UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

CURSO 2017

Caché transparente para protocolos P2P

Autor:
Rodrigo Tartajo Martínez

Tutor:
Francisco Serradilla

Julio 2017

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Protocolos centralizados y protocolos P2P

El uso medio de ancho de banda por hogar conectado a la red Internet no ha hecho sino incrementarse con cada año que pasa. Del mismo modo que la cantidad de datos consumido por usuario ha ido incrementándose, el perfil de este tráfico también ha evolucionado. Los ISPs (proveedores de servicios de internet), empresas cuyo negocio es la venta de ancho de banda al por menor tanto a hogares como a negocios, usan diferentes técnicas para intentar extraer el mayor margen de beneficio por contrato, intentan comprar el menor ancho de banda necesario para dar servicio a sus clientes. Para compensar este hecho, es común el uso de diferentes técnicas:

- Asimetría de ancho de banda: los contratos con los carriers (proveedores de ancho de banda al por mayor, y enlaces internacionales) que establecen los ISPs suelen poseer diferentes precios para el tráfico entrante y para el saliente. Los ISP intentando mantener una balanza equilibrada de consumo de tráfico (lo que se traduce en un precio medio menor para el tráfico) ofrece a las conexiones destinadas al hogar un ancho de banda de subida menor que el de bajada, pues el perfil de consumo de ancho de banda de los hogares es principalmente de bajada. Dado que el ISP compra capacidad de red de forma simétrica, el ancho de banda de subida restante es vendido a empresas de alojamiento Web o servicios de internet, que al ser el origen de la mayor

parte del tráfico consumido por una conexión de un hogar, tiene un perfil opuesto y complementario: principalmente de subida.

- Sobre-subscripción: los ISP al comprar ancho de banda de los carriers, y vender conexiones a los hogares, no suele comprar suficiente ancho de banda para proveer del total del ancho de banda contratado a todos su clientes simultáneamente. Esto es debido a que el perfil de consumo de ancho de banda de un hogar suele ser muy irregular y aunque puntualmente puede que cada cliente del ISP esté en uso de todo el ancho de banda contratado, este uso máximo no se suele mantener durante mucho tiempo ni ser simultáneo entre todos sus clientes. Un ejemplo de esto es que durante las horas nocturnas, el uso de ancho de banda de los hogares suele ser mínimo.

- Acuerdos de interconexión: el lugar físico de encuentro entre los ISP y los carrier suele realizarse en unos pocos lugares para una misma región, normalmente nacional. Es en estos lugares, denominados "Puntos neutros" es donde los diferentes ISPs y carriers interconectan sus redes para realizar el intercambio de tráfico. Dado que todo tráfico que un carrier acarrea es facturado, incluso si este tráfico no sale del punto neutro al tener tanto su origen como destino en dos ISP presentes en el punto neutro, muchos ISP interconectan sus redes directamente para intercambiar tráfico entre ellas sin que un carrier medie en ellas, lo que conlleva una reducción en sus facturas.

- Uso de cachés a nivel ISP: históricamente, algunos ISP han intentado usar cachés, principalmente para el protocolo HTTP, para así eliminar la descarga múltiple del mismo elemento desde el servidor remoto para múltiples clientes. Por desgracia esto rompe ciertos comportamientos del protocolo HTTP (contenido dinámico o dependiente de sesión), y no es aplicable al resto de protocolos.

Los protocolos encontrados en el tráfico de Internet puede ser clasificado principalmente en dos categorías:

- Cliente-Servidor: la mayor parte del volumen actual en el perfil de tráfico de una conexión de un hogar es del tipo cliente-servidor. Este tipo de tráfico

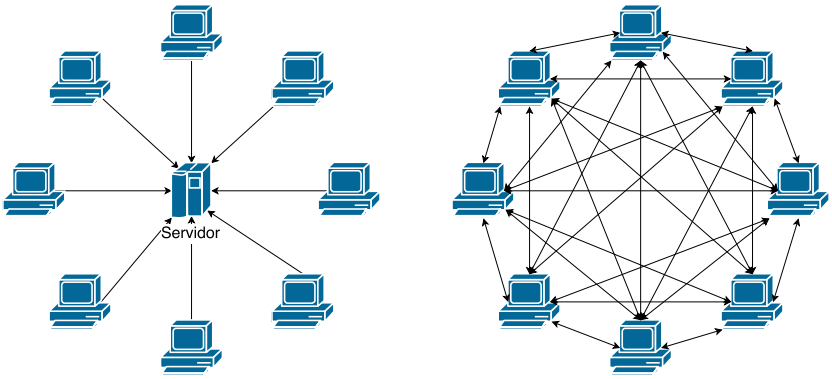

Figura 1.1: Representación de los pares en una red cliente-servidor, y para una red P2P.

se caracteriza por ser asimétrico tanto en su contenido (donde cada extremo de la conexión realiza un papel diferente) como en su volumen (mayoritariamente tráfico de servidor a cliente, donde el cliente suele ser el hogar). En este tipo caben destacar los protocolos de streaming multimedia y web. Aunque los clientes suelen ser cualquier hogar, los servidores con mayor consumo abarcan una lista muy reducida, como puede ser Youtube, Netflix o Facebook.

- P2P: a diferencia del anterior este tráfico se caracteriza por su simetría: cada extremo de una conexión se comporta idénticamente con el otro extremo y la proporción entre tráfico entrante y saliente suele ser equivalente. En este caso no existe una agrupación del grueso de tráfico por parte de ningún servidor o segmento de la red.

Tanto los principales proveedores de contenido por volumen, como los ISP, intentan reducir el tráfico que intercambian con los carriers para reducir el coste monetario que les supone operar. Para ello han aparecido las redes de distribución de contenido (CDN), las cuales crean copias locales de los contenidos más populares en puntos cercanos a las redes de los ISPs, sin que existan carriers como intermediarios, lo que permite a los ISP obtener estos contenidos sin tener que pagar a los carriers. Los principales creadores de contenido usas sus propios CDN privados, mientras que otros más humildes usan los servicios CDN ofertados por terceros. Esta técnica solo es posible para protocolos cliente-servidor, al

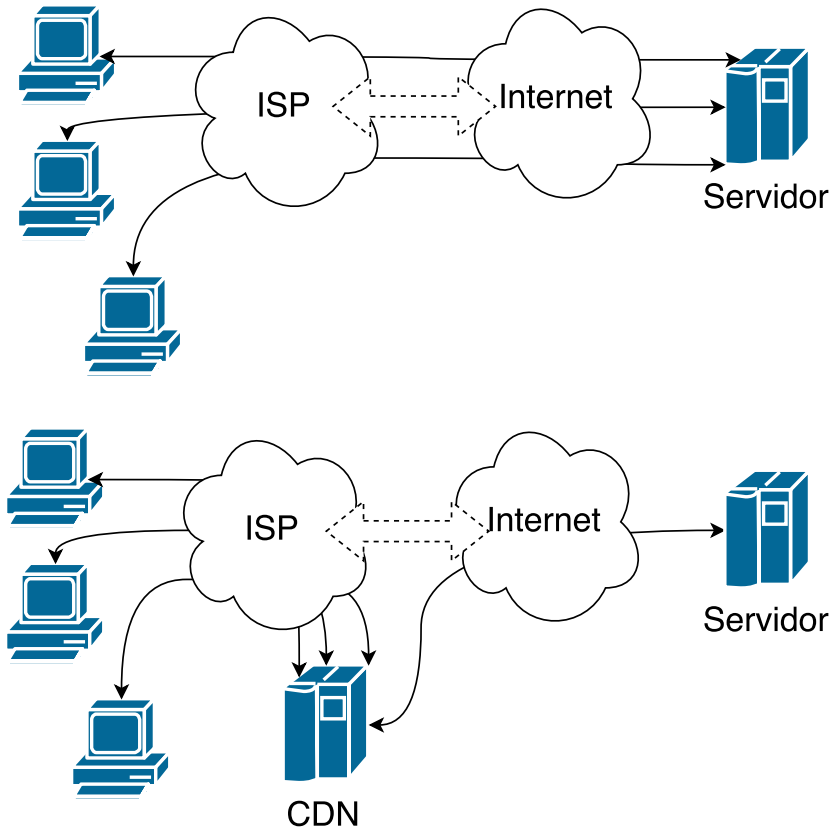

Figura 1.2: Representación del uso de CDNs para la distribución de contenidos digitales.

ser redireccionada a la instancia del CDN local, parte de las peticiones que serían enviadas al servidor principal.

En cambio para protocolos P2P la situación es la opuesta, pues al tratarse de protocolos descentralizados no existe una concentración de conexiones en ningún servidor, y dado que históricamente estos protocolos han sido usados para la transferencia de ficheros de dudosa legalidad, el uso de puertos no estándar ha sido una de las primeras técnicas usadas para intentar dificultar la detección del protocolo.

1.2. Evolución del uso de la red Internet

Con el aumento del ancho de banda ofertado en las conexiones a Internet de los hogares, el patrón de uso del ancho de banda se ha ido transformando, habi-

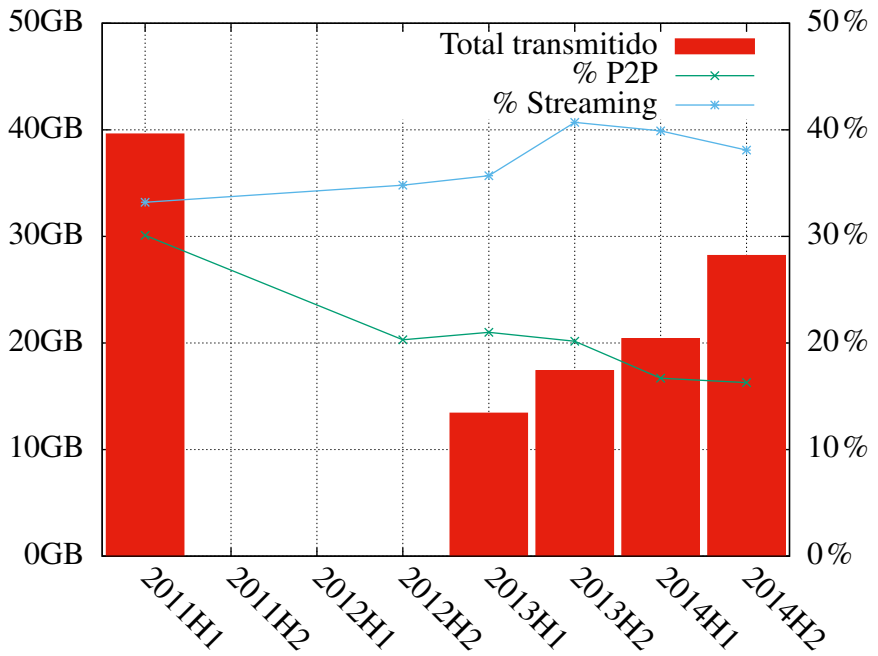

Figura 1.3: Evolución del consumo medio mensual de tráfico de Internet por hogar europeo [10].

litando nuevos servicios que anteriormente habrían sido demasiado limitados/inconvenientes con conexiones de menor ancho.

Si bien el uso de las conexiones domésticas pronto alcanzó los requisitos para la compartición de ficheros por Internet, fue la llegada de las distintas generaciones de protocolos P2P lo que permitió que se extendiera su uso.

Finalmente cuando las conexiones alcanzaron la suficiente velocidad, los servicios de streaming multimedia (Netflix, Youtube) pudieron irrumpir en el hogar ofreciendo una calidad más que aceptable de vídeo en tiempo real, lo que supuso una competencia directa con las redes P2P para la distribución de este tipo de contenidos.

Capítulo 2

Objetivos

2.1. Objetivo

El objetivo del proyecto descrito en este documento consiste en una solución software que implementa una caché para protocolos P2P. La solución, mediante la intercepción de tráfico en red, permitirá reducir el volumen de tráfico intercambiado entre los pares de una conexión P2P, sin que por ello se vea afectada de forma aparente que la comunicación entre los pares ha sido modificada. Las operaciones de modificación de la conexión P2P consistirán en:

- Inclusión de las partes presentes en la caché al intercambiar los conjuntos de partes ya descargadas: Cuando dos pares intercambian la lista de partes que cada uno tiene disponibles para ser compartidas, la caché modificará el mapa enviado al par en el lado interno, de modo que este contenga tanto la lista que el par en el lado externo, como las que están presentes en la caché.

- Interceptación y envío de partes en la caché: cuando el par en el lado interior envía una petición de descarga de una parte presente en la caché, esta petición será eliminada de la conexión entre los pares, y la parte correspondiente a esta petición será inyectada para ser recibida por el par. Peticiones de descargas desde el par en el lado exterior o partes no presentes en la caché, serán enviadas sin modificación.

- Adquisición transparente de contenido: cuando una petición no ha sido in-

terceptada, la respuesta, si se ve conveniente, será copiada a la caché para ser usada para satisfacer futuras peticiones de otros pares de la misma descarga.

El tráfico correspondiente a otros protocolos no soportados por la caché será enviado sin ningún tipo de procesado.

2.2. Especificación de la solución

Dadas las características del tráfico P2P, el único modo de poder cachear archivos compartidos es mediante un sistema en línea, que intercepte la comunicación entre pares y se encargue de leer e inyectar las partes cacheadas de los ficheros compartidos.

2.3. Características de los protocolos P2P

A diferencia de otros protocolos, donde cada extremo de la comunicación sirve un papel muy determinado, los pares en una conexión de un protocolo P2P se comportan de forma completamente simétrica, enviando y recibiendo los mismos mensajes. Otra característica es que al no ser ninguno de los extremos un servidor, no existe una concentración de conexiones a o desde ningún par. En el caso específico del protocolo BitTorrent, un par no intenta tener una conexión con cada uno de los otros pares conocidos compartiendo la misma descarga, sino que intenta mantener un subconjunto de estos que presentan un comportamiento más generoso al compartir partes de la descarga.

2.4. Entorno de despliegue

El entorno de despliegue es un entorno Linux, versión mínima 2.6.23, con al menos dos interfaces Ethernet. Estas dos interfaces serán enlazadas por el servicio de cachéo P2P, haciendo que todo tráfico que entre por una, será enviado por la otra. La velocidad y medio usado por los interfaces Ethernet no es relevante, aunque estas han de estar configurados con una MTU no superior a 1500 bytes y a ser

posible, ambas interfaces han de operar a la misma velocidad. Estas dos interfaces son configuradas como interior y exterior en la aplicación, proporcionándose el servicio de cachéo solo a los extremos de la conexión que se encuentran del lado de la interfaz marcada como interior.

La caché también hace uso de del espacio en disco para almacenar las partes de los ficheros extraídos de la red P2P para posteriormente inyectarlos. Para aliviar la carga de red a tratar por la caché P2P, aunque esta se encuentra optimizada para usar el menor número de recursos para la gestión de conexiones no relacionadas con el tráfico P2P, solo ha de desviarse hacia la caché P2P el tráfico que no ha podido ser clasificado como ningún otro protocolo o servicio conocido: mediante clasificación por número de puerto (HTTP en el puerto TCP 80, DNS en el puerto UDP 53, etc...) y subredes conocidas (servidores de servicios con gran uso de ancho de banda: Youtube, Netflix, Akamai)

2.5. Interacción con la red

La caché P2P analizara los paquetes entrantes de las conexiones durante su inicio, tanto para evitar posibles incompatibilidades como para detectar si estas conexiones pertenecen o no a protocolos soportados. Una vez una conexión es clasificada como en uso de un protocolo soportado, esta pasa a ser gestionada por la caché. La caché puede modificar el contenido de la conexión, pero nunca modificará su estado: desconectando o manteniendo la conexión como conectada de forma artificial.

Capítulo 3

Teoría

3.1. La pila de protocolos

Dado que la comunicación a gestionar se realiza mediante el uso de redes informáticas, los datos transportadas por estas se encuentran encapsulados en múltiples protocolos, cada uno encargado de una tarea diferente. A continuación se enumeran la pila de protocolos que se espera encontrar para el correcto funcionamiento de la aplicación.

3.2. Ethernet

El estándar Ethernet, definido y refinado el la familia de documentos técnicos IEEE 802.3 [1], se encarga de las tareas asignadas a los protocolos de capa 1 (capa física) y capa 2 (capa de enlace). Dado que de los detalles de la capa física se encarga tanto el hardware como el controlador de la interfaz de red, los detalles que nos interesa es el formato de la PDU y sus limitaciones. El marco Ethernet está formado por una cabecera con las direcciones Ethernet de destino y origen (6 octetos cada una) y dos octetos más usados para indicar la longitud total del marco. Dado que versiones más modernas del estándar Ethernet permiten detectar la longitud del marco a nivel de capa física, este campo es reusado para almacenar un indicador del protocolo en la capa inmediatamente superior.

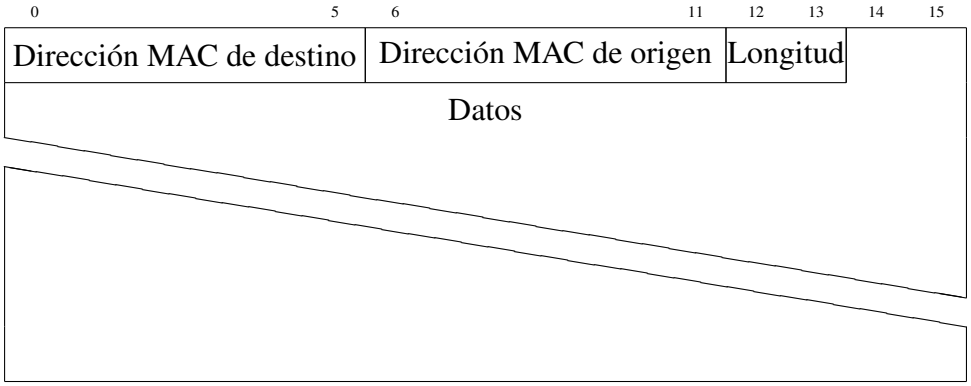

Figura 3.1: Estructura de un marco Ethernet

3.3. IP

El protocolo de internet (IP) [7] es el equivalente a la capa de red de la pila OSI, encargado de enumerar los pares de una red, y el enrutado de los paquetes entre estos. En el caso del protocolo IP en su versión 4, el espacio de direcciones es de 2^{32} distintos valores. El protocolo IP no asegura la recepción de los datagramas, ni evita su posible duplicación durante su propagación.

3.3.1. Estructura de la cabecera IP

Descripción de los campos:

Versión: Indica la versión del protocolo. Solo se han especificado las versiones 4 y 6.

IHL: Indica la longitud de la cabecera IP: incluye las opciones pero no los datos, en número de palabras de 32 bits.

DSCP: Indicador de tipo de servicio, más detalles en [5].

ECN: Indicador explícito de congestión, con uso no muy extendido. Para más detalles véase [9].

Longitud total: Longitud total del datagrama, incluyendo cabecera, opciones y datos.

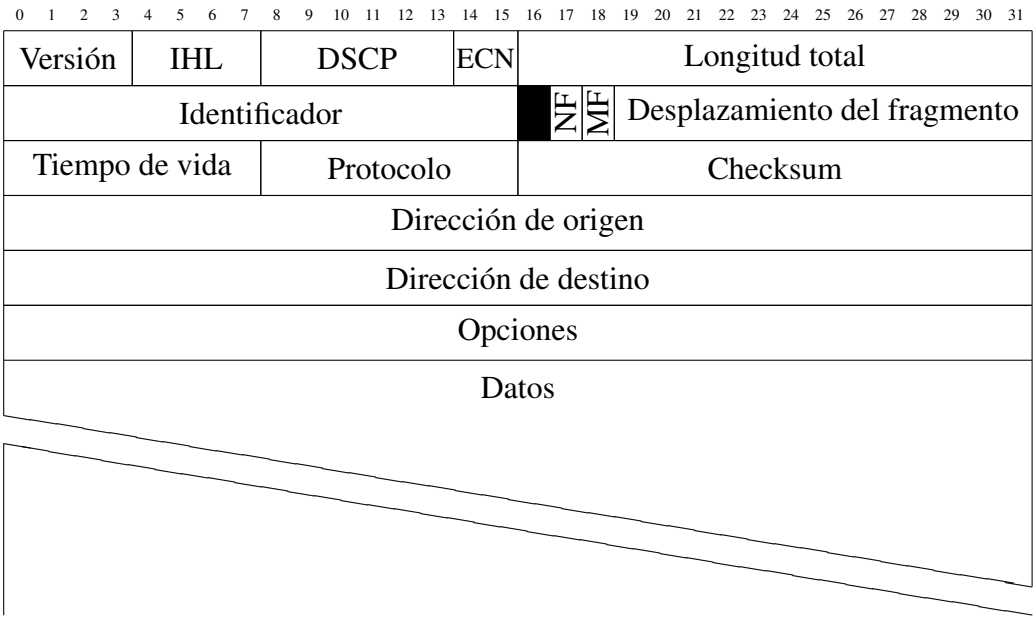

Figura 3.2: Cabecera de un datagrama IP

Identificador: Identificador único del datagrama. Común a todos los fragmentos.

Tiempo de vida: Contador de saltos.

Protocolo: Identifica el protocolo en la capa superior.

Checksum: Usado para detectar la posible corrupción del datagrama.

Dirección de origen: Dirección IP de origen del datagrama

Dirección de destino: Dirección IP de destino del datagrama.

Opciones: campo opcional de longitud variable con las opciones IP presentes

Datos: Posibles datos transportados en el datagrama.

Más información sobre los detalles de uso de los campos se pueden encontrar en los diferentes RFC.

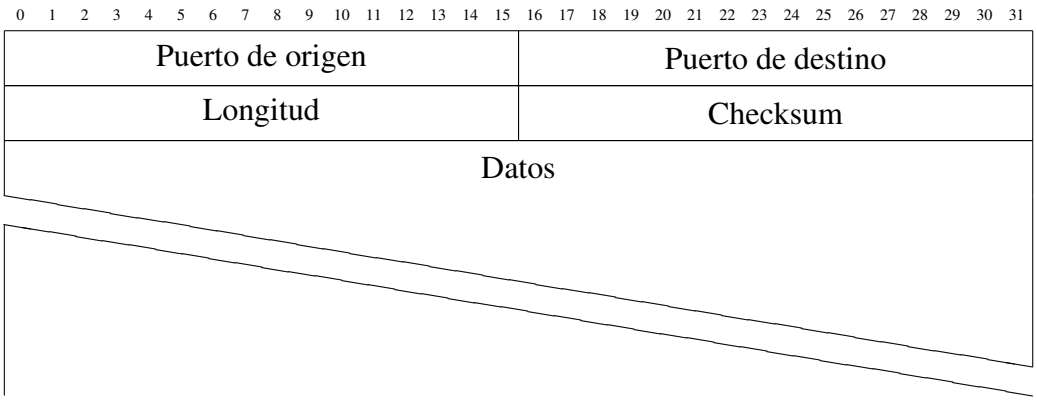

Figura 3.3: Estructura de un datagrama UDP

3.4. UDP

El protocolo más simple que se puede encontrar sobre la capa IP es el protocolo UDP [6]. Este protocolo no provee de ningún mecanismo para asegurar la detección de la posible pérdida, reordenación o duplicación de paquetes, pero es muy simple de implementar y muy ligero de usar. Sus características son:

- No orientado a conexión: no es necesario establecer una conexión antes de empezar a transmitir datos.
- Orientado a mensajes: los datos escritos en una única operación serán recibidos sin ser divididos o fusiones con los datos de operaciones de escritura anteriores o posteriores.
- El uso de números de puerto permite establecer múltiples canales de comunicación independientes entre dos mismos pares.
- Es posible su utilización en redes “multicast”.

Descripción de los campos, todos los enteros de 2 octetos se codifican en formato Big-Endian (octeto de mayor peso primero):

Puerto de origen : entero de 16 bits indicando el puerto de origen del datagrama.

Puerto de destino : entero de 16 bits indicando el puerto de destino.

Longitud : longitud de los datos de usuario en el datagrama.

Checksum : código de control para detectar la corrupción de datagrama durante su transmisión.

3.5. TCP

Si el protocolo IP proveía de enrutamiento de tráfico entre puntos en la red, el protocolo TCP [3] [8] provee la capacidad de crear flujos de datos entre ellos. Algunas características del protocolo TCP son:

- Fiable: permite la detección de errores tales como: pérdidas, reordenación, repetición y corrupción en el flujo de datos.
- Orientado a conexión: antes de la transmisión de datos, la conexión ha de negociarse, permitiendo así comprobar que ambos extremos de la conexión están listo, y qué opciones han de usarse.
- Transmite flujo de datos: al no orientarse a mensajes, aunque se asegura que todos los datos enviados serán recibidos, no así se respetaran los tamaños de las escrituras de datos en su recepción.
- Abstracción a las capas superiores: aísla a las aplicaciones de los detalles de la implementación: fragmentación, reenvío de datos perdidos o corruptos se gestionan de forma transparente.
- Prioriza fiabilidad sobre latencia: algunos de los comportamientos y algoritmos aconsejados en la implementación del protocolo puede incidir en un aumento de la latencia de transferencia para asegurar la correcta transmisión de datos.

Descripción de los campos, todos los enteros de 2 o 4 octetos se codifican en formato Big-Endian (octeto de mayor peso primero):

Puerto de origen : entero de 16 bits indicando el puerto de origen del segmento.

Puerto de destino : entero de 16 bits indicando el puerto de destino.

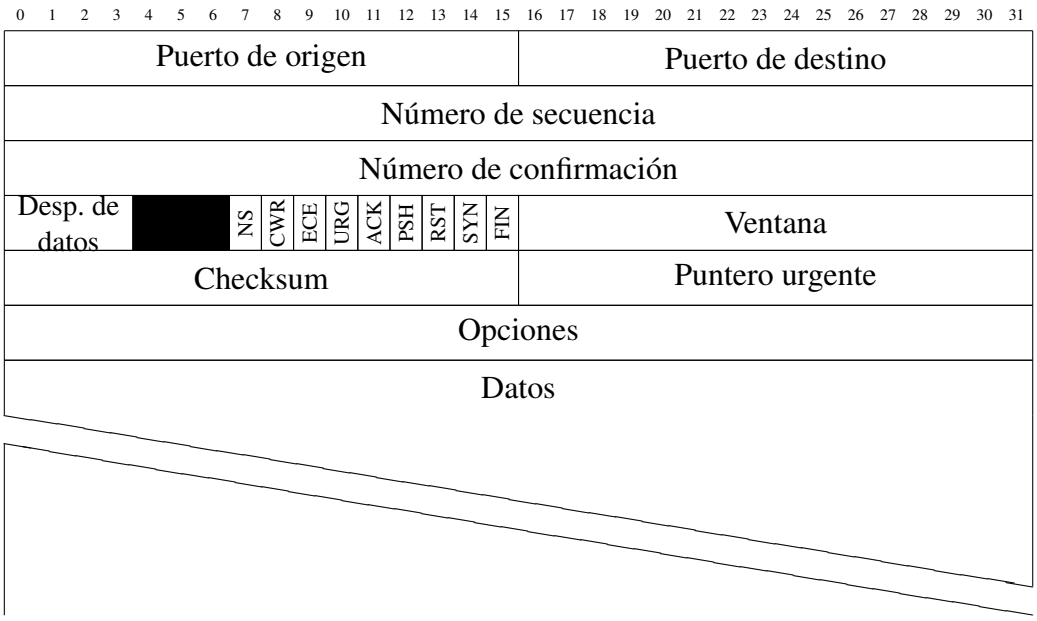

Figura 3.4: Cabecera de un segmento TCP

Número de secuencia : indica el número de secuencia del segmento, usado para ordenar y detectar la pérdida de segmentos.

Número de secuencia de confirmación : cuando la bandera "ACK" está activa, este campo contiene el número de secuencia del próximo segmento que se espera recibir.

Desplazamiento de datos : cuenta en número de palabras de 32 bits, el tamaño total de la cabecera TCP.

Bits sin uso : 3 bits que actualmente no tiene uso.

Banderas NS, CWR y ECE : banderas para la gestión de la notificación explícita de congestión [11] [9].

URG : bandera usada para indicar que el puntero de datos urgentes está en uso.

ACK : bandera que indica el uso del campo de número de secuencia de confirmación.

PSH : bandera que indica que los datos han de ser enviados inmediatamente, reduciendo el uso de bufferes intermedios. En el emisor esto significa no esperar a llenar el buffer de envío antes de volcarlo a la red, en el receptor significa que la lectura del buffer es inmediata.

RST : bandera de reset, terminación inmediata de la conexión.

SYN : bandera usada para indicar que se está acordando el establecimiento de una nueva conexión.

FIN : bandera que indica la finalización ordenada de un extremo de la conexión.

Ventana : indica cuántos octetos este extremo de la conexión está dispuesto a aceptar.

Checksum : código de comprobación para la detección de datos corruptos.

Puntero urgente : indica el desplazamiento dentro de los datos del segmento, de los datos urgentes. Los datos marcados como urgente pueden ser tratados fuera de orden.

Opciones : campo opcional con las opciones del segmento. Para asegurar que el tamaño total de las opciones es múltiplo de 32 bits, se rellenan los octetos sin uso con el valor 0x00.

3.5.1. Ciclo de vida

Como ya se indicó en su descripción, el protocolo TCP es orientado a conexión, por lo que es necesario seguir una serie de pasos para el establecimiento y destrucción de una nueva conexión.

Establecimiento de conexión

El establecimiento de conexión se realiza mediante el intercambio de tres paquetes:

1. Un paquete con la bandera “SYN” puesta a 1, sin la bandera “ACK”, un número de secuencia aleatorio y sin datos es enviado por el extremo que desea establecer una nueva conexión con otro par en la red IP.

2. En caso de que el otro par acepte la conexión, responderá con un paquete con las banderas “SYN” y “ACK” puestas a 1, un número de secuencia aleatorio, un número de confirmación igual al número del primer paquete, pero incrementado en uno.

3. Finalmente el par que inició la conexión responde al segundo paquete con un paquete de confirmación a su número de secuencia, más uno.

Tras el establecimiento de la conexión ya se puede proceder al envío de datos. Los paquetes con la bandera ACK puesta a uno indican en su campo “Ventana” la cantidad máxima de datos que el extremo de la conexión están dispuestos a aceptar.

Finalización ordenada de la conexión

Cuando un par decide finalizar una conexión, se usa un paquete con la bandera FIN puesta a uno. Esto indica que el extremo de la conexión ya no enviará más datos, pero no así el que no vaya a aceptar más. La detección de la bandera FIN incrementa el número de confirmación en uno. Solo cuando ambos extremos han enviado un paquete con la bandera FIN y estos han sido confirmados, es cuando se puede marcar la conexión como finalizada y sus recursos liberados.

Finalización anómala de la conexión

En caso de que se detecte un error irreconciliable en la conexión, la conexión puede ser abortada inmediatamente enviando un paquete con la bandera “RST”, lo que provoca la inmediata destrucción de la conexión y la liberación de sus recursos. El paquete con la bandera “RST” debe ir en secuencia para ser aceptado.

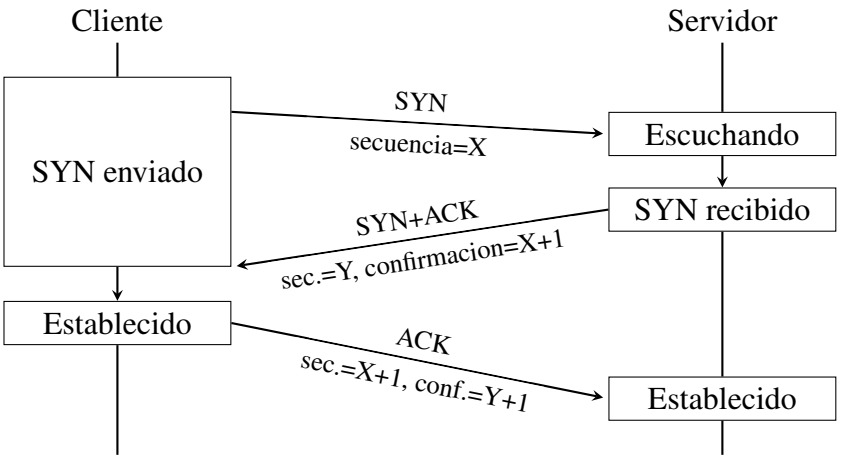

Figura 3.5: Diagrama de estados durante el establecimiento de una conexión TCP.

3.5.2. Control de flujo

A continuación se explica la lógica usada para asegurar que los datos transmitidos son recibidos sin problemas por los pares, asegurando la ordenada y correcta recepción de datos. El campo con el número de secuencia es usado para poder calcular el orden de los datos de un segmento en el flujo, así como la posible reordenación de segmentos sin datos. Este número inicializado a un valor aleatorio en cada extremo de la conexión al establecerla es incrementado con por cada octeto recibido en orden, así como por las banderas “SYN” y “FIN”. Cuando nuevos datos son recibidos en orden, el campo de número de secuencia de confirmación es usado para indicar cual es el número de secuencia del próximo octeto que se espera recibir.

La recepción de segmentos cuyos datos ya se habían confirmado son ignorados, pero si se reciben segmentos con un número de secuencia que para estar en orden nos hace falta algún segmento anterior, se interpreta como una posible pérdida en la red, y para forzarla retransmisión desde el otro extremo se envían un segmento de confirmación (sin datos, pero con la bandera “ACK” establecida) para indicar con el campo de confirmación el número de secuencia que parece haber ser perdido. Si el otro extremo recibe múltiples segmentos de confirmación con el mismo número de confirmación que ya se había enviado, se procede a retransmi-

tir todos los segmentos empezando por aquel con el octeto cuya secuencia se está pidiendo.

Dado que los extremos de la conexión pueden no estar consumido los datos a la misma velocidad a la que se reciben, el campo de ventana se usa para controlar cuántos datos está el par de la conexión listo para recibir. Este campo almacena un valor absoluto, pero junto con el número de confirmación puede ser usado para calcular la secuencia del último octeto que se está listo para recibir. Cuando la aplicación consume datos del buffer de recepción, un paquete de confirmación es transmitido con el nuevo valor del campo ventana.

Técnicas adicionales se han especificado para intentar aliviar una red congestionada [2], cuando se detecta una pérdida en la red.

3.6. El protocolo P2P BitTorrent

A diferencia de anteriores protocolos P2P, BitTorrent separa en protocolos muy diferentes las distintas fases de la distribución de una descarga en su red. La especificación oficial del protocolo se puede encontrar en [4].

3.6.1. El fichero de metadatos

Este fichero contiene la información necesaria acerca de una descarga disponible en la red P2P. Se suele distribuir como un fichero con extensión ".torrent" El contenido de este fichero tiene forma de "diccionario" con claves del tipo cadena alfanumérica con un valor asociado que puede ser de diferente tipo. La codificación seguida se denomina "Bencoding"

Reglas de codificación en Bencoding

Tipos simples:

Entero El valor numérico expresado en base 10 irá prefijado por el carácter "i" y como sufijo el carácter "e"

Cadena de texto Las cadenas de texto estarán siempre codificadas en UTF-8 e irán precedidas por su longitud expresada en base 10, seguida del carácter

‘:” y finalmente el valor de la cadena.

Tipos complejos:

Lista una lista usa como prefijo el carácter “l” seguidos de los elementos que la componen, y es terminada por el carácter “e”

Diccionario Un diccionario se codifica con el prefijo “d” seguido de un número par de elementos, para cada pareja de elementos, el primero será usado como clave y el segundo como valor asociado. El diccionario se finaliza con el carácter “e”

3.6.2. Estructura del fichero de metadatos

El fichero de metadatos consiste en un diccionario codificado mediante Bencoding. Si aparecen entradas no válidas estas son ignoradas, pero siempre han de aparecer:

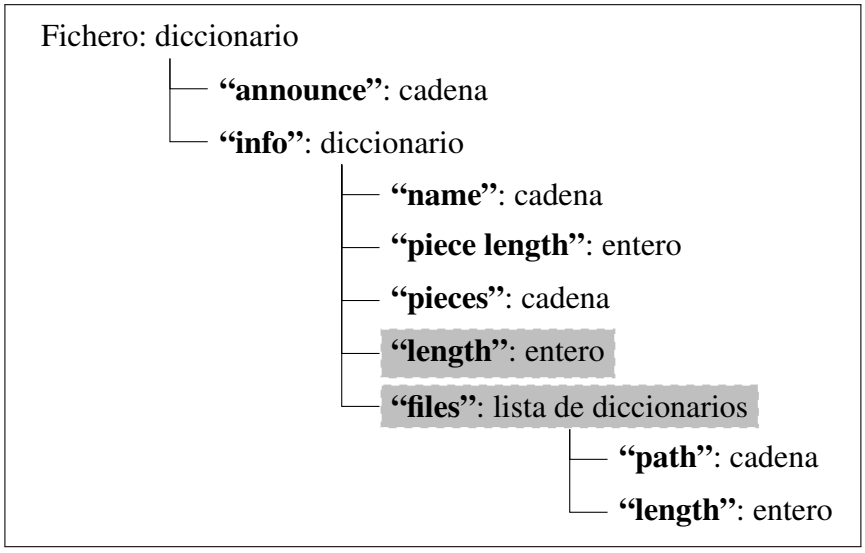

Figura 3.6: Diagrama de la organización de los datos en un fichero “.torrent”

“announce” : esta clave tiene asociada una cadena con la URL del rastreador. Más detalles en la sección dedicada a la búsqueda de pares.

“info” : esta clave acompaña a un diccionario con la información relativa a los ficheros contenidos en la descarga. Entradas a encontrar en este diccionario:

“piece length” : el entero al que acompaña indica el tamaño de los segmentos de datos a verificar por cada hash parcial. Esta longitud es siempre mayor o igual a 16 KiB.

“pieces” : tiene por valor asociado una cadena de longitud múltiplo de 20, cada subcadena de 20 caracteres es hash mediante SHA-1 de cada una de los segmentos de datos.

“length” : en caso de que la descarga solo contenga un fichero, este campo tiene asociado un entero que indica la longitud del fichero. Su presencia es incompatible con la clave “files”

“files” : no siendo posible que aparezca esta entrada, si la entrada “length” existe, tiene asociada una lista de diccionarios con al menos dos parejas de clave-valor:

“length” : con entero indicando la longitud del fichero individual

“path” : que se asocia a una lista de cadenas, siendo cada cadena un elemento en la ruta del fichero: de 0 a N directorios y finalmente el nombre del fichero.

“name” esta clave tiene asociada una cadena, que en el caso de una descarga individual sería el nombre del fichero a descargar. En caso de una descarga múltiple este valor es el nombre del directorio del que todos los ficheros son descendientes.

3.6.3. El rastreador

También conocido como “tracker”, es un servidor HTTP encargado de almacenar y transmitir la lista de pares involucrados en la distribución de una descarga. La URL del tracker es obtenida del campo “announce” del fichero de metadatos. Esta URL es accedida mediante el protocolo HTTP usando el método HTTP GET y los siguientes parámetros y valores:

```
{
    'announce':'http://tracker.example.com/path',
    'info':{
        'length':10,
        'name':'example',
        'piece length':32768,
        'pieces': ...
    }
}
```

Figura 3.7: Representación del contenido de un fichero .torrent en formato JSON.

info_hash : este parámetro va acompañado de una cadena de 20 octetos: del identificador de la descarga. Este identificador se calcula al hacer el hash SHA-1 del diccionario asociado a la clave "info" del fichero de metadatos.

peer_id : este parámetro tiene asociado una cadena de 20 octetos, cuyo uso es el de identificador único del par conectando al rastreador. Esta cadena es calculada como un valor aleatorio por el par al iniciar la descarga de un archivo.

ip : parámetro opcional para indicar la IP a registrar para este par en lugar de la de origen de la conexión al rastreador.

port : parámetro con el que el par indica el puerto TCP al que otros pares han de usar para conectarse a él.

uploaded : parámetro con el que el par indica la cantidad de datos cargados a la red de la descarga.

downloaded : parámetro con el que el par indica la cantidad de datos descargados de la red para la descarga.

left : parámetro que contiene la cantidad de datos restantes para completar la descarga.

event : este parámetro opcional indica el estado del par. Posibles valores:

started : este estado se usa cuando se inicia la descarga y se conecta por primera vez al rastreador.

completed : este estado se usa si el par ha completado la descarga.

stopped : el estado indica que el par ha interrumpido la descarga.

empty : valor por defecto cuando se omite el parámetro. Indica el estado normal del par descargando.

La respuesta del rastreador es un diccionario codificado mediante bencode que contienen las siguientes entradas:

failure : si la petición produjo un error en el rastreador aparecerá esta clave y solo esta clave. Tiene asociada una cadena con la descripción del error.

interval : tiene asociado un valor numérico indicando el periodo en segundos con el que se ha de enviar llamadas al rastreador para actualizar el estado del par.

peers : esta clave lleva asociada una lista de diccionarios, con las siguientes entradas:

peer id : identificador del par.

ip : dirección IP del par.

por : puerto TCP a usar para conectar al par.

3.6.4. Comunicación entre pares

Este es el protocolo usado cuando dos pares se conectan entre sí mediante una conexión TCP. La principal función de este protocolo es el intercambio de partes de la descarga entre los dos pares.

Formato

Por regla general y a menos que se indique lo contrario, todos los enteros son de 4 octetos, en formato big-endian. Todas las longitudes se indican en octetos. El formato de los mensajes es el siguiente:

- Longitud: un entero que indican la longitud del mensaje. Esta longitud no incluye este mismo campo.
- Tipo: un único octeto indicando el tipo del mensaje.
- El cuerpo del mensaje: cuyo contenido depende del tipo de mensaje.

Tipos de mensajes

- Pulsación

  Longitud: 0

  Tipo: Sin octeto de tipo

  Formato: No contiene campos adicionales

  Descripción: Este mensaje sin tipo es enviado cada dos minutos para indicar que el par sigue conectado.

- Bloquear

  Longitud: 1

  Tipo: 0x01

  Formato: Sin campos adicionales

  Descripción: Cuando un par envía este mensaje, indica que no aceptará peticiones de partes de la descarga.

- Desbloquear

  Longitud: 1

  Tipo: 0x02

  Formato: sin campos adicionales

  Descripción: Cuando un par envía este mensaje, está indicando que se encuentra listo para satisfacer peticiones de partes.

- Interesado

Longitud: 1

Tipo: 0x02

Formato: Sin campos adicionales

Descripción: Este mensaje es usado por el par para indicar que está interesado en realizar peticiones de partes al otro par.

- Desinteresado

  Longitud: 1

  Tipo: 0x03

  Formato: Sin campos adicionales

  Descripción: Con este mensaje el par indica que no tiene intención de realizar ninguna petición de ninguna parte al otro par.

- Tengo

  Longitud: 5

  Tipo: 0x04

  Formato: Un único entero indicando el índice de una parte.

  Descripción: Este mensaje indica que el par ha completado satisfactoriamente la descarga de la parte cuyo índice se indica, siendo posible su petición.

- Mapa de disponibilidad

  Longitud: Variable

  Tipo: 0x05

  Formato: Una cadena de octetos indicando que partes ya se han descargado satisfactoriamente. El bit más alto del primer octeto indica el estado de la primera parte.

  Descripción: Este mensaje solo se envía al inicio de la conexión, permitiendo a los pares saber que partes están disponibles en el otro par.

- Petición

  Longitud: 13

  Tipo: 0x06

  Formato: Tres enteros de 4 octeto en big-endian.

  Descripción: Este mensaje se usa para pedir la transferencia de un par. Los tres enteros indican: índice de la parte, desplazamiento desde el inicio de la parte y cantidad de datos a transferir.

- Parte

  Longitud: Variable

  Tipo: 0x07

  Formato: dos enteros de 4 octetos en formato big-endian, seguidos de un número de octetos hasta completar la longitud.

  Descripción: Este mensaje contiene la descarga de una parte requerida mediante un mensaje "petición". Los dos enteros indican el índice del aparte y desplazamiento de los datos respecto al inicio de la parte.

- Cancelar

  Longitud: 13

  Tipo: 0x08

  Formato: Tres enteros de 4 octetos en big-endian.

  Descripción: Siguiendo el mismo formato que el mensaje "petición" sirve para cancelar la petición con los mismo parámetros anteriormente enviados.

Establecimiento de la conexión

Cuando se establece la conexión TCP entre dos pares, se realiza la inicialización del protocolo, siguiendo una serie de pasos. Primero desde el par iniciando la conexión y después en la dirección opuesta. Los pasos a seguir son:

1. Envío del identificador del protocolo: el octeto 0x13 seguido de la cadena "BitTorrent protocol".

2. 8 octetos con la máscara de bits enumerando las extensiones del protocolo soportadas.

3. 20 octetos con la identificación de la descarga

4. 20 octetos con el identificador del par.

Tras repetir ambos pares esta secuencia, se envía un mensaje del tipo "Mapa de disponibilidad", para indicar que partes están disponibles en cada par. Tras esto, el estado inicial es que ambas partes es "desinteresado" y "bloqueando", por lo que en caso de que un par quiera empezar a realizar peticiones antes ha de actualizar su estado.

Comportamiento de los pares

Para facilitar la transferencia entre pared, una serie de comportamiento son propuestos a seguir por las diferentes implementaciones del protocolo:

- Cola de peticiones: para evitar tiempos ociosos, múltiples peticiones han de realizarse de forma consecutiva, asegurando así que no se deja el canal sin uso por falta de trabajo, pero el número de peticiones no satisfechas no ha de superar la cuenta de 5.

- Tamaño de la petición: Cada petición individual no ha de superar más de 16KiB de datos.

- Selección de parte a descargar: por regla general, y para mejorar la distribución de la descarga, han de pedirse a los pares las partes menos comunes de entre las disponibles por el conjunto de pares conectados, que no se hayan pedido ya a ningún otro par. En caso de que múltiples partes cumplan este requisito, una de ellas será seleccionada aleatoriamente.

- Algoritmo "fin de juego": aunque los umbrales son dependientes de cada implementación, este algoritmo se ejecuta cuando quedan pocas partes para

completar la descarga, y consiste en pedir todas las partes restantes a todos los pares conectados. De este modo evitamos bloquear la descarga innecesariamente si las ultimas partes a completar han sido pedidas a pares lentos, cuando otros más rápidos están disponibles.

- Bloqueo: un par puede tener múltiples conexiones a otros pares compartiendo la misma descarga, pero intentar enviar a todos los pares a la vez puede degradar la velocidad de transferencia y dificultar la distribución de esta. Como regla general han de desbloquearse pares que se han comportado de forma altruista o han mantenido una buena estadística de compartición de la descarga, fomentando una mentalidad de "lo comido por lo servido".

- Desbloqueo optimista: dado que por aplicación de la regla anterior un par puede no obtener muchas partes al estar todas los otros pares no enviándole por ser un par con un pobre historial de compartición, aleatoriamente y de vez en cuando, un par con bajo índice de compartición le es permitido realizar la descarga de una parte, para así darle la oportunidad de mejorar su índice de compartición y entrar a formar parte del conjunto de pares en el "comido por lo servido".

- Súper semilla: cuando un par tiene la descarga completa no suele informar de ello a los otros pares a los que está conectado, sino que les informa de que no posee ninguna parte. Entonces este par selecciona alguna de las partes menos comunes y mediante un mensaje "Tengo", informa a cada par de una parte diferente. Después observando el estado de los pares comprueba la velocidad de distribución de estas partes poco comunes en el enjambre de pares. De este modo se puede descubrir que pares de la red son más propensos a ayudar a distribuir las partes y se les da prioridad a la hora de seleccionar a que pares enviar partes.

Capítulo 4

Ejecución

4.1. Entorno de desarrollo y despliegue

El entorno de desarrollo y despliegue está basado el el sistema operativo Linux, siendo necesaria una versión igual o superior a 2.6.23, usando interfaces Ethernet configuradas a una MTU igual a 1500 bytes, no siendo importante el medio usado para la transmisión. El entorno de despliegue ha de constar de dos interfaces Ethernet entre las cuales se intercambiará el tráfico gestionado.

4.2. Diseño general

La caché P2P funciona como un puente entre dos interfaces Ethernet, enviando por una lo que recibe por la otra, pero al mismo tiempo, detecta y modifica conexiones P2P para el protocolo BitTorrent, reduciendo el tráfico intercambiado entre los pares mediante la inyección de partes del fichero compartido desde la caché, todo esto sin que los pares tengan ningún papel activo en este proceso. Dado que la caché recibe paquetes Ethernet correspondientes a la capa 2 de la pila OSI, pero la gestión y modificación del protocolo P2P pertenece a la capa 7 de la pila OSI, múltiples niveles de esta pila están presente en la solución.

La razón por la que se ha elegido un diseño de puente ethernet en lugar de simplemente emular un par en la red P2P es debido a que en el caso concreto del protocolo BitTorrent, cada par que descarga un único fichero ni conoce ni llega

a conectarse a todos los pares que forman el enjambre, lo que propiciaría que nuestro par de la caché pudiera ser no contactado por los pares a los que queremos dar servicio.

4.3. Capa 2: Enlace

Dado que la capa 1 ya está gestionada por la propia interfaz Ethernet, la caché recibe paquetes Ethernet desde las interfaces configuradas. Para ello se hace uso de la funcionalidad PACKET_MMAP presente en el kernel Linux. Esta funcionalidad permite el envío y recepción de paquetes de forma "cruda" desde interfaces, minimizando el número de copias entre espacio de kernel y usuario. Esta funcionalidad permite la reserva de una región de memoria organizada por subsegmentos contiguos de tamaño fijo, compuestos por una cabecera y a continuación el paquete Ethernet. En la cabecera se almacena tanto información sobre el paquete capturado (marca de tiempo, posibles banderas de error, longitud del paquete), así como una serie de banderas que permite indicar desde espacio de kernel a espacio de usuario, que un nuevo paquete está listo, y reseteándolo, el usuario indica al kernel que ese segmento ya ha sido procesado y puede usarse para capturar un nuevo paquete. Este comportamiento es análogo tanto en captura como en inyección de paquetes. Para poder configurar las dos regiones de memoria, una para captura y otra para inyección, es necesario primero abrir un descriptor de fichero asociado a la interfaz. Este descriptor se usa tanto para la espera asíncrona en al recepción, como para indicar la presencia de nuevos paquete para inyección.

4.4. Capa 3: Red

Tras la recepción de un paquete crudo, este no es más que un conjunto de bytes, por lo que se procede a su inspección. Dado que solo se soportan interfaces de tipo Ethernet, estos bytes capturados se empiezan a interpretar como un paquete Ethernet de 14 bytes: dos direcciones de 6 bytes para la dirección física de envío y recepción, y dos bytes para indicar la longitud o como etiqueta que indica el siguiente protocolo anidado. Dado que el único protocolo P2P soportado por

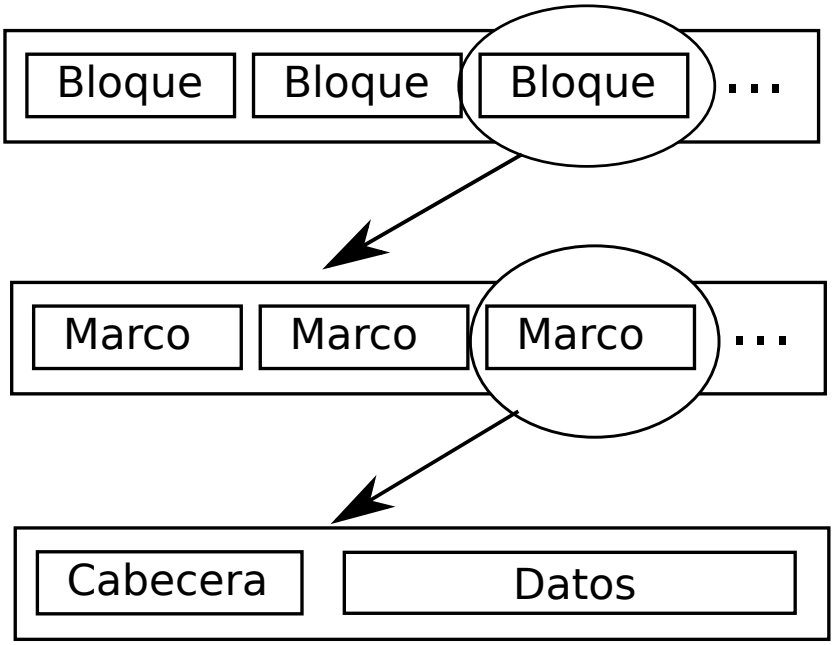

Figura 4.1: Disposición de la memoria en un anillo de captura de paquetes

la caché es BitTorrent, y este usa un protocolo de transporte TCP, si durante la inspección del paquete capturado no se detecta la pila de protocolos Ethernet + IP + TCP, el paquete es etiquetado para ser enviado de inmediato sin ningún tipo de procesado. Cuando se detecta que el paquete si posee los protocolos IP+TCP, se envía a la tabla de conexiones para su procesamiento. La tabla de conexiones organiza estas mediante dos niveles de acceso: una tabla hash y un árbol. Tanto el valor usado para acceder a la tabla hash como al árbol se obtienen de identificador de la conexión, compuesto por el identificador del protocolo (TCP o UDP), y los extremos de la conexión (IP+puerto) ordenados por el valor numérico de la IP. Esto es así para poder generar el mismo identificador para los paquetes que viajan en ambos sentidos. Dado que no todos los protocolos poseen una señalización explícita para indicar que la conexión ha concluido, existe un hilo ejecución que realiza la tarea recurrente de comprobar que conexiones no han procesado ningún tráfico en un periodo de tiempo configurable por conexión, lo que permite eliminar aquellas que han caducado.

Para protocolos orientados a conexión (TCP) solo se creará una nueva entradas cuando se detecta el establecimiento de una nueva conexión, ignorándose el tráfico

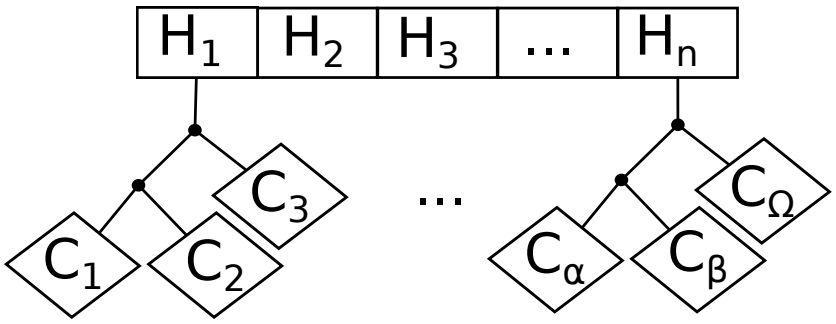

Figura 4.2: Representación de la estructura encargada de almacenar las conexiones

si el identificador de conexión no se encuentran en la tabla de conexiones. Para protocolos no orientados a conexión (UDP), cualquier paquete creará una nueva entrada en la tabla.

Al ser enviado un paquete a la tabla de conexiones, se espera como resultado dos posibles operaciones:

- Enviar: se permite al paquete ser enviado por la otra interfaz de captura.
- Desechar: el paquete es eliminado sin ser enviado.

La operación por defecto es la de enviar, que se puede dar porque el paquete no presenta ningún protocolo soportado, no existe conexión para el paquete, la conexión no tiene parser, o el parser indica que el paquete debe ser enviado. Para ser desechado, solo el parser puede decidirlo.

4.5. Capa 4: Transporte

Como era de esperar, la tabla de conexiones está poblada por conexiones, que representan cada sesión individual de tráfico entre dos pares de la red. El objeto “connection” carece de especialización para los dos protocolos de transporte soportados (TCP y UDP), y sirve simplemente de intermediario entre el flujo de paquetes asociados a cada conexión individual, y el opcionalmente presente “parser”: clase especializada en el tratamiento de los protocolos de capa aplicación.

La selección del parser asociado a una conexión puede ocurrir en dos momentos:

- En la creación de la conexión: si el protocolo de aplicación utiliza un puerto estándar, es posible asignar el parser en el momento de crear una nueva conexión.

- Con el primer paquete de datos: si la inspección del primer paquete con datos detecta la firma de un protocolo conocido, se usa este para asignar el parser.

4.5.1. Adaptación al transporte TCP

Para facilitar la modificación del protocolo de transporte TCP necesario para la operación del protocolo BitTorrent, se ha implementado una clase encargada de mantener y modificar el estado de esta. Usando una cola de órdenes, permite la manipulación del flujo de segmentos TCP de una conexión soportando las siguientes operaciones:

- Acumular tráfico: espera a que la cantidad indicada de tráfico esté lista para ser leída. Cuando esta condición se cumple, se señaliza a la capa superior para que se proceda al tratamiento de este tráfico.

- Permitir tráfico: permite una cierta cantidad de tráfico desde un par al otro.

- Desechar tráfico: elimina una cierta cantidad de tráfico.

- Inyectar tráfico: añade nuevo tráfico a un sentido de la conexión.

Reconstrucción del flujo

Para asegurar que solo el tráfico entre los pares de la conexión TCP monitorizada ya ha sido inspeccionado y tratado, todos los paquetes de la conexión son interceptados y enviados al otro extremo solo tras su procesamiento.

En cada extremo de la conexión se dispone de dos colas de paquetes:

- Cola de entrada: encargada de ordenar los paquetes de entradas, detectando ausencias, repeticiones y reordenaciones de los segmento mediante la comparación de números de secuencia. Solo cuando el flujo de segmentos

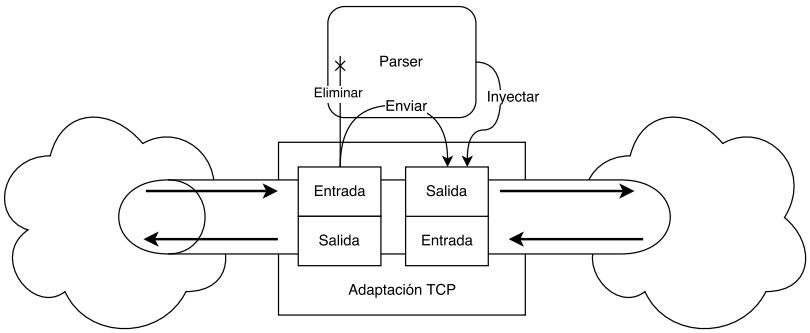

Figura 4.3: Representación del adaptador TCP interaccionando con el flujo de paquetes de una conexión

se encuentran reconstruido de forma correcta, se procede al tratamiento de este según las órdenes en la pila de órdenes. Dado que esta cola controla cual es el siguiente número de secuencia para la recepción en orden, es la encargada de actualizar el número de secuencia de confirmación a enviar al extremo de la conexión. Los paquetes de esta cola serán eliminados cuando una orden para permitir o eliminar tráfico es ejecutada. En el caso de permitir el tráfico, los paquetes son copiados a la cola de salida.

- Cola de salida: En esta cola se encuentran los paquetes con tráfico permitido desde la cola de entrada, o aquellos creados por datos que han sido inyectados. A esta cola solo serán añadidos paquetes en orden sin posibilidad de reordenaciones, repeticiones o agujeros en los números de secuencia. El hecho de añadir un paquete a esta lista implica que estos serán enviados cuando la ventana decepción del otro par así lo permita, y cuando paquetes con un número de confirmación así lo indique, se irán eliminando los paquetes confirmados.

Interpretación de las órdenes

Las órdenes en la cola de operaciones permite interaccionar con las colas de entrada y salida que existen para cada dirección de la conexión. Solo cuando una order “Acumular” es completada, se señaliza al parser para que este procese el tráfico y añada más órdenes, pues una cola vacía es una situación no soportada. Las órdenes pueden ser:

Acumular : cuando suficiente tráfico está listo en la cola de entrada, métodos adicionales para acceder al tráfico son usados por el parser para inspeccionarlo.

Permitir : paquetes con datos son movidos de la cola de entrada a la de salida. Si es necesario, estos serán divididos si no todos los datos de aplicación en el segmento TCP son afectados por esta orden. Solo si la ventana de recepción los permite, es ser añadidos a la cola de salida, estos son aviados. Si la orden involucra más segmentos de los que hay disponibles en la cola de entrada, según más paquetes son añadidos en orden estos son movidos a la de salida.

Copiar : en una mezcla de los dos anteriores comandos, se permite el tráfico, mientras los datos de este son copiados a un buffer.

Desechar : Como la operación anterior, esta orden elimina paquetes en la cola de entrada, pero no los añade a la de salida. Esto implica una modificación de los números de secuencia que aceptar a los siguientes segmentos a copiar en la cola de salida.

Inyectar : esta orden acarrea consigo un buffer de datos, el cual será usado para crear los paquetes en secuencia que serán añadidos a la cola de salida. Al introducir nuevos datos no procedentes del par original, los números de secuencia son modificados para asegurar que el estado de la conexión se mantiene intacto.

Interacción entre los comandos y el flujo TCP

De los comandos anteriormente listados solo Acumular, Permitir y Copiar no alteran el estado entre los extremos de la conexión monitorizada, pues no modifican los datos en el flujo TCP.

Los comandos que sí modifican los parámetros del flujo son:

Desechar : Este comando disminuye el desplazamiento a aplicar a los números de secuencia de los segmentos TCP cuando se permiten datos. El tope de ventana de recepción es modificado para simular que los datos leídos han sido consumidos por el par.

Inyectar : Al contrario del anterior comando, este aplica un aumento a los números de secuencia TCP para el tráfico al que se deja pasar. El tope de ventana de recepción no es reducido pues esto no se permite según la especificación TCP, sino que se congela en su valor hasta que los datos inyectados son confirmados por el par TCP al que se envían.

Interacción con el estado de la conexión TCP

Dado que la capa de adaptación está diseñada para modificar conexiones preexistentes, segmentos TCP que se detecten con banderas de modificación del estado de la conexión (SYN,FIN,RST), no podrán ser eliminados, y su presencia será notificada al parser que usa el adaptador.

4.6. Capa 7: Aplicación

La capa de aplicación, abstraída en la clase "parser", se encarga de dos tareas relacionadas con los protocolos de aplicación:

- Detección de los protocolos conocidos.
- Gestión de conexiones de protocolos conocidos

4.6.1. Detección de los protocolos

Tras la creación de una nueva conexión en la tala de conexiones, se intenta la detección del protocolo que transporta y la asignación de un parser sí así es necesario. Las dos técnicas para la asignación del parser son:

- Por número de puerto de destino: si la conexión se establece hacia un número de puerto para el que se ha registrado un parser de un protocolo, se procede a la asignación de una instancia de este parser.
- Si no se asignó un parser por número de puerto, cuando el primer paquete con datos de aplicación es visto, se intenta asignar un parser inspeccionando estos datos.

Son los propios parser los que al ser cargados registran que números de puertos de qué protocolos (TCP o UDP) quieren que se creen instancias de sí cuando se detectan conexiones usándolos. Asimismo también registran los patrones en los datos que han de encontrarse para asignar el parser por inspección de datos.

4.6.2. Tratamiento

Una vez un parser de un protocolo es asignado a una conexión, este se encarga de procesar el tráfico que es asignada a la conexión. Si el parser es del tipo más básico, los paquetes que son asignados a la conexión, son enviados al parser, decidiendo este si a cada paquete se le permite ser enviado por la interfaz de salida, o si es simplemente eliminado. Como ya se ha explicado anteriormente, también existe la posibilidad de usar el adaptador TCP, lo que permite usar una interfaz más orientada a flujo de bytes en lugar de paquetes individuales.

4.6.3. El parser BitTorrent Caché

El núcleo de la aplicación consiste en el parser encargado de no solo gestionar las conexiones del protocolo BitTorrent, sino que además permite la interacción con la caché en disco. Este parser se basa en el adaptador TCP, por lo que no trata con paquetes directamente, sino que interacciona con el flujo TCP como si de un flujo constante de bytes se tratase.

Establecimiento de la conexión

Durante el establecimiento de la conexión, el identificador de la descarga es almacenada, pues se usa para acceder a la caché, y el intercambio de los mapas de disponibilidad es modificado, solo en el sentido hacia la red interna, añadiéndose las partes que ya están disponibles en la caché. Los campos de estados (bloqueado e interesados) de cada par son inicializados.

Descubrimiento de la longitud de una parte

Como ya se indicó durante la descripción del protocolo BitTorrent, la descarga se encuentra verificada parcialmente mediante hashes que se corresponden con

partes de longitud fija, siendo siempre la longitud de estas partes una potencia de 2 mayor o igual a 16KiB. Esta longitud de una parte verificable de la descarga está indicada en el fichero de metadatos de la descarga, pero no se transmite entre los pares de una descarga, al ser un dato ya conocido por ambos. Dado que el parser desconoce la información contenida por el fichero de metadatos, parte de esta es leída directamente desde la conexión (como es el identificador de la descarga), pero al no transmitirse de forma explícita el tamaño de estas partes, esta es detectada como el mayor posición de offset transmitida en una descarga, si esta es una potencia de 2. Cuando el tamaño de parte es descubierta, este dato es almacenado en la caché para no tener que repetir este proceso.

Adquisición de partes para la caché

Una vez se ha inicializado la comunicación entre los dos pares, se procede a la monitorización de peticiones de descarga, independientemente de la dirección en que se realiza una petición de una parte. Cuando se detecta la transmisión de esta se comprueba si la caché está interesada en ella, y de ser así, se copia a la caché. Dado que la caché usa el mismo mapa de disponibilidad que la conexión entre pares, una parte solo será marcada como disponible si se ha transmitido de forma completa y ordenada en una única conexión.

Inyección de partes desde la caché

La inyección de partes desde la caché sucede tras un proceso mucho más complejo que la adquisición de partes. Este proceso empieza al monitorizar las peticiones de partes, especialmente aquellas desde la red de clientes hacia internet. Según se observan las peticiones, se comprueba si estas pueden ser satisfechas desde la caché, y de ser así, son eliminadas de la conexión y se obtienen los datos correspondientes a esta petición desde la caché. Dado que la petición de una parte y el mensaje con la parte van en sentidos diferentes en la conexión, es necesaria la coordinación el estado del parser para cada sentido. Cuando los datos de la parte son recibidos desde la caché, se crea el buffer con el mensaje completo que se va a insertar en el flujo, y si el otro sentido de la conexión se encuentra entre mensajes, este nuevo mensaje con la parte es inyectado inmediatamente. En caso de que el

otro sentido de la conexión no esté en un estado entre mensajes, se almacena el nuevo mensaje temporalmente, y se indica al estado del parser en el otro sentido para que realice la inyección en cuanto sea posible.

Dado que la inyección de partes se ha de realizar en el mismo orden en que se detectaron las peticiones, existe una cola de peticiones, tanto las que se han dejado enviar como las que han sido eliminadas. Para aquellas que se han dejado enviar, todas las peticiones consecutivas permitidas son enviadas, estas son eliminadas según las partes pedidas con correspondidas con un mensaje con dicha parte en detectado. Del mismo modo, todas las peticiones que pueden ser satisfecha mediante inyección desde la caché, lo son para todas las peticiones consecutivas que así lo son. Cuando en la cola de peticiones detectamos que una petición y la siguiente son una a permitir y otra a satisfacer desde la caché, tenemos que eliminar la primera tras completar su tratamiento antes de gestionar la siguiente de distinto tipo.

4.6.4. La caché de partes

Dado que el parser se encarga de la manipulación del protocolo, la caché es la encargada de almacenar las partes de cada descarga individual, permitiendo añadir más partes desde el parser, así como almacenar nuevas partes.

Datos y metadatos en la caché

Las entradas en la caché están indexadas por el hash identificador de cada descarga y un identificador de protocolo. Para cada entrada en la caché, se hace disponible la siguiente información:

- Mapa de disponibilidad de las partes presentes en la caché.
- El tamaño descubierto de las partes.
- Los datos de las partes ya cacheadas.

Interfaz con el parser

Cuando un parser detecta el intercambio de mensajes inicial entre pares, copia el identificador de la descarga que pasa a ser usado para obtener la entrada en la caché para dicha descarga. Una vez se tiene la entrada a la caché correspondiente se le pide que fusione su mapa de disponibilidad con el que está presente en el mensaje de disponibilidad, pero solo en el sentido hacia el par en el lado interior, de este modo se aumenta el número de partes disponibles con las presentes en la caché.

Si se trata de una nueva entrada en la caché, no solo el mapa estará vació, sino que el tamaño de las partes de está descarga será desconocido. Por ello cada vez que se ve una petición conde el siguiente octeto estaría en una posición potencia de 2 y mayor que 16KiB, este tamaño se sugiere como posible longitud de la parte. A este proceso pueden contribuir de forma concurrente los parsers de múltiples conexiones en las que se identifica la misma descarga.

Una vez el tamaño de las partes es conocido, se puede proceder a añadir partes desde el parser (si la parte no está todavía presente) o a tomar partes desde la caché si lo están. Dado que la entrada en la caché es común a todos los parser que tienen el mismo identificador de descarga, partes obtenidas desde una conexión pueden ser inyectadas en otra.

Formato en disco

El contenido de la caché es persistente en disco, no solo los datos de las partes, sino también los metadatos. Para ello en un directorio, cuya ruta es configurable, se crea un sub-directorio con nombre el identificador de la descarga. Dentro del sub-directorio se encuentran los siguientes ficheros:

- El fichero “metadata” con los metadatos de la descarga: identificador de protocolo, longitud de las partes y mapa de disponibilidad.
- Un fichero por cada parte cacheada: con nombre su índice de la parte en la descarga.

Capítulo 5

Pruebas

Durante la implementación del proyecto se han ido añadiendo una serie de pruebas unitarias para certificar el correcto funcionamiento de los diferentes componentes que lo forman. Una vez el proyecto empezó a ofrecer una mínima funcionalidad las pruebas de integración ofrecieron una imagen más detallada de las limitaciones y rendimiento que la solución mostraría en un entorno de despliegue real .

5.1. Entorno

El entorno de las pruebas es el mismo usado para el desarrollo, usando dos máquinas virtuales para simular las dos redes entre las cuales se despliega la solución. Estas máquinas virtuales ejecutan un entorno Linux con la distribución Debian en su versión 8. El único software instalado es aquel necesario para la ejecución de pruebas:

- Iperf v3.0.7: generador de tráfico TCP y UDP, el contenido de este tráfico es irrelevante. Es posible configurar el número de conexiones a usar en cada prueba, pero estas solo ofrecen el envío de tráfico en una única dirección.
- Cliente BitTorrent BitTornado v0.3.18: cliente BitTorrent en python. Incluye cliente y tracker HTTP, así como las herramientas necesarias para crear fichero .torrent. Este cliente no se usa para pruebas de rendimiento sino para comprobar el correcto funcionamiento cuando se gestionan conexiones

usadas por pares usando clientes del protocolo BitTorrent totalmente funcionales.

- Script fakebt.py: cliente BitTorrent simulado que genera descargas BitTorrent basadas en parámetros configurables, así como la posibilidad de repetir en bucle la descarga tantas veces como se especifique.

5.2. Diseño

La intención de las pruebas de integración y rendimiento es poder observar el correcto funcionamiento de la solución, así como su comportamiento bajo distintas situaciones, y observar el impacto que tiene sobre el rendimiento los distintos componentes de esta.

El diseño de las pruebas de rendimiento imita el camino que el tráfico sigue dentro de la solución, observando como para cada nueva capa de protocolo a procesar, qué impacto tiene esta en el rendimiento. Dado que solo la última capa trata con el protocolo de aplicación P2P, las pruebas de protocolos en capas inferiores de la pila usan tráfico TCP/UDP cuyo contenido es ignorado, intentando saturar la capacidad de procesamiento de tráfico de la solución. Las configuraciones del entorno a probar serán:

- Puente: un puente ethernet, creado por el módulo del kernel Linux, conecta las dos interfaces de red.
- Sin conexiones: La solución copia paquetes de una interfaz a la otra.
- Con conexiones: La solución clasifica los paquetes en conexiones y lo envía por la otra interfaz, pero no aplica ningún procesado posterior.
- Parser TCP transparente: Para cada conexión TCP gestionada por la solución, un parser TCP es asignado con la única tarea de dejar pasar todo el tráfico.
- Parser BitTorrent con caché: Tras detectar el protocolo BitTorrent, las conexiones son gestionadas manipulando las partes requeridas y enviadas entre los pares a cada extremo de la conexión.

El tráfico usado en las pruebas se clasifica en los siguientes grupos:

- Tráfico de saturación: tráfico UDP usado para medir el impacto de la captura de paquetes y clasificación por conexión. El protocolo UDP carece de control de flujo, por lo que pérdidas de paquetes no afectan al tráfico subsiguiente.
- Flujos TCP sin protocolo: tráfico TCP usado para medir el rendimiento de la pila TCP implementada. Un parser TCP envía todo el tráfico de extremo a extremo sin aplicar ningún otro tratamiento.
- Tráfico P2P sintético: para poder medir la carga que supone el procesado del protocolo BitTorrent. Dado que la generación de este tráfico es más compleja, no es posible alcanzar altas tasas de transferencia, por lo que este tráfico no se usa para las pruebas de rendimiento de las capas inferiores de la pila.

5.3. Resultados

En la primera figura 5.1 se puede observar el ancho de banda procesado por la solución en diferentes configuraciones y para distintos perfiles de tráfico. Este tráfico ha sido generado con la utilidad "iperf3", generando tráfico UDP y TCP.

	UDP			TCP		
	Unidir.	Bidir.	Bidir. x10	Unidir.	Bidir.	Bidir. x10
Puente	266 Mb/s	256 Mb/s	272 Mb/s	202 Mb/s	210 Mb/s	228 Mb/s
Sin conexiones	171 Mb/s	174 Mb/s	170 Mb/s	121 Mb/s	122 Mb/s	123 Mb/s
Con conexiones	176 Mb/s	165 Mb/s	172 Mb/s	64 Mb/s	86 Mb/s	120 Mb/s
Parser TCP				25 Mb/s	23 Mb/s	42 Mb/s

Cuadro 5.1: Rendimiento de la solución para tráfico UDP y TCP sin protocolo procesado.

Se puede observar que en el caso de tráfico UDP, la tasa de transferencia soportada se mantiene estable en cada entorno probado, a pesar de las diferencias en las características del tráfico generado. Esto se debe a que con tráfico UDP, la pérdida

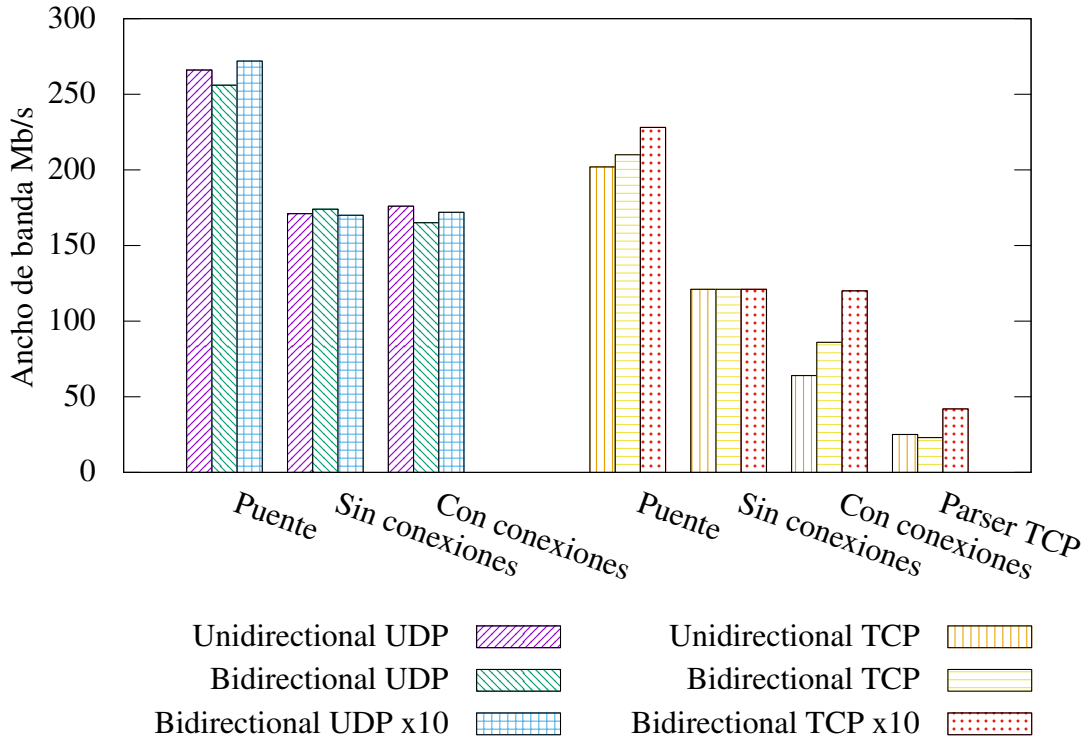

Figura 5.1: Rendimiento de la solución para tráfico UDP y TCP sin protocolo procesado.

de un paquete no implica una retransmisión, y por lo tanto una reducción en la tasa de transferencia. La caída de rendimiento entre el entorno usando el puente ethernet y los entornos usando la solución se debe a necesidad de identificar los protocolos en cada paquete y a que realizamos la copia del paquete desde la interfaz de recepción a la de envío, siendo ambos pasos no necesarios en el caso del puente: este no necesita detectar los protocolos presentes en los paquetes, y es capaz de enviar por la interfaz de envío los paquetes sin realizar ninguna copia, al tener acceso directo a los bufferes de recepción y envío de paquetes usados por el kernel. Por esta misma razón, La variación en la tasa de transferencia para el tráfico UDP cuando está presente la solución, no ofrece mucha variación entre el caso con y sin clasificación en conexiones.

En el caso del tráfico TCP la situación presentada es bien distinta, pues al detectarse pérdidas de paquetes, se procede a la retransmisión de estos, limitando

el ancho de banda a usar por la conexión e incrementándolo según se produzca la correcta transmisión de datos a mayor velocidad. Otra característica a tener en cuenta es que cada conexión gestionada por la solución solo trata un paquete cada vez, lo que causa contención al no poder procesar un paquete de cada sentido de una única conexión simultáneamente. Cuando el número de conexiones concurrentes aumenta, este problema se atenúa.

El uso del parser TCP supone otra bajada de rendimiento, pues la comunicación entre las pilas TCP existente en cada extremo de la conexión ya no es directa, sino que el parser aplica una gestión más activa de los paquetes.

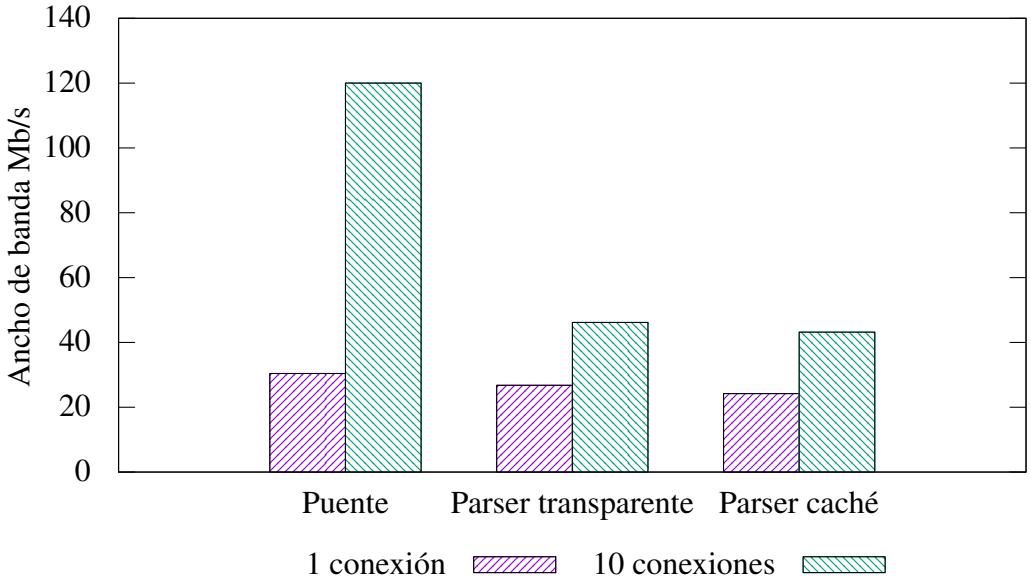

Figura 5.2: Rendimiento de la solución para tráfico BitTorrent, falso cliente.

	1 Cliente	10 Clientes
Puente	30,39 Mb/s	120,01 Mb/s
Parser transparente	26,77 Mb/s	46,15 Mb/s
Parser caché	24.17 Mb/s	42,16 Mb/s

Cuadro 5.2: Rendimiento de la solución para tráfico BitTorrent, falso cliente.

En la siguiente figura 5.2 se puede observar el ancho de banda procesado cuando se usa un falso cliente de BitTorrent: generando peticiones y respondiéndolas

con partes en un bucle sin límite. Esto permite generar un flujo de mensajes BitTorrent constante y con un comportamiento más estable. Se puede observar que la tasa de transferencia apenas varía entre el uso del parser TCP transparente y el encargado de cachear el protocolo BitTorret, esto se debe a que la mayor limitación se encuentra en la implementación del protocolo TCP y no el procesamiento del protocolo BitTorrent.

Capítulo 6

Conclusiones

Como se ha podido constatar, tanto por la descripción teórica como por los detalles de la implementación y pruebas realizadas, la solución software implementada ofrece la funcionalidad deseada de reducir el tráfico entre pares en comunicación mediante protocolos P2P, si no así mostrando un gran rendimiento debido a las limitaciones de reimplementación de la pila de protocolos, requisito necesario dada las peculiaridades de despliegue necesario para la interceptación de tráfico.

Dadas las características de los protocolos P2P, la implementación de una solución capaz de reducir el volumen de tráfico entre pares requiere del uso de una arquitectura bastante diferente a la esperada en soluciones de cacheo de protocolos cliente-servidor:

- Dado que no existen servidores concentrando las conexiones, y que el uso de puertos específicos no es aplicable a protocolos P2P, se ha de usar un análisis del contenido de las conexiones para identificar aquellas que hacen uso de un protocolo P2P.

- Dada una única descarga, no todos los pares involucrados en su distribución están conectados a todos los otros pares del enjambre, esto obliga a que para maximizar el número de pares que la caché gestiona, la caché ha de localizarse en un punto intermedio entre los pares, en lugar de aparentar ser un par más del enjambre.

A consecuencia de estos dos requerimientos, la solución se ve obligada a usar la arquitectura poco convencional de un sistema DPI capaz de analizar todo el tráfico entre dos interfaces de red. Dado que la solución no usa conexiones gestionadas por el sistema operativo, sino accediendo directamente a los paquetes capturados por las interfaces de red, debe separar aquellos que corresponden a cada conexión. Usando patrones de tráfico para detectar, por su contenido, que conexiones son usadas por protocolos P2P. Una vez detectadas y dado que se acceden a los paquetes directamente, se ha de reconstruir el flujo TCP y gestionar tanto su contenido como su estado. Esta tarea, nada simple, es realizada por la solución sin poder usar la pila TCP, bastante optimizada, ya existente en el sistema operativo.

Sobre esta base, se implementa tanto el gestor del protocolo específico (BitTorrent) como una caché capaz de guardar a disco las partes de una descarga que posteriormente serán inyectadas en conexiones interceptadas.

El hecho de perder la gestión de los protocolos realizada por el sistema operativo se muestra como una espada de doble filo: las conexiones ignoradas son más fácilmente gestionadas (para cada paquete recibido, enviarlo por el otro interfaz), pero en el caso contrario, cuando se detecta un protocolo soportado que usa como transporte TCP, el soporte para flujo TCP ofrecida por la solución muestra más limitaciones que la existente en el sistema operativo.

Capítulo 7

Futuros proyectos

Este proyecto ha implementado toda la pila de protocolos necesaria para llevar a cabo el objetivo pretendido, pero esta misma pila puede ser reutilizada para poder implementar todo un abanico de posible soluciones orientadas al tratamiento y modificación de tráfico en tiempo real. Otros posibles proyectos basados en este podrían ser:

- P4P: en lugar de cachear el tráfico entre pares, mediante la interceptación de comunicación entre pares y trackers, es posible modificar la lista de pares obtenida para dar prioridad a aquellos dentro del mismo ISP. Esto haría que al estar formado el conjunto de pares conocidos por pares en el mismo ISP, reducir el tráfico de interconexión.

- Política de reemplazo en la caché: la caché entregada no implementa ninguna política de reemplazo para las partes cacheadas. Aunque la interfaz para la eliminación "en caliente" de descargas en la caché está presente, habría que estudiar y probar qué política sería la más idónea para este caso. Ha de tenerse en cuenta, que a diferencia de otros entornos, los recursos usados por cada entrada en la caché no se mantienen constantes en el tiempo.

- Modularización de la caché: separando la caché del procesador de tráfico, sería posible extraer la caché como un servicio externo e implementarlo como un servicio común a múltiples instancias del procesador de tráfico.

- Análisis y monitorización de la red: dado que la solución interacciona con

los pares de las conexiones de forma transparente a estos, es posible usar este como base para soluciones orientados al análisis del tráfico: detección de tráfico anómalo (IDS), antivirus online, detección de spam, recolección del perfil de uso de la red.

Bibliografía

[1] (2012). *IEEE Standard for Ethernet*. The Institute of Electrical and Electronics Engineers, Inc.

[2] Allman, M., Paxson, V., and Stevens, W. (1999). TCP Congestion Control. RFC 2581 (Proposed Standard). Obsoleted by RFC 5681, updated by RFC 3390.

[3] Cerf, V. G. and Khan, R. E. (1974). A protocol for packet network intercommunication. *IEEE TRANSACTIONS ON COMMUNICATIONS*, 22:637–648.

[4] Cohen, B. (2013). *The BitTorrent Protocol Specification*. BitTorrent.org.

[5] Nichols, K., Blake, S., Baker, F., and Black, D. (1998). Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard). Updated by RFCs 3168, 3260.

[6] Postel, J. (1980). User Datagram Protocol. RFC 768 (INTERNET STANDARD).

[7] Postel, J. (1981a). Internet Protocol. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864.

[8] Postel, J. (1981b). Transmission Control Protocol. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528.

[9] Ramakrishnan, K., Floyd, S., and Black, D. (2001). The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard). Updated by RFCs 4301, 6040.

[10] Sandvine Inc (2011-2014). *Global Internet Phenomena Report*.

[11] Spring, N., Wetherall, D., and Ely, D. (2003). Robust Explicit Congestion Notification (ECN) Signaling with Nonces. RFC 3540 (Experimental).