a la ruta que siguen los paquetes que se envían del host H9 hacia el host H1, se puede observar que la conexión MPTCP emplea dos rutas distintas:

- Ruta 1: H9 - R30005 - R20005 - R10001 - R20001 - R30001 - H1.
- Ruta 2: H9 - R30005 - R20006 - R10004 - R20002 - R30001 - H1.

En particular, se presenta el patrón de que los subflujos asociados a la conexión MPTCP se reparten entre estas dos posibles rutas. La Figura 4.2 expone las rutas mencionadas anteriormente.

Análisis de resultados

Se comprueba el funcionamiento del ambiente de trabajo que se define para la posterior ejecución de casos de prueba.

A su vez, se pone a prueba el funcionamiento de los subflujos de tipo *backup*, resultando en que éstos operan según lo esperado. Sin embargo, se detecta que el proceso de pasar a utilizar subflujos de *backup* implica un retardo considerable por parte de la conexión MPTCP. Parte de este retardo puede corresponder al tiempo que le insume al protocolo detectar que sus subflujos de tipo *no backup* dejaron de estar operativos.

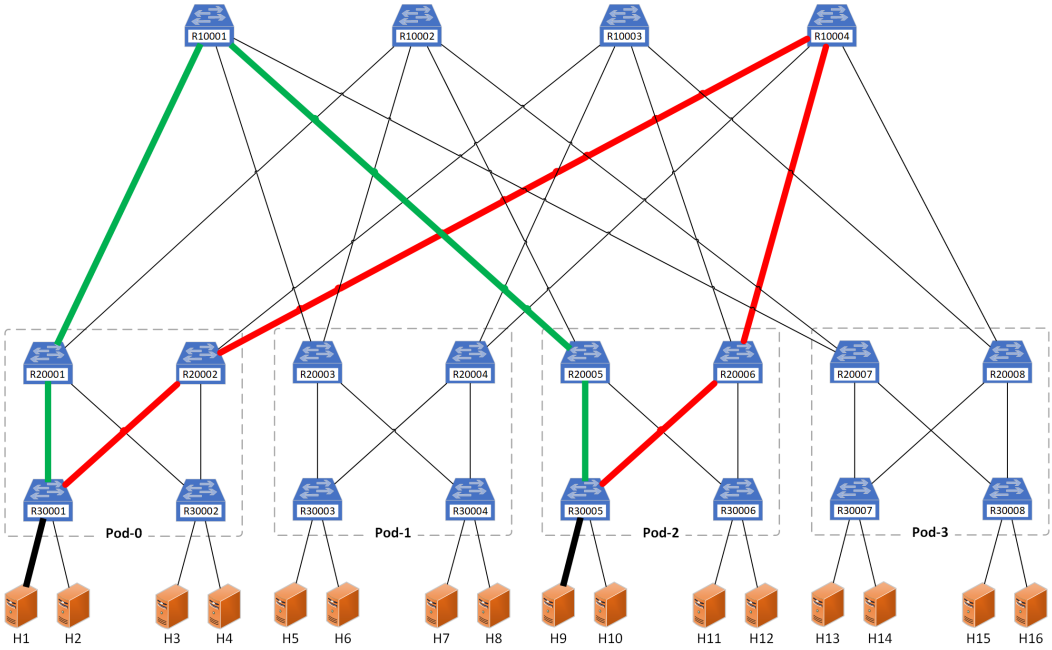

Figura 4.2: Rutas empleadas por una conexión MPTCP de origen H9 y destino H1.

En el contexto de una conexión MPTCP, el protocolo opera de mejor manera cuando cada host dispone de un rol estrictamente asumido. Esto implica que el host servidor se limite a anunciar sus direcciones disponibles, mientras que el host cliente se encargue de establecer los nuevos subflujos.

Finalmente, se destaca que el algoritmo de enrutamiento utilizado en la topología fat-tree permite que distintos subflujos de una conexión MPTCP empleen rutas mayoritariamente disjuntas. Esto implica una mejor utilización de la infraestructura de red por parte de las conexiones MPTCP.

4.5. Caso de prueba 1: Competencia de ancho de banda

Cuando varias conexiones TCP compiten por un mismo recurso de red, se espera que éste sea repartido entre las distintas conexiones TCP siguiendo una tendencia equitativa. Sin embargo, una conexión MPTCP puede llegar a disponer de varios subflujos que a su vez se modelan como conexiones TCP individuales.

Considerando que el ancho de banda se reparte a nivel de conexión TCP, resulta intuitivo asumir que una conexión MPTCP, con la suficiente cantidad de subflujos, podría llegar a superar en ancho de banda a una simple conexión TCP con la que compita.

Descripción del caso de prueba

Se desea comparar los anchos de banda que experimentan una conexión TCP y una conexión MPTCP que ocurren al mismo tiempo, sobre los mismos hosts de origen y destino.

Se define H1 como host servidor y H9 como host cliente, y se establece el siguiente procedimiento:

- *Paso 1*: para cada valor entero de n, con $1 \leq n \leq 9$:

  Realizar 10 mediciones del ancho de banda obtenido en cada caso por una conexión TCP y una conexión MPTCP con exactamente n subflujos, siendo que ambas se originan al mismo tiempo en el host H9 con destino el host H1. Cada una de las conexiones

dispone de una duración de 30 segundos. Finalmente, obtener un promedio a partir de las 10 mediciones obtenidas.

- *Paso 2*: determinar para cuál valor de n se obtuvieron mejores resultados de promedio de ancho de banda en el *Paso 1*, respecto a las conexiones MPTCP.

  Utilizando dicho valor de n, realizar 5 mediciones del ancho de banda obtenido en cada caso por una conexión TCP y una conexión MPTCP con exactamente n subflujos. Considerar conexiones con duraciones de 60, 300, 600, 1800 y 3600 segundos. Finalmente, promediar las mediciones obtenidas.

Resultados obtenidos

En la Figura 4.3 se presenta una tabla con los promedios obtenidos a partir de la realización del *Paso 1* del caso de prueba. Notar que cada fila representa un promedio calculado sobre 10 mediciones independientes. A su vez, la tabla también expone el promedio de volumen de datos que cada conexión logra trasportar durante su duración de 30 segundos.

Cantidad de Subflujos MPTCP	Promedio de mediciones de 30 segundos			
	TCP		MPTCP	
	Mbytes	Kbits/sec	Mbytes	Kbits/sec
1	1,703	432,7	1,703	427
2	1,315	330	2,064	531
3	1,1025	275,5	2,366	607,4
4	1,0785	266,3	2,39	611,8
5	1,0395	256	2,378	613,4
6	1,0275	258,1	2,353	615,1
7	1,065	263,4	2,328	601,7
8	1,0645	265,6	2,364	611,5
9	1,077	267,5	2,328	601,3

Figura 4.3: Promedios de ancho de banda registrados para TCP y MPTCP.

Considerando los resultados obtenidos, la Figura 4.4 muestra una tabla que representa el promedio del porcentaje de ancho de banda que registra cada conexión en el host servidor H1. Esto se calcula como el promedio de ancho de banda registrado en cada caso, dividido entre el ancho de banda máximo del enlace (1000 Kbits/seg) y multiplicado por 100.

Cantidad de Subflujos MPTCP	Promedio de porcentaje de ancho de banda 100% = 1000 Kbits/sec			
	TCP	MPTCP	MPTCP + TCP	MPTCP / TCP
1	43,27	42,7	85,97	0,99
2	33	53,1	86,1	1,61
3	27,55	60,74	88,29	2,2
4	26,63	61,18	87,81	2,3
5	25,6	61,34	86,94	2,4
6	25,81	61,51	87,32	2,38
7	26,34	60,17	86,51	2,28
8	26,56	61,15	87,71	2,3
9	26,75	60,13	86,88	2,25

Figura 4.4: Promedios de porcentajes de ancho de banda registrados para TCP y MPTCP.

La Figura 4.4 también expone el porcentaje de ancho de banda total que utilizan las conexiones TCP y MPTCP en conjunto. Además, muestra la relación entre los porcentajes de ancho de banda que experimentan ambas conexiones.

Por otro lado, la Figura 4.5 detalla un gráfico construido a partir de los promedios de ancho de banda registrados.

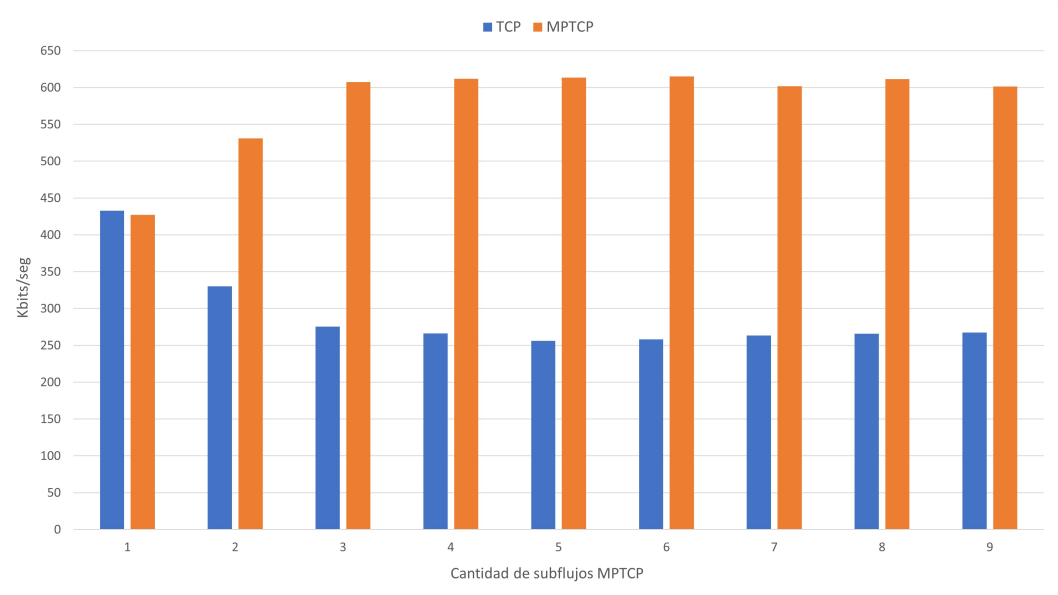

Figura 4.5: Gráfico de promedios de ancho de banda.

De forma análoga, la Figura 4.6 corresponde a un gráfico que en este caso representa el promedio de volúmenes de datos trasportados.

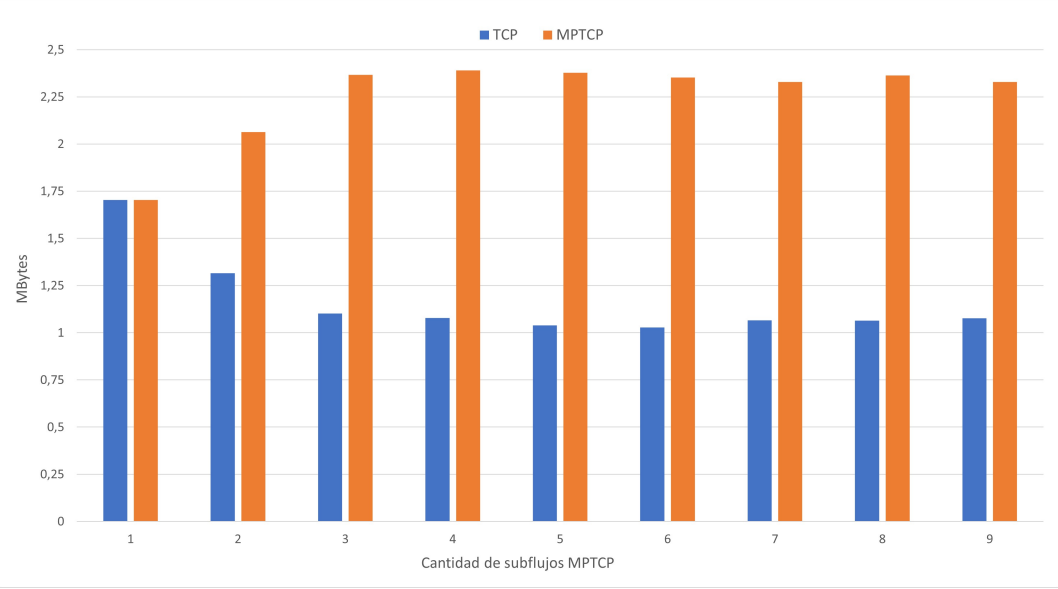

Figura 4.6: Gráfico de promedios de volúmenes de datos trasportados.

A continuación, en la Figura 4.7 se muestran tablas que registran los promedios obtenidos a partir de la realización del *Paso 2* del caso de prueba. En particular, para la realización

de este paso se definió utilizar conexiones MPTCP con exactamente 5 subflujos, puesto que se entiende que tal cantidad implicó mejores resultados en el *Paso 1*.

60sec				300sec			
TCP		MPTCP		TCP		MPTCP	
Mbytes	Kbits/sec	Mbytes	Kbits/sec	Mbytes	Kbits/sec	Mbytes	Kbits/sec
1,9	249,2	4,58	613,6	7,88	217	23,46	650,6

600sec				1800sec			
TCP		MPTCP		TCP		MPTCP	
Mbytes	Kbits/sec	Mbytes	Kbits/sec	Mbytes	Kbits/sec	Mbytes	Kbits/sec
15,02	208,4	45,9	638,8	45,08	209,4	146	680,2

3600sec			
TCP		MPTCP	
Mbytes	Kbits/sec	Mbytes	Kbits/sec
83,66	194	294,4	685,4

Figura 4.7: Promedios de ancho de banda al utilizar 5 subflujos MPTCP.

Finalmente, utilizando los resultados obtenidos en el *Paso 2*, la Figura 4.8 detalla el volumen de datos MPTCP dividido por el volumen de datos TCP, expresado en función de la duración de las conexiones.

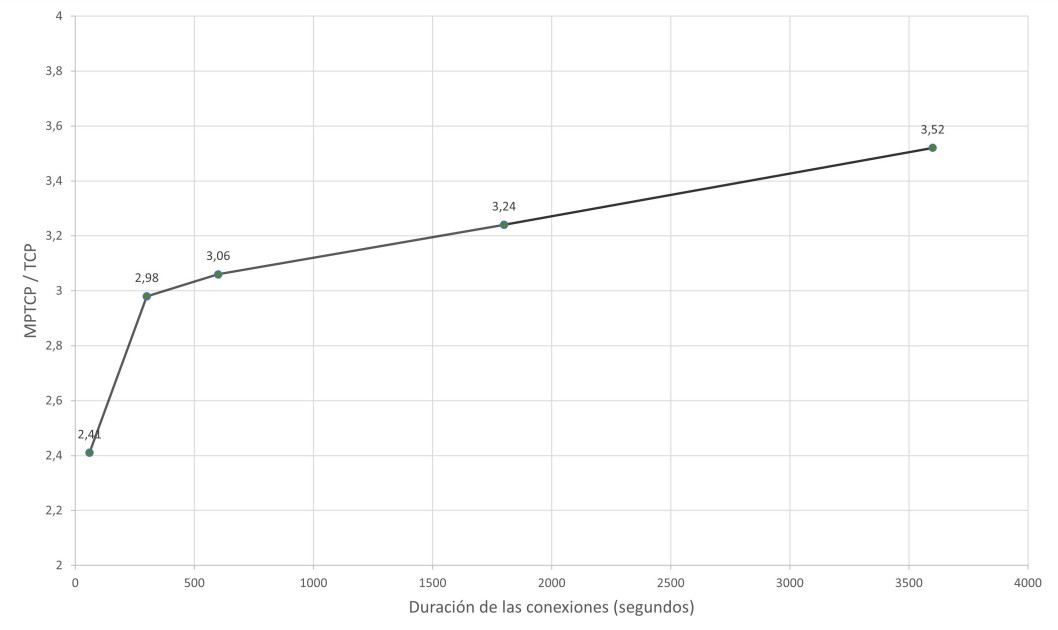

Figura 4.8: Volumen de datos MPTCP dividido por volumen de datos TCP.

Análisis de resultados

Al competir una conexión TCP con una conexión MPTCP de un solo subflujo, se observa que el ancho de banda se reparte prácticamente de igual manera entre las dos conexiones. En este contexto, los resultados indican que TCP supera por una mínima diferencia a MPTCP en ancho de banda.

Cuando la conexión MPTCP dispone de 2 subflujos, se evidencia un mayor ancho de banda obtenido por esta conexión. En consecuencia, la conexión TCP consigue un menor ancho de banda. Este comportamiento se incrementa cuando incluso se consideran 3 subflujos MPTCP. A partir de los 3 subflujos en adelante, MPTCP continua superando en ancho de banda a TCP pero manteniendo prácticamente la misma ventaja. En este punto, aumentar la cantidad de subflujos no impacta significativamente en la cantidad de ancho de banda que MPTCP pueda llegar a adquirir.

MPTCP con 5 subflujos resulta ser la configuración que obtiene mejores resultados en el *Paso 1* del caso de prueba. Con esta configuración MPTCP logra obtener un ancho de banda 2,4 veces superior al de TCP.

Cuando compite una conexión TCP con una conexión MPTCP de 5 subflujos en intervalos de duración mayores a 300 segundos, MPTCP obtiene un ancho de banda ligeramente superior al registrado en el *Paso 1* con 5 subflujos. En estos escenarios, el volumen de datos transportado por MPTCP alcanza a triplicar el de TCP.

4.6. Caso de prueba 2: Utilización de subflujos

Cuando una conexión Multi-Path TCP dispone de varios subflujos establecidos de tipo *no backup*, surge la interrogante de cómo se distribuye el tráfico de paquetes enviados entre éstos.

En relación a los resultados obtenidos durante el *caso de prueba 1* en la sección 4.5, se desea averiguar si la limitación de ancho de banda impuesta sobre la topología corresponde a un factor que condiciona la proporción de tráfico de datos que transmite cada subflujo.

Descripción del caso de prueba

Se busca comparar la utilización de los distintos subflujos asociados a una conexión MPTCP, y determinar si existe una relación con respecto al ancho de banda disponible en la topología fat-tree.

Se define H1 como host servidor y H9 como host cliente. Por otro lado, se establece que cada conexión MPTCP creada disponga de la máxima cantidad de subflujos posibles. Esta cantidad corresponde a 9 subflujos y está determinada por la propia implementación de MPTCP. Adicionalmente, para la realización de las mediciones implicadas, se consideran distintas velocidades máximas con respecto a todas las interfaces de las topología fat-tree. El procedimiento para este caso de prueba corresponde al siguiente:

- *Paso 1*: sea $v \in \{100, 1000, 10000, 100000, 1000000\}$ y $d \in \{5, 10, 30, 60\}$, para cada combinación posible (v, d) proceder a:

  Inicializar una topología fat-tree donde la velocidad de carga y descarga de todas sus interfaces esté limita a v Kbits/seg. A continuación, establecer una conexión MPTCP con exactamente d segundos de duración. Obtener una captura de tráfico de toda la conexión MPTCP.

- *Paso 2*: haciendo uso de la herramienta Wireshark, extraer el número de paquetes enviados a través de cada subflujo. Considerando el total de paquetes transmitidos por toda la conexión MPTCP, calcular el porcentaje de paquetes que cada subflujo se encargó de transmitir.

- *Paso 3*: calcular para cada conexión MPTCP la varianza con respecto al porcentaje de paquetes que fueron enviados a través de cada subflujo.

Resultados obtenidos

En la Figura 4.9 se exponen cuatro tablas que representan los resultados obtenidos a partir de la ejecución del caso de prueba. Estas tablas se organizan en función de las duraciones establecidas para las conexiones MPTCP. En su interior, cada una dispone de cinco filas que representan a las conexiones MPTCP realizadas sobre topologías fat-tree con distintos anchos de banda.

Ancho de banda (Kbits/seg)	5 segundos										
	Total paquetes	Porcentaje de paquetes por subflujo									
		1	2	3	4	5	6	7	8	9	Varianza
100	430	16,3	17,2	10,7	13,5	10,2	10,2	9,3	7,9	4,7	15,8
1000	824	18,3	11,8	10,8	9	14	17,1	8,9	5,2	5	22,3
10000	5584	22,8	25,1	19	14,8	10,2	4,4	1,2	0,7	1,7	93,7
100000	6960	28,5	20,8	17,1	8,8	8,4	7,8	4,4	1,8	2,3	83,1
1000000	6350	33	28,9	16,9	13,7	4,1	2	0,4	0,5	0,3	164

Ancho de banda (Kbits/seg)	10 segundos										
	Total paquetes	Porcentaje de paquetes por subflujo									
		1	2	3	4	5	6	7	8	9	Varianza
100	548	11,9	16,4	14,2	12,8	11,9	14,1	8	6,2	4,6	15,8
1000	922	21,7	18,7	24,9	7,6	7,4	4,6	6,5	5,2	3,5	67,8
10000	11716	29,7	24,3	21	12	4,7	4,4	2,4	0,9	0,7	124,1
100000	18459	26,2	22,8	24,3	12,8	5,8	2,7	2,7	1	1,7	112,7
1000000	13914	29	26,1	20,5	14,4	5,4	2,7	0,8	0,9	0,1	134,9

Ancho de banda (Kbits/seg)	30 segundos										
	Total paquetes	Porcentaje de paquetes por subflujo									
		1	2	3	4	5	6	7	8	9	Varianza
100	689	13,1	19,2	16,3	13,4	10,9	9,7	5,2	4,6	7,7	24,1
1000	2258	25,4	19,3	26,2	15	7,3	2,3	2,3	1,5	0,7	110,6
10000	30766	26,1	23,5	21,6	15,7	8,2	2,3	1,2	1,1	0,4	113,6
100000	51590	24	21,1	19,1	13	8,9	5,9	3,9	2,5	1,6	72,7
1000000	42664	28,9	23,3	21,4	12,1	6,2	3,3	2,8	1,3	0,8	116,1

Ancho de banda (Kbits/seg)	60 segundos										
	Total paquetes	Porcentaje de paquetes por subflujo									
		1	2	3	4	5	6	7	8	9	Varianza
100	932	15,2	13,7	12,4	17,3	14,5	8,9	6,1	5,7	5,8	20,3
1000	3947	20,4	19,3	16,4	21,1	16,3	2,4	2	1,1	0,8	84,2
10000	58157	24,2	22,4	19,3	16,4	9,3	4,6	1,8	0,8	1,1	91,6
100000	97270	24,2	22,3	19	13,6	8,1	4,8	3,2	2,6	2,1	78,8
1000000	88353	26,6	24	21,8	13,2	6,9	3,8	2,1	1,2	0,5	111

Rango	0 <= x < 5	5 <= x < 10	10 <= x < 15	15 <= x < 20	20 <= x
Color					

Figura 4.9: Utilización de subflujos en distintas conexiones MPTCP.

Para cada conexión MPTCP se especifica el total de paquetes enviados, el porcentaje de paquetes que fue enviado a través de cada subflujo y la varianza calculada. En particular, cada subflujo se denota con un número que especifica el orden en que fue establecido en la conexión MPTCP.

A modo de brindar un mejor análisis sobre los datos obtenidos, se procede a aplicar un color a cada celda que especifique un porcentaje de paquetes enviado a través de un subflujo. En el extremo inferior la Figura 4.9 se expone el criterio de colores utilizado.

Análisis de resultados

Predomina la tendencia de que los primeros subflujos en establecerse son los que más paquetes trasmiten. Sin embargo, esta tendencia no parece corresponder al tiempo que tarda la conexión MPTCP en crear todos los subflujos ya que al variar la duración de las conexiones no se ve afectado el resultado a gran escala.

Al observar las varianzas, dicha tendencia aparenta agudizarse a medida que el ancho de banda aumenta. Se puede observar que las distribuciones más disparejas son las asociadas a conexiones con el mayor ancho de banda.

Los resultados obtenidos cuando el ancho de banda equivale a 1000 Kbits/seg y las conexiones MPTCP duran 30 o más segundos demuestran que los primeros 5 subflujos son los que transmiten la mayor cantidad de paquetes. Esto puede explicar por qué en el *caso de prueba 1* al considerar más de 5 subflujos no se logra adquirir un mayor ancho de banda en la conexión MPTCP.

4.7. Caso de prueba 3: MPTCP con un solo subflujo

El protocolo Multi-Path TCP se construye a través del protocolo TCP, donde cada uno de sus subflujos en realidad se modela como una conexión TCP individual. Luego, a nivel lógico se establece que cada subflujo en realidad forma parte de una conexión de mayores dimensiones, es decir, la propia conexión MPTCP.

En este caso de prueba se propone comparar el desempeño de una conexión TCP con respecto al de una conexión MPTCP que solo dispone de un subflujo asociado. En otras palabras, interesa comparar el despeño de las conexiones cuando se encuentran en igualdad de condiciones.

Descripción del caso de prueba

Se desea comparar el desempeño que experimentan una conexión TCP y una conexión MPTCP de un solo subflujo que ocurren al mismo tiempo, sobre los mismos hosts de origen y destino. En estas condiciones, se prevé que una conexión MPTCP de un solo subflujo implique un mayor gasto a nivel de recursos de cómputo (*overhead*) que una conexión TCP estándar.

Este caso de prueba considera dos enfoques distintos. Uno es a través del envío datos por volumen, y el otro corresponde al envío de datos por duración de conexión.

Se establece H1 como host servidor y H9 como host cliente. En ambos hosts, se define que toda conexión MPTCP solo pueda disponer de un subflujo asociado.

El procedimiento del caso de prueba se presenta a continuación:

- *Paso 1*: para cada $t \in \{10, 30, 60, 300, 600\}$:

  Realizar 5 mediciones del volumen de datos enviados y del ancho de banda obtenido en cada caso por una conexión TCP y una conexión MPTCP con un solo subflujo, siendo que ambas se originan al mismo tiempo en el host H9 con destino el host H1. Cada una de las conexiones dispone de una duración exacta de t segundos. Promediar las 5 mediciones obtenidas.

- *Paso 2*: para cada $v \in \{1, 2, 5, 10, 20\}$:

  Realizar 5 mediciones del tiempo empleado en cada caso por una conexión TCP y una conexión MPTCP con un solo subflujo, siendo que ambas se originan al mismo tiempo en el host H9 con destino el host H1. Cada una de las conexiones debe trasmitir exactamente v MBytes. Realizar un promedio de las 5 mediciones obtenidas.

Resultados obtenidos

La Figura 4.10 expone los resultados obtenidos con respecto a la realización del *Paso 1*. En esta se detalla el desempeño de los protocolos en función de cada duración de conexión estipulada.

Duración de conexión (segundos)	TCP		MPTCP 1 Subflujo		MPTCP - TCP	
	Mbytes	Kbits/sec	Mbytes	Kbits/sec	Mbytes	Kbits/sec
10	0,7	446,8	0,7	438,6	0	-8,2
30	1,68	427,6	1,73	434	0,05	6,4
60	3,3	437,6	3,23	424,6	-0,07	-13
300	15,72	434,2	14,94	418,8	-0,78	-15,4
600	31,38	435	30,14	417,4	-1,24	-17,6

Figura 4.10: Desempeño de conexiones en función de su duración.

Por otro lado, la Figura 4.11 muestra los resultados del *Paso 2*, es decir, el desempeño de los protocolos cuando se varía la cantidad de volúmenes de datos que deben transferir.

Volumen de datos (MB)	TCP	MPTCP 1 Subflujo	MPTCP - TCP
	Segundos	Segundos	Segundos
1	19,26	19,45	0,19
2	38,34	39,1	0,76
5	98,04	99,19	1,15
10	194,53	198,41	3,88
20	390,83	398,2	7,37

Figura 4.11: Desempeño de conexiones en función del volumen de datos a transmitir.

Análisis de resultados

Los resultados obtenidos se muestran favorables con respecto a las conexiones TCP. En comparativa, una conexión TCP parece obtener un mejor ancho de banda cuando compite con una conexión MPTCP de un solo subflujo. Esto provoca que una conexión TCP pueda transportar un mayor volumen de información que una conexión MPTCP de un solo subflujo. De esta forma, el desempeño de las conexiones TCP se torna notoriamente más favorable en la medida que aumenta la duración de las conexiones.

Una posible explicación con respecto a los resultados obtenidos puede consistir en que MPTCP es un protocolo que requiere una mayor utilización de recursos de cómputo para poder operar. Típicamente, MPTCP incluye más información de control que TCP en la cabecera de sus segmentos. A su vez, MPTCP debe procesar dicha información de control cuando se recibe un segmento en un host.

En esencia, MPTCP con un solo subflujo puede entenderse como una versión de TCP con mayor carga. Esta carga deriva de la inclusión de más información y de la ejecución de las operaciones que la utilizan. Para aprovechar de las propiedades brindadas por el protocolo

MPTCP, se debe considerar una cantidad de subflujos estrictamente mayor que uno.

4.8. Caso de prueba 4: Volúmenes de datos pequeños

Como bien se comprueba a través del *caso de prueba 1* en la sección 4.5, el protocolo Multi-Path TCP se desempeña de mejor manera que el protocolo TCP cuando dispone de varios subflujos y la conexión implica la transmisión de un volumen de información mayor o igual que 1 MByte.

Este caso de prueba pretende extender los resultados obtenidos en el *caso de prueba 1*, considerando conexiones TCP y MPTCP que transmitan pequeños volúmenes de información. De esta forma, interesa investigar si los desempeños de los protocolos están condicionados al volumen de datos que deben transmitir.

Descripción del caso de prueba

Se desea comparar el desempeño que experimentan una conexión TCP y una conexión MPTCP que ocurren al mismo tiempo, sobre los mismos hosts de origen y destino, considerando escenarios que implican transmitir volúmenes de datos pequeños.

Se define H1 como host servidor y H9 como host cliente. En ambos hosts, se limita la cantidad máxima de subflujos por conexión MPTCP a un máximo de 5. Esta convención se define considerando los resultados obtenidos en el *caso de prueba 1*.

El procedimiento a seguir para este caso de prueba es el siguiente:

- Para cada $v \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$:

  Realizar 5 mediciones del tiempo empleado y del ancho de banda obtenido en cada caso por una conexión TCP y una conexión MPTCP con un máximo de 5 subflujos, siendo que ambas se originan al mismo tiempo en el host H9 con destino el host H1. Cada una de las conexiones debe transmitir exactamente v KBytes. Promediar las 5 mediciones obtenidas.

Resultados obtenidos

La Figura 4.12 muestra los resultados obtenidos luego de promediar las mediciones realizadas.

Volumen de datos	TCP		MPTCP		MPTCP - TCP	
(KB)	Segundos	Kbits/sec	Segundos	Kbits/sec	Segundos	Kbits/sec
1	0,00062	14556	0,00922	1179,6	0,0086	-13376,4
2	0,0015	20558	0,0076	3565,2	0,0061	-16992,8
4	0,0007	54800	0,00926	5568	0,00856	-49232
8	0,00054	110200	0,25016	262,2	0,24962	-109937,8
16	0,14836	891,4	0,18156	795,8	0,0332	-95,6
32	0,55344	478,8	0,46506	587,2	-0,08838	108,4
64	1,30538	412	1,08646	488,2	-0,21892	76,2
128	2,66094	394,4	2,29968	457	-0,36126	62,6
256	5,14952	408,2	3,86768	524,6	-1,28184	116,4
512	10,55182	398,6	7,29234	577,2	-3,25948	178,6
1024	20,88896	402,2	14,53512	579,2	-6,35384	177
2048	40,70576	412,2	27,99896	600,6	-12,7068	188,4

Figura 4.12: Desempeño de conexiones TCP y MPTCP con volúmenes de datos pequeños.

En el caso de las conexiones MPTCP realizadas, corresponde destacar que para los escenarios en que los volúmenes de datos a transmitir son menores que 128 KBytes, estas conexiones MPTCP solo logran establecer un único subflujo. Igualmente, se registraron unas pocas excepciones que en estas condiciones sí lograron establecer hasta tres subflujos como máximo.

Por el contrario, a partir de los 128 KBytes en adelante ocurre exactamente lo opuesto, es decir, las conexiones MPTCP logran establecer la cantidad máxima de subflujos permitida.

Análisis de resultados

El protocolo TCP obtuvo un mejor desempeño en la transmisión de volúmenes de datos menores o iguales a 16 KBytes. De forma contraria, el protocolo MPTCP con 5 subflujos obtuvo mejores resultados con la transmisión de volúmenes de datos mayores o iguales a 32 KBytes.

Los resultados obtenidos implican que Multi-Path TCP no resulta ser una mejor opción que el protocolo TCP cuando el volumen de datos a enviar se mantiene muy pequeño.

4.9. Caso de prueba 5: Recuperación frente a fallas

En el contexto de la topología fat-tree, se desea averiguar cómo reacciona una conexión cuando uno de los enlaces que interviene en ésta deja de estar operativo. Específicamente, interesa comparar las respuestas de las conexiones TCP y Multi-Path TCP.

Descripción del caso de prueba

Se procede a generar una conexión a la vez y, antes de que finalice, se inhabilita un enlace de la topología que esté siendo utilizado por la conexión. El enlace a inhabilitar debe ser elegido de tal forma que en la topología fat-tree exista al menos un camino desde el host de origen hacia el host de destino que no utilice dicho enlace.

Se define H1 como host servidor y H9 como host cliente de la conexión. En ambos hosts, se limita la cantidad máxima de subflujos por conexión MPTCP a un máximo de 5. A su vez, se establece que en cada conexión se transmiten 2 MBytes de datos.

El procedimiento de este caso de prueba es el siguiente:

- *Paso 1*: realizar 10 conexiones de tipo TCP, ejecutando solo una por vez. Medir la duración y velocidad promedio de las conexiones.
- *Paso 2*: idem *Paso 1*, pero utilizando conexiones de tipo Multi-Path TCP con 5 subflujos asociados.
- *Paso 3*: correr la herramienta ifstat en el *edge switch* R30005. Realizar 10 conexiones de tipo TCP, estableciendo solo una por vez. Antes de que finalice la conexión, mientras se emplea la herramienta ifstat para el monitoreo de interfaces, proceder a inhabilitar el enlace que cumpla las siguientes propiedades:
  - Es utilizado por la conexión TCP para el envío de paquetes de H9 a H1.
  - Conecta el *edge switch* R30005 con el *aggregation switch* R20005 o con el *aggregation switch* R20006.

  Medir la duración y velocidad promedio de las conexiones.
- *Paso 4*: correr la herramienta ifstat en el *edge switch* R30005 y proceder a realizar 10 conexiones de tipo MPTCP con 5 subflujos, ejecutando solo una por vez.

Teniendo en cuenta el algoritmo de enrutamiento empleado, se prevé que cada conexión MPTCP utilice durante el envío de paquetes los dos enlaces que conectan el *edge switch* R30005 con los *aggregation switches* R20005 y R20006.

De esta forma, antes de que finalice cada conexión MPTCP, inhabilitar el enlace que conecta el *edge switch* R30005 con el *aggregation switch* R20006. Medir la duración y velocidad promedio de las conexiones.

Resultados obtenidos

En la Figura 4.13 se puede observar una tabla con los promedios obtenidos tras la ejecución del caso de prueba.

Sin caidas de enlace			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
22,25	754,3	21,04	797,5

Con caidas de enlace			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
22,58	743,9	21,09	796,2

Figura 4.13: Promedios obtenidos con y sin considerar la caída de enlace.

Análisis de resultados

Los resultados obtenidos con respecto a las conexiones TCP evidencian que el algoritmo de enrutamiento se encarga de redistribuir el tráfico de estas conexiones hacia rutas que disponen de enlaces activos. De lo contrario, se esperaría que las conexiones TCP no pudiesen trasportar el volumen de datos definido.

Se observa que la redistribución de tráfico por parte del algoritmo de enrutamiento no impacta significativamente en la duración de las conexiones TCP y MPTCP. Particularmente, los resultados registrados exponen que el desempeño de las conexiones TCP y las conexiones MPTCP con 5 subflujos resultan similares cuando el volumen de datos a trasportar corresponde a 2 MBytes. Se recuerda que en este contexto, las conexiones no compiten por ancho de banda ya que ocurren en momentos separados.

4.10. Caso de prueba 6: Rendimiento ante congestión

Se desea conocer qué tipo de conexión es la que mejor opera ante un contexto de congestión global en una topología de red fat-tree. En particular, se busca averiguar si el resultado también depende del tipo de conexiones que producen la congestión.

Descripción del caso de prueba

En a una topología congestionada por conexiones donde todas son de tipo TCP o de tipo MPTCP, se establece una nueva conexión sobre la que se toman mediciones. Además, se desea que las conexiones establecidas específicamente para la generación de congestión se distribuyan de forma equitativa entre los enlaces de la red. Estas características se modelan de la siguiente forma:

- Cada host de la topología mantiene n conexiones activas con el único propósito de generar congestión en la red. De esta forma, cada host toma el rol de cliente en $n/2$ conexiones, mientras que a su vez toma el rol de servidor en las restantes $n/2$ conexiones.
- Se designan dos hosts que, además de mantener n conexiones activas cada uno, proceden a establecer un nueva conexión sobre la cual se obtienen las mediciones.
- Todas las conexiones se realizan entre hosts pertenecientes a distintos pods.
- Las conexiones generadas para la congestión son todas del mismo tipo entre sí, y siempre están presentes durante el transcurso de las conexiones sobre las que se toman las mediciones.

Para asegurar el cumplimiento de las características expuestas y facilitar el desarrollo de las pruebas, se automatiza la generación de matrices aleatorias que especifican las distintas conexiones a realizarse entre los hosts.

Se define H1 como host servidor y H9 como host cliente de la conexión sobre la que se realizarán las mediciones. Específicamente, se establece que toda conexión de tipo Multi-Path TCP cuente con 5 subflujos.

El procedimiento asociado a este caso de prueba es el siguiente:

- *Paso 1*: para cada $b \in \{1, 2, 4, 8, 16\}$:

  Sin congestión en la red, realizar 5 mediciones separadas de conexiones de tipo TCP y MPTCP que transmitan b MBytes de información. Registrar la duración y velocidad media de cada conexión, promediando los resultados asociados a cada valor de b.

- *Paso 2*: para cada $c \in \{1, 2, 3\}$:

  Se establece congestión en la red de tal forma que cada host es cliente de c conexiones TCP, mientras que a su vez también es servidor de otras c conexiones TCP. Realizar 5 mediciones separadas de conexiones de tipo TCP y MPTCP que transmitan 1 MByte de información. Registrar la duración y velocidad media de cada conexión para luego proceder a calcular promedios.

- *Paso 3*: idem *Paso 2*, pero utilizando conexiones de tipo Multi-Path TCP para generar la congestión en la red.

Resultados obtenidos

En la Figura 4.14 se pueden apreciar los promedios obtenidos tras ejecutar el *Paso 1*. Se observa que a medida que aumenta el volumen de datos a transmitir, también se incrementa la duración de las conexiones en aproximadamente la misma proporción. Esta característica recién mencionada ocurre tanto para las conexiones de tipo TCP, como también para las de tipo MPTCP. Es por esto que para los próximos pasos solo se realizan mediciones para un único volumen de datos a transmitir.

Por otro lado, la Figura 4.15 expone los resultados correspondientes a la ejecución del *Paso 2*. En esta se muestran los desempeños de las conexiones TCP y MPTCP en el contexto de una red congestionada con conexiones TCP.

Finalmente, a través de la Figura 4.16 se presentan los resultados obtenidos una vez realizado el *Paso 3*. En este caso, se puede apreciar el desempeño de las conexiones TCP y MPTCP cuando la red se encuentra congestionada por conexiones MPTCP de 5 subflujos.

Sin congestión - 1 MB			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
11,44	734,2	10,90	771,4

Sin congestión - 2 MB			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
21,92	766,4	21,14	795

Sin congestión - 4 MB			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
44,25	758,6	36,02	846,2

Sin congestión - 8 MB			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
89,45	750,6	80,31	833,6

Sin congestión - 16 MB			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
174,47	769,4	164,34	817,2

Figura 4.14: Promedios obtenidos con conexiones TCP y MPTCP en una red sin congestión.

Conexiones de tipo TCP para la congestión - 1 par de conexiones por cada host			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
28,78	291,4	15,88	529,6

Conexiones de tipo TCP para la congestión - 2 pares de conexiones por cada host			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
43,34	194,4	21,10	399,6

Conexiones de tipo TCP para la congestión - 3 pares de conexiones por cada host			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
65,94	127,2	35,28	239

Figura 4.15: Resultados obtenidos en una red congestionada con conexiones TCP.

Conexiones de tipo MPTCP para la congestión - 1 par de conexiones por cada host			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
82,45	102,34	37,82	222,6

Conexiones de tipo MPTCP para la congestión - 2 pares de conexiones por cada host			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
191,06	43,98	70,83	120,8

Conexiones de tipo MPTCP para la congestión - 3 pares de conexiones por cada host			
TCP		MPTCP	
Duración (sec)	Velocidad (Kbits/sec)	Duración (sec)	Velocidad (Kbits/sec)
291,33	28,92	110,68	76,12

Figura 4.16: Resultados obtenidos en una red congestionada con conexiones MPTCP.

Análisis de resultados

Las conexiones Multi-Path TCP con 5 subflujos desempeñan de forma más eficiente que las conexiones TCP en todos los escenarios considerados para este caso de prueba.

En general, las conexiones TCP y MPTCP se presentan más veloces cuando la congestión de red se origina a partir de conexiones TCP. Esto se debe a que cada conexión MPTCP dispone de 5 subflujos, siendo que cada uno se modela como una conexión TCP. En otras palabras, la congestión que una conexión MPTCP aporta puede llegar a compararse con la congestión que implican 5 conexiones TCP.

4.11. Caso de prueba 7: Redistribución de carga ante congestión

En situaciones que implican congestión en la red, una conexión Multi-Path TCP debe ser capaz de redistribuir la carga de tráfico de datos hacia sus subflujos que presentan menor congestión. Este caso de prueba propone validar la presencia de esta funcionalidad del protocolo en la implementación utilizada.

Descripción del caso de prueba

Dada una topología fat-tree, se desea generar un escenario en el cual inicialmente existe una conexión MPTCP transmitiendo datos entre dos hosts, para luego proceder a congestionar parte de la red que esté siendo utilizada por algunos de sus subflujos. En respuesta a esta congestión, se espera observar cómo MPTCP redistribuye su carga hacia los subflujos que no se ven afectados por la congestión de red inducida.

Se definen H1 y H12 como hosts servidores. Por otro lado, se establecen H9 y H10 como hosts clientes. Además, en cada host se limita la cantidad máxima de subflujos por conexión MPTCP a un máximo de 2.

El procedimiento de este caso de prueba es el siguiente:

- *Paso 1*: establecer una conexión MPTCP entre los hosts H9 y H1. Esta conexión debe disponer de exactamente 2 subflujos asociados, y su duración debe ser lo suficientemente extensa como para registrar resultados mientras se realiza el *Paso 2*.

  En un instante mientras se ejecuta la conexión MPTCP, registrar los anchos de banda que experimentan las interfaces de todos los componentes del pod número 2.

- *Paso 2*: mientras permanece activa la conexión MPTCP que se establece durante el *Paso 1*, establecer 9 conexiones TCP entre los hosts H10 y H12.

  Una vez establecidas estas nuevas conexiones, registrar nuevamente los anchos de banda que experimentan las interfaces de todos los componentes del pod número 2.

Considerando el algoritmo de enrutamiento definido, se prevé que durante la realización del *Paso 1* el tráfico saliente desde H9 sea redirigido hacia la interfaz *eth3* del *edge switch* R30005. En este punto, los subflujos de la conexión MPTCP originada en H9 se reparten entre las interfaces *eth1* y *eth2* del *edge switch* R30005. Al establecerse una conexión MPTCP con dos subflujos, cada interfaz *eth1* y *eth2* del *edge switch* R30005 procederá a transmitir el tráfico de exactamente un solo subflujo. La Figura 4.17 (a.) expone el escenario que se genera tras la ejecución del *Paso 1*.

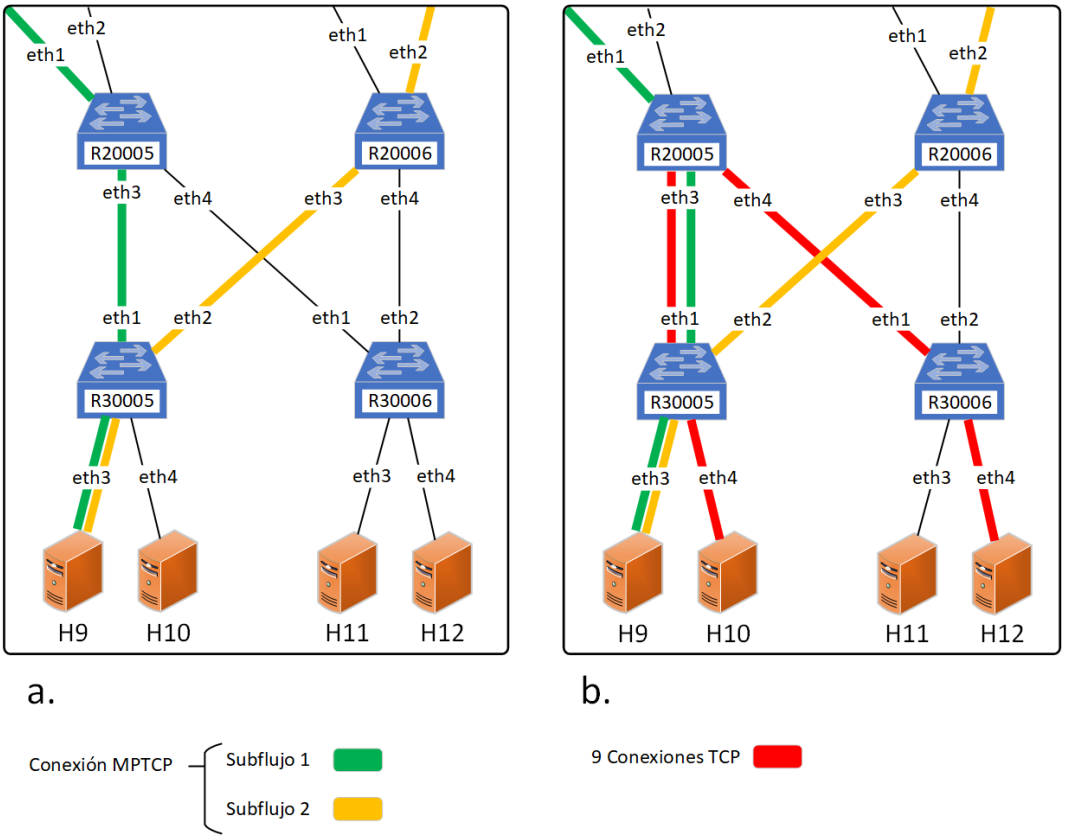

Figura 4.17: Escenarios generados a partir de la ejecución de este caso de prueba.

Teniendo en cuenta nuevamente el algoritmo de enrutamiento utilizado, se prevé que al realizar el *Paso 2* el tráfico proveniente de H10 será redirigido hacia la interfaz *eth4* del *edge switch* R30005. Luego, el tráfico de cada una de las 9 conexiones TCP será transmitido a partir de las interfaz *eth1* del *edge switch* R30005.

Frente a esta situación, la interfaz *eth1* del *edge switch* R30005 comenzará a experimentar una congestión como consecuencia de estar transmitiendo el tráfico de 10 flujos de transporte (9 flujos TCP y un subflujo de la conexión MPTCP). Por otro lado, la interfaz *eth2* del *edge switch* R30005 continuará transmitiendo únicamente el tráfico de un solo subflujo MPTCP. La Figura 4.17 (b.) muestra el escenario que se genera tras la ejecución del *Paso 2*.

Resultados obtenidos

En relación a los anchos de banda registrados durante el *Paso 1*, se observa que el host H1 recibe tráfico mientras que el host H9 se encarga de enviar tráfico. Los hosts H10 y H12 no envían ni reciben tráfico puesto que aún no se han establecido las conexiones TCP del *Paso 2*.

Desde el punto de vista de los switches, el tráfico proveniente del host H9 se reparte entre las interfaces *eth1* y *eth2* del *edge switch* R30005. Los anchos de banda de salida registrados para estas dos interfaces se mantienen en torno a los 500 Kbits/seg.

Tras la ejecución del *Paso 2* se obtienen nuevas mediciones de anchos de banda. Se reconoce tráfico saliente desde el host H9 al H1 y también desde el host H10 al H12. En este caso, el tráfico detectado entre H10 y H12 se debe a las 9 conexiones TCP establecidas entre éstos.

A su vez, se manifiesta un incremento en los anchos de banda de las interfaces *eth1* y *eth2* del *edge switch* R30005. El incremento que experimenta su interfaz *eth1* se debe a que ahora transmite el tráfico de 10 flujos de transporte. Por otro lado, el incremento que ocurre en su interfaz *eth2* corresponde a una redistribución de carga que la conexión MPTCP realiza como consecuencia de la congestión detectada en la interfaz *eth1* de este mismo *edge switch*.

Se aprecia entonces que el ancho de banda saliente en la interfaz *eth2* del *edge switch* R30005 se mantiene la mayor parte del tiempo por encima de los 500 Kbits/seg, experimentando ráfagas que superan los 800 Kbits/seg.

Análisis de resultados

Se comprueba la capacidad el protocolo Multi-Path TCP de redistribuir la carga en función de la congestión de red detectada. Esta funcionalidad del protocolo Multi-Path TCP se encuentra condicionada principalmente por las características de la infraestructura de red, y por el algoritmo de enrutamiento que sea empleado.

En otras palabras, la existencia de múltiples rutas mayoritariamente disjuntas que conecten cualquier par de hosts implicados, y la presencia de un algoritmo de enrutamiento que permita que los subflujos establecidos utilicen dichas rutas, consisten en factores claves que permiten aprovechar esta funcionalidad de mejor manera.

4.12. Caso de prueba 8: Manejador de rutas a nivel de usuario

En la subsección 3.2.3 se describe la implementación realizada con respecto a un manejador de rutas que opera a nivel de usuario. En esencia, este manejador de rutas implementado es capaz de recrear las mismas operaciones que realiza el manejador de rutas que es provisto por la implementación de Multi-Path TCP en el kernel de Linux.

Sin embargo, existen determinadas funcionalidades que presentan una mayor flexibilidad cuando se realizan desde un manejador de rutas que opera a nivel de usuario. En este caso de prueba se desea validar tales funcionalidades.

Descripción del caso de prueba

Una de las capacidades que más diferencian a un manejador de rutas que opera a nivel de usuario con respecto a uno que opera a nivel de kernel, consiste en la flexibilidad que este primero presenta durante la creación de subflujos.

El manejador de rutas que opera a nivel del usuario es capaz de establecer más de 9 subflujos por cada conexión MPTCP. En adición, éste es capaz de establecer dichos subflujos repitiendo los pares de direcciones IP local e IP remota. En otras palabras, es capaz diferenciar los subflujos en función de sus números de puertos utilizados, en lugar de utilizar sus direcciones IP.

Por otro lado, un manejador de rutas a nivel de usuario puede utilizar operaciones básicas de MPTCP para dar lugar a otras operaciones más complejas. La definición de políticas de gestión de subflujos es un ejemplo de esto. Cada política definida funciona a través del mismo principio, el cual consiste en ejecutar la operación MPTCP denominada como *Change Subflow Priority* para cambiar las prioridades de los distintos subflujos.

En este caso de prueba interesa evidenciar estos aspectos recién descritos. Considerando que todas las políticas de gestión de subflujos definidas operan a través de la misma operación MPTCP, se opta por exponer únicamente una de éstas.

Se define H1 como host servidor y H9 como host cliente de la conexión. Adicionalmente, se configuran los hosts H1 y H9 para que utilicen el manejador de rutas a nivel de usuario que se expone en la subsección 3.2.3. Este manejador de rutas se ejecuta de forma individual en cada uno de estos hosts.

El procedimiento asociado a este caso de prueba es el siguiente:

- *Paso 1*: establecer una conexión MPTCP de 9 subflujos entre el host H9 y el host H1, para luego aplicar la política de gestión de subflujos denominada como *vector de prioridades*. Realizar capturas de la *consola de manejo* antes y después de aplicar dicha política.
- *Paso 2*: establecer una conexión MPTCP de 20 subflujos entre el host H9 y el host H1. Registrar una captura de la *consola de manejo*.
- *Paso 3*: iniciar una conexión MPTCP entre el host H9 y el host H1. A continuación, establecer varios subflujos asociados a la conexión, utilizando pares de direcciones IP local e IP remota repetidos. Registrar una captura de la *consola de manejo* que muestre los subflujos establecidos.

Resultados obtenidos

La Figura 4.18 (a.) expone la conexión MPTCP de 9 subflujos establecida durante el *Paso 1*, antes de aplicar la política de gestión de subflujos de tipo *vector de prioridades*.

La conexión MPTCP se inicializa con una política de gestión de subflujos de tipo *manual*, siendo que todos sus subflujos comienzan siendo de tipo *no backup* (también denominados como *activos*). Por otro lado, se observa que cada subflujo registrado dispone de su propia prioridad numérica en cuanto a la política de *vector de prioridades*. Se recuerda que estas prioridades se obtienen a partir del *archivo externo de prioridades*.

En la Figura 4.18 (b.) se puede apreciar el estado de los subflujos de la conexión MPTCP, una vez que se aplica la política de *vector de prioridades*. En este punto, el único subflujo que permanece como *no backup* (o *activo*) corresponde al que tiene la prioridad en valor numérico 1. El resto de los subflujos que disponen de prioridades más bajas pasan a ser de tipo *backup*.

El cambio de política recién descrito implicó la ejecución de ocho cambios de prioridades de subflujos. En otra palabras, la operación MPTCP conocida como *Change Subflow Priority* se ejecutó una vez por cada subflujo sobre el cual se solicitó un cambio de prioridad a *backup*.

a.

Subflujos establecidos:

Indice	Direccion Local	Direccion Remota	Prioridad Local	Prioridad remota	Prioridad Vector
1	200.0.0.100 : 10275	200.0.8.100 : 38534	Activo	Activo	6
2	200.0.0.100 : 15771	200.0.8.100 : 36801	Activo	Activo	6
3	200.0.0.100 : 15771	200.0.8.101 : 25009	Activo	Activo	4
4	200.0.0.100 : 15771	200.0.8.102 : 34768	Activo	Activo	2
5	200.0.0.100 : 15771	200.0.8.103 : 16339	Activo	Activo	1
6	200.0.0.100 : 15771	200.0.8.104 : 65488	Activo	Activo	5
7	200.0.0.101 : 40343	200.0.8.100 : 29165	Activo	Activo	9
8	200.0.0.101 : 40343	200.0.8.101 : 23466	Activo	Activo	5
9	200.0.0.101 : 40343	200.0.8.102 : 30098	Activo	Activo	3

b.

Subflujos establecidos:

Indice	Direccion Local	Direccion Remota	Prioridad Local	Prioridad remota	Prioridad Vector
1	200.0.0.100 : 10275	200.0.8.100 : 38534	Backup	Activo	6
2	200.0.0.100 : 15771	200.0.8.100 : 36801	Backup	Activo	6
3	200.0.0.100 : 15771	200.0.8.101 : 25009	Backup	Activo	4
4	200.0.0.100 : 15771	200.0.8.102 : 34768	Backup	Activo	2
5	200.0.0.100 : 15771	200.0.8.103 : 16339	Activo	Activo	1
6	200.0.0.100 : 15771	200.0.8.104 : 65488	Backup	Activo	5
7	200.0.0.101 : 40343	200.0.8.100 : 29165	Backup	Activo	9
8	200.0.0.101 : 40343	200.0.8.101 : 23466	Backup	Activo	5
9	200.0.0.101 : 40343	200.0.8.102 : 30098	Backup	Activo	3

Figura 4.18: Ejecución de política de gestión de subflujos de tipo *vector de prioridades*.

En la Figura 4.19 se muestra el resultado obtenido a partir del *Paso 2*. En este caso, se comprueba la capacidad del manejador de rutas a nivel de usuario de establecer más de 9 subflujos por cada conexión MPTCP.

Subflujos establecidos:

Indice	Direccion Local	Direccion Remota	Prioridad Local	Prioridad remota	Prioridad Vector
1	200.0.8.100 : 62645	200.0.0.100 : 10275	Activo	Activo	6
2	200.0.8.100 : 12715	200.0.0.100 : 37340	Activo	Activo	6
3	200.0.8.101 : 42892	200.0.0.100 : 37340	Activo	Activo	4
4	200.0.8.102 : 7064	200.0.0.100 : 37340	Activo	Activo	2
5	200.0.8.103 : 32163	200.0.0.100 : 37340	Activo	Activo	6
6	200.0.8.104 : 2434	200.0.0.100 : 37340	Activo	Activo	10
7	200.0.8.100 : 16798	200.0.0.101 : 55728	Activo	Activo	4
8	200.0.8.101 : 42956	200.0.0.101 : 55728	Activo	Activo	5
9	200.0.8.102 : 50608	200.0.0.101 : 55728	Activo	Activo	4
10	200.0.8.103 : 35303	200.0.0.101 : 55728	Activo	Activo	2
11	200.0.8.104 : 23473	200.0.0.101 : 55728	Activo	Activo	10
12	200.0.8.100 : 26079	200.0.0.102 : 50566	Activo	Activo	5
13	200.0.8.101 : 37846	200.0.0.102 : 50566	Activo	Activo	3
14	200.0.8.102 : 20428	200.0.0.102 : 50566	Activo	Activo	6
15	200.0.8.103 : 35752	200.0.0.102 : 50566	Activo	Activo	1
16	200.0.8.104 : 53680	200.0.0.102 : 50566	Activo	Activo	10
17	200.0.8.100 : 37826	200.0.0.103 : 54234	Activo	Activo	1
18	200.0.8.101 : 23505	200.0.0.103 : 54234	Activo	Activo	1
19	200.0.8.102 : 60907	200.0.0.103 : 54234	Activo	Activo	1
20	200.0.8.103 : 10686	200.0.0.103 : 54234	Activo	Activo	7

Figura 4.19: Establecimiento de 20 subflujos en una misma conexión MPTCP.

En adición, la Figura 4.20 expone la ejecución del *Paso 3*, mostrando el establecimiento de varios subflujos con los mismos pares de IP local e IP remota. Específicamente, desde el host cliente se generan 6 subflujos que utilizan la IP local 200.0.8.102 junto con la IP remota 200.0.0.100. Estos subflujos se diferencian a partir de los números de puertos locales.

Subflujos establecidos:

Indice	Direccion Local	Direccion Remota	Prioridad Local	Prioridad remota	Prioridad Vector
1	200.0.8.100 : 37041	200.0.0.100 : 10275	Activo	Activo	6
2	200.0.8.101 : 46476	200.0.0.100 : 59307	Activo	Activo	4
3	200.0.8.100 : 39381	200.0.0.100 : 59307	Activo	Activo	6
4	200.0.8.102 : 33226	200.0.0.100 : 59307	Activo	Activo	2
5	200.0.8.102 : 24513	200.0.0.100 : 59307	Activo	Activo	2
6	200.0.8.102 : 5511	200.0.0.100 : 59307	Activo	Activo	2
7	200.0.8.102 : 30117	200.0.0.100 : 59307	Activo	Activo	2
8	200.0.8.102 : 16822	200.0.0.100 : 59307	Activo	Activo	2
9	200.0.8.102 : 16321	200.0.0.100 : 59307	Activo	Activo	2

Figura 4.20: Establecimiento de subflujos con pares de direcciones IP repetidos.

Análisis de resultados

Se exponen las capacidades descritas de las cuales dispone el manejador de rutas a nivel de usuario. Estas capacidades se derivan del hecho de que un manejador de rutas de este tipo es programable por el usuario. De esta forma, a través de la programación se pueden eludir determinadas restricciones impuestas que sí son respetadas por el manejador de rutas que opera a nivel de kernel.

Un manejador de rutas a nivel de kernel optará por establecer subflujos con pares de direcciones IP no repetidos, respetando la cantidad máxima de subflujos permitidos. Esta cantidad máxima podrá tomar el valor 9 como máximo, ya que el comando *ip mptcp* así lo determina. Sin embargo, un manejador de rutas a nivel de usuario puede configurarse para ignorar todos estos parámetros, y operar en base a valores arbitrarios provistos por el usuario.

Se comprueba el funcionamiento de la operación MPTCP denominada como *Change Subflow Priority*. Se destaca que a través del manejador de rutas que opera a nivel de kernel no ha sido posible inducir a que se ejecute esta operación MPTCP. Por otro lado, el manejador de rutas a nivel de usuario puede ser programado para que ejecute esta operación cuando el usuario lo desee.

4.13. Caso de prueba 9: Topología fat-tree modificada

Manteniendo la restricción de que todas las interfaces de la topología disponen de la misma velocidad máxima de ancho de banda, en este caso de prueba se busca modificar la topología fat-tree para evitar que necesariamente todo el tráfico saliente y entrante se concentre en los enlaces que conectan cada host con su respectivo *edge switch*. De esta forma, se procede a validar si tal modificación impacta positivamente en el desempeño de MPTCP.

Descripción del caso de prueba

Se compara el rendimiento de conexiones de tipo Multi-Path TCP sobre la topología fat-tree con $k = 4$ original, frente a otra topología fat-tree con $k = 4$ modificada. Esta modificación consta de que todos los *edge switches* pertenecientes al pod i se conecten con todos los hosts que a su vez pertenecen al mismo pod. Por medio de este cambio, se logra que para cada par de hosts existan dos caminos completamente disjuntos que los conectan entre sí. La Figura 4.21 muestra una representación de esta nueva configuración asociada a cada pod.

Por otro lado, también se busca determinar si el rendimiento de las conexiones Multi-Path TCP se ve condicionado por el tipo de manejador de rutas que sea empleado, en el contexto de esta nueva topología fat-tree.

Se define H1 como host servidor y H9 como host cliente. Cada conexión MPTCP consta de exactamente 4 subflujos y se emplean tanto la topología fat-tree con $k = 4$ original, como la topología fat-tree con $k = 4$ modificada.

El procedimiento de este caso de prueba es el siguiente:

- *Paso 1*: para cada $b \in \{1, 5, 10\}$:

  Utilizando la topología fat-tree con $k = 4$ original y el manejador de rutas a nivel de kernel, establecer 5 conexiones MPTCP que transmitan b MBytes. En cada caso, registrar su duración y velocidad media para luego proceder a calcular promedios.

- *Paso 2*: idem *Paso 1*, pero empleando la topología fat-tree con $k = 4$ modificada y el manejador de rutas a nivel de kernel.

- *Paso 3*: idem *Paso 1*, pero empleando la topología fat-tree con $k = 4$ modificada y el manejador de rutas a nivel de usuario.

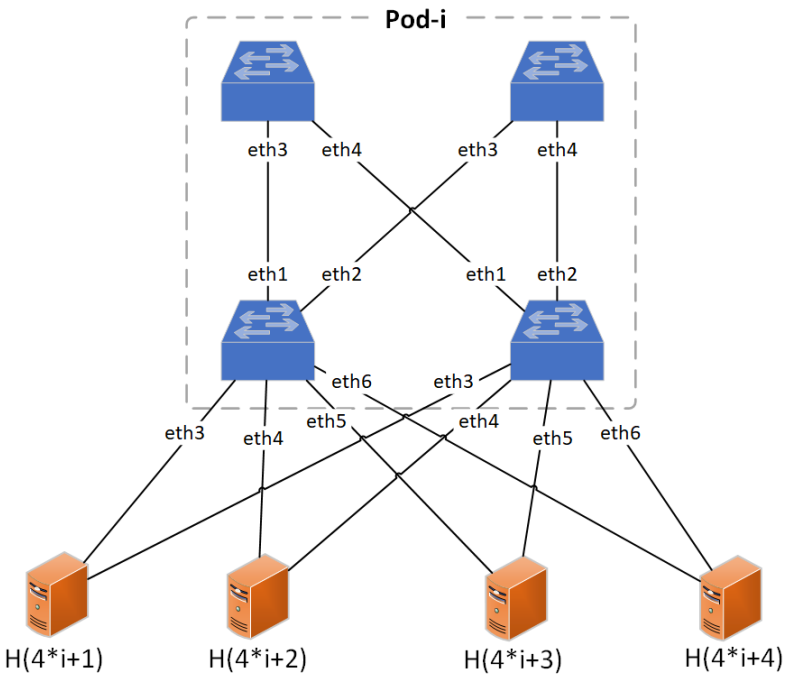

Figura 4.21: Estructura del pod número i de la topología fat-tree modificada.

Resultados obtenidos

En la Figura 4.22 se muestran tres tablas con los promedios de las mediciones realizadas para las distintas combinaciones de volumen de datos, topología y manejador de rutas que se utilizaron.

1 MB					
Topología Fat-Tree original		Topología Fat-Tree modificada			
		Manejador de rutas kernel		Manejador de rutas usuario	
Duración (sec)	Velocidad (Kbits/s)	Duración (sec)	Velocidad (Mbits/s)	Duración (sec)	Velocidad (Mbits/s)
10,87	771,80	6,88	1,23	7,31	1,15

5 MB					
Topología Fat-Tree original		Topología Fat-Tree modificada			
		Manejador de rutas kernel		Manejador de rutas usuario	
Duración (sec)	Velocidad (Kbits/s)	Duración (sec)	Velocidad (Mbits/s)	Duración (sec)	Velocidad (Mbits/s)
50,90	827,00	32,56	1,29	33,54	1,26

10 MB					
Topología Fat-Tree original		Topología Fat-Tree modificada			
		Manejador de rutas kernel		Manejador de rutas usuario	
Duración (sec)	Velocidad (Kbits/s)	Duración (sec)	Velocidad (Mbits/s)	Duración (sec)	Velocidad (Mbits/s)
99,35	844,60	64,16	1,31	64,31	1,31

Figura 4.22: Promedios obtenidos de las pruebas realizadas.

Análisis de resultados

En todos los casos, el protocolo Multi-Path TCP presenta un mejor rendimiento en la topología fat-tree con $k = 4$ modificada que en la original. Esta mejora se debe a que en la topología fat-tree original se cumplen las siguientes condiciones:

- La velocidad máxima de ancho de banda en todas la interfaces de cada componente es la misma.
- Cada conexión entre hosts implica una concentración de tráfico sobre los enlaces que conectan dichos hosts con sus respectivos *edge switches*. En adición, estos enlaces son los únicos que siempre experimentan este tipo de concentración.

En contraparte, en la topología fat-tree modificada siempre existen dos caminos completamente disjuntos que conectan cualquier par de hosts entre sí. De esta forma, la concentración de tráfico previamente mencionada pasa a ser repartida entre dos enlaces diferentes que pueden operan a la misma velocidad máxima de ancho de banda.

Al comparar los promedios de las velocidades logradas en la topología modificada, no se observa una diferencia significativa entre la utilización del manejador de rutas a nivel de kernel y el manejador de rutas a nivel de usuario.

5. Conclusiones y trabajo futuro

En el presente proyecto se estudió la implementación del protocolo Multi-Path TCP provista por el kernel de Linux, utilizando ambientes emulados. Inicialmente, se enfatizó en adquirir una comprensión global con respecto a la naturaleza y funcionamiento del protocolo para luego comenzar con una fase de experimentación.

Considerando las aptitudes que promete el protocolo, se estableció como marco de estudio una topología de red fat-tree, bajo la premisa de que Multi-Path TCP es capaz de aprovechar de forma más eficiente los recursos que ésta ofrece en comparación a otros protocolos de transporte tradicionales.

La topología de tipo fat-tree es característica de infraestructuras de red que corresponden a centros de datos masivos, implicando que entonces el empleo de Multi-Path TCP en estos contextos podría suponer un impacto positivo en cuanto a desempeño.

Al no disponer de una infraestructura de red real de tipo fat-tree sobre la cual realizar pruebas, se recurrió a la virtualización. De esta forma, la emulación ocupó un rol importante a lo largo del desarrollo del proyecto.

Otro tópico de interés sobre el que se invirtió esfuerzo, además de la mera experimentación con el protocolo, corresponde a la investigación de un medio que permitiese gestionar las conexiones de transporte Multi-Path TCP. Específicamente, se profundizó en la definición de un programa que opera como manejador de rutas siguiendo convenciones arbitrarias que pueden ser impuestas por un usuario.

Los principales resultados que se obtuvieron a través de experimentar con la implementación del protocolo Multi-Path TCP, utilizando un ambiente de trabajo emulado, corresponden a los siguientes:

- La implementación del protocolo Multi-Path TCP aún se encuentra en proceso de desarrollo. Durante su estudio, se pudo comprobar que sus operaciones denominadas como *Multipath Capable*, *Join Connection*, *Data Sequence Signal*, *Add Address*, *Remove Address*, y *Change Subflow Priority* se encuentran actualmente implementadas.

- El desempeño del protocolo Multi-Path TCP está fuertemente condicionado a la infraestructura de red sobre la cual opera. Aquellas que presentan múltiples rutas potencialmente disjuntas entre sus pares de hosts son las que permiten aprovechar de mejor manera los servicios que brinda este protocolo.

  A su vez, también se requiere de la utilización de un correcto algoritmo de enrutamiento que permita distribuir eficientemente los flujos de transporte que sean establecidos entre las distintas rutas disponibles.

- Al comparar los desempeños de los protocolos Multi-Path TCP y TCP estándar, en relación a la topología fat-tree con parámetro $k = 4$ y un ancho de banda limitado a 1000 Kbits/seg, surgieron los siguientes resultados:

  - En general, las conexiones MPTCP de un solo subflujo no desempeñan de forma más eficiente que las conexiones TCP.

    A su vez, las conexiones MPTCP con cualquier cantidad de subflujos asociados no resultan más eficientes que las conexiones TCP cuando el volumen de datos a enviar es menor a 32 KBytes.

  - Cuando la red no presenta congestión, una conexión MPTCP con varios subflujos tiende a obtener una mayor proporción de ancho de banda que una conexión TCP.

    Este comportamiento se registra cuando ambas conexiones ocurren al mismo tiempo y compiten entre sí, y también cuando ambas conexiones ocurren en distintos intervalos de tiempo sin que hayan solapamientos.

  - Cuando la red presenta congestión, independientemente de que ésta sea ocasionada por conexiones MPTCP o TCP, las conexiones MPTCP con varios subflujos logran un desempeño notoriamente superior con respecto a la conexiones TCP.

Esto puede deberse, entre otros aspectos, a la capacidad del protocolo MPTCP de redistribuir la carga como consecuencia de detectar congestión en la red.

- La velocidad máxima de ancho de banda que se permite en la infraestructura sobre la cual opera una conexión MPTCP puede afectar la distribución de tráfico de datos entre sus subflujos. Se observa la tendencia de que entre mayor sea el ancho de banda permitido, mayor resultará la cantidad de tráfico concentrado entre los primeros subflujos de la conexión en haber sido establecidos.

  Esta condición implica que, a partir de determinado punto, la incorporación de nuevos subflujos para una conexión MPTCP puede resultar innecesaria, o incluso contraproducente, ya que estos nuevos subflujos se limitarán a transportar una cantidad insignificante de tráfico de datos.

- Un manejador de rutas que opera a nivel de usuario resulta especialmente útil cuando se desean instaurar políticas arbitrarias sobre las conexiones Multi-Path TCP. En otras palabras, un manejador de rutas de este tipo permite personalizar con mayor detalle aspectos específicos que están relacionados con la operativa de las conexiones Multi-Path TCP, permitiendo la implementación y ejecución de distintas políticas de calidad de servicio.

  Al ser programable, un manejador de rutas a nivel de usuario puede incluso configurarse para eludir restricciones que sí le son impuestas al manejador de rutas que opera a nivel de kernel.

En términos generales, Multi-Path TCP consiste en un protocolo relativamente nuevo que promete un gran potencial producto de su capacidad de operar mediante múltiples flujos de transporte en forma simultánea. Su desempeño se ve favorecido cuando la infraestructura red involucrada permite que sus flujos de transporte se mapeen a rutas efectivamente disjuntas.

Al tratarse de un protocolo reciente y en proceso de desarrollo, sobre el cual no existe una cantidad extensa de información, los principales desafíos que se presentaron durante la realización de este estudio consistieron en encontrar una implementación funcional del protocolo y capacitarse para operar con ésta. Por otro lado, la investigación e implementación de un programa manejador de rutas también correspondió a una de las tareas más desafiantes, insumiendo un cantidad considerable de esfuerzo.

En este punto, conviene reconocer que se logró realizar un correcto estudio del protocolo Multi-Path TCP. Se adquirieron conocimientos sólidos en cuanto a la definición y operativa del protocolo que luego fueron empleados para la especificación y ejecución de diversos casos de prueba. Estos casos de prueba posicionaron al protocolo Multi-Path TCP en distintos escenarios que permitieron evaluar su desempeño. Adicionalmente, se profundizó en comprender la naturaleza implicada en la gestión de las conexiones Multi-Path TCP, logrando así confeccionar un programa que fuese capaz de sobrellevar esta tarea en base a convenciones impuestas de forma arbitraria.

No obstante, también se consideraron otros enfoques de estudio que no pudieron ser finalmente realizados debido a que las emulaciones implicaron más tiempo del estimado. En particular, se destaca la simulación como uno de los enfoques que quedaron sin ser explorados.

La simulación puede entenderse como el proceso de diseñar un modelo de un sistema real para luego proceder a experimentar con éste y así lograr describir, explicar y predecir el comportamiento del sistema real (Wainer, s.f.). En relación a Multi-Path TCP, este enfoque resulta interesante debido a que brinda la posibilidad de estudiar el protocolo prescindiendo de su implementación. En otras palabras, permite la construcción y estudio de un modelo que se fundamente en las especificaciones del protocolo y no en las capacidades y limitaciones de sus implementaciones vigentes.

De esta forma, a modo de complementar los resultados expuestos en este proyecto, se propone como trabajo futuro la implementación y ejecución de simulaciones que modelen el comportamiento del protocolo Multi-Path TCP.

Por otro lado, también se plantea como una posible investigación a futuro el estudio de distintas topologías de red que aprovechen de forma óptima las propiedades de Multi-Path TCP.

Finalmente, se propone la implementación de un manejador de rutas estable que opere a nivel de usuario. Sin embargo, se entiende que para la realización correcta de esta tarea se deberá esperar a que la implementación del protocolo Multi-Path TCP esté más avanzada.

Las implementaciones realizadas durante el desarrollo de este proyecto pueden encontrarse en (Braica y Bruzzese, 2023) para ser empleadas como punto de partida de nuevas investigaciones.

Referencias

Abeni, P. (2020). *mptcp-tools.* https://github.com/pabeni/mptcp-tools.

Alberro, L., Castro, A., y Grampin, E. (2022). Experimentation environments for data center routing protocols: A comprehensive review. *Future Internet*, *14*(1). Descargado de https://www.mdpi.com/1999-5903/14/1/29 doi: 10.3390/fi14010029

Baerts, M., y cols. (2022). *Linux mptcp upstream project.* https://github.com/multipath-tcp/mptcp_net-next/.

Braica, M., y Bruzzese, N. (2023). *Proyecto grado mptcp.* https://gitlab.fing.edu.uy/mauricio.braica.alcalde/proyecto_grado_mptcp/-/tree/main/.

Canonical Ltd. (2019a). *ip-mptcp - mptcp path manager configuration.* https://manpages.ubuntu.com/manpages/jammy/man8/ip-mptcp.8.html.

Canonical Ltd. (2019b). *namespaces - overview of linux namespaces.* https://manpages.ubuntu.com/manpages/jammy/man7/namespaces.7.html.

Canonical Ltd. (2023). *Download ubuntu desktop.* https://ubuntu.com/download/desktop.

Cisco. (2022). *What is a data center?* https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html. (Último acceso: 23-11-2022)

Combs, G., y cols. (2022). *Wireshark.* https://www.wireshark.org/.

Eddy, W. (2022, Agosto). *Transmission Control Protocol (TCP)* (RFC n.º 9293). Internet Requests for Comments. Descargado de https://www.rfc-editor.org/rfc/rfc9293.txt

Ford, A., y cols. (2020, Marzo). *TCP Extensions for Multipath Operation with Multiple Addresses* (RFC n.º 8684). Internet Requests for Comments. Descargado de https://www.rfc-editor.org/rfc/rfc8684.txt

FRRouting Project Contributors. (2022). *Frrouting.* https://frrouting.org/.

Kurose, J. F., y Ross, K. W. (2010). *Redes de computadoras: Un enfoque descendente.* Madrid, España: Pearson.

Kuznetsov, A. (2015). *Ifstat.* https://man7.org/linux/man-pages/man8/ifstat.8.html.

Martineau, M., y Othman, O. (2020). Using upstream mptcp in linux systems.

McMahon, R. (2021). *Iperf2.* https://sourceforge.net/projects/iperf2/.

Medhi, D., y Ramasamy, K. (2017). Network routing. En (cap. 12). Morgan Kaufmann.

Mininet Project Contributors. (2022). *Mininet: An instant virtual network on your laptop (or other pc).* http://mininet.org/.

Mulhollon, V. (2004). *Wondershaper.* https://manpages.ubuntu.com/manpages/jammy/man8/wondershaper.8.html.

Othman, O., y cols. (2023). *Multipath tcp daemon.* https://github.com/multipath-tcp/mptcpd.

Raiciu, C., y cols. (2011). Improving datacenter performance and robustness with multipath tcp.

Red Hat. (2023). *¿qué es una api y cómo funciona?* https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces.

The kernel development community. (2023). *Mptcp sysfs variables.* https://dri.freedesktop.org/docs/drm/networking/mptcp-sysctl.html.

The Tcpdump Group. (2023). *Tcpdump.* https://www.tcpdump.org/.

UCLouvain. (2022). *Multipath tcp - linux kernel implementation.* https://www.multipath-tcp.org/.

VMware. (2023). *Vmware workstation pro.* https://www.vmware.com/latam/products/workstation-pro/workstation-pro-evaluation.html.

Wainer, G. A. (s.f.). *Introducción a la simulación de sistemas de eventos discretos.* https://twiki.cern.ch/twiki/pub/Main/DEVSTheoryLinks/APUNT.PDF.

Wikipedia. (2022). *Emulador.* https://es.wikipedia.org/wiki/Emulador.

Anexos

Anexo 1: Topología fat-tree Utilizada para Pruebas