UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO DE UN ALGORITMO DE AQM PARA DISPOSITIVOS DE ENRUTAMIENTO QUE SEA JUSTO EN BASE AL RTT DE LAS CONEXIONES TCP

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

FELIPE IGNACIO FREDES MUÑOZ

PROFESOR GUÍA:
CLAUDIO ESTÉVEZ MONTERO

MIEMBROS DE LA COMISIÓN:
CÉSAR AZURDIA MEZA
JAIME ANGUITA GARCÍA

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
POR: FELIPE IGNACIO FREDES MUÑOZ
FECHA: 2020
PROF. GUÍA: CLAUDIO ESTÉVEZ MONTERO

DISEÑO DE UN ALGORITMO DE AQM PARA DISPOSITIVOS DE ENRUTAMIENTO QUE SEA JUSTO EN BASE AL RTT DE LAS CONEXIONES TCP

El control de congestión en Internet es un problema que está lejos de estar solucionado. Más aún, la investigación de esta disciplina no ha demostrado muchos avances significativos en la última decada. Esto constrasta con el progreso constante a nivel de capas física y de acceso, lo cual transforma progresivamente a la capa de transporte en un cuello de botella, especialmente en el contexto de conexiones intercontinentales a nivel global. Este ritmo de avance podría ser crítico, ya que en la actualidad es posible apreciar incremento continuo del número de aplicaciones que hacen uso de Internet para entregar servicios a sus clientes. Lo anterior implica una evolución continua del tráfico de datos, lo cual lleva a pensar que es posible que las técnicas necesarias para lidiar con la congestión eventualmente podrían volverse incapaces de soportar las cambiantes necesidades de la red que conocemos como Internet.

Es posible distinguir dos puntos donde es posible realizar control de congestión. Uno de ellos corresponde a la capa de transporte, donde el Protocolo de Control de Transmisión (TCP, Transmission Control Protocol) emplea un algoritmo para determinar adaptativamente un valor correcto para la variable de estado conocida como ventana de congestión. El valor de esta variable se utiliza para limitar la tasa de transmisión de datos en respuesta a eventos de congestión. El segundo punto corresponde a los routers, los cuales pueden implementar algoritmos de Active Queue Management (AQM) para determinar que existe congestión antes de que sus buffers se saturen por completo, lo cual permite evitar latencia adicional excesiva y fenómenos como la sincronización global.

En el presente documento se presenta una nueva propuesta para realizar control de congestión a nivel de routers, la cual procura aliviar el sesgo que existe en contra de las conexiones de alta latencia. Esta se denomina RTT-Based Fair Active Queue Management (RBF-AQM), y se basa en el uso del Round-Trip Time (RTT) para descartar paquetes de una forma más justa, aprovechandose del comportamiento de TCP tradicional para limitar la latencia adicional por encolamiento en un nivel razonable, y a la vez garantizar justicia en cuanto al throughput alcanzado por cada una de las conexiones que compiten por un determinado ancho de banda, sin disminuir significativamente el throughput total del sistema.

En particular, se estudia el rendimiento de la propuesta cuando se toma como referencia la probabilidad de pérdidas utilizada por el algoritmo denominado Random Early Detection (RED). El algoritmo resultante es comparado con las versiones tradicional y adaptativa de RED (ARED), demostrando un desempeño muy similar a nivel de utilización total, pero logrando una mejora significativa en cuanto al Fairness Index (FI) obtenido. Adicionalmente, diversas alternativas de implementación son consideradas, las cuales se caracterizan por ser logrables a través de cambios menores a nivel de software, y que además no requieren el uso de variables de estado por conexión, lo cual favorece la escalabilidad de la propuesta.

A mi familia, quienes hicieron posible que llegara a este punto.

Agradecimientos

En primer lugar me gustaría agradecer a mi familia por brindarme las condiciones necesarias para llegar al punto donde me encuentro actualmente, y apoyarme durante mi paso por la carrera.

A Francisco Leiva, quien fue de gran ayuda durante la investigación inicial de las variantes de TCP, lo cual sería la base que me llevaría a investigar el tema que aborda esta tesis.

A Freddy Pesántez, quien contribuyó de forma significativa en el desarrollo de simulaciones y proponiendo ideas relevantes para el análisis.

A mi profesor guía, Claudio Estévez, quien siempre tuvo una actitud muy colaborativa y comprensiva, desde el inicio hasta el final de esta investigación.

Finalmente, me gustaría agradecer a los académicos César Azurdia y Jaime Anguita por acceder a ser integrantes de mi comisión, así como también por su retroalimentación sobre el trabajo realizado.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Siglas

ACK Acknowledgement

AIMD Additive Increase Multiplicative Decrease

AQM Active Queue Management

ARED Adaptive Random Early Detection

BBR Bottleneck Bandwidth and Round-trip propagation time

BDP Bandwidth-Delay Product

BGP Border Gateway Protocol

BIC Binary Increase Congestion control

CHOKe Choose and Keep/Kill

CHOKeH CHOKeHold on unfairness

CoDel Controlled Delay

DCCP Datagram Congestion Control Protocol

DCTCP Datacenter TCP

DOCSIS Data Over Cable Service Interface Specification

DRED Dynamic RED

DUPACK Duplicate Acknowledgement

ECN Explicit Congestion Notification

FAST Fast Active queue management Scalable TCP

FI Fairness Index

FIFO First In First Out

FRED Flow Random Early Detection

FTP File Transfer Protocol

GAIMD General Additive Increase Multiplicative Decrease

gCHOKe Geometric CHOKe

H-TCP Hamilton-TCP

HSTCP Highspeed TCP

HTTP Hypertext Transfer Protocol

IETF Internet Engineering Task Force

IP Internet Protocol

ISO International Organization for Standardization

ISP Internet Service Provider

MIMD Multiplicative Increase Multiplicative Decrease

NLRED Nonlinear RED

OSI Open Systems Interconnection

OSPF Open Shortest Path First

PCC Performance-oriented Congestion Control

PIE Proportional Integral controller Enhanced

QoE Quality of Experience

RBF-AQM RTT-Based Active Queue Management

RBF-ARED RTT-Based Fair ARED

RBF-RED RTT-Based Fair RED

RED Random Early Detection

REM Random Early Marking

RFC Request For Comments

RRED Robust RED

RSFB Resiliente Stochastic Fair BLUE

RTT Round Trip Time

SACK Selective Acknowledgement

SFB Stochastic Fair BLUE

SRED Stabilized RED

STCP Scalable TCP

TCP Transmission Control Protocol

TTL Time To Live

UDP User Datagram Protocol

VPN Virtual Private Network

WWW World Wide Web

Capítulo 1

Introducción

1.1. Motivación

En la actualidad, la mayoría del tráfico que atraviesa la internet hace uso del Protocolo de Control de Transmisión (TCP, Transmission Control Protocol) a nivel de capa de transporte, y esto es algo que no se prevee que cambie en el futuro cercano [4]. Esto se debe en gran parte a la capacidad que tiene el protocolo para garantizar que la información transmitida por un emisor sea recibida sin errores en el receptor. Sin embargo, a un usuario no solo le importa que una aplicación mantenga la integridad de la información enviada (pues a estas alturas es algo que se da por hecho), sino que también es crucial que esto se realice en el menor tiempo posible. Para lograr esto último existen dos aspectos fundamentales: transmitir una gran cantidad de información por unidad de tiempo, y que esta comunicación no experimente una latencia excesiva, entendiendo esta última como la suma de los retrasos que afectan a la transmisión. En el primer caso, la métrica de interés corresponde al throughput de la conexión, que corresponde a la tasa de transmisión de datos en bits por segundo[1]. En el segundo caso lo que usualmente se mide es el Round-Trip Time (RTT), el cual consiste en el tiempo que transcurre entre el envío de un paquete y la recepción del paquete que reconoce su llegada en el receptor.

Lamentablemente, dada la naturaleza de la red que conocemos como Internet, no es posible garantizar un nivel de servicio al usuario en cuanto a los dos aspectos previamente mencionados. Esto se debe a que el servicio que provee el Protocolo de Internet (IP, Internet Protocol), a nivel de capa de red, tiene un comportamiento del tipo mejor esfuerzo. Esto quiere decir que IP hace lo mejor posible para transmitir la información desde un punto a otro, pero no se proveen garantías respecto al orden en que llegan los paquetes, la latencia que experimentan o incluso la recepción de los mismos. Los problemas no acaban allí, pues el crecimiento masivo de Internet y del número de aplicaciones que hacen uso de esta implican un aumento significativo en el tráfico que fluye en la red a través del tiempo.

[1]En el resto del texto se mantiene el uso del término throughput. Esto se debe en parte a que es un término comúnmente utilizado en el contexto de redes, y adicionalmente evita la verborrea causada por su equivalente en español (tasa de transmisión de datos).

Un aumento del tráfico no sería un problema si existiera la posibilidad de tener recursos ilimitados. No obstante, este nunca es el caso, y por lo tanto es posible que los recursos existentes no den abasto para todo el tráfico que originan los usuarios. La consecuencia de lo anterior es que existan problemas de congestión en la red, lo cual se puede traducir en una experiencia desagradable desde el punto de vista del usuario, quien siempre espera que sus aplicaciones funcionen de forma expedita. Específicamente, el usuario puede percibir tasas de transmisión menores a las usuales, un alto nivel de latencia y un alto porcentaje de pérdida de paquetes (lo cual se traduce en desperdicio de recursos).

Para solucionar lo anterior, se han desarrollado diversas propuestas a través del tiempo en la disciplina denominada control de congestión. Gran cantidad de estos avances se han realizado a nivel de capa de transporte, con propuestas que modifican las acciones que realiza TCP en base a eventos de congestión [5]. Sin embargo, también se han desarrollado otros algoritmos para abordar este problema a nivel de capa de red, los cuales se implementan en los routers (los dispositivos que se encargan del enrutamiento de paquetes). En particular, estos algoritmos permiten notificar a los protocolos de transporte que existe congestión antes de que las colas de paquetes, que se generan durante eventos de congestión, llenen por completo los buffers de los routers. Lograr lo anterior permite evitar latencia excesiva, tasas de pérdidas altas y el fenómeno conocido como sincronización global [6]. Este último hace referencia a la ocurrencia de bajas simultáneas y sincronizadas de las tasas de transmisión de datos debido a múltiples pérdidas consecutivas forzadas por un buffer lleno.

La disciplina que aborda este problema a nivel de routers es denominada Active Queue Management (AQM), y una de las propuestas más estudiadas en la literatura se denomina Random Early Detection (RED) [7]. Sin embargo, y a pesar de que el uso de este algoritmo fue inicialmente recomendado en un Request For Comments (RFC) [8], actualmente la Internet Engineering Task Force (IETF) no recomienda ningún algoritmo en particular, sino que simplemente se dan algunas directrices generales sobre el tema, además de promover la investigación en el área [6, 9]. No obstante, desde la propuesta original se ha trabajado extensivamente para comprender el comportamiento de RED y mejorar su rendimiento, lo cual ha derivado en una gran cantidad de variantes basadas en el algoritmo propuesto originalmente [10, 11, 12, 13].

1.2. Descripción del problema

A pesar de que existen herramientas para lidiar con la congestión, los avances en esta disciplina no han sido muchos en la última década [14]. En particular, el protocolo predominante a nivel de TCP es uno desarrollado hace más de 10 años [15], lo cual contrasta con la situación que se da para las capas de acceso y física, donde constantemente se generan avances concretos a nivel de records en las tasas de transmisión de datos alcanzadas [16, 17, 18]. Lo anterior progresivamente convierte a la capa de transporte en un cuello de botella en este contexto, en especial cuando se consideran enlaces intercontinentales, donde la latencia adicional en conjunto a la alta capacidad física disponible constituyen un escenario que ha demostrado ser complejo para TCP en el pasado [19].

Adicionalmente, aún existen aspectos que están lejos de poder ser considerados como resueltos. Uno de los más recurrentes que se estudia en la literatura es el de la justicia (o fairness) entre conexiones, la cual en términos generales se puede entender como la idea de distribuir equitativamente los recursos de la red. Sin embargo, en la práctica no es tan sencillo definir un único objetivo deseable ni medir el rendimiento en este aspecto, pues existen conexiones con distinta latencia, que pueden hacer uso de distintos protocolos de transporte, y adicionalmente atravesar un número variable de enlaces y routers congestionados [20]. Adicionalmente hay ciertas condiciones donde solo algunos protocolos pueden lograr una buena utilización del enlace [21, 22, 15], en cuyo caso se da un conflicto entre desear que el ancho de banda se distribuya equitativamente y no desperdiciar la capacidad del enlace. En efecto, si solo se toma en consideración lo primero, entonces se llegaría a una solución que sería subóptima en cuanto al uso eficiente de los recursos disponibles, logrando una utilización total que podría ser significativamente menor a la que permite la capacidad del canal.

Uno de los estudios que se realiza comúnmente en la literatura de variantes de TCP se enfoca en la capacidad que tiene un nuevo algoritmo para conseguir que dos conexiones que lo emplean se distribuyan equitativamente el ancho de banda de un enlace en el cual compiten [23, 24]. En particular, el enlace de interés corresponde al del cuello de botella experimentado por ambas conexiones, el cual se caracteriza por ser el de menor capacidad en el camino que siguen los paquetes a través de la red.

Por otra parte, existe un concepto basado específicamente en la justicia que se obtiene cuando compiten conexiones que utilizan una variante de TCP tradicional y otras variantes de TCP. En este caso no se habla de justicia sino que de TCP-friendliness, pues se piensa en qué tan abusivos pueden llegar a ser estos algoritmos con respecto al control de congestión tradicional [25]. Esto es importante para toda nueva propuesta, pues debe apoyar la idea de un despliegue gradual durante el cual sin duda existirá competencia con otras conexiones que hacen uso del protocolo original. Esto se dará en parte debido a que los mecanismos utilizados por el protocolo original son los únicos mecanismos que forman parte de un estándar de la IETF [26], y que permiten tener una base mínima para tener un control de congestión que evite un colapso como el visto a finales de la decada de los ochenta [27]. Por otro lado, muchas de las variantes más recientes hacen uso de la dinámica de TCP tradicional como una base para operar durante periodos de alta congestión [15, 21, 23], con el fin de poder lograr ser más amistosos con TCP tradicional, lo cual de cierta forma refuerza la omnipresencia del protocolo original.

Otro estudio que se realiza frecuentemente se enfoca en el análisis de la distribución del ancho de banda cuando las conexiones tienen distintos niveles de latencia [15, 28, 29, 30], y corresponde al tema en torno al cual se centra el presente trabajo. La necesidad de este estudio se basa en una injusticia inherente a la dinámica del estado estacionario que caracteriza al control de congestión empleado por TCP tradicional [31, 32]. Sin embargo, este problema también lo experimentan la mayoría de las nuevas variantes de TCP [33].

Por otra parte, a nivel de AQM no existen muchas propuestas que aborden el problema. En particular, si bien existen propuestas que logran combatir el problema [11, 34, 35], sus ventajas son limitadas y no logran eliminar el sesgo en contra de las conexiones de alta latencia. Adicionalmente, los avances prácticos en esta disciplina son menores en comparación

a aquellos de la capa de transporte, lo cual ha motivado a que la IETF vuelva a promover la investigación con la actualización de [8], sin cambiar los objetivos del documento original [6]. Esto se puede atribuir en parte a factores como la complejidad que radica en la configuración de los parámetros de un algoritmo como RED para lograr un buen rendimiento en un gran número de escenarios [2, 3], así como también puede ser que la comunidad que trabaja en redes no sienta que exista esta necesidad.

1.3. Hipótesis

Un algoritmo de AQM puede lograr ser justo considerando en su operación el valor del RTT de las conexiones TCP. Por justicia se considera que, por medio del algoritmo propuesto, todas las conexiones que fluyen por el dispositivo de enrutamiento del cuello de botella obtienen la misma tasa de transmisión de datos, independiente de sus RTTs. A su vez, se espera que esto no implique una disminución relevante de la utilización del canal una vez implementado el algoritmo propuesto, el cual debe conocer las latencias para operar, en comparación al caso que estas son ignoradas.

1.4. Objetivos

1.4.1. General

- Diseñar y explorar las ventajas y desventajas de un algoritmo de AQM que logre ser justo mediante el conocimiento del RTT de las conexiones TCP.

1.4.2. Específicos

- Profundizar el estudio del estado del arte relacionado al control de congestión del punto de vista de TCP y de AQM.
- Investigar métricas y criterios de fairness utilizados comúnmente para conexiones de TCP.
- Analizar alternativas de implementación que permitan disponibilizar el valor del RTT para realizar un descarte diferenciado de paquetes.
- Estudiar el rendimiento obtenido por TCP tradicional al interactuar con routers que hacen uso de RED y ARED. Por rendimiento entiéndase que se medirá el nivel de utilización total del canal y el Fairness Index (FI).
- Evaluar el cambio en el FI y la utilización total del canal cuando se hace uso de la propuesta sobre RED y ARED.
- Analizar posibles mejoras al algoritmo de encolamiento en base al desempeño observado de RED y ARED.

1.5. Alcances

La propuesta que se presenta está diseñada específicamente para lograr fairness en base al comportamiento en estado estacionario de TCP tradicional. De tal forma, si bien la idea general resulta válida para otras implementaciones, teóricamente debería rediseñarse matemáticamente el algoritmo y su implementación para aspirar a lograr una mayor justicia.

Por otra parte, las simulaciones solo consideran tráfico en una dirección, y solo se estudian tráficos de alta duración. En particular, el origen del tráfico son sesiones de File Transfer Protocol (FTP) entre clientes y servidores.

1.6. Estructura de la tesis

La estructura del presente documento es la siguiente:

- En el Capítulo 2 se describen conceptos teóricos básicos para la comprensión del documento, así como también se aborda la evolución del control de congestión y el estado del arte.
- El Capítulo 3 describe brevemente la propuesta, explicando su motivación teórica y justificando la elección de algunas constantes.
- El Capítulo 4 presenta la metodología empleada para la evaluación de la propuesta, lo cual incluye tanto la descripción de los escenarios simulados, así como también los valores utilizados para los parámetros que definen el comportamiendo de RED y ARED.
- El Capítulo 5 se muestran y analizan los resultados obtenidos a partir de las simulaciones realizadas.
- El Capítulo 6 presenta las conclusiones del trabajo, y se indica el posible trabajo futuro.

Capítulo 2

Marco Teórico

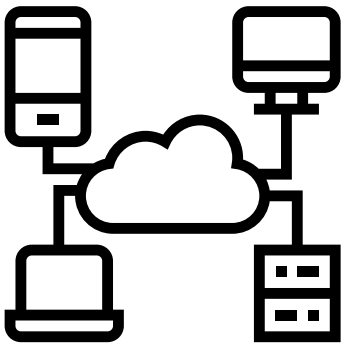
2.1. Conceptos básicos de telecomunicaciones

En esta sección se explican brevemente algunos conceptos fundamentales para la comprensión del contexto en el cual se sitúa el tema que aborda la presente tesis. Esto incluye una noción básica sobre cómo se transmite la información entre dispositivos a través de Internet, el concepto de protocolo, y el uso de capas para facilitar la comprensión de este complejo sistema de comunicaciones.

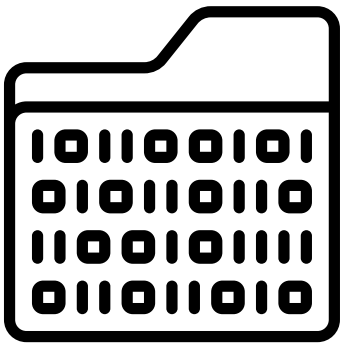
2.1.1. Internet y transmisión de información digital

Hoy en día, gran parte de nuestra vida gira en torno a lo que se conoce como Internet, lo cual consiste basicamente en una red de redes que permiten el intercambio de información entre dispositivos conectados a esta (ver Figura 2.1a). En particular, esta información es digital, lo cual quiere decir que se representa como una secuencia de unos y ceros. Debido a lo anterior se dice que el sistema es binario, pues esta secuencia se puede descomponer en unidades básicas que solo pueden tomar dos valores. A esta unidad básica se le denomina bit (abreviación de binary digit), y usualmente también suele ocuparse el término byte, el cual se utiliza para denominar un conjunto de 8 bits. Lo anterior se representa esquemáticamente en la Figura 2.1b.

Adicionalmente, si lo que se desea transmitir es lo suficientemente grande, entonces no se transmite la secuencia completa, sino que esta se divide en bloques de bits, los cuales componen lo que se denomina un paquete. Estos bloques de datos pueden viajar a través de distintos caminos para llegar al receptor, y los componentes de Internet encargados de dirigir a los paquetes se denominan routers. En particular, estos dispositivos poseen conexiones a múltiples enlaces (medio físico a través del cual viajan los paquetes) y cuando reciben un paquete proveniente de uno de estos, su función es determinar cual es el siguiente enlace a través del cual debe viajar este paquete (para llegar a su destino) y posteriormente realizar el redireccionamiento correspondiente a nivel de hardware.



(a) Internet como un medio para interconectar dispositivos que se desean comunicar entre sí.



(b) Un archivo como una secuencia de unos y ceros. Un byte es una secuencia de 8 bits.

Figura 2.1: Internet y concepto de información digital.

2.1.2. Stack de Protocolos de Internet y el Modelo OSI

Si bien lo anterior permite hacerse una idea del funcionamiento general de Internet, es necesario introducir un concepto clave: el de los protocolos. No es posible tener un sistema de comunicación funcional si es que no hay un acuerdo entre los participantes que permita la comprensión de los mensajes transmitidos. Este es precisamente el rol de los protocolos, los cuales se pueden entender como un set de reglas o pasos a seguir para que el intercambio de información sea factible.

Con el fin de simplificar el diseño de protocolos y darle algo de estructura a la comunicación digital, se utiliza el concepto de "capas" para definir conceptualmente las distintas etapas a través de las cuales debe pasar la información para llegar de un dispositivo a otro. Uno de los modelos más típicos se conoce como stack de protocolos de Internet, el cual posee 5 capas: aplicación, transporte, red, enlace y física. Esto es muy similar al modelo de referencia desarrollado por la Organización Internacional de Normalización (ISO, International Organization for Standardization), denominado modelo de interconexión de sistemas abiertos (también conocido como modelo OSI, por Open System Interconnection), el cual tiene 7 capas. Estas incluyen las 5 mencionadas previamente, pero se tienen adicionalmente las capas de sesión y de presentación. Una comparación entre ambos modelos mencionados se presenta en la Figura 2.2.

A continuación se presenta una breve descripción de cada una de las capas del stack de protocolos de Internet:

- **Capa de aplicación**: es la capa donde se originan los datos, y en la cual operan los protocolos asociados a las aplicaciones que se ejecutan en los dispositivos que hacen uso de la red. Un ejemplo de estos protocolos corresponde a Hypertext Transfer Protocol (HTTP), el cual estandariza el intercambio de documentos en la World Wide Web (WWW), y es clave para el funcionamiento de un navegador. Adicionalmente, esta capa se caracteriza por no tener una cabecera, la cual es utilizada en el resto del proceso para implementar las funcionalidades particulares de cada capa.

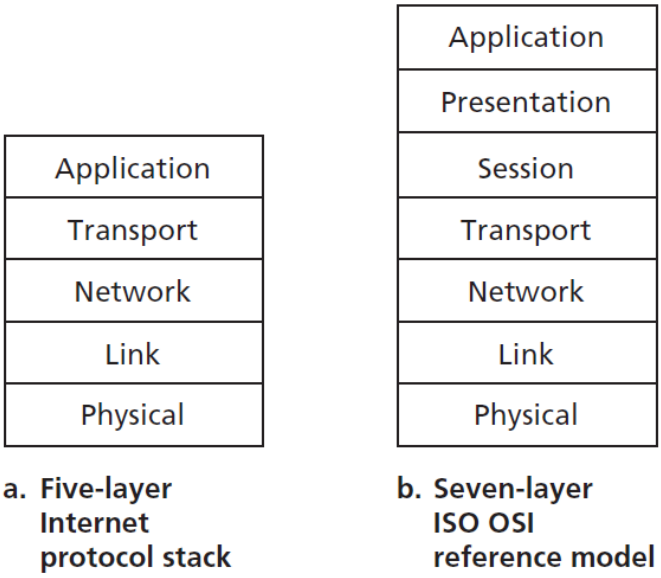


Figura 2.2: Stack de protocolos de Internet vs modelo OSI.

- **Capa de transporte**: su principal función radica en transmitir los datos provenientes de la capa de aplicación al destinatario respectivo. En particular, es esta capa la que establece la comunicación entre las aplicaciones o procesos que se ejecutan en los dispositivos emisor y receptor. Es importante notar que dado que un dispositivo suele ejecutar múltiples aplicaciones simultáneamente, es necesaria la identificación de los mismos en base a lo que se denomina como puertos.

  Un aspecto relevante que aborda la capa de transporte es el uso de técnicas para asegurar la correcta transmisión de los datos, o al menos evitar la recepción de datos corruptos. Para lo primero se hace uso de reconocimientos, y para lo segundo se envían datos de redundancia que posteriormente se pueden utilizar para verificar si los datos recibidos son los mismos que se transmitieron.

  Por otra parte, en la capa de transporte se hace uso de técnicas para controlar la magnitud del flujo de datos. En el caso del control de flujo, la idea es evitar saturar un buffer en el receptor, mientras que cuando se habla de control de congestión el objetivo es prevenir o contrarrestar la congestión a nivel de la red, la cual se manifiesta en la forma de colas de paquetes que se ubican en buffers de los routers.

  Existen dos protocolos responsables de la funcionalidad a nivel de capa de transporte para la mayoría del tráfico. El primero corresponde a TCP, el cual se caracteriza por requerir el establecimiento de una conexión previo al flujo de datos, así como también por asegurar la eventual recepción correcta de los datos mediante el uso de reconocimientos y retransmisiones. El segundo protocolo corresponde a User Datagram Protocol (UDP), el cual solo proporciona funcionalidades básicas como permitir la comunicación entre procesos y utilizar bits de redundancia. En este caso no se emplean retransmisiones ni se requiere el establecimiento de una conexión.
- **Capa de red**: se encarga de permitir la comunicación entre dispositivos en base al uso de direcciones que deben ser únicas a nivel global para poder direccionar ded forma correcta los paquetes de datos a través de la red. El principal protocolo de esta capa es

Internet Protocol (IP), en base al cual se habla de direcciones IP. Otros protocolos que operan en esta capa son aquellos que se encargan de el proceso de enrutamiento, tales como Open Shortest Path First (OSPF) y Border Gateway Protocol (BGP).

- **Capa de enlace**: a veces también denominada capa de acceso, es responsable de la transmisión de datos desde un nodo a otro a través de un determinado enlace perteneciente a la ruta que debe seguir el paquete de datos. En este caso, el protocolo a utilizar depende del tipo de enlace, y una de las funcionalidades implementadas en la capa puede ser la tranmisión confiable de datos. Es decir, es posible utilizar bits de redundancia y retransmisiones para evitar la recepción de datos corruptos en el otro extremo del enlace respectivo.
- **Capa física**: se preocupa de la transmisión correcta de cada bit a través del enlace, en vez del paquete completo como es el caso de la capa anterior. En esta capa es donde se ubican temas como la modulación y multiplexación de la información.

2.2. Control de congestión

2.2.1. ¿Cómo se genera congestión en Internet?

La infraestructura que compone internet consta principalmente de un gran número de enlaces y dispositivos de enrutamiento. Estos últimos se suelen denominar como routers, y tienen como objetivo el direccionar de forma correcta los paquetes de datos. En particular, a un router se conectan múltiples enlaces, de tal forma que cuando llega un paquete por uno de estos, es necesario determinar a través de protocolos de enrutamiento cuál es el correcto enlace por el cual se debe enviar el paquete para que pueda llegar a su destino. En general, los routers se comunican constantemente entre ellos para mantener actualizada una tabla de enrutamiento, en la cual se asocian direcciones IP de destino a determinados enlaces. Por lo tanto, a nivel práctico lo que sucede es que se lee la IP de destino, se revisa la tabla, y se envía el paquete por el enlace indicado en esta.

Sin embargo, es posible que al momento de haber determinado el enlace de salida, este último esté ocupado transmitiendo otro paquete. En dicho caso, se procede a ubicar al paquete en una cola (un buffer de datos), con el fin de poder transmitirlo una vez que haya llegado su turno, ya que es posible que la cola se encuentre ocupada con paquetes que llegaron previamente.

Para que se generen colas basta que el flujo de datos que ingresa al router con un determinado destino sea mayor al flujo de salida del enlace respectivo. En particular, hay dos factores que influyen en la ocurrencia de lo anterior:

- La capacidad del enlace de salida es menor a la del enlace de entrada. Esto se puede dar incluso cuando solo hay una ruta posible (un enlace de entrada y uno de salida).
- Llegan paquetes por múltiples enlaces de entrada a un mismo enlace de salida. Esto permite se pueda generar congestión sin necesitar una diferencia en la capacidad de los enlaces.

En la práctica se da una combinación de las dos situaciones previamente mencionadas, lo cual se ilustra en la Figura 2.3. Allí, es posible notar que el tráfico proveniente de los nodos emisores (N1 y N2) se transmite a una tasa mayor a la del enlace de salida común (el cual se dirige al nodo receptor N3), lo cual sumado al hecho de que puede darse un arrivo simultáneo de paquetes habilita la generación de congestión. En particular, basta que un RTT sea un múltiplo del otro para que se den transmisiones simultáneas de ventanas de datos de forma periódica.

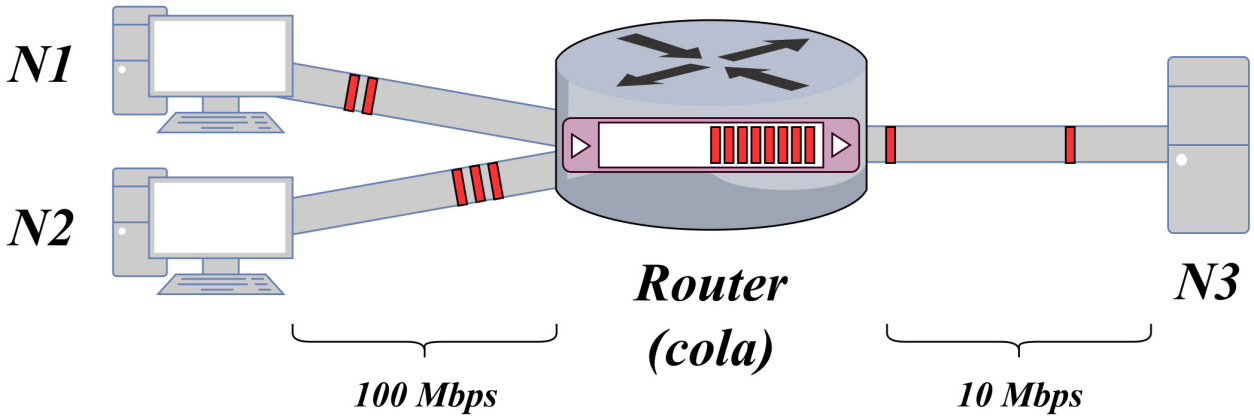


Figura 2.3: Fenómeno de congestión manifestado a nivel de router como una cola de paquetes esperando su turno para ser transmitidos.

No obstante, la situación ilustrada previamente sigue siendo una simplificación, puesto que en la práctica también es posible utilizar múltiples colas y algoritmos diseñados para manejar el fenómeno de la congestión en base al trato diferenciado de paquetes con distintos niveles de prioridad [36]. En particular, el sistema ilustrado en la Figura 2.3 se denomina First In First Out (FIFO), y se caracteriza por generar un retraso temporal (en la transmisión del paquete) que es directamente proporcional a la cantidad de paquetes en la cola.

Naturalmente, se debe cuestionar qué es lo que sucede cuando la cola se llena por completo, puesto que en dicho momento no existe capacidad física en el buffer para almacenar más paquetes. En este caso existen dos alternativas: descartar el paquete que intenta entrar a la cola, o bien alguno que ya se encuentra inserto en esta. Sea cual sea la opción que se tome, el solo hecho de descartar paquetes en el router no es suficiente para solucionar el evento de congestión, y tal como se describirá posteriormente es necesario que los dispositivos que transmiten datos regulen sus tasas de transmisión en base a la detección de estos eventos.

Por lo tanto, se pueden distinguir las siguientes consecuencias asociadas a la ocurrencia de congestión:

- Se generan retrasos adicionales que contribuyen a la latencia que experimentan los paquetes de datos durante su transmisión.
- Se pueden generar pérdidas de paquetes cuando las colas se llenan por completo. Esto equivale a un desperdicio de los recursos que fueron utilizados para la transmisión de los paquetes descartados, así como también implica que la transmisión efectiva de datos (la cual considera solo aquello que se recibe en el receptor) se ve afectada negativamente.

2.2.2. Orígenes del control de congestión

El control de congestión surge como una solución a un evento en particular experimentado por Internet en la decada de los ochenta, en particular en el año 1986. Durante este incidente, se observaron disminuciones en los throughputs medidos en el orden de mil veces menor a los obtenidos previo a este fenómeno. Esto derivó en que Van Jacobson publicara en 1988 [27] una propuesta que describe los mecanismos fundamentales del control de congestión recomendados por la IETF en la actualidad (RFC 5681 [26]), y son la base de lo que se conoce como TCP tradicional.

Sin embargo, este no fue el primer trabajo en abordar el tema. De hecho, el conocido comportamiento del tipo Additive Increase Multiplicative Decrease (AIMD) que caracteriza a TCP (en particular, a congestion avoidance) fue propuesto previo al trabajo de Jacobson. No obstante, este último ha demostrado lograr una robustez notable en el tiempo, permaneciendo operativo hasta nuestros días, a pesar de la existencia de nuevas variantes del protocolo original. A pesar de esto, el rendimiento de TCP tradicional es conocido por no ser capaz de obtener resultados óptimos en condiciones donde el ancho de banda disponible y la latencia son altos [21].

Lo anterior es uno de los principales factores que ha motivado el desarrollo de nuevas propuestas. La mayoría de estos protocolos estan diseñados para seguir un comportamiento basado en la ocurrencia de eventos de congestión, los cuales se asocian a la pérdida de paquetes. En particular, reaccionan frente a estos realizando ajustes a su ventana de congestión, usualmente denotada como *cwnd*. Esta corresponde a una variable de estado controlada por TCP, y se encarga de limitar la tasa de transmisión de datos con el fin de evitar congestionar la red.

Es importante mencionar que, además de la necesaria disminución de la ventana de congestión frente a un evento de congestión, los algoritmos deben ser capaces de adaptarse a las cambiantes condiciones de Internet. Esto se debe a que existe la posibilidad de que el ancho de banda disponible en un enlace aumente debido al término de algunas conexiones, así como también luego de una transición de un estado congestionado a uno no congestionado (considerar la posible disminución simultánea de la tasa de transmisión de múltiples conexiones). En ambos casos es claro que, para poder hacer uso eficiente de los recursos disponibles (en este caso, el ancho de banda liberado), los emisores deben aumentar su caudal de datos de forma conservadora (al menos durante un tiempo prudente), con el fin de prevenir congestionar el enlace y generar oscilaciones a nivel de carga del sistema.

2.2.3. Evolución de TCP tradicional y nuevos avances en la capa de transporte

2.2.3.1. TCP tradicional

TCP tradicional es un concepto que hace referencia a una familia de algoritmos, los cuales nacen a partir de la evolución de la propuesta inicial de V. Jacobson [27]. Esta se conoce como TCP Tahoe, y se caracteriza por el uso de 3 algoritmos; slow start, congestion avoidance y fast retransmit, los cuales controlan el valor de la ventana de congestión y se basan en un principio de conservación de paquetes. Este control de la ventana se basa en dos tipos de eventos: la recepción de paquetes de reconocimiento (ACKs, Acknowledgements) y la ocurrencia de eventos de congestión (pérdida de paquetes o un timeout).

Los paquetes de reconocimiento son una parte fundamental de TCP, necesaria para garantizar la recepción correcta de la información, y cada uno de estos se encarga de confirmar la recepción correcta de una cierta cantidad de datos. Esto último se toma como una indicación de que la red no está congestionada, y por lo tanto se considera apropiado el aumentar el tamaño de la ventana de congestión. Es importante tener en cuenta que es absolutamente necesario que la ventana sea dinámica, pues en cualquier instante de tiempo se puede dar que ciertas conexiones que pasan por la red terminen de transmitir, lo cual deja disponible el ancho de banda previamente utilizado por estas.

En el caso de eventos de congestión, estos pueden ser detectados de forma implicita o explícita. Lo más común es inferirlos de forma implícita, y en esta categoría hay dos formas de proceder. En primer lugar, se puede utilizar la recepción de ACKs duplicados (DUPACKs, Duplicate Acknowledgments), lo cual se refiere a la recepción de paquetes de reconocimiento que no reconocen más datos que uno previamente recibido. Sabiendo esto, es posible asociar la recepción de múltiples de estos paquetes a la ocurrencia de una pérdida. En segundo lugar, es posible utilizar una cierta ventana de tiempo apropiada durante la cual se esperará por el reconocimiento de los paquetes enviados. Si este tiempo es superado se dice que ocurre un timeout, y se asume que el paquete se perdió debido a congestión en la red. La detección de eventos de congestión de forma explícita es algo que surgió posteriormente, y el caso más emblemático es el de Explicit Congestion Notification (ECN), un mecanismo que se basa en la utilización de un par de bits en la cabecera de IP para que los routers puedan notificar de forma directa a los extremos sobre la existencia de congestión [37].

Sea cual sea la forma en que se detecten eventos de congestión, la acción apropiada luego de estos es la reducción de la ventana de congestión, ya que tal como se explicó en la Sección 2.2.1 la congestión se da cuando la tasa de ingreso de datos a un determinado enlace es mayor a la tasa de salida del mismo. En el caso en que se detectan pérdidas mediante la recepción de DUPACKs la ventana sufre una baja multiplicativa, mientras que cuando ocurre un timeout suele disminuirse al tamaño que tenía al iniciar la transmisión. Esto se debe a que en general un timeout solo ocurre cuando existe un nivel de congestión significativo.

Dicho esto, para comprender cómo operan los tres algoritmos previamente mencionados, se describirá brevemente en qué momento se utiliza cada uno y cómo manejan la ventana

de congestión. Luego, sea una conexión TCP que está comenzando a transmitir datos, esta partirá con una ventana de congestión con un cierto tamaño inicial pequeño [38] (para el caso de TCP Tahoe este corresponde a 1 paquete). Teniendo en cuenta esto, se comenzará a transmitir datos y la ventana de congestión se actualizará mediante slow start, mecanismo que plantea un incremento exponencial de la ventana. En particular, esta se incrementa en un paquete por cada ACK recibido, efectivamente duplicando la ventana cada vez que es reconocida por completo (lo cual toma aproximadamente un RTT). Luego de esto, existen dos formas de pasar desde slow start a congestion avoidance. Por una parte, si el tamaño de la ventana supera un determinado umbral *ssthresh* (slow start threshold), entonces se considera que el incremento exponencial es muy agresivo y se comienza a utilizar congestion avoidance. Por otra parte, si se detecta un evento de congestión también se considera que es prudente pasar a un incremento mas conservador. Específicamente, la ventana de congestión durante congestion avoidance se incrementa en un paquete por ventana completa reconocida (es decir, un paquete por RTT aproximadamente). Finalmente, fast retransmit lo que plantea es inferir la ocurrencia de una pérdida de paquete en base a la recepción de 3 DUPACKS, lo cual permite evitar esperar a que ocurra un timeout para realizar la retransmisión del paquete y reducir la ventana.

La inclusion de fast recovery, dos años más tarde, es lo que marca la diferencia entre TCP Tahoe y la nueva versión del algoritmo, el cual se denomina TCP Reno. El mecanismo como tal permite una rápida transición a congestion avoidance luego de la retransmisión que se genera producto de la recepción de 3 DUPACKs, disminuyendo la ventana de congestión inmediatamente a la mitad de su valor previo. A diferencia de lo anterior, en el caso de TCP Tahoe, donde no existe fast recovery y se invoca un slow start luego de la retransmisión del paquete (ya sea que se haya producido por timeout o por la recepción de DUPACKs). El comportamiento de TCP Reno se presenta en la Figura 2.4, donde se presenta la evolución de la ventana de congestión en el tiempo. Allí, W corresponde a un valor arbitrario para la ventana, el cual simplemente se utiliza para mostrar claramente la disminución multiplicativa (a la mitad del valor que se tenía previo a la pérdida) que utiliza TCP Reno.

La figura anterior no toma en consideración los pequeños ajustes realizados a la ventana durante la etapa de fast recovery, donde se incrementa la ventana en una unidad por cada DUPACK adicional que es recibido luego de los 3 DUPACKs que gatillan fast retransmit. En general, fast recovery dura una fracción de un RTT, y los incrementos realizados se revierten una vez que se recibe un ACK que reconoce nuevos datos. Una descripción detallada de la operación de TCP Reno, incluyendo los cuatro mecanismos mencionados (slow start, congestion avoidance, fast retransmit y fast recovery) se puede encontrar en [39].

Uno de los problemas que aqueja a TCP Reno corresponde a la ocurrencia de múltiples pérdidas en una misma ventana. Si esto ocurre se pueden experimentar múltiples divisiones de la ventana de congestión o incluso gatillar un timeout. Luego, para lidiar con estos problemas se desarrollaron dos propuestas que se conocen como TCP New Reno y TCP SACK, las cuales introducen mejoras significativas en el rendimiento del algoritmo cuando se experimentan altas tasas de pérdida de paquetes.

En la actualidad, New Reno corresponde a la opción recomendada cuando no es posible utilizar TCP SACK [40], ya sea porque no esté disponible en uno de los nodos o bien que

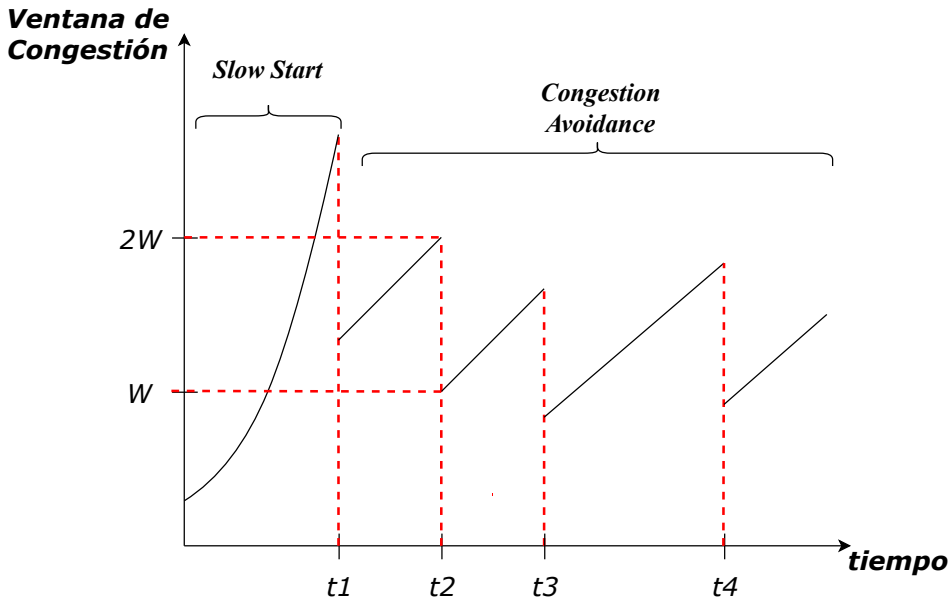


Figura 2.4: Evolución de la ventana de congestión en el tiempo cuando se utiliza TCP Reno. El comportamiento se caracteriza por dos etapas, las cuales están asociadas a los algoritmos slow start y congestion avoidance. En el caso en que no existen múltiples pérdidas en una misma ventana, este comportamiento representa correctamente a la familia de algoritmos denominada como TCP tradicional (Reno, New Reno y SACK).

alguno de estos manifieste la voluntad de no utilizarlo (lo cual no es recomendable). El algoritmo como tal lo que propone es una respuesta inteligente frente a lo que se denomina como ACK parcial. Este tipo de ACKs se presentan cuando existe mas de 1 pérdida por ventana, en cuyo caso el reconocimiento de la retransmision del primer paquete perdido no reconocerá todos los paquetes que estaban en vuelo al momento de detectarse la pérdida (lo cual sucedería en caso de una perdida única y en ausencia de reordenamiento de paquetes), sino que solo una parte de ellos (origen del término "parcial"). Esta respuesta incluye dos respuestas importantes: la primera de ellas corresponde al manejo apropiado de la ventana y la segunda está asociada a la retransmisión inmediata del primer paquete que no fue reconocido por el ACK parcial (en otras palabras, donde se detuvo el reconocimiento).

Si bien lo anterior es un avance significativo sobre la versión de fast recovery utilizada para TCP Reno, estas decisiones se hacen en base a información escasa. Una forma de lograr una respuesta más apropiada es captar mayor información respecto de lo que está sucediendo mediante la utilización de reconocimientos selectivos. Este tipo de reconocimientos se denominan Selective ACKs (SACKs y ellos junto a un manejo apropiado de esta información es lo que compone al algoritmo TCP SACK [41].

El conjunto de protocolos que incluye a TCP Reno, New Reno y SACK es lo que se suele llamar TCP tradicional o estándar. Esta familia ha demostrado una robustez significativa frente a diversas condiciones a lo largo de la evolución de Internet. Sin embargo, los avances tecnológicos de las últimas decadas han hecho posible el uso de enlaces capaces de soportar tasas de transmisión muy altas. Esto, combinado a enlaces de larga distancia (como los intercontinentales) generan un escenario donde TCP tradicional no es capaz de adaptarse de

forma efectiva, debido a su incremento conservador de la ventana de congestión durante la etapa de congestion avoidance [21].

2.2.3.2. Otras variantes de TCP

Si bien muchas de las propuestas que se han desarrollado en esta disciplina se basan en la ocurrencia de eventos de pérdida de paquetes, existen métodos alternativos que toman en consideración otros aspectos tales como el retraso o latencia adicional causada por la congestión. Una de las primeras propuestas que utiliza esta perspectiva corresponde a TCP Vegas [42], un algoritmo de control de congestión que se basa en el incremento del RTT para poder detectar congestión. Lamentablemente, algunos algoritmos que se basan en este tipo de métricas suelen tener problemas al competir con aquellos que se basan en la ocurrencia de pérdidas, pues suelen reaccionar de forma prematura a la congestión en comparación a aquellos protocolos basados en pérdidas [43], lo cual resulta en la obtención de throughputs que pueden ser significativamente menores [44].

Otro trabajo temprano que es interesante es el análisis generalizado del comportamiento AIMD propuesto para TCP tradicional, razón por la cual se denomina General Additive Increase Multiplicative Decrease (GAIMD). A través de esta parametrización se genera un modelo del throughput obtenido en estado estacionario y se deriva analíticamente la relación que debe existir entre los factores de incremento aditivo y decrecimiento multiplicativo (α y β respectivamente) para que el algoritmo resultante sea amistoso con TCP tradicional. En particular, se encuentra que el par de valores que permite esto corresponde a $\alpha = 0{,}31$ y $\beta = 0{,}875$, lo cual permite adicionalmente reducir la fluctuación de la ventana en base a una reducción menos brusca de la ventana luego de un evento de congestión.

Por otra parte, la necesidad por utilizar efectivamente los nuevos enlaces de alta velocidad dio paso a un número significativo de propuestas cuyo principal objetivo es aumentar la capacidad de TCP para aprovechar los recursos que entregan este tipo de enlaces. Algunas de estas que se basan en aplicar control de congestión en función de pérdidas incluyen protocolos como Highspeed (HSTCP) [21], Scalable (STCP) [22], y Hamilton (H-TCP) [23], aunque sin duda el trabajo de más impacto nace a partir del desarrollo de Binary Increase Congestion control (BIC), un protocolo que no solo es capaz de aprovechar dichos enlaces (utilizando una búsqueda binaria para el incremento de la ventana de congestión), sino que además logra hacerlo tomando en consideración el problema del RTT fairness dentro de su operación. Tal sería su aceptación que en el año 2004 pasa a ser el protocolo de control de congestión utilizado por defecto en el kernel de Linux. Lo anterior motiva el diseño de una versión mejorada de BIC denominada CUBIC [15] tan solo un año más tarde, y que posteriormente en 2006 tomaría el lugar de BIC como el protocolo utilizado por defecto en Linux [45]. El principal cambio desde BIC a CUBIC yace en la introducción de una función cúbica para la evolución temporal de la ventana de congestión.

De esta ola de protocolos también hay otros que incorporan en su operación una componente temporal asociada a la latencia. TCP Illinois [46] hace uso de lo anterior para modificar el algoritmo AIMD de TCP tradicional con el fin de producir una curva cóncava en la evolución de la ventana de congestión en el tiempo. Otra propuesta icónica que realiza algo similar

es Compound TCP [25], un algoritmo desarrollado por Microsoft que utiliza el throughput de TCP tradicional como una base sobre la cual se añade un throughput secundario que es controlado completamente por datos de latencia. Este protoloco fue utilizado como protocolo por defecto en algunos de sus sistemas operativos [47], aunque recientemente ha sido reemplazado por CUBIC en Windows 10 [48]. En cuanto a propuestas que solamente utilicen la componente temporal, una de ellas que se inspira en TCP Vegas corresponde a Fast Acive Queue Management Scalable TCP (FAST) [49], la cual propone variabilidad en la tasa a la cual se actualiza la ventana dependiendo de la distancia entre el estado actual y el punto de equilibrio.

Existen otras instancias donde se han propuesto protocolos para abordar problemas en un determinado contexto o bien apuntan a un objetivo específico distinto a la escalabilidad de los throughputs en enlaces de alta velocidad. Un ejemplo de lo primero es Datacenter TCP (DCTCP) [50], una propuesta que extiende la funcionalidad de ECN para efectuar reducciones de la ventana con una mayor cantidad de información (en particular, la proporción de datos que experimentó congestión con respecto al total). Respecto a lo segundo, TCP Libra [51] es una propuesta diseñada específicamente con el fin de abordar el problema del RTT fairness, y se basa en la modificación de los factores aditivo y multiplicativo de TCP tradicional. Específicamente, estos se ven afectados por el valor del RTT, una función de control y constantes que ajustan la sensibilidad de ciertas componentes.

En cuanto a tiempos recientes, se ha generado interés por algoritmos basados en algún tipo de aprendizaje en vez de la definición heurística de reglas que parezcan razonables para alcanzar un cierto objetivo. Una de las propuestas que se basa en esta idea es TCP Ex Machina [52], un trabajo en el cual se desarrolla un programa que permite generar algoritmos de control de congestión en base a ciertas suposiciones sobre la red sobre la cual operará el algoritmo, un modelo de tráfico y una función objetivo. Otro algoritmo que se podría considerar en esta categoría corresponde a Performance-oriented Congestion Control (PCC) [53], aunque en este caso el aprendizaje ocurre durante la operación del algoritmo. En particular, se analiza constantemente el resultado de distintas acciones (como aumentar y disminuir la tasa de transmisión) frente a eventos de congestión, con el fin de validar empíricamente la mejor alternativa. Es decir, solo despues de probar ambos caminos es que se toma la decisión final, en vez de utilizar una respuesta predeterminada como se hace en TCP tradicional, lo cual puede ser inapropiado si es que por ejemplo la pérdida no se debe a congestión en la red. Esta idea es la base del algoritmo, y esto combinado a conceptos de optimimzación utilizados en aprendizaje de máquinas resultó en una variante de mejorada del algoritmo original [54].

Un algoritmo que se debe destacar es Bottleneck Bandwidth and Round-trip propagation time (BBR) [55], pues corresponde a un trabajo muy reciente que ya posee una implementación en el kernel de Linux. Esto se debe en gran parte al hecho de haber sido desarrollado por investigadores de Google, lo cual facilitó además la evaluación del mismo en la infraestructura privada de la empresa [56], donde se afirma que este logra superar a CUBIC en cuanto al throughput obtenido por un margen significativo (al menos 2 veces mayor), lo cual se atribuye a la tolerancia del algoritmo frente a distintos niveles de pérdida. Esta propiedad proviene del diseño del algoritmo, el cual no utiliza las pérdidas como una señal de congestión, sino que se basa en la estimación del ancho de banda del cuello de botella y de la componente de propagación de la latencia que experimentan los paquetes.

En la Tabla 2.1 se presenta una línea temporal que permite ubicar temporalmente a las propuestas mencionadas previamente. En la actualidad, el protocolo dominante es CUBIC, el cual desde el inicio de esta década ya mostraba indicios de su gran aprobación [57]. Esto, sumado a la reciente adopción del mismo en Windows 10, establece a una propuesta de hace una década como la principal representante del estado del arte. No obstante, a pesar de que no se hayan logrado grandes avances a nivel de implementación de nuevas propuestas en el último tiempo, este no es un problema olvidado [14, 58]. Un proyecto muy relevante en este contexto y que fue fundado recientemente se denomina Pantheon [59][60], el cual tiene como objetivo servir como plataforma para la investigación del control de congestión en TCP.

Tabla 2.1: Propuestas de TCP a través del tiempo. [1]

1988	TCP Tahoe [27].
1990	TCP Reno* [61].
1994	TCP Vegas* [42].
1995	TCP New Reno* [62].
1996	TCP SACK [63].
2000	TCP GAIMD [64].
2003	TCP Highspeed* [21], , Scalable TCP* [22].
2004	TCP BIC* [65], H-TCP* [23], FAST TCP [49].
2005	TCP CUBIC* [15].
2006	Compound TCP [25], TCP Ilinois* [46].
2007	TCP YeAH* [66], TCP Libra [51].
2011	DCTCP* [50].
2013	TCP Ex Machina [52].
2015	PCC Allegro [53].
2016	TCP BBR* [55].
2018	PCC Vivace [54].

[1] **Disponibles en el kernel de linux*

2.2.4. Evolución de AQM como un complemento a TCP

2.2.4.1. Drop Tail, Random Drop y Early Random Drop

Cuando no se tiene ningún algoritmo de AQM para lidiar con la congestión en el router, entonces lo que sucederá es que el buffer del mismo se llenará gradualmente hasta donde no exista más espacio en este, y el router se verá obligado a descartar el paquete que está ingresando. Este comportamiento es lo que se denomina como Drop Tail.

Un problema directo de lo anterior es que se permite que el buffer se llene por completo, lo cual implica una latencia adicional que puede ser significativa (dependiendo del tamaño del buffer). En particular, existe una regla usualmente utilizada para determinar el tamaño de los buffers que se basa en el comportamiento de TCP tradicional. Esta recomienda utilizar un tamaño del buffer igual al Bandwidth-Delay Product (BDP) del enlace [67], el cual se determina a partir de la multiplicación del RTT y la capacidad del canal.

Otro efecto algo más indirecto corresponde a aquel fenómeno que se conoce como sincronización global, el cual consiste en pérdidas sincronizadas de múltiples conexiones en un cierto periodo de congestión, lo cual posteriormente deriva en la baja simultánea de las ventanas de múltiples conexiones, generando posteriormente una infrautilización del canal [68]. Por otro lado, existe otro problema que yace en la naturaleza de Drop Tail y es el hecho de que su comportamiento genera un sesgo en contra de las conexiones que tienen un tráfico caracterizado por ráfagas de paquetes. Esto, pues en el momento en que hay congestión y el buffer está apunto de llenarse, es altamente probable que una ráfaga de paquetes experimente una pérdida comparado a un tráfico que es más espaciado en el tiempo.

Algunas de las primeras propuestas diseñadas para lidiar con la congestión incluyen: Source Quench [69], Fair Queueing [70] y Congestion Indication [71]. No obstante, las diversas limitaciones de estas llevaron a la IETF a considerar otras alternativas. Lo anterior derivó en la creación del algoritmo denominado Random Drop, el cual propone el decarte de un paquete aleatorio de la cola cuando el buffer se llena por completo (en vez del paquete que ingresa, como en el caso de Drop Tail). Esto permite brindar un trato un poco más justo, pues las conexiones que tengan una mayor cantidad de paquetes en el buffer tienen más probabilidades de experimentar una pérdida. No obstante, más allá de la mejora a nivel de fairness para conexiones homogéneas, el algoritmo no demostró otras ventajas notables. De hecho, se observó que una conexión con RTT alto obtuvo más perdidas con el uso del algoritmo, mientras que las conexiones de RTT bajo practicamente mantuvieron su rendimiento.[72]

Random Drop, al igual que Drop Tail, es incapaz de prevenir la congestión, ya que solo ataca el problema una vez que la congestión ya existe. Una versión modificada del algoritmo es estudiada en [68], la cual se denomina Early Random Drop, y que se basa en aplicar Random Drop de forma prematura, con el fin de evitar las altas pérdidas causadas cuando el buffer se llena por completo. En particular, Early Random Drop se basa en el uso de un umbral, luego del cual se asume que existe congestión, y se hace uso de una cierta probabilidad de pérdidas para aplicar Random Drop en los paquetes del buffer. En este trabajo tanto el umbral como la probabilidad de pérdidas son fijos, pero se indica que lo ideal sería

desarrollar algoritmos que permitan ajustar estos parámetros de forma dinámica. Los resultados encontrados demostraron que Early Random Drop obtiene resultados iguales o mejores que Random Drop. Esto se debe en parte al hecho de que Early Random Drop opera como Random Drop en el caso en que es incapaz de prevenir la congestión.

Early Random Drop de cierta forma plantea las bases del algoritmo más conocido en la disciplina de AQM; Random Early Detection (RED), el cual también pretende prevenir la congestión, haciendo uso de una lógica levemente más compleja, la cual permite ajustar dinámicamente la probabilidad de pérdidas y además incluye un estimador que evita generar un sesgo contra conexiones con tráfico del tipo ráfaga.

2.2.4.2. Random Early Detection (RED)

Random Early Detection es un algoritmo simple, el cual se basa en el descarte aleatorio de paquetes en función de un valor estimado del tamaño promedio de la cola, el cual se denotará como avg_q. En particular, se utilizan como referencia un umbral inferior y superior (min_{th} y max_{th} respectivamente), los cuales definen una función de probabilidad de pérdidas que depende de avg_q, la cual se presenta en la Figura 2.5. En esta se puede apreciar que la probabilidad de descartar un paquete es nula cuando avg_q es menor a min_{th}. Una vez que se supera el umbral inferior y hasta llegar al umbral superior, la probabilidad aumenta de forma lineal hasta llegar a max_p. Posteriormente, en el caso de la propuesta original existe una discontinuidad en max_{th} y la probabilidad pasa a ser constante y unitaria, lo cual implica que luego de este punto todos los paquetes son descartados [7]. Por otra parte, existe una variante gentil del algoritmo (propuesta en [73]) la cual plantea reemplazar la discontinuidad por un nuevo incremento lineal hasta una probabilidad de pérdidas unitaria, la cual se da solo al llegar al doble del umbral superior.

Por lo tanto, la diferencia entre la variante gentil y la original es clave para determinar el punto a partir del cual todos los paquetes comienzan a ser descartados. Esto es relevante, pues luego de esto básicamente se comienza a operar como Drop Tail, lo cual conlleva a permitir los problemas previamente mencionados que se dan cuando se tiene un buffer completamente lleno.

Es importante aclarar que la probabilidad de pérdidas real utilizada por RED no solo considera aquella que se obtiene de la función presentada en la Figura 2.5. Esto se debe a que los autores de la propuesta determinaron que la utilización directa de lo anterior implica una distribución no uniforme de las pérdidas en el tiempo. Luego, se propone la amplificación de la probabilidad por un factor que depende del número de paquetes que han ingresado a la cola desde la última pérdida. Sea *count* esta última cantidad, p_b la probabilidad obtenida a partir de la función presentada previamente, y p_a la probabilidad real empleada por RED, la relación entre estas variables se presenta en la siguiente ecuación:

$$p_a = \frac{p_b}{1 - count \cdot p_b}. \tag{2.1}$$

Algo que no se ha mencionado es el método utilizado para obtener el tamaño promedio de

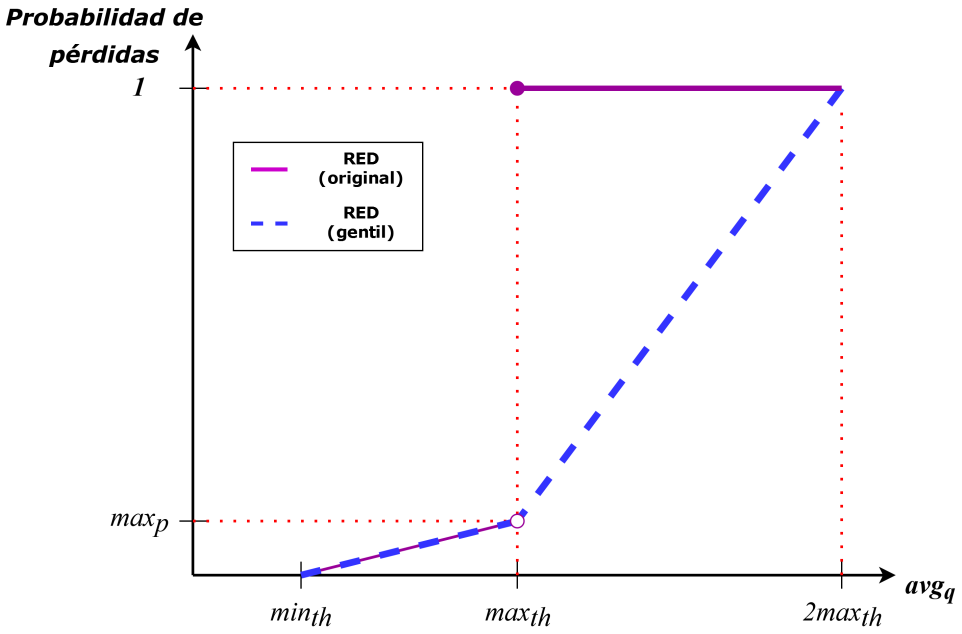


Figura 2.5: Probabilidad de pérdidas utilizada por RED como función del tamaño promedio de la cola, la cual se define por partes tomando como referencia los umbrales min_{th} y max_{th}. Se hace la distinción entre la propuesta original y la versión gentil, pues esto determina el comportamiento una vez que se supera el umbral superior.

la cola avg_q. Este se basa en el uso de un filtro del tipo Exponential Weighed Moving Average (EWMA), y tal como su nombre lo indica se caracteriza por un decrecimiento exponencial del peso de las muestras pasadas en el valor actual del estimado. Sea w_q el peso a utilizar en el filtro y q el valor instantáneo (no filtrado) del tamaño de la cola, entonces el nuevo valor para el estimado se obtiene a partir de la siguiente expresión:

$$avg_q \leftarrow avg_q + w_q \cdot q. \tag{2.2}$$

Dicho lo anterior, resulta útil considerar brevemente lo que respecta a la selección de los valores que deben tomar los parámetros max_p, min_{th}, max_{th} y w_q. A continuación se indican brevemente algunas consideraciones para la configuración de cada uno de ellos:

- $\boldsymbol{max_p}$: tal como se puede apreciar en la Figura 2.5, este parámetro determina la agresividad de RED. Luego, a medida que aumenta su valor, es más difícil que el tamaño promedio de la cola se acerque o supere a max_{th}. Si bien en el trabajo original se utilizaron valores de 0,02 para este parámetro, posteriormente se recomendó un valor de 0,1 en base a las probabilidades de pérdidas medidas posteriormente. Adicionalmente, se argumenta que no tiene sentido utilizar valores más altos, pues si existe tal nivel de congestión entonces es posible que hayan problemas a nivel de ingeniería de tráfico. [74].
- $\boldsymbol{min_{th}}$: dado que el umbral inferior determina el punto a partir del cual se comienzan a descartar paquetes, el valor específico a utilizar fija de cierta forma un nivel de tolerancia frente a tráfico del tipo ráfaga [7], dado un cierto valor de w_q.

No obstante, el aspecto más relevante a considerar para determinar un valor apropiado corresponde al tamaño promedio de la cola que se considera como deseable, y esto depende tanto de la capacidad del canal como del nivel de latencia promedio que se considere aceptable. Dicho esto, el valor recomendado previo a la versión adaptativa del algoritmo es de 5 paquetes, aunque se afirma que es una buena idea explorar valores superiores siempre y cuando esto no signifique tener una latencia adicional comparable a aquella asociada a los procesos de transmisión y propagación de los datos.

- $\boldsymbol{max_{th}}$: al igual que en el caso del umbral inferior, el valor a escoger se relaciona directamente con el tamaño promedio de la cola que se considere como deseable. Adicionalmmente, puesto que la agresividad del algoritmo incrementa de forma considerable una vez que se ha pasado este punto, lo ideal es que exista una distancia prudente entre este umbral y el inferior. Es por esta razón que la recomendación es utilizar el triple de min_{th}.

  Por otra parte, es conveniente tener en cuenta que el valor a utilizar se relaciona al tamaño del buffer necesario o suficiente para implementar el algoritmo de forma correcta. Un aspecto evidente es que se requiere que el tamaño del buffer sea al menos igual a max_{th} en el caso de utilizar la propuesta original, o $2 \cdot max_{th}$ en el caso de implementar la variante gentil. Sin embargo, puesto que existe un retraso asociado a la actualización del promedio con respecto al valor real, utilizar un valor mayor es lo que permitiría que avg_q efectivamente sea capaz de llegar al punto donde la probabilidad de pérdida es unitaria.

- $\boldsymbol{w_q}$: en este caso resulta conveniente considerar que existen dos casos extremos. Si se utiliza un valor muy grande (cercano a 1), entonces el valor estimado se parece mucho al actual, lo cual no permite mucha tolerancia para recibir ráfagas de paquetes. En el caso contrario, si el valor utilizado es muy pequeño, puede darse que la estimación tarde mucho en revelar que se está generando congestión, lo cual podría derivar en la operación de RED como si fuera Drop Tail.

  Luego, una forma de abordar el problema de determinar un valor apropiado es determinando límites. En particular, en [7] se establece un límite superior que permite garantizar cierto nivel de tolerancia frente a ráfagas de paquetes, dado un cierto valor para min_{th}, así como también un límite inferior en base al número de paquetes necesarios para que el valor estimado logre actualizarse desde un cierto nivel a uno nuevo. En base a lo anterior es que se decide utilizar $w_q = 0{,}002$.

Haciendo uso del comportamiento previamente descrito, RED las principales ventajas que logra RED con respecto a los algoritmos desarrollados previamente son las siguientes [7]:

- Evitar el fenómeno de sincronización global entre conexiones que pasan por la misma cola.
- Reducir la tasa de pérdidas de paquetes, y por lo tanto mejorar el throughput.
- Minimizar el sesgo en contra de tráficos del tipo ráfaga.
- Reducir la latencia adicional ocasionada por la espera en la cola.

A modo de ejemplo, se presentan en la Tabla 2.2 resultados obtenidos en [1] para la latencia promedio experimentada por una conexión TCP transmitiendo datos mediante FTP, cuando el ancho de banda del cuello de botella varía entre 0.5 y 1.5 Mbps, en una topología del tipo

Ancho de banda	Latencia - Drop Tail	Latencia - RED
0.5 Mbps	379 ms	67 ms
0.75 Mbps	262 ms	47 ms
1.25 Mbps	161 ms	36 ms

Tabla 2.2: Comparación de la latencia experimentada por una conexión TCP cuando el router del cuello de botella hace uso de Drop Tail y RED respectivamente. Resultados provienen del trabajo realizado en [1], donde se utiliza una simulación de una topología del tipo dumbbell de 2 conexiones.

dumbbell con dos conexiones. En esta, se puede apreciar una clara reducción en la latencia cuando se comparan los valores obtenidos con RED a aquellos alcanzados con Drop Tail. Adicionalmente, se puede notar que las latencias disminuyen a medida que aumenta el ancho de banda, lo cual tiene sentido pues disminuye el tiempo necesario para transmitir cada paquete en la cola.

2.2.4.3. Versiones adaptativas de RED

En el año 1997 los autores de [75] reportan un aumento en la tasa de pérdidas en dicha época, lo cual se atribuyó en parte a un incremento en la demanda, sumada a la incapacidad de la red para notificar oportunamente a las fuentes de tráfico sobre un fenómeno de congestión. Para mitigar esto, la IETF consideró fomentar el uso de algoritmos de AQM tales como RED [8]. Sin embargo, este último no es un algoritmo que carezca de debilidades tal como se demostraría prontamente. En particular, una de ellas se asocia a la incapacidad del algoritmo para lidiar con cambios considerables en el nivel de congestión de la red, lo cual impide que el algoritmo sea capaz de mostrar ser eficaz en múltiples escenarios con una configuración única de sus parámetros [2]. A continuación se describen algunas de las propuestas que abordan este problema de adaptabilidad.

En el año 1999, en [2] se propone un algoritmo adaptativo basado en RED y ECN. La motivación de este artículo yace en uno de los problemas que afecta a RED, la dependencia del rendimiento con respecto a los parámetros del algoritmo. En particular, en este documento se demuestra que una configuración estática de estos no es capaz de obtener un buen rendimiento cuando se enfrenta a distintos escenarios de congestión (donde cambia el número de conexiones activas que pasan por una cola).

En base a esto, se presenta una variante de RED, la cual hace uso de una configuración en línea del parámetro max_p en función del número de conexiones activas, y se utiliza ECN como método de notificación a las fuentes de tráfico. Este algoritmo permite lograr una menor cantidad de pérdidas, manteniendo además una alta utilización del canal.

El algoritmo utilizado para ajustar el valor de max_p se muestra en la Figura 2.6, el cual permite notar que el objetivo es controlar el nivel de avg_q y apuntar a mantenerlo en el rango entre min_{th} y max_{th}, en base a cambios del estilo Multiplicative Increase Multiplicative Decrease (MIMD).

```
Every avg_q update:
    if (min_th < avg_q < max_th)
        status = Between;
    if (avg_q < min_th and status != Below)
        status = Below;
        max_p ← max_p/α;
    if (avg_q > max_th and status != Above)
        status = Above;
        max_p ← max_p * β;
```

Figura 2.6: Pseudo-código del algoritmo adaptativo de RED propuesto por Feng et al. [2]. Se basa en un control del tipo MIMD

Posteriormente, en el año 2001, S.Floyd y otros investigadores presentan una mejora al algoritmo previamente descrito. La principal característica de esta propuesta, al igual que la anterior yace en el hecho de que es capaz de configurar dinámicamente el valor de max_p. Sin embargo, parte del valor de esta propuesta yace también en el hecho de que permite configurar automáticamente el resto de parámetros que caracterizan a RED. No obstante, esta nueva propuesta adaptativa de RED deja a libertad de quien implementa el algoritmo la elección de un nivel deseado para el tamaño promedio de la cola. Este último dependerá del nivel de importancia que se le asigne a la utilización del canal y a la latencia adicional asociada a colas con una cantidad significativa de datos en su buffer.

A continuación se describen brevemente los parámetros utilizados cuando se utiliza el modo automático de esta propuesta [3]:

- $\boldsymbol{max_{th}}$: para el caso del umbral superior, la configuración automática sigue la recomendación previa para RED, utilizando el triple del umbral inferior:

$$max_{th} = 3 \cdot min_{th}. \tag{2.3}$$

- $\boldsymbol{min_{th}}$: en este caso, el valor del umbral inferior se determina a partir de su relación con el tamaño promedio objetivo en la cola. Este está directamente relacionado a la latencia objetivo, la cual se denota como $delay_{target}$. La idea detrás de la configuración automática de min_{th} se basa en fijar el tamaño de la cola promedio en el punto medio entre ambos umbrales. Luego, considerando que el umbral superior es el triple del umbral inferior, se cumple que el punto medio entre los umbrales es equivalente al doble del umbral inferior.

  Sea C la capacidad del canal en paquetes/segundos, y considerando además un valor mínimo para el umbral inferior de 5 paquetes, entonces el valor de este último en paquetes se determina mediante la siguiente ecuación:

$$min_{th} = Max\left[5, \frac{delay_{target} \cdot C}{2}\right]. \tag{2.4}$$

- $\boldsymbol{w_q}$: se determina su valor en base al uso de una constante de tiempo de 1 segundo, asumiendo un RTT igual a 100 ms. Dada la relación logaritmica de la constante de

tiempo, la expresión resultante para la constante del filtro depende de una función exponencial de la capacidad C del canal:

$$w_q = 1 - exp(1/C). \tag{2.5}$$

Otra diferencia con respecto a la propuesta anterior yace en el hecho de que el ajuste se realiza de la misma forma que opera TCP tradicional; mediante AIMD. El pseudo-código específico para esto se presenta en la Figura 2.7, donde $target = [min_{th} + 0{,}4 * (max_{th} - min_{th}), min_{th} + 0{,}6 * (max_{th} - min_{th})]$.

```
Every interval seconds:
    if (avg > target and max_p ≤ 0,5)
        max_p ← max_p + α;
    else if (avg < target and max_p ≥ 0,01)
        max_p ← max_p * β;
```

Figura 2.7: Pseudo-código del algoritmo adaptativo de RED propuesto por Floyd et al. [3]. Se basa en un control del tipo AIMD.

La capacidad adaptativa de max_p permite en este caso tener una función de pérdidas que cambia en el tiempo, pues el valor de este parámetro es el que controla el trazado de ambas rectas, cuando se hace uso del modo gentil (ver Figura 2.8). Luego, max_p se comporta como un regulador dinámico de la agresividad del algoritmo en base al incremento o decrecimiento de las probabilidades de pérdidas. Además, al igual que en el caso de RED, las probabilidades de pérdidas reales siguen cumpliendo la relación expresada en la Ecuación 2.1.

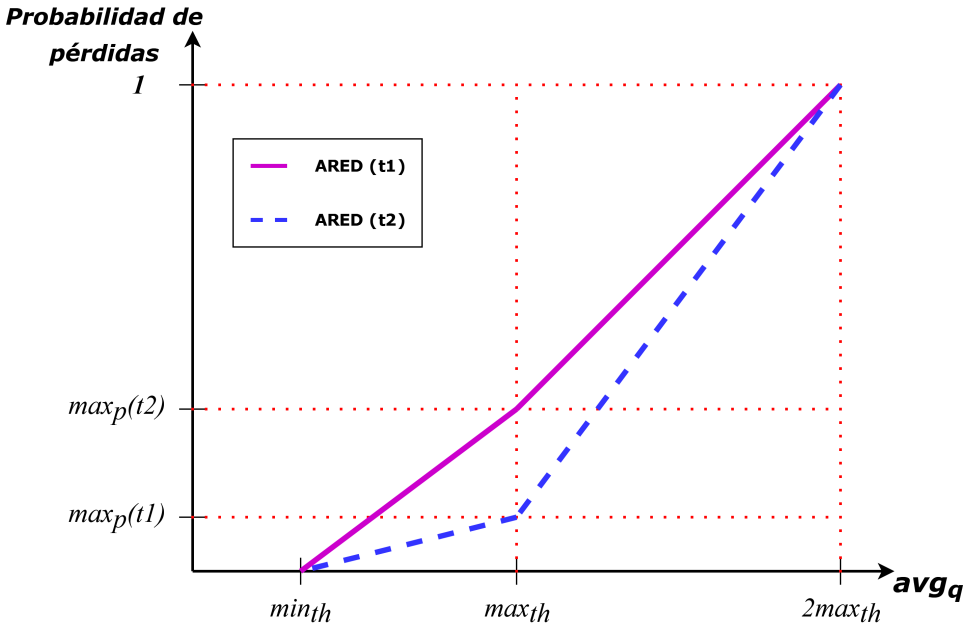


Figura 2.8: Función de probabilidades de pérdida para el algoritmo ARED. En este caso, la función cambia dinámicamente en el tiempo en base a la evolución de max_p. Esta evolución depende de la magnitud del nivel de congestión, lo que se refleja en el valor de avg_q, el cual es monitoreado periódicamente con el fin de aplicar los ajustes necesarios.

Finalmente se desea destacar que desde este punto en adelante se denominará como ARED a la propuesta de [3], mientras que el trabajo de [2] se denominará Adaptive RED (misma convención utilizada en ns-3 [76]).

2.2.4.4. Otros algoritmos de AQM

Además de las variantes adaptativas de RED, a través del tiempo este algoritmo ha motivado el diseño de múltiples variantes de todo tipo. Una de las propiedades deseables en un algoritmo de AQM corresponde a la estabilidad en torno a un cierto punto o rango de operación, independiente del número de conexiones que pasan por la cola. Esto es lo que buscan algoritmos como Stabilized RED (SRED)[77] y Dynamic RED (DRED) [78], donde el primero de estos propone la estimación del número de conexiones sin guardar variables de estado para cada conexión, mientras que el segundo se enfoca en el uso de teoría de control de sistemas para lograr este objetivo en base a control integral. Otro algoritmo que destaca por su simpleza es Nonlinear RED (NLRED) [13], el cual propone un cambio de la función de probabilidad de pérdidas de una lineal a una cuadrática, con el fin de lograr mayor gentileza cuando la congestión es baja y más agresividad en caso contrario, apuntando a operar en un rango deseable para el tamaño promedio de la cola.

También existen trabajos con motivaciones más específicas, tales como el diseño de un marco de trabajo que apunta a proveer un nivel de throughput en torno a un cierto valor objetivo en presencia de congestión, lo cual se aborda mediante RED with In/Out bit (RIO) [11], un algoritmo que se basa en una etiqueta (de 1 bit) para determinar la prioridad de cada paquete. Otra propuesta de este estilo es denominada Flow RED (FRED) [10], la cual se centra en ajustar la operación de RED en base al tipo de tráfico, el cual se caracteriza en función de la cantidad de espacio que ocupa en la cola. Un caso mucho más particular es el de Robust RED (RRED) [12], algoritmo que se enfoca en aumentar la robustez frente a ataques de denegación de servicio.

Si bien la literatura está en gran parte dominada por variantes de RED, también existen otros algoritmos que se utilizan otras ideas que son independientes o bien se desvían significativamente de aquello propuesto por RED (en especial en tiempos recientes). Uno de estos algoritmos es Random Early Marking (REM) [79], cuya principal característica corresponde a la separación de métricas asociadas a congestión y aquellas que permiten evaluar el rendimiento. En particular, se mide la congestión en base a la tasa que ingresan el total de datos a la cola, mientras que el tamaño de la cola se utiliza para evaluar qué tan lejos se encuentra del valor objetivo. Otro algoritmo cuyo acercamiento al problema es similar corresponde a BLUE [34], ya que tampoco opta por utilizar el tamaño promedio de la cola como una métrica de congestión, sino que en cambio utiliza la tasa de pérdida de paquetes y la utilización del enlace. Este algoritmo tiene algunas variantes como Stochastic Fair BLUE (SFB), que apunta a proteger a conexiones TCP del tráfico que no responde frente a eventos de congestión, así como también se desarrolló Resilient Stochastic Fair Blue (RSFB) [80], otra variante del algoritmo cuyo objetivo es mitigar el efecto de ataques de denegación de servicio distribuídos.

Otra familia de algoritmos que ha resultado fructífera es aquella que se origina en la

propuesta denominada CHOose and Keep/Kill (CHOKe) [81], diseñado para poder brindar un trato justo mediante una operación sencilla, lidiando de forma efectiva con el tráfico que no responde frente a congestión. Esto se logra mediante una leve modificación a RED, incorporando una etapa adicional previa al cálculo de la probabilidad de pérdidas que realiza RED. En particular, lo que se hace es verificar si el paquete que está entrando a la cola pertenece a la misma conexión que uno seleccionado de forma aleatoria de la misma. Si esto último es cierto, entonces se descartan ambos paquetes, y en caso contrario se prosigue con el funcionamiento tradicional de RED (descartar o insertar paquete en cola en base a probabilidad de pérdidas). Esto es generalizado en geometric CHOKe (gCHOKe) [82], donde se permite realizar una comparación adicional cada vez que los paquetes pertenecen a una misma conexión, limitando la máxima cantidad de veces que se puede realizar este proceso mediante el parámetro *maxcomp* (de tal forma que CHOKe corresponde a gCHOKe con $maxcomp = 1$). Una variante similar es CHOKeHold on unfairness (CHOKeH) [35], donde en vez de realizar múltiples comparaciones secuenciales se opta por realizar una comparación entre un número dinámico de paquetes seleccionados de la cola.

Por otra parte, el interés por el desarrollo de algoritmos que utilicen teoría de control de sistemas ha derivado en propuestas que incorporan tanto la componente proporcional como la integral [83]. Una propuesta reciente que se basa en lo anterior es Proportional Integral controller Enhanced (PIE) [84, 85], un algoritmo que adicionalmente es capaz de configurar sus parámetros de forma automática, lo cual facilita su adopción y la capacidad para operar en una gran variedad de escenarios, una propiedad altamente valorable en esta disciplina. Una variante de este algoritmo es requerida para la subida de datos en Data Over Cable Service Interface Specification (DOCSIS) 3.1, la cual se ha modificado específicamente para ajustarse a este tipo de tráfico y tecnología [86].

En la tabla 2.3 se presentan algunas de las principales propuestas de AQM en el último tiempo. A pesar de que no existen pruebas de un despliegue masivo de AQM, es posible que en el futuro cercano se generen avances significativos, pues se ha formado una comunidad en torno al problema denominado como bufferbloat [87] [88], el cual hace referencia a la acumulación de latencia en routers que componen Internet. Otro factor que podría generar un efecto similar es la reciente aparición de RFCs sobre PIE y Controlled Delay (CoDel) [89, 90]. Este último es un algoritmo que utiliza como métrica de congestión el tiempo que se quedan en la cola los paquetes. En particular, se toma en cuenta el mínimo medido sobre un cierto intervalo de tiempo, y las pérdidas se generan cuando se supera un valor predefinido para esta métrica. Luego de esto, el tiempo que pasa hasta la siguiente pérdida se determina en función de la cantidad de pérdidas que han ocurrido desde que se entró al estado congestionado. Este mecanismo puede ser combinado con un sistemas de múltiples colas, lo cual es conveniente para separar distintos tipos de tráfico [91].

Tabla 2.3: Propuestas de AQM a través del tiempo

1989	Random Drop & Early Random Drop [68].
1993	RED [7].
1997	FRED [10].
1998	RIO [11].
1999	Adaptive Red [2], BLUE/SFB [34], SRED [77].
2000	CHOKe [81].
2001	ARED [3], REM [79], PI [83], DRED [78].
2006	NLRED [13].
2009	RSFB [80].
2010	RRED [12].
2012	CoDeL [89].
2013	PIE [84], gCHOKe [82].
2017	RFC8033 (PIE) [85], RFC8034 (DOCSIS-PIE) [86].
2018	RFC8289 (CoDel) [90], RFC8290 (FQ-Codel) [91], CHOKeH [35].

2.3. El Problema del Fairness

En general la idea de justicia surge de manera natural en cualquier problema de distribución de recursos. Este no está ligado específicamente a las telecomunicaciones, y por lo tanto algunas de las métricas o criterios que serán mencionados prontamente pueden resultar válidos en el contexto de otras disciplinas.

En el caso de las telecomunicaciones, no resulta trivial definir un objetivo deseable a nivel de fairness, razón por la cual sigue siendo un tema de investigación en tiempos recientes

[92]. A primera vista pareciera sencillo el definir de forma ingenua el procurar igualar los throughputs de todas las conexiones que hacen uso de una cierta red. Sin embargo, esto puede tener un precio a nivel de utilización de recursos [93], como sucede en el caso en que se tienen múltiples enlaces congestionados. Lo que sucede en estos casos, es que aquellas conexiones que cruzan múltiples enlaces congestionados, en estricto rigor hacen uso de más recursos de la red, por el hecho de que el flujo de paquetes se replica en cada salto a través de los routers. De hecho, este es uno de los argumentos que puede ser utilizado en contra de la distribución equitativa de la capacidad de un cuello de botella cuando existen conexiones de alta latencia compitiendo en este [20, 15].

Existen múltiples casos donde el término fairness hace referencia a contextos significativamente distintos, lo cual requiere que el lector de artículos de la literatura tenga precaución al momento de enfrentarse a un análisis de fairness. Posiblemente el contexto más común esté asociado a la medición del fairness como una métrica que permita determinar si dos conexiones de la misma naturaleza (homogéneas) son capaces de repartirse de forma equitativa la capacidad del canal donde compiten. Un segundo caso un tanto más especial hace referencia al fairness que se obtiene cuando compite una cierta variante de TCP con TCP tradicional, en cuyo caso se suele utilizar un término más específico denominado TCP friendliness. El último caso corresponde a aquel donde lo que se analiza es la capacidad de un protocolo o esquema para lograr throughputs relativamente similares cuando se tienen distintos niveles de latencia (caracterizada por el RTT), en cuyo caso se suele utilizar el término RTT fairness.

2.3.1. Criterios y Métricas

Una de las propuestas con mayor aceptación (y que se sigue utilizando en tiempos recientes [30]) corresponde al **Fairness Index (FI)** propuesto por R.Jain [94], el cual presenta características atractivas tales como: continuidad, independencia con respecto a la escala, y el hecho de estar acotado entre los valores 0 y 1 (razón por la cual a veces se interpreta en forma de porcentajes). En estricto rigor el **FI** se puede utilizar en base a los throughputs normalizados [95], lo cual permite añadir en el análisis criterios como el de **Max-Min Fairness** [96]. En términos matemáticos este índice depende de la razón entre la desviación estandar y el promedio de los throughputs alcanzados por el sistema considerado, En particular, es posible relacionarlo al **coeficiente de variación**, el cual también puede ser utilizado como una métrica de fairness [97]. Sea COV el coeficiente de variación, σ la desviación estándar, μ el promedio de los throughputs normalizados T_i (asociado a la conexión i), y N el número de conexiones, entonces se cumple lo siguiente:

$$Fairness\ Index = \frac{\left(\sum_i T_i\right)^2}{N\sum_i T_i^2} = \frac{1}{1+COV}, \tag{2.6}$$

$$COV = \frac{\sigma}{\mu}. \tag{2.7}$$

Otra alternativa de métrica cuantitativa es el **Product Measure**, el cual, tal como su nombre lo indica, hace uso de la multiplicación directa de todos los throughputs [98], tal como puede notarse en la Ecuación 2.8. Esta métrica tiene la particularidad de ser bastante sensible cuando hay escenarios donde hay conexiones que alcanzan throughputs significativamente bajos respecto al resto. Adicionalmente, es posible demostrar que en el caso donde hay competencia en un único router congestionado, entonces esta se maximiza cuando todos los throughputs son iguales.

$$Product\ Measure = \prod_i T_i. \tag{2.8}$$

En cuanto a criterios, uno que ya fue mencionado es el de **Max-Min Fairness** [99], el cual se basa en la idea de que se busca maximizar el mínimo throughput alcanzable por cada una de la conexiones. Es decir, se busca que el throughput más pequeño sea lo más grande posible, y que el siguiente más pequeño cumpla el mismo criterio, y así sucesivamente. Dicho de otra forma, no es posible aumentar un throughput sin que esto implique una reducción de otro que sea menor al que se está aumentando. Este criterio es relevante en topologías donde existen multiples cuellos de botella, y la forma de encontrar la distribución que cumple el criterio es aumentar gradualmente todos los throughputs hasta el punto donde uno de los cuellos resulta limitante. Luego, se fijan los throughputs de las conexiones que pasan por el cuello de botella y se continúa el proceso con el resto de las conexiones.

Otro criterio corresponde a **Proportional Fairness** [100], el que se basa en la idea de que se logra este tipo de justicia si es que no existe otro set de throughputs tal que la suma de los cambios proporcionales sea mayor a cero. Sea T_i^* el throughput asignado a la conexión i y que es parte de la distribución de recursos que se desea evaluar mediante el criterio, y T_i el throughput derivado de cualquier otra distribución, entonces la desigualdad que permite validar el criterio tiene la siguiente forma:

$$\sum_i (T_i^* - T_i)/T_i \geq 0. \tag{2.9}$$

Los criterios y métricas son algunas de las alternativas existentes en la literatura, pero no existe una solución utilizada universalmente, razón por la cual sigue siendo un tema de investigación en la actualidad [101, 102].

2.3.2. RTT Fairness

El problema del RTT fairness es el tema en torno al cual gira el presente trabajo, y es uno que ha demostrado persistir desde el inicio del control de congestión como disciplina. El origen inicial de este yace en el modo de operación de TCP tradicional, cuyas bases fueron sentadas por el trabajo de Jacobson [27] luego de los eventos de congestión del fin de los ochenta. Este trabajo plantea algoritmos que han demostrado ser increíblemente robustos frente a congestión, y se basan en un principio de conservación de paquetes. En base a este

es que se plantea el uso de la ventana de congestión y de un incremento moderado de la misma para lograr adaptarse a los cambios en la capacidad disponible (por ejemplo, debido al término de una determinada congestión). En particular, el factor clave está asociado al hecho de que el incremento de las ventanas es independiente de la latencia de las conexiones, mientras que el throughput alcanzado por las mismas depende de la razón inversa del RTT [103, 31].

No obstante, no es tan solo TCP tradicional el que exhibe este problema, sino que múltiples protocolos diseñados para enlaces de alta velocidad también presentan un sesgo en contra de las conexiones de alta latencia [33]. En particular, TCP CUBIC, a pesar de tener la capacidad para emplear ventanas de distinto tamaño en función del RTT de la conexión, no es capaz de eliminar completamente este problema [15]. Lo anterior ha motivado el nacimiento de propuestas que apuntan a mejorar la capacidad de CUBIC para lidiar con este problema [30].

Adicionalmente, existe un conflicto entre la búsqueda por mantener un buen nivel de TCP friendliness y el RTT fairness. El problema en este caso está asociado al hecho de que muchos de los protocolos más recientes operan de tal forma que su comportamiento simula la agresividad de TCP tradicional cuando hay un nivel significativo de congestión (lo que se traduce en un alto nivel de pérdidas). Esto dificulta significativamente la posibilidad de lidiar con el problema a nivel de capa de transporte en estos casos, lo cual motiva fuertemente la idea de hacer uso de AQM para abordarlo.

Ahora bien, existen múltiples trabajos que avalan el hecho de que RED es capaz de mejorar los resultados obtenidos por Drop Tail, principalmente con throughputs que se acercan más a una relación lineal con respecto a la razón inversa de los RTTs [51, 65, 104]. No obstante, el efecto es limitado y el sesgo persiste en contra de las conexiones de alta latencia.

2.4. Propuestas que abordan el problema del RTT Fairness

A través del tiempo se han generado diversas propuestas que abordan el problema del RTT fairness hasta cierto punto. En esta sección se mencionarán algunos de los trabajos que lo abordan a nivel de TCP y AQM.

En cuanto a las propuestas a nivel de capa de transporte, [33] evalúa el rendimiento de múltiples protocolos en cuanto a distintos aspectos. En particular, se encuentra que H-TCP [23] es el único que demuestra ser más justo que TCP tradicional, lo cual tiene sentido pues por diseño se espera que logre atenuar esta injusticia. Por otro lado, BIC, STCP, HSTCP y FAST TCP obtienen peores resultados que TCP tradicional. En el caso de BIC, el resultado no es sorprendente, pues si bien esta propuesta le brinda una gran importancia al problema del RTT fairness, los resultados del artículo original también demuestran que si bien tiene mejores resultados en comparación protocolos como HSTCP y STCP, de todas formas no supera la justicia obtenida por TCP tradicional [65]. Los resultados de FAST sorprenden un poco más debido a que por diseño también debería ser capaz de mitigar el problema, sin

embargo se ha mostrado que su rendimiento puede ser limitado en función de la topología o si sus parámetros no son los apropiados [105].

Un protocolo diseñado especialmente para abordar el problema del RTT fairness es TCP Libra [51], cuyo control de la ventana de congestión permite teóricamente a un throughput que garantiza la distribución justa de los recursos disponibles. En particular, en [28] se muestra que logra un fairness similar a aquel obtenido por TCP Hybla [106], pero a diferencia de este no muestra el comportamiento agresivo en contra de conexiones que utilizan TCP tradicional. Tanto Hybla como Libra muestran superar significativamente a Vegas, CUBIC y TCP tradicional.

En el caso de CUBIC, el RTT fairness obtenido es mejor que el de BIC, y su buen rendimiento se basa en la idea de presentar funciones de respuesta distintas dependiendo del RTT. Teóricamente esto permite que, con el mismo nivel de pérdidas, se tengan ventanas más grandes cuando se experimenta latencia alta y viceversa. Sin embargo, los resultados obtenidos por los autores del algoritmo muestran que el sesgo sigue existiendo [15]. Esto ha motivado el trabajo de [30], donde se analiza el origen de la injusticia y se propone un ajuste en función del RTT para el parámetro K utilizado por CUBIC para ajustar su ventana.

En el contexto de data-centers, se ha propuesto DCTCP como una alternativa diseñada específicamente para lidiar en este tipo de ambientes. No obstante. dado que la versión original de este protocolo mantiene el sesgo en contra de conexiones de alta latencia, en [104] se muestra que es posible lograr una mejora en base a una leve modificación a la respuesta frente a paquetes que indican congestión. En particular, se propone una disminución paulatina por cada ACK, en vez de hacer una disminución instantánea una vez por ventana.

Una de las variantes de TCP más recientes y relevantes corresponde a TCP BBR [107], un protocolo desarrollado por Google y que se encuentra disponible en el kernel de Linux. Lo anterior, sumado al buen desempeño dentro de la infraestructura privada de Google [56] lo posiciona como un buen candidato para competir con CUBIC, el protocolo dominante en la actualidad. No obstante, se ha mostrado que el rendimiento a nivel de RTT fairness cuando se hace uso de BBR puede deteriorarse de forma significativa dependiendo del tamaño de los buffers y la capacidad del enlace [24].

Respecto a AQM, el algoritmo denominado Flow Random Early Drop (FRED) [10] demuestra lograr ajustar de forma drástica el throughput alcanzado por una conexión de latencia alta que compite con múltiples conexiones de baja latencia. Sin embargo, esta propuesta tiene una desventaja asociada a requerir el uso de contadores para cada conexión. A pesar de esto, este problema no resulta tan grave puesto que su diseño permite que el costo de esto no dependa del número de conexiones, sino que solo del tamaño del buffer.

Otro algoritmo que aborda el problema parcialmente es RIO [11], el cual hace uso de etiquetado de paquetes en función de un umbral de transferencia de datos, con el fin de proveer servicios diferenciados. En particular, se discrimina en contra de los paquetes que se asocian a la superación de los umbrales, empleando un algoritmo RED para cada tipo de paquetes (diferenciados por la etiqueta). Los resultados obtenidos por el algoritmo demuestran la capacidad para aliviar el problema del RTT fairness, la cual es significativamente mejor cuando se mueve el mecanismo de etiquetado a la capa de transporte del usuario.

Por otra parte, los autores de [34] muestran que Stochastic Fair Blue (SFB) obtiene un fairness muy similar al obtenido por RED cuando compiten conexiones con una gran variedad de RTTs, lo cual mantiene el sesgo a favor de las conexiones de baja latencia. Solo en el caso en que existe un número limitado de conexiones con baja latencia es que el algoritmo es capaz de aliviar un poco este problema.

Una propuesta más reciente corresponde a CHOKeH [35], cuya modificación al algoritmo original logra mitigar la ventaja de las conexiones de RTT alto, y de tal forma nivela de mejor forma los throughputs alcanzados por conexiones de distinta latencia, tanto en simulaciones donde todas las conexiones son TCP, como en otras donde se añaden flujos que no responden frente a eventos de congestión.

2.5. Importancia del desarrollo del control de congestión del futuro

El control de congestión es un tema delicado, pues si no se desarrolla con seriedad y responsabilidad se podrían promover prácticas que eventualmente lleven la red a un colapso por congestión, tal como ocurrió en el pasado.

Uno de los factores fundamentales de la posibilidad de dicha ocurrencia, es que la justicia entre flujos no es algo que le preocupe al usuario promedio, el cual podría considerarse egoísta por naturaleza. Este egoísmo nace a partir de intereses personales, muchas veces económicos y es peligroso dejar a disposición del usuario la posibilidad de utilizar un nuevo tipo de algoritmo que le provea de tasas de transmisión mejores que las anteriores, ya que esto puede deberse a la agresividad del algoritmo y no a un mejor uso de los recursos de la red.

Una mayor agresividad se puede lograr a partir de la modificación del algoritmo de control de congestión de TCP, así como también haciendo uso de múltiples conexiones, como lo que hacen algunos navegadores web. Estos últimos suelen tener la capacidad de utilizar múltiples conexiones simultáneas con un servidor común, y a pesar de que se recomienda limitar este número a 2 conexiones [108], la mayoría de los navegadores supera este número considerablemente (desde 6 hacia arriba). Estos problemas son conocidos desde hace 2 décadas [8].

El tema es incluso más delicado si es que eventualmente un algoritmo gana popularidad en base a un buen rendimiento inicial, el cual podría verse perjudicado si eventualmente el despliegue se vuelve masivo. Luego, es fundamental que las propuestas sean validadas en base al conocimiento exhaustivo del estado del arte, el nivel de despliegue de las propuestas, el rendimiento de la red considerando lo anterior y las causas de los problemas que podría tener este último.

Una idea que resulta muy atractiva consiste en comenzar a promover el diseño de algoritmos de TCP y AQM en conjunto, logrando una mayor coherencia en cuanto a las consideraciones que se deben tomar en cuenta en ambos puntos para lograr un cierto objetivo. No obstante, esto no debe dejar de lado la idea de que la implementación de estas soluciones

deben soportar un despliegue gradual, en base a lo cual no puede existir una dependencia total entre ambos algoritmos propuestos.

Finalmente, se desea destacar la idea de que no se debe permitir que la complejidad del problema ahuyente a posibles investigadores. Al contrario, esto debería evitarse en base a lo atractivo que sería lograr seguir aumentando las tasas de transmisión y reducir el nivel de latencia que experimentan las conexiones, siendo esto último fundamental cuando se considera la cantidad de aplicaciones que se verían beneficiadas de una latencia pequeña.

Capítulo 3

Descripción de la Propuesta

3.1. El problema del RTT fairness utilizando TCP tradicional

Un problema que afecta a algoritmos como RED, es el hecho de que estos no logran ser justos en cuanto al throughput que obtienen conexiones con distintos valores del RTT [10, 11]. Esto tiene sentido, pues nunca fue uno de los objetivos del algoritmo proveer esta funcionalidad [7], sin embargo es importante comprender el por qué se genera esta injusticia.

Para ello, es fundamental conocer dos factores fundamentales sobre la dinámica del estado estacionario de TCP tradicional:

- En primer lugar, es posible modelar el throughput promedio que alcanza una conexión que utiliza TCP tradicional en función de su tasa de pérdidas p, lo cual se denotará como $B(p)$. Esta derivación se realiza en [31] y la función obtenida para el throughput en paquetes por unidad de tiempo se presenta en la Ecuación 3.1,

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min(1, 3\sqrt{\frac{3b}{8}}p(1+32p^2))}, \tag{3.1}$$

  donde se puede apreciar una dependencia con respecto a p, el RTT de la conexión y las constante T_0 y b. La primera de estas constantes corresponde al tiempo que transcurre entre el envío de un paquete y su retransmisión por timeout, mientras que la última representa la cantidad de paquetes que son reconocidos por cada ACK. Se debe notar además que las unidades de RTT y T_0 deben ser idénticas, y se debe recordar que las unidades de $B(p)$ dependen de lo anterior.

  En el caso de ocupar ACKs acumulativos b puede tomar valores mayores a 1, pero durante este trabajo no se hace uso de este mecanismo. También es posible interpretar el valor de b como la cantidad de ventanas completas que deben ser reconocidas para aumentar la ventana de congestión en un paquete.
- Ignorando el factor que considera los timeouts, la expresión anterior se convierte en la

Ecuación 3.2, donde es posible identificar aquello que representa la ventana en función de la tasa de pérdidas $W(p)$ y el RTT, considerando que el throughput $Th(RTT, p)$ se obtiene de la división de ambas cantidades. Esto último tiene su origen en el comportamiento de TCP, el cual solo envía una ventana por RTT, pues hace uso de los reconocimientos como una herramienta para mantener el principio de conservación de paquetes descrito V. Jacobson en [27].

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + o(1/\sqrt{p})} \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}}} = Th(RTT, p) = \frac{W(p)}{RTT}. \tag{3.2}$$

Este último resultado del modelo de Padhye et al. también se ha derivado y evaluado en otros [103, 109], razón por la cual se toma como referencia para el diseño del algoritmo. En particular, si se considera este modelo y se asume que dos conexiones experimentan la misma tasa de pérdidas p, entonces la razón entre el throughputs de ambas conexiones depende únicamente de la razón inversa de los RTTs. En efecto, sea Th_i el throughput de la conexión i, y RTT_i el RTT respectivo, entonces lo anterior se puede comprobar en la Ecuación 3.3:

$$\frac{Th_1}{Th_2} = \frac{Th(RTT_1, p)}{Th(RTT_2, p)} \approx \frac{\frac{W(p)}{RTT_1}}{\frac{W(p)}{RTT_2}} = \frac{RTT_2}{RTT_1}. \tag{3.3}$$

3.2. Solución propuesta

Dicho lo anterior, el problema yace en que un algoritmo como RED utiliza una función de probabilidad que depende principalmente del tamaño promedio de las colas y esta no discrimina entre una conexión u otra. En consecuencia, los throughputs a los cuales convergen las conexiones con distinta latencia serán inherentemente distintos debido a la relación presentada en la Ecuación 3.2.

En estricto rigor no es completamente correcto asumir que las conexiones experimentan una probabilidad de pérdidas constante, como se asume en dicha ecuación, pues una cola es un sistema dinámico, con un tamaño variante en el tiempo, lo cual implica que la tasa de pérdidas cambia constantemente. Sin embargo, y tal como se verá posteriormente, es una aproximación lo suficientemente buena como para ser una referencia.

Considerando que la injusticia teórica se basa en el trato equitativo de las conexiones, nace la motivación para el diseño de un nuevo algoritmo de AQM, el cual utilice los datos de latencia para lograr una mayor justicia entre flujos con distintos niveles de latencia. En particular, se desea modificar las pérdidas de forma apropiada para que se logren igualar los throughputs. Por lo tanto, igualando la expresión teórica de los throughputs en base a lo presentado en la Ecuación 3.2 de dos conexiones se obtiene lo siguiente:

$$Th_1 = Th(p_1, RTT_1) = \frac{1}{RTT_1\sqrt{\frac{2bp_1}{3}}} = \frac{1}{RTT_2\sqrt{\frac{2bp_2}{3}}} = Th(p_2, RTT_2) = Th_2, \tag{3.4}$$

$$\iff \frac{p_1}{p_2} = \Big(\frac{RTT_2}{RTT_1}\Big)^2. \tag{3.5}$$

Luego, para que exista igualdad se necesita una proporcionalidad inversa entre las probabilidades y el cuadrado de los RTTs:

$$p_1 RTT_1^2 = p_2 RTT_2^2 = k \Rightarrow p_j = \frac{k}{RTT_j^2}. \tag{3.6}$$

Esto nos indica que mientras las probabilidades mantengan una forma como la que se presenta en la expresión anterior, entonces teóricamente la igualdad de los throughputs se tendrá para cualquier par de conexiones, logrando una distribución equitativa del ancho de banda. Por lo tanto, k se puede entender como un nivel de probabilidad base a partir del cual se obtendrán las probabilidades reales, en función del valor del RTT de una conexión j.

Resulta útil modificar un poco la fórmula anterior para facilitar la comprensión del efecto que se tendrá sobre un término común compartido por todas las conexiones, en función del RTT de la conexión. Así, se plantea como forma alternativa aquella presentada en la Ecuación 3.7 para una conexión j. En esta, $\overline{RTT}$ corresponde al RTT promedio de las conexiones que pasan por la cola de interés y p_{ref} corresponde a la probabilidad que experimentará una conexión con un RTT igual a $\overline{RTT}$. En otras palabras, se define el término común como aquella probabilidad asociada a una conexión cuyo RTT se ubica en el promedio, en base a lo cual se está utilizando a este como una referencia.

$$p_j = \Big(\frac{RTT_{ref}}{RTT_j}\Big)^2 p_{ref} = \Big(\frac{\overline{RTT}}{RTT_j}\Big)^2 p_{ref}. \tag{3.7}$$

En consecuencia, la probabilidad de pérdidas efectiva será el resultado de la multiplicación entre la probabilidad de pérdidas de referencia (p_{ref}) y un factor que depende de la razón entre el RTT de referencia (RTT_{ref}) y el de la conexión. En particular, las conexiones con un RTT por debajo de RTT_{ref} experimentarán probabilidades de pérdidas más altas con el fin de compensar su mayor agresividad (en el sentido que incrementan su ventana más rápido debido al incremento aditivo de un paquete por RTT durante Congestion Avoidance), y se verá el efecto contrario cuando el RTT esté por encima de RTT_{ref}.

Con el fin de validar que una probabilidad de pérdidas de esta forma logra apuntar a un objetivo común para todas las conexiones, a continuación se hace el ejercicio de reemplazar la expresión anterior en la fórmula derivada por Padhye [31]:

$$Th(RTT_j, p_j) = \frac{1}{RTT_j\sqrt{\frac{2b\left(\frac{\overline{RTT}}{RTT_j}\right)^2 p_{ref}}{3}}} = \frac{1}{\overline{RTT}\sqrt{\frac{2bp_{ref}}{3}}} = Th(\overline{RTT}, p_{ref}). \tag{3.8}$$

Por supuesto, el valor de este throughput objetivo no es algo que sea calculable, pues depende del valor de p_{ref}, el cual dependerá de la congestión experimentada en un cierto instante de tiempo. De hecho, lo que debería suceder idealmente en la práctica es que los throughputs logren igualarse gracias a esta propuesta, donde el valor específico que alcanzarán dependerá de la utilización del canal que logra el algoritmo utilizado, lo cual nunca será exactamente igual al máximo, el cual tiene la forma C/n, donde C es la capacidad del canal y n el número de conexiones repartiéndoselo de forma equitativa.

Es importante aclarar que, si bien previamente se mencionó la posibilidad de utilizar el promedio de los RTTs, esto no es necesario en la práctica, pues las probabilidades de pérdidas de referencia deberían ser dinámicas. Esto quiere decir que si se considerara por ejemplo un $RTT_{ref} = \frac{\overline{RTT}}{10}$, entonces lo que se puede hacer es considerar ese factor 10 al cuadrado como un modificador de la agresividad original del otro factor de la probabilidad ($p'_{ref} = \frac{p_{ref}}{100}$).

Por lo tanto puede resultar conveniente añadir a la definición un factor adicional, el cual se denota mediante la constante λ y que tiene como fin permitir hacer ajustes de agresividad que se estimen convenientes, lo cual se presenta en la Ecuación 3.9.

$$p_j = \left(\frac{RTT_{ref}}{RTT_j}\right)^2 \frac{p_{ref}}{\lambda}. \tag{3.9}$$

No obstante, es importante notar que este factor no debería tener mucha influencia en algoritmos adaptativos como ARED. Probablemente lo que se esperaría en estos casos es que el tiempo de convergencia se modifique, pero el punto de convergencia no debería moverse de forma significativa.

3.2.1. Definiciones de p_{ref}, RTT_{ref} y λ en el presente trabajo

En el contexto de esta investigación, en primer lugar se decide definir $RTT_{ref} = 0{,}1$ [s], ya que representa un nivel de latencia razonable [110], lo cual permitirá hacerse una idea rápida del valor del factor cuadrático en base al RTT de una conexión j. Por otra parte, se decide utilizar un valor unitario para λ, puesto que de esta forma la agresividad del algoritmo de referencia no es afectada, y de tal forma, la probabilidad de pérdidas experimentada por una una conexión cuyo RTT es igual a RTT_{ref} sería la misma tanto para el caso donde se utiliza la propuesta, como en aquel donde se utiliza directamente el algoritmo de referencia. Dicho de otra forma, esta conexión no se vería afectada por el uso de la propuesta, siempre y cuando las condiciones de congestión sean las mismas.

Finalmente, y si bien el análisis teórico indica que existe libertad para el valor de p_{ref}, se opta por no invertir tiempo en definir nuevos métodos complejos para realizar AQM.

Al contrario, en el contexto del presente estudio se decide hacer uso de las probabilidades que caracterizan a RED (y su forma adaptiva; ARED), ya que posee un comportamiento sencillo y existen múltiples trabajos que han estudiado su rendimiento [1, 10]. No obstante, teóricamente existe la libertad de utilizar otros algoritmmos de AQM como referencia.

Dicho esto, en este trabajo se tomarán como referencia los algoritmos RED y ARED, cuyas probabilidades de pérdida reemplazarían a p_{ref} en la Ecuación 3.9. Luego, tomando esto en consideración y la elección de los parámetros RTT_{ref} y λ, la forma de las probabilidades de pérdidas experimentadas por una conexión con RTT_j queda dada por la siguiente expresión:

$$p_j = \left(\frac{0{,}1}{RTT_j}\right)^2 p_{AQM} \quad , \quad AQM = \{RED, ARED\}. \tag{3.10}$$

Dado que la derivación de la expresión anterior depende fuertemente del RTT de las conexiones, y que la motivación subyacente corresponde a la justicia, la propuesta se denomina RTT-Based-Fair Active Queue Management (RBF-AQM). Esta solución se caracteriza por ser del tipo capa cruzada, pues es posible considerar la operación de AQM como un control de enlace lógico en la capa de acceso de los routers, el cual en el caso de la propuesta se realiza en base a la información del RTT que es estimado por TCP en la capa de transporte. Lo anterior requiere que exista una forma de entregar este dato a los routers, lo cual se puede efectuar mediante el uso de un campo en la cabecera de IP. Esto se explica en más detalles en la siguiente sección, donde se habla sobre aquello que se requiere para la implementación de la propuesta.

3.2.2. Alternativas para la implementación de la propuesta

El algoritmo propuesto requiere el uso del RTT para equiparar los throughputs de conexiones con distinta latencia. En la actualidad, este no es un dato que se encuentra disponible en los routers, sino que es algo estimado internamente por TCP para el manejo apropiado de los timeouts [111].

La primera alternativa considerada es aquella emulada en las simulaciones de este trabajo, y se basa en el uso directo del valor estimado por TCP, transmitiendo este dato en la cabecera del paquete IP como una opción, lo cual ciertamente requiere una inclusión al estándar de IP antes de que pueda ser utilizado en la práctica. Esto es muy similar a lo que se propone para el Datagram Congestion Control Protocol (DCCP) [112], con la diferencia de que en este caso la opción es de capa IP y no es necesario una granularidad significativamente alta. Por ejemplo, se estima que para esta propuesta el uso de 2 bytes son suficientes, puesto que permiten abordar un rango de latencia entre 0 y 3 segundos con una granularidad de un décimo de milisegundo.

Un problema con lo anterior es que existe la posibilidad de que un usuario modifique de forma maliciosa el campo para abusar del algoritmo empleado en el router, con el fin de mejorar su tasa de transmisión. Sin embargo, esto es similar al caso en que un usuario modifique TCP para actuar de forma agresiva con el mismo objetivo [6]. Adicionalmente, es posible

mitigar este problema acotando de forma apropiada los valores máximos y mínimos admitidos para este campo, con el fin de mantener la probabilidad de pérdida en un determinado intervalo. Esto limitaría la ventaja que se podría obtener un usuario que intenta aparentar poseer un RTT exageradamente alto con el fin de disminuir su tasa de pérdidas.

Una segunda alternativa plantea el inferir el RTT a partir de las direcciones IP de origen y destino del paquete. La determinación de esta estimación podría depender en base a la distancia física aproximada entre el remitente y destinantario, o bien podría calcularse a partir de datos históricos que manejan los proveedores de Internet (ISP, Internet Service Provider) [113, 114]. Esta alternativa tiene la gran ventaja de no requerir ningún cambio a las especificación actual del formato del paquete IP, aunque presenta la desventaja de utilizar estimaciones que podrían diferir de forma importante del valor real (más cercano al estimado por TCP en los extremos). Otro factor, independiente de la relación que se establezca entre direcciones IPs y el RTT, es el uso de Virtual Private Networks (VPNs), cuyo proceso de encapsulamiento y ruteo de paquetes generaría la infraestimación del RTT percibido en el router. Esto, pues el RTT total se divide en dos partes: donde una de ellas está asociada al camino entre el emisor y el servidor de la VPN, y la restante se asocia al camino entre este último y el receptor.

Una característica atractiva que comparten ambas alternativas mencionadas previamente corresponde al hecho de que su implementación no requiere cambios significativos en el firmware de los routers. Adicionalmente, tampoco se necesita mantener variables de estado por conexión, lo cual le brinda a la propuesta una gran escalabilidad, pues se evitan problemas de procesamiento o memoria cuando aumenta el número de conexiones que atraviesan el router.

Otra alternativa considerada fue el uso del campo Time-To-Live (TTL) como una forma de inferir el RTT, pero existen múltiples problemas con esta idea. En primer lugar, la estandarización del TTL utilizado por defecto por sistemas operativos no es consistente, lo cual se debe a que solo existe una recomendación respecto a su valor, pero no existe una obligación para hacer uso de esta [115]. Adicionalmente, el intentar relacionar directamente un valor de TTL con un cierto RTT no es sencillo, pues existe enlaces de muy alta latencia (como los intercontinentales) o aquellos de muy baja latencia (redes locales o zonas urbanas) tienen el mismo efecto neto sobre el valor del TTL (la disminución de su valor previo en una unidad). Luego, no solo sería necesario promover la estandarización de aquel valor por defecto en todos los sistemas operativos, sino que además sería necesario idear un método que evite errores de estimación generados por una correlación baja entre número de enlaces y el RTT de las conexiónes. El primer punto puede ser conflictivo, pues elimina la libertad de hacer uso de un valor personalizado, lo cual podría ser útil para un protocolo o aplicación que haga uso de un valor personalizado (como el conocido comando traceroute). Por otra parte, el intentar generar políticas que permitan eliminar los errores puede ser algo complejo, y podría ser poco robusto frente a la evolución de la topología de la red.

Una forma de abordar los problemas de la propuesta anterior sería el proponer un nuevo campo que cumpla con un comportamiento similar al TTL, el cual permita abordar los problemas mencionados en cuanto al error de estimación. Esto evitaría además generar conflictos con la funcionalidad actual del TTL, aunque seguiría manteniendo una dependencia con la topología.

Capítulo 4

Metodología

Para evaluar el rendimiento del algoritmo propuesto, se hace uso de Riverbed Modeler [116] como plataforma de simulaciones de eventos discretos. En esta se montan un total de 3 escenarios distintos para evaluar el rendimiento de la propuesta descrita en el Capítulo 3. Por otra parte, una vez obtenidos los datos crudos de las simulaciones, se hace uso de Matlab [117] para el posterior procesamiento de los datos.

En el presente capítulo se describe la metodología empleada en este trabajo, lo cual incluye una breve descripción de la topología y parámetros que describen cada uno de los escenarios construidos, así como también las métricas que se utilizaron para la evaluación del rendimiento de los algoritmos estudiados.

4.1. Algoritmos estudiados

Tal como se explicó en el capítulo anterior, la propuesta se basa en agregar un factor multiplicativo a la probabilidad de pérdidas que experimentan los paquetes al momento de ingresar a la cola de un router. En particular, se propone la utilización de esto sobre el funcionamiento de RED (ya sea la propuesta original o una adaptativa), dada su simplicidad y el hecho de que corresponde a una propuesta vastamente estudiada.

Dado lo anterior, los algoritmos en torno a los cuales girará este estudio serán RED y ARED (con el fin de considerar la capacidad adaptiva dentro del estudio). Luego, en cada uno de los escenarios que se mencionarán a continuación se realizarán simulaciones con 4 algoritmos distintos: RED, ARED, RBF-RED (RTT-Based Fair RED) y RBF-ARED (RTT-Based Fair ARED), donde el prefijo "RBF" indica que la probabilidad de pérdidas tiene la forma descrita en la Ecuación 3.10.

4.2. Parámetros utilizados

Existen algunos parámetros que se mantienen constantes a lo largo de todo el trabajo y estos corresponden a algunos valores claves utilizados para determinar las probabilidades de pérdidas de RED y ARED.

En primer lugar, es importante considerar que la capacidad del cuello de botella de todos los escenarios es de 100 Mbps. Tal como se indicó en la Sección 2.2.4.3, lo anterior permite escoger un valor más apropiado de min_{th} en base a un valor deseable para la latencia. Esta última cantidad se denota como $delay_{target}$, y en este caso se decide adoptar el valor por defecto de 5 [ms] [76]. Utilizando esto en conjunto a la Ecuación 2.4, y considerando además un tamaño de paquetes de 1500 bytes, se puede determinar un valor aproximado de 21 paquetes para min_{th}. Luego, haciendo uso de la Ecuación 2.3, se obtiene un valor de 63 paquetes para max_{th} [3].

Por otra parte, para el valor de max_p se decide utilizar las recomendaciones respectivas para cada algoritmo. Es decir, para RED se utiliza un valor igual a 0.1 [7, 74], mientras que para ARED el valor inicial de max_p se fija en 0.02 [3, 76]. Es decir, ARED partirá las simulaciones siendo menos agresivo que RED, lo cual puede cambiar dependiendo de la magnitud de la congestión.

Finalmente, en cuanto a w_q se decide mantener el valor recomendado inicialmente para RED [7, 74]. Si bien se podría haber utilizado la recomendación que se hace en [3], la cual toma en consideración la capacidad del enlace, las pruebas realizadas con el valor resultante resultan en un filtro que es incapaz de lidiar con la agresividad de múltiples Slow Start. Adicionalmente, resulta conveniente utilizar un valor de w_q constante, pues de esta forma ambos algoritmos tienen un comportamiento consistente a nivel de estimador del tamaño de la cola.

A modo de resumen, se presentan todos estos parámetros en la Tabla 4.1:

Parámetro	Valor	Referencia
$delay_{target}$	5 [ms]	[3, 76]
min_{th}	21 [paquetes]	[3]
max_{th}	63 [paquetes]	[3]
$RED : max_p$	0.1	[7, 74]
$ARED : max_p(t0)$	0.02	[3, 76]
w_q	0.002	[7, 76]

Tabla 4.1: Parámetros definidos para la operación de los algoritmos RED y ARED.

Por otro lado, tal como se mencionó en la Sección 2.2.4.2, tanto para RED como ARED se hace uso del modo gentil, lo cual evita tener una discontinuidad en la curva que define las probabilidades de pérdidas como función de avq_q (ver Figuras 2.5 y 2.8).

Otro aspecto fundamental es que la aplicación que utilizan las conexiones TCP son transmisiones FTP, lo cual quiere decir que el tráfico de datos solo fluye desde servidores a clientes.

Esto simplifica el comportamiento de la ventana de congestión, ya que la transmisión de datos no depende de una demanda intermitente a nivel de capa de aplicación, y por lo tanto se transmiten datos constantemente siempre que la ventana lo permita, dándole validez a los modelos de estado estacionario del throughput alcanzado por TCP durante Congestion Avoidance. Dicho esto, en el sentido opuesto solo se tendrán paquetes del tipo ACK, los cuales no experimentarán ninguna perdida, ni tampoco se verán sometidos al fenómeno conocido de compresión de ACKs [118]. Este último consiste en la acumulación de estos paquetes en las colas debido a congestión, lo cual genera la recepción de estos reconocimentos uno después de otro en el emisor (como una cadena de bits), generando incrementos consecutivos de la ventana de congestión y contribuyendo así a generar rafagas de paquetes en el sentido en que fluyen los paquetes de datos.

Respecto a la política de acceso al canal para los paquetes que ingresan a la cola, se respeta el esquema First-In First-Out (FIFO) durante todas las simulaciones del trabajo. Esto implica que la latencia adicional causada en una cola es directamente proporcional a la cantidad de paquetes presentes en ella, pues el último paquete que entra debe esperar la transmisión de todos aquellos que le anteceden en la cola (ya que debe ser el último en salir).

Una observación importante corresponde al hecho de que solamente se utilizan datos que se ubican en el intervalo temporal donde todas las conexiones están compitiendo. Esto se debe a que la dinámica del sistema podría cambiar de forma considerable y afectar de forma significativa a las métricas, lo cual podría llevar a sacar conclusiones erróneas. Esto deriva a definir la máxima cantidad de datos que puede transmitir un nodo durante las simulaciones, pues esto tiene una relación directa con la duración de las mismas. Esto es equivalente en efectos prácticos al tamaño del archivo a transmitir mediante FTP, y los valores definidos para este parámetro para cada escenario se presentan en la Tabla 4.2. Allí se puede apreciar que para el último caso se utiliza un valor mayor, lo cual se debe al hecho de que la topología de este escenario permite tasas de transmisiones más altas, y por lo tanto se requiere un archivo más grande para que la duración de la transmisión que termina primero no disminuya significativamente.

Escenario	Tamaño de archivo FTP	Equivalente en paquetes de 1500 MB
1	80 [MB]	53333 [paquetes]
2	80 [MB]	53333 [paquetes]
3	500 [MB]	333333 [paquetes]

Tabla 4.2: Tamaños del archivo a transmitir por FTP en cada escenario. Valor utilizado es mayor en tercer escenario debido a que el ancho de banda disponible para las conexiones TCP es mayor.

4.3. Postura frente a fairness

En la comunidad científica no existe un consenso respecto a un objetivo concreto que represente un “buen fairness"[20]. En este trabajo se tiene como objetivo la repartición equitativa del ancho de banda del cuello de botella, tanto para los escenarios que tienen topología

tipo dumbbell donde solo hay un enlace congestionado, como en el último escenario en el cual se utiliza una topología del tipo parking-lot, donde se puede presentar congestión en más de un enlace.

El hecho de que las topologías a utilizar tengan un cuello de botella por el cual pasan todas las conexiones contribuye a la simplificación de este tópico, pues esto implica que incluso si se soluciona el problema de optimización que plantea el criterio Max-Min Fairness, se llegaría a la conclusión de que la distribución de throughputs que satisface dicho criterio es aquella donde la capacidad del cuello de botella se reparte de forma equitativa.

4.4. Escenario inicial: 2 conexiones TCP y tráfico de fondo a una tasa constante

El primer escenario construido se caracteriza por su simpleza, lo cual contribuyó significativamente a la rápida obtención de resultados preliminares del rendimiento de la propuesta en las etapas tempranas de la investigación.

4.4.1. Descripción de la Topología

Este escenario hace uso de una topología estilo dumbbell, donde 2 conexiones TCPs con distintos niveles de latencia compiten por el ancho de banda del cuello de botella. Estas transmiten el tráfico desde servidores (S) a clientes (W), y se caracterizan por RTTs de 50 y 150 ms. Esto se representa en la Figura 4.1, donde adicionalmente se puede apreciar la existencia de dos componentes adicionales denominados **source** y **sink**, los que se utilizan para añadir un tráfico de fondo que se caracteriza por el tiempo de interarribo entre paquetes. Esta última cantidad corresponde al inverso de la tasa a la cual se transmiten los paquetes en el **source**, los cuales son destruidos posteriormente en el **sink**.

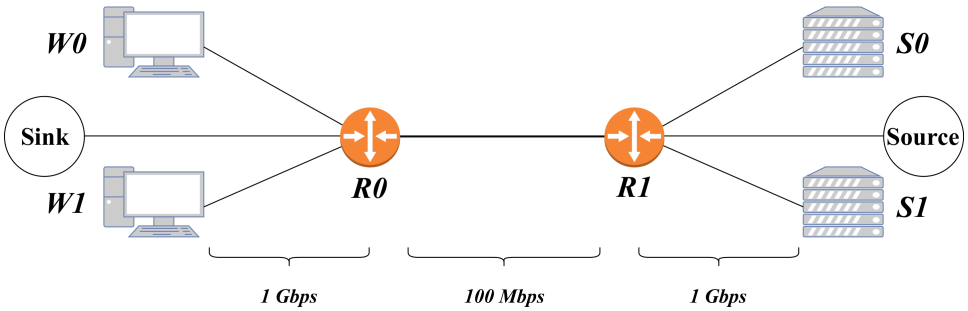


Figura 4.1: Primera topología: dumbbell con 2 conexiones TCP y tráfico de fondo que se transmite a una tasa constante.

Dado que la propuesta hace uso del RTT como dato conocido, se decide darle un valor ficticio $RTT_{bg} = 100\ [ms]$ a los paquetes del tráfico de fondo [110], con el fin de tener una base sobre la cual calcular la probabilidad de pérdidas que afectará a estos paquetes en el

router del cuello de botella ($R1$).

En este caso, es importante notar que existirá una relación directa entre el tiempo de interarribo y el ancho de banda que podría podría potencialmente utilizar el tráfico de fondo, en ausencia de pérdidas. Esta relación se presenta en la Figura 4.2, donde se puede apreciar que en general el ancho de banda que quedaría disponible para la competencia de las 2 conexiones TCP se mueve entre 5 Mbps y 15 Mbps aproximadamente, considerando la ausencia de pérdidas para el tráfico de fondo.

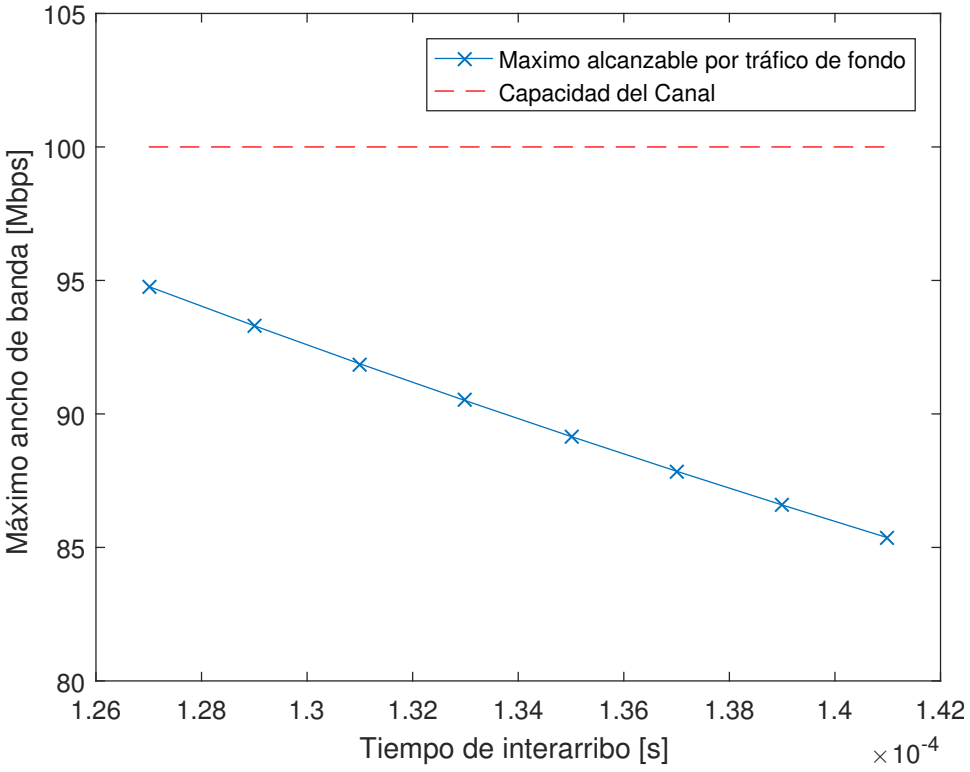


Figura 4.2: Relación entre el tiempo de interarribo y el máximo ancho de banda que puede utilizar el tráfico de fondo, el cual se alcanzaría en ausencia de pérdidas.

A nivel práctico, este tipo de tráfico de fondo es análogo a un flujo UDP que no posee implementado un algoritmo de control de congestión y que transmite a una tasa constante. Considerando esto y que es un problema conocido para AQM el tener que lidiar con este tipo de tráfico [6], lo que se espera observar es que este tráfico efectivamente restringirá el ancho de banda disponible para las conexiones TCP de los servidores $S0$ y $S1$ en un rango muy cercano a 5-15 Mbps, aun cuando el tráfico de fondo sufra algunas pérdidas por el algoritmo de AQM implementado.

4.4.2. Métricas a utilizar

Considerando que en este caso solo se tienen 2 conexiones TCP, hay 2 métricas claves que permitirán hacerse una idea rápida sobre el rendimiento de la propuesta a nivel de fairness. La primera de ellas corresponde simplemente a la razón entre los throughputs obtenidos por

cada conexión ($\frac{Th_1}{Th_2}$, donde Th_1 hace referencia al throughput que logra $S0$ en la topología, mientras que Th_2 se refiere a aquel que logra $S1$), lo cual además de ser fácil de interpretar permite analizar cómo se comparan los resultados empíricos a la relación teórica descrita por la Ecuación 3.3.

En segundo lugar, se pretende comparar los throughputs en el "fairness plane", un plano que tiene como ejes los throughputs alcanzados por cada una de las conexiones. En este se podrá apreciar desde otro punto de vista qué tan lejos se está de una justicia ideal (donde el throughput es repartido de forma equitativa entre ambas conexiones, y que se representa visualmente como la recta que divide en partes iguales al plano) y si existe una tendencia consistente a medida que se hace variar el tiempo de interrarribo. Adicionalmente, permite hacerse una idea sobre si existen diferencias significativas a nivel de utilización total, considerando que en este plano las rectas con pendiente negativa y unitaria representan un cierto valor de utilización total (suma de Th_1 y Th_2), dependiendo de su posición. En particular, la utilización total es directamente proporcional a la distancia entre la recta respectiva y el origen.

Finalmente, se hará uso del Fairness Index (ver Ecuación 2.6) para tener una métrica común a través de todos los escenarios, la cual es frecuentemente utilizada en la literatura [119, 51, 89]. Es importante notar que en este caso la normalización de los throughputs es innecesaria, pues se vuelve irrelevante cuando se tiene como objetivo la distribución equitativa de los throughputs. Esto se debe a que matemáticamente el factor de normalización se cancelaría, debido a su aparición como factor común en el numerador y denominador. Lo anterior también implica que la métrica obtiene el mismo valor si los throughputs se igualan, independiente de la magnitud de los throughputs obtenidos. Es decir, el resultado del índice no depende de la utilización total, la segunda métrica que evaluará en los tres escenarios, con el fin de poder validar la hipótesis del presente trabajo.

4.5. Segundo escenario: 10 conexiones TCP en competencia con topología dumbbell

El segundo escenario tiene como objetivo simular exclusivamente la interacción entre un mayor número de conexiones TCP, de tal forma que ya no existe una fuente de tráfico a tasa constante, sino que todo tráfico proviene de 10 conexiones TCP que experimentan distintos niveles de latencia.

4.5.1. Descripción de la Topología

La topología utilizada en este escenario es practicamente idéntica a la anterior, con la diferencia de que no existe un *source* o un *sink*. En cambio, se tienen 8 conexiones TCP adicionales con respecto al primer escenario, tal como se aprecia en la Figura 4.3. Estas conexiones poseen distintos RTTs, los cuales se mueven entre 37.5 [ms] y 150 [ms], lo cual

debería permitir observar una injusticia considerable para las conexiones que se ubican en el extremo de RTTs altos.

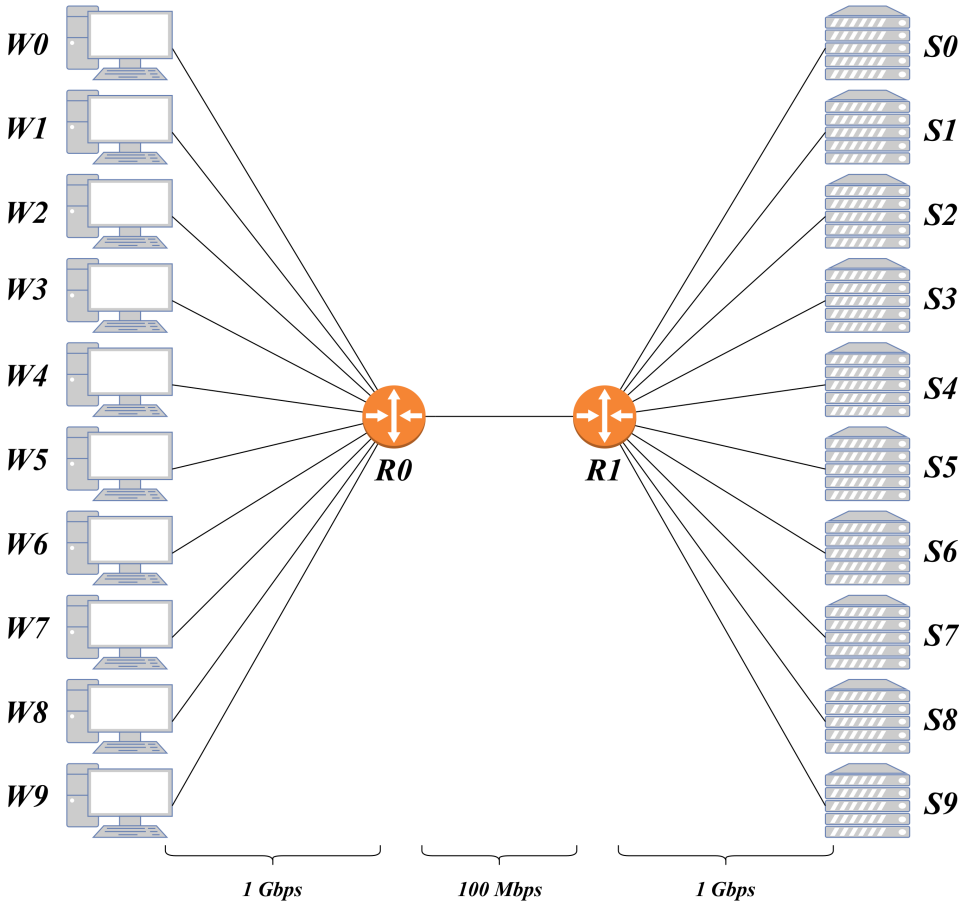


Figura 4.3: Segunda topología: dumbbell con 10 conexiones TCP.

4.5.2. Métricas a utilizar

Lo primero que se debe notar es que en este caso no es posible utilizar la razón entre los throughputs como un indicador representativo de fairness, pues en este caso el rendimiento del sistema depende de qué tan buena es la repartición del ancho de banda entre todas las conexiones que compiten y no basta considerar solo un par de ellas.

De tal forma, el análisis se basará en el Fairness Index y la utilización total que alcanzan las 10 conexiones en conjunto. En particular, al igual que en el resto de los escenarios, lo que se buscará en primer lugar es analizar la mejora que se logra a nivel de fairness, y luego comprobar si existe un efecto negativo a nivel de utilización total. La intuición detrás de la posibilidad de que suceda lo anterior proviene del viene de la hipotesis que considera que es posible que en el caso tradicional (sin el uso de la propuesta),las conexiones que se benefician de la injusticia sean capaces de captar el ancho de banda disponible de forma más eficiente en base a su mayor agresividad. Esta propiedad podría contribuir a mantener un alto nivel de utilización del canal, y se podría perder si la utilización de estas conexiones se mantiene

acotada para obtener una mejor justicia.

4.6. Tercer escenario: 4 conexiones TCP en competencia con topología estilo parking-lot

El tercer escenario también considera exclusivamente tráfico proveniente de conexiones que utilizan TCP tradicional. No obstante, a diferencia del caso anterior, el número de conexiones es reducido a 4, lo cual permitirá que se alcancen throughputs aproximadamente 2 veces más altos. Por otra parte, el rango de latencias es más acotado, con RTTs entre 60 [ms] y 100 [ms], donde adicionalmente 2 conexiones comparten un valor de RTT igual a 80 [ms].

4.6.1. Descripción de la Topología

La topología del último escenario es del tipo parking-lot, donde se tienen 4 conexiones TCP que atraviesan un número variable de enlaces de latencia constante igual a 10 [ms], lo cual permite que se tengan distintos valores de RTT. Esta se presenta en la Figura 4.4, donde se puede apreciar la existencia necesaria de más de 2 routers, lo cual permite teóricamente que se dé congestión en un router distinto al del cuello de botella. En particular, en este caso el router que maneja el tráfico en el cuello de botella corresponde a $R2$ y es posible que se genere algo de congestión en $R3$.

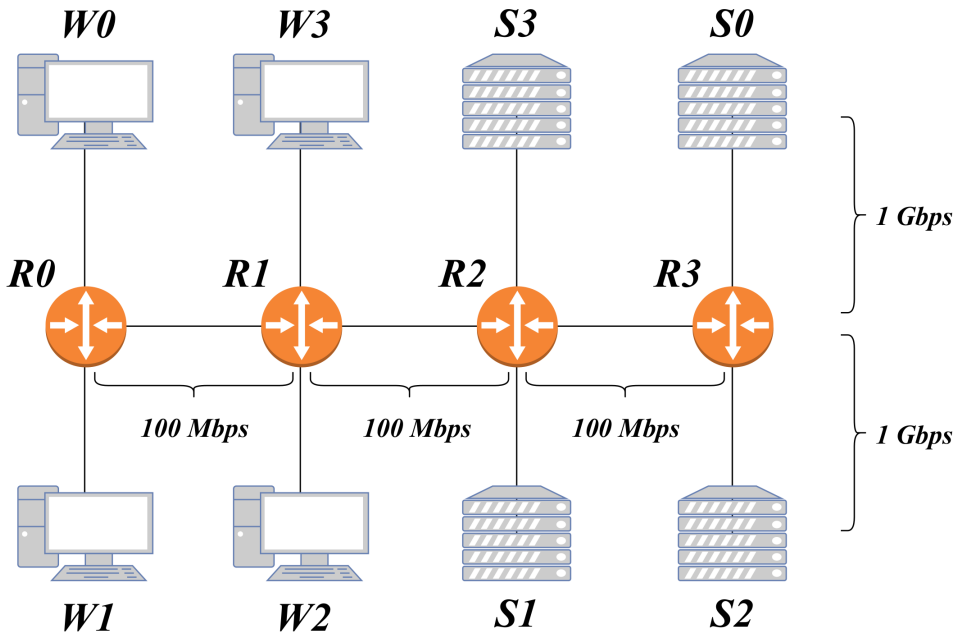


Figura 4.4: Tercera topología: parking-lot con 4 conexiones TCP.

4.6.2. Métricas a utilizar

En este escenario, las métricas a utilizar serán las mismas que fueron utilizadas en el escenario anterior, aunque se pondrá énfasis en un par de puntos claves. En primer lugar, se desea observar si existe alguna diferencia significativa en la utilización que obtienen ambas conexiones que tienen el mismo RTT. Esto, pues una de ellas pasa por $R3$, quien podría experimentar eventos de congestión menores. Finalmente, otro aspecto que parece relevante es el hecho de que en este caso todos los RTTs se ubican a un mismo lado del valor de referencia mencionado en 3.2, lo cual permitiría demostrar con un ejemplo concreto que esto no implica que aumenten las probabilidades de pérdidas experimentadas por el sistema (a pesar de que el factor de ajuste del algoritmo siempre sea mayor a 1 en la Ecuacion 3.10).

Capítulo 5

Resultados y Análisis

En el presente capitulo se presentan los resultados obtenidos mediante las simulaciones realizadas durante la investigación, junto a un análisis que hace uso de las métricas mencionadas en el capítulo previo para evaluar el rendimiento de cada uno de los algoritmos estudiados. En particular, las métricas más relevantes y que se utilizan en todos los escenarios corresponden a la utilización total alcanzada por el conjunto de conexiones TCP y el Fairness Index.

5.1. Primer escenario: 2 conexiones TCP y tráfico de fondo a una tasa constante

Para el primer escenario, dado que se explora un rango de tiempos de interarribo entre paquetes del tráfico de fondo, las figuras que se presentarán a continuación se obtienen en base a datos calculados como el promedio del resultado de tres simulaciones independientes con distintas semillas. Esto permite disminuir parcialmente la posibilidad de realizar un análisis sesgado en base a casos que pueden ser poco frecuentes.

Adicionalmente, en el contexto de este escenario se entenderá a la conexión 1 como aquella establecida entre $S0$ y $W0$ en la topología de la Figura 4.1. Ignorando la latencia adicional causada por las colas, esta conexión se caracteriza por un RTT de 50 ms, mientras que la conexión 2, establecida entre $S1$ y $W1$ tiene un RTT igual a 150 ms.

5.1.1. Análisis de la justicia obtenida

El aspecto más interesante del presente estudio corresponde al nivel de justicia que es posible alcanzar con un algoritmo como el propuesto en funcionamiento. En la Figura 5.1 se presenta la razón entre los throughputs obtenidos por ambas conexiones en el intervalo en el cual compiten (antes de que se acabe una de estas transmisiones). Allí se observa que la razón

para los datos obtenidos por RED y ARED se ubican de forma consistente bastante cerca del valor esperado, el cual está dado por la razón de los RTTs que experimentan las conexiones (ver Ecuación 3.3). No obstante, se aprecia que la injusticia es levemente menor cuando se emplea RED. Por otra parte, las curvas asociadas a los algoritmos que implementan la propuesta muestran una clara mejora, ubicandose bastante cerca de la curva ideal (alcanzable cuando ambos throughputs son iguales), aunque no logran eliminar la injusticia por completo, y de hecho se ubican de forma consistente por sobre la curva, lo cual quiere decir que es una injusticia que siempre discrimina a la conexión de mayor latencia. Finalmente, y tal como es esperable, no se observan tendencias notables a medida que cambia el valor del tiempo de interarribo.

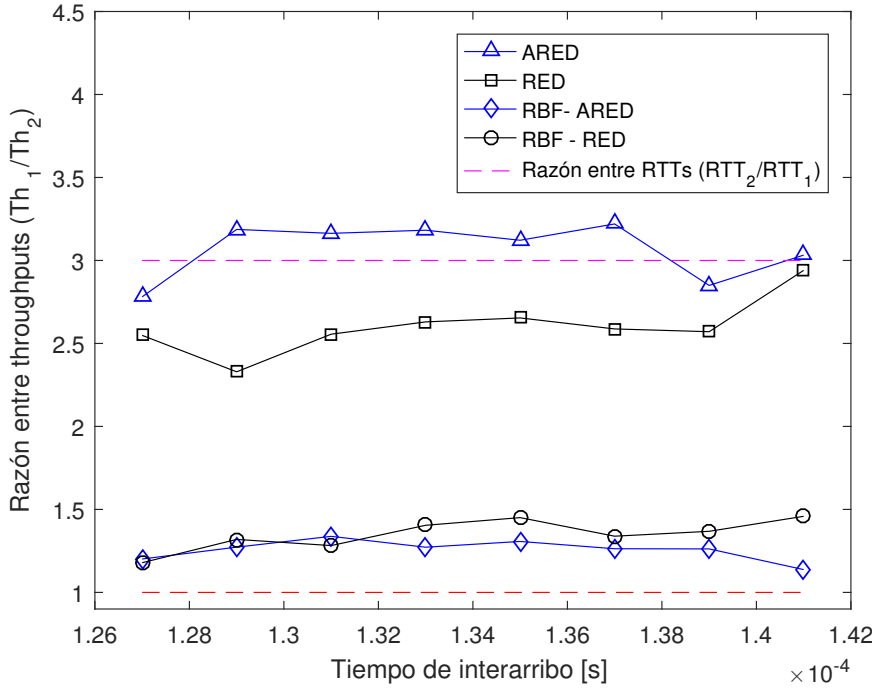


Figura 5.1: Razón entre los throughputs promedios obtenidos por las conexiones 1 y 2, versus el tiempo de interarribo de paquetes del tráfico de fondo. Se muestran 4 curvas, una por cada algoritmo evaluado.

En la Figura 5.2 se grafica el throughput de la primera conexión versus el throughput de la segunda. En este gráfico, una recta desde el origen con una pendiente fija determina un cierto nivel de fairness (la razón entre los throughputs es constante), donde el caso ideal es una recta con pendiente unitaria, la cual representa la igualdad entre ambos throughputs. Luego, y dado los resultados de la Figura 5.1, tiene sentido que en este se observe que la mayoría de los puntos asociados a un algoritmo determinado en general parecieran ubicarse en torno a una recta trazable desde el origen.

Por otro lado, se puede notar que en general el set de puntos asociado a un tiempo de interarribo fijo se ubica en una recta trazable de forma perpendicular a la recta ideal (es decir, tienen pendiente unitaria y negativa). Estas rectas definen un nivel de utilización conjunto de ambas conexiones es constante, y por lo tanto todo punto que pertenezca a una de ellas cumple que la suma de ambas coordenadas (el valor en ambos ejes) es constante. Esto se puede comprobar en la Figura 5.3, donde se observa que la suma de los throughputs obtenidos (Th_1

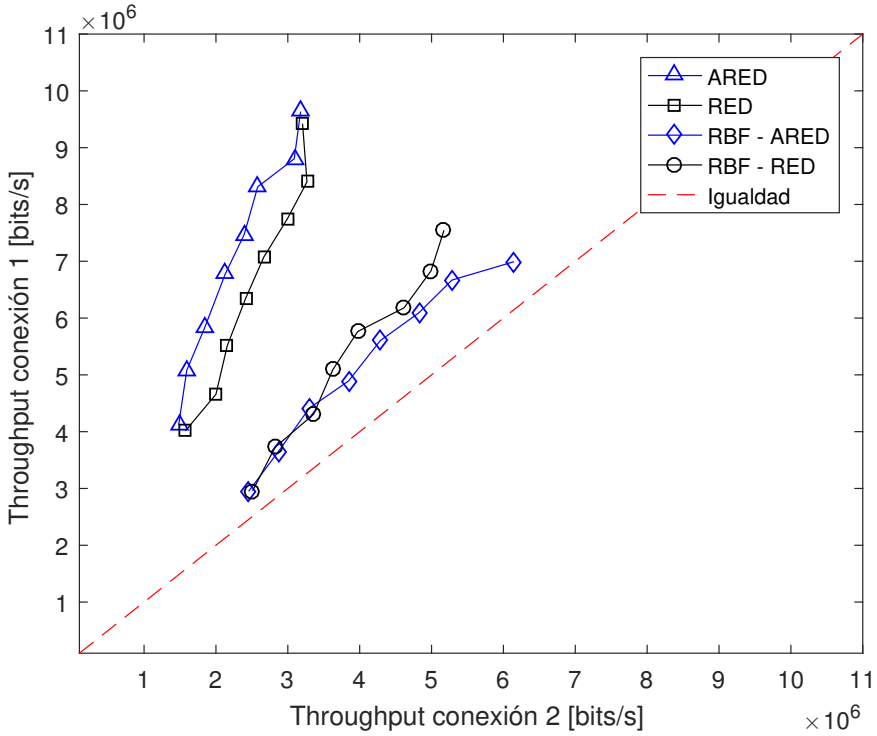


Figura 5.2: Fairness plane: muestra throughput de conexión 1 versus throughput de la conexión 2. Se muestran 4 curvas, una por cada algoritmo.

y Th_2, perteneciente a conexiones 1 y 2 respectivamente) suele mantenerse en un intervalo bastante acotado, como insinuaban los resultados previos. Además, los valores alcanzados en cada caso coinciden de forma aproximada con lo que se esperaría, considerando que el tráfico de fondo no responde frente a congestión, de tal forma que este último alcanza un throughput cercano al máximo presentado en la Figura 4.2. Esto efectivamente corrobora la incapacidad de un algoritmo como RED para lidiar con este tipo de tráfico, lo cual no es sorprendete considerando que este es uno de los problemas más desafiantes de la disciplina de AQM.

En este caso, para hacer el cálculo del Fairness Index lo que se hace es realizar primero un cálculo independiente para cada valor del tiempo de interarribo, y para cada uno de los algoritmos. Luego, estos resultados son promediados, y el producto de dicho procedimiento se presenta en la Figura 5.4, donde lo que se grafica en estricto rigor es el Fairness Index promedio alcanzado. En estos resultados se puede comprobar que existe una clara mejora una vez que se hace uso de la propuesta, lo cual ya era evidente en base a lo visto en las Figuras 5.1 y 5.2. No obstante, la diferencia entre los resultados alcanzados por RED y ARED no es clara o consistente, pues se puede notar que en el caso cuando se utilizan los algoritmos originales es RED el que logra brindar mayor justicia, mientras que ocurre lo contrario en las variantes generadas a partir de la propuesta.

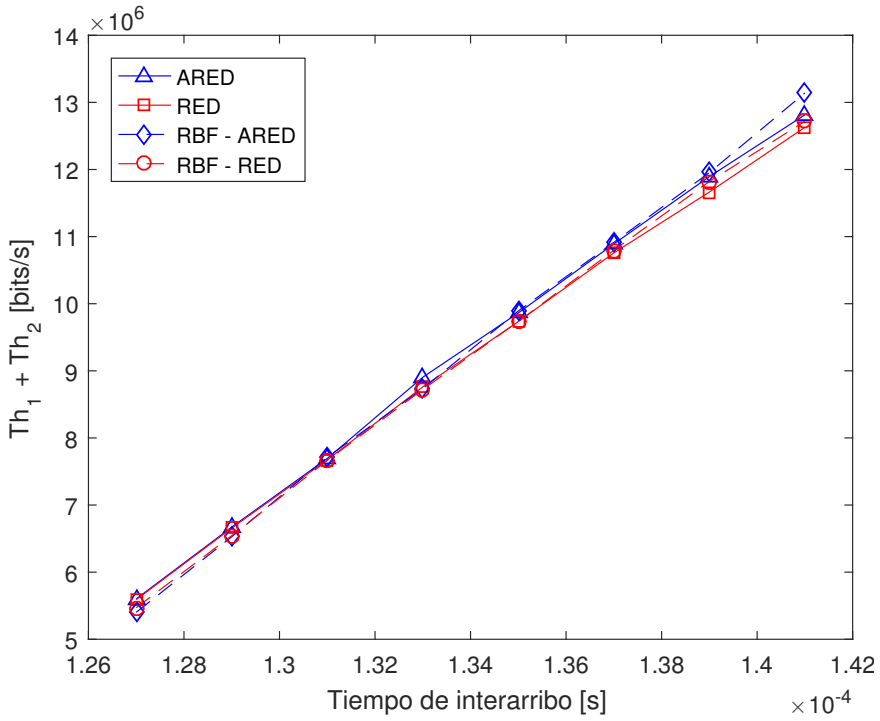


Figura 5.3: Throughput total obtenido por ambas conexiones TCP como función del tiempo de interarribo.

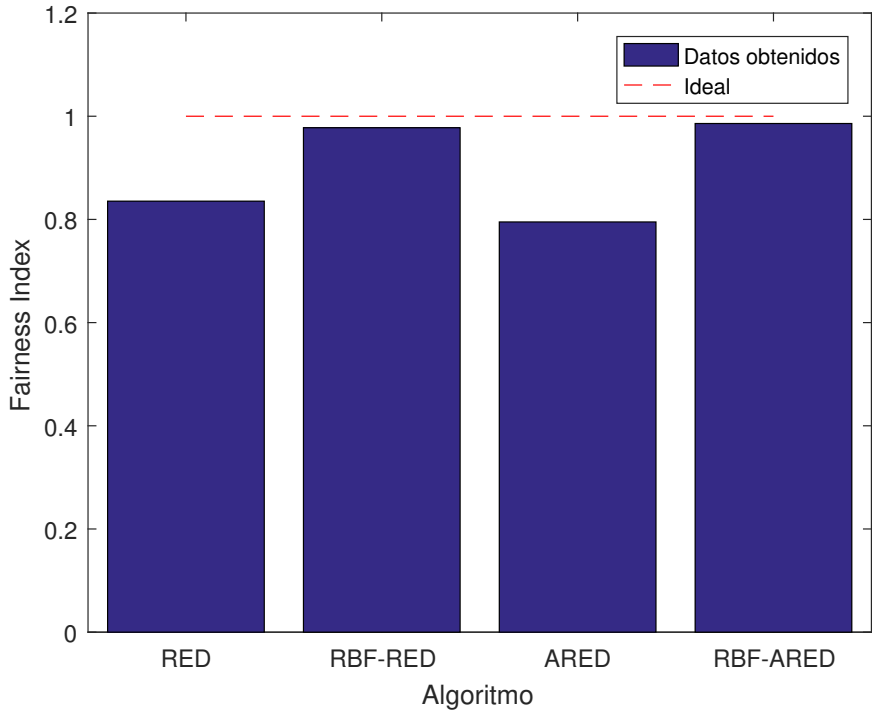
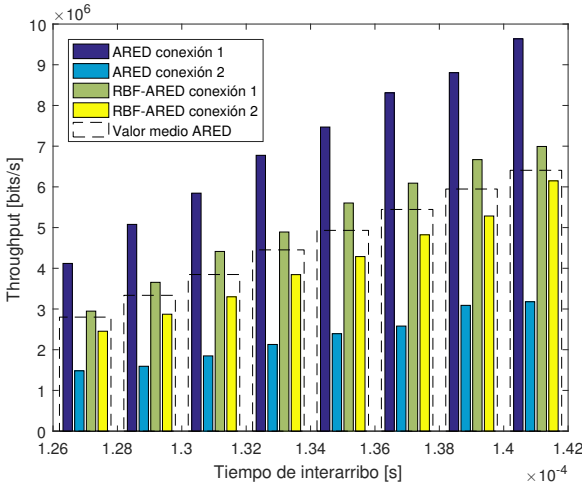


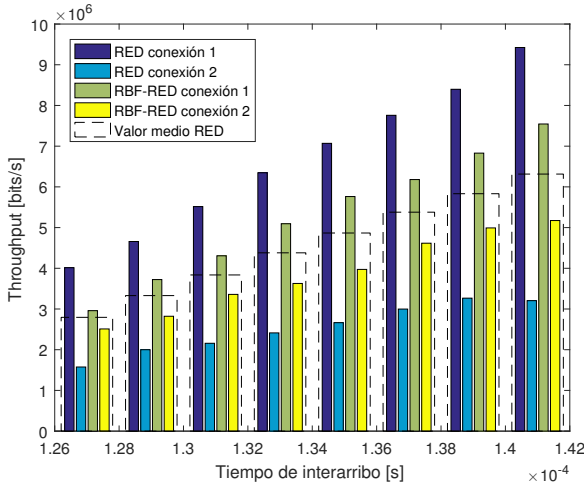
Figura 5.4: Fairness Index obtenido por cada algoritmo, en escenario donde compiten 2 conexiones TCP en un cuello de botella por el cual pasa tráfico de fondo que no responde frente a congestión.

5.1.2. Análisis de la evolución del throughput y la utilización total alcanzada

Los resultados anteriores permiten visualizar fácilmente la mejora que existe en cuanto a fairness, sin embargo no es tan evidente el efecto individual que se obtuvo para un ancho de banda y un algoritmo fijo (RED o ARED). Para abordar esto último se procede a utilizar un gráfico de barras para analizar la evolución de los throughputs luego de implementar RBF-AQM sobre RED y ARED, lo cual se puede ver en las Figuras 5.5a y 5.5b. Allí, además de graficarse los throughputs en forma de barras, se muestra como referencia el valor medio entre los throughputs obtenidos sin el uso de la propuesta, para tener una idea del comportamiento ideal que uno podría esperar del nuevo algoritmo.



(a) Evolución del throughput mediante el uso de RBF-ARED.



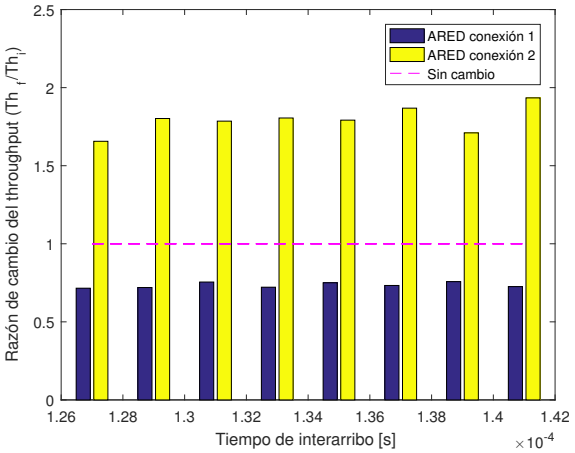
(b) Evolución del throughput mediante el uso de RBF-RED.

Figura 5.5: Evolución del throughput obtenido luego de aplicar la propuesta sobre ARED y RED.

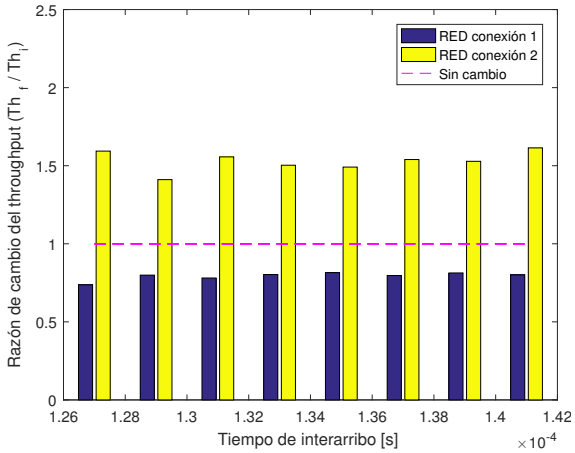
Adicionalmente se presentan las razones entre los throughput que se obtienen al aplicar la propuesta y previo a esto, sobre ARED y RED. Estos resultados se presentan en las Figuras 5.6a y 5.6b respectivamente. En estas es posible notar que el cambio porcentual experimentado por la conexión de latencia alta es significativamente más alto que aquel experimentado por la conexión de latencia pequeña. Esto tiene sentido, pues si se asume un trafico de fondo con utilización constante, entonces el throughput que pierde la conexión 1 es utilizado por la conexión 2. Dada que la diferencia es constante en ambos casos (con un signo distinto), la variación porcentual siempre será superior para la conexión que tuvo un throughput inicial más pequeño.

Lo anterior es relevante cuando se piensa en terminos de Quality of Experience (QoE), pues si el algoritmo propuesto fuera implementado se tendría que el usuario con latencia alta notaría una mejora significativa en el rendimiento de su conexión, debido al gran incremento relativo, mientras que el usuario con conexión baja solo notaría un rendimiento levemente menor y por lo tanto menos perceptible.

El análisis previo es sencillo para el caso de 2 conexiones, pero la idea general de que un cambio de cierta magnitud es más perceptible para aquel con un throughput inicialmente más pequeño es siempre cierta. Dicho esto, resulta interesante cuestionarse qué es lo que sucede en el caso general donde se tienen múltiples conexiones compitiendo entre sí. En particular, si se asume una distribución inicial de throughputs en base a la razón inversa de los RTTs (proveniente del modelo del throughput alcanzado en estado estacionario por TCP tradicional), y se asume que estos se logran igualar, entonces es posible demostrar matemáticamente que el efecto promedio que se percibe es positivo, independiente de la distribución de los RTTs o del número de conexiones consideradas en el análisis (ver Anexo A).



(a) Razón de cambio de los throughputs luego de aplicar la propuesta sobre ARED.



(b) Razón de cambio de los throughputs luego de aplicar la propuesta sobre RED.

Figura 5.6: Razón entre los throughputs luego de aplicar la propuesta y aquellos que se obtienen previo a esto.

La última métrica de interés en cuanto a throughputs corresponde a la utilización total promedio. Los resultados de esta se presentan en la Figura 5.7, y se obtienen realizando un proceso análogo al utilizado para el Fairness Index. De tal forma, se considera un valor ideal para la utilización de aproximadamente un 10 % de la capacidad total (100 Mbps), en base a la relación presentada previamente entre el tiempo de interarribo y el throughput máximo alcanzable por el tráfico de fondo. Los resultados no se desvían mucho de este valor ideal ni entre sí, pero si es posible observar una diferencia clara entre el nivel alcanzado por ARED y el de RED. Esto es una indicación de que la agresividad promedio de ARED es menor, lo cual no depende del uso de la propuesta. De hecho, se puede apreciar que ni para el caso de ARED ni para RED se observa una variación significativa en la utilización total al aplicar RBF-AQM.

5.1.3. Análisis de la probabilidad de pérdidas

Otro aspecto relevante sobre los resultados corresponde al nivel de pérdidas que experimentaron las conexiones cuando se utilizan distintos algoritmos. En las Figuras 5.8 y 5.9 se presentan las probabilidades de pérdida promedio vistas en el router para cada conexión

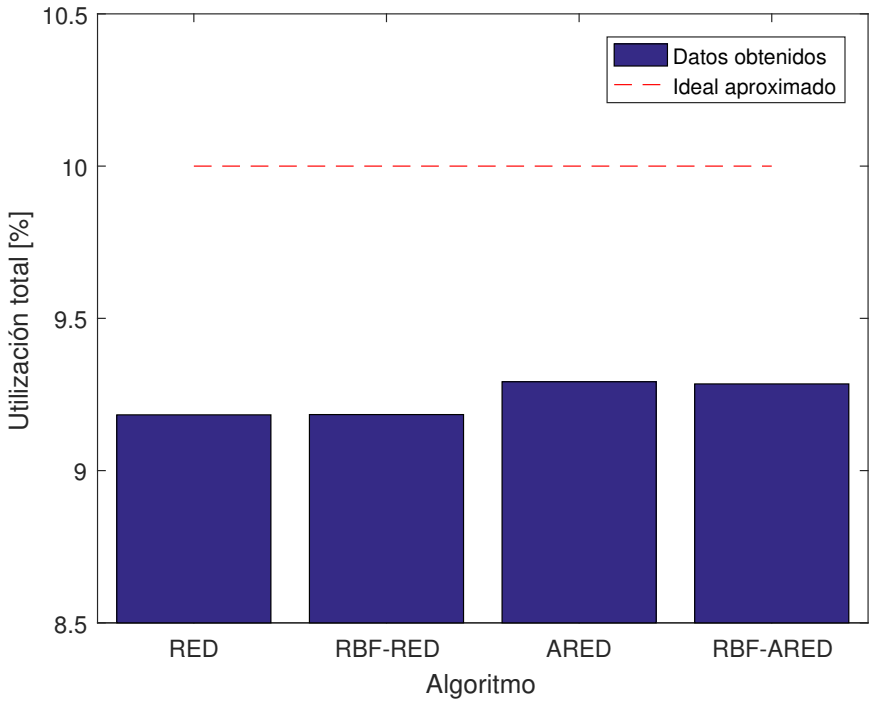


Figura 5.7: Utilización total promedio obtenido por ambas conexiones TCP como función del tiempo de interarribo, en un escenario donde compiten 2 conexiones TCP en un cuello de botella por el cual pasa tráfico de fondo que no responde frente a congestión.

en función del tiempo de interarribo, cuando se utiliza ARED y RED respectivamente. En estos se puede apreciar que cuando no se hace uso de la propuesta, las probabilidades de pérdida que experimentan ambas conexiones son en general muy parecidas. Esto es esperable, pues en este caso no existe discriminación entre una conexión y otra para el cálculo de la probabilidad. En caso contrario, existe un claro distanciamiento entre las probabilidades de pérdida por conexión cuando se hace uso la latencia como un diferenciador para el cálculo de la probabilidad.

Una tendencia clara que se puede apreciar en estos resultados es que existe una disminución en las probabilidades de pérdida a medida que aumenta el tiempo de interarribo. Esto es completamente razonable, pues un tiempo de interarribo mayor es equivalente a un throughput menor para el tráfico de fondo. Ahora bien, la eficiencia de la propuesta no depende directamente de la magnitud de las probabilidades de pérdida de las conexiones, sino que el punto clave corresponde a la razón entre estas (ver Ecuación 3.3).

Lo anterior se puede visualizar más fácimente en la Figura 5.10, donde se muestra la razón promedio entre las probabilidades de pérdidas que ven ambas conexiones al pasar por el router. Se entregan además como referencia una recta constante con valor unitario y otra con valor igual al necesario teóricamente para igualar los throughputs, que corresponde al cuadrado de la razón entre los RTTs. De tal forma, se puede notar que el uso de la propuesta logra aumentar la razón de pérdidas bajo estas condiciones, logrando ubicarse bastante cerca del valor esperado. Adicionalmente se aprecia que la razón entre las pérdidas la ausencia de la propuesta ronda el valor unitario, siendo consistente con lo observado en las Figuras 5.8 y 5.9.

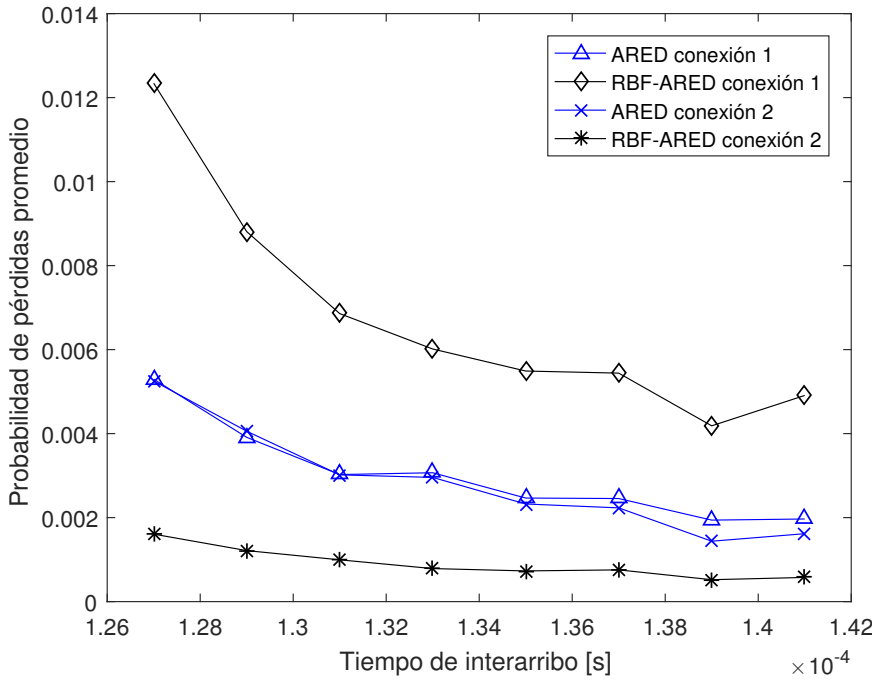


Figura 5.8: Probabilidades de pérdida promedio para cada conexión versus el tiempo de interarribo, haciendo uso de ARED.

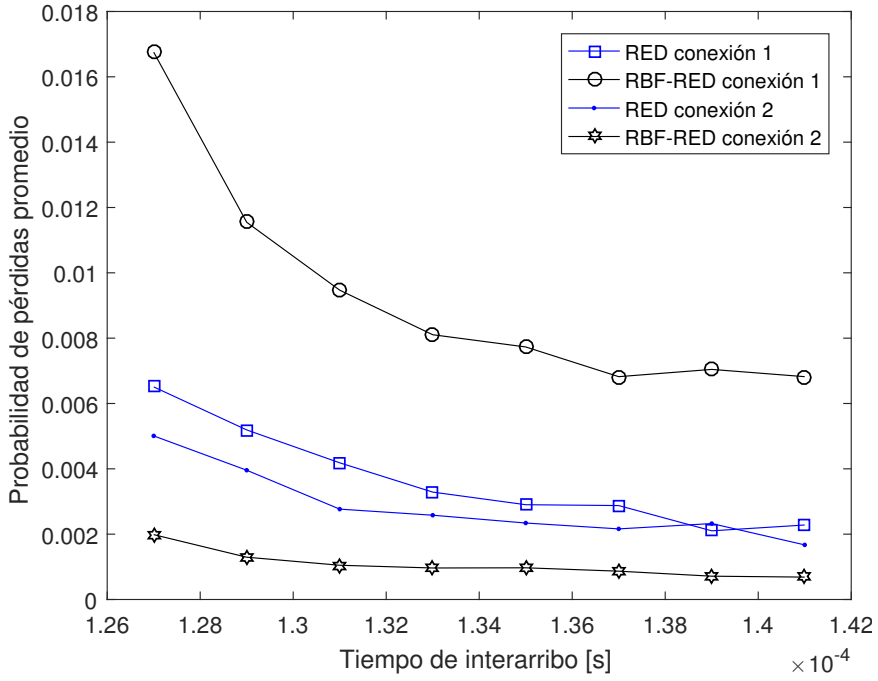


Figura 5.9: Probabilidades de pérdida promedio para cada conexión versus el tiempo de interarribo, haciendo uso de RED.

5.1.4. Análisis del tamaño promedio de la cola

Además de lo que tiene que ver con justicia, se desea observar qué sucede a nivel de tamaño promedio de la cola que se genera en el buffer del router $R0$. Los resultados de esto

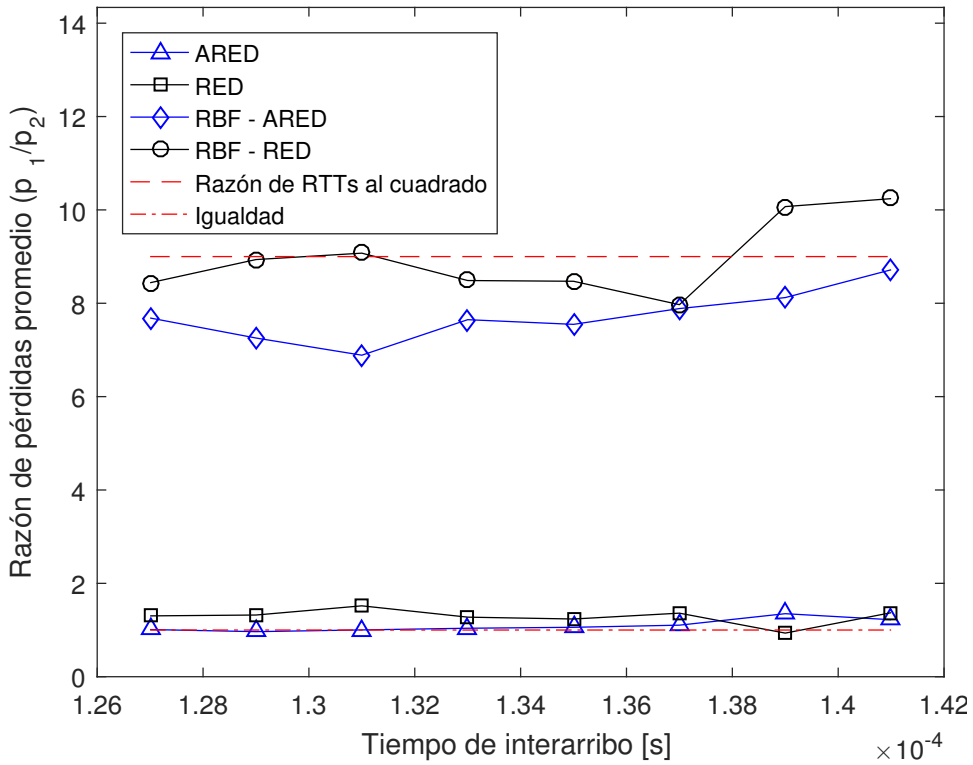


Figura 5.10: Razón de las probabilidades de pérdidas experimentadas por ambas conexiones.

se presentan en la Figura 5.11, donde es posible notar que el tamaño promedio muestra una tendencia de decrecimiento a medida que aumenta el tiempo de intearribo, lo cual demuestra que, a pesar de que las ventanas del tráfico proveniente de las conexiones TCP son mayores (y por consiguiente sus throughputs también), la agresividad del tráfico total disminuye. Esto es lo esperado, pues las conexiones TCP se caracterizan por tener un tráfico de volumen oscilatorio, un comportamiento inevitable producto de la necesidad de tener una capacidad adaptativa y simultáneamente poder responder frente a eventos de congestión.

Otro aspecto llamativo de estos resultados es la clara separación entre las curvas asociadas a RED y ARED. Esto valida la hipótesis previamente planteada durante el análisis de la utilización total sobre la menor agresividad de ARED durante su operación.

5.2. Segundo escenario: 10 conexiones TCP

5.2.1. Análisis de la utilización por conexión

En este caso, ya que se tienen más de 2 conexiones, resulta imposible utilizar la razón entre throughputs como una métrica confiable del rendimiento del algoritmo. Luego, el primer foco de atención se dirige hacia los throughputs promedio alcanzados por cada una de las conexiones. Estos se presentan en las Figuras 5.12 y 5.13 para los algoritmos RED y ARED respectivamente, donde cada barra está asociada a la conexión cuyo RTT se indica en el

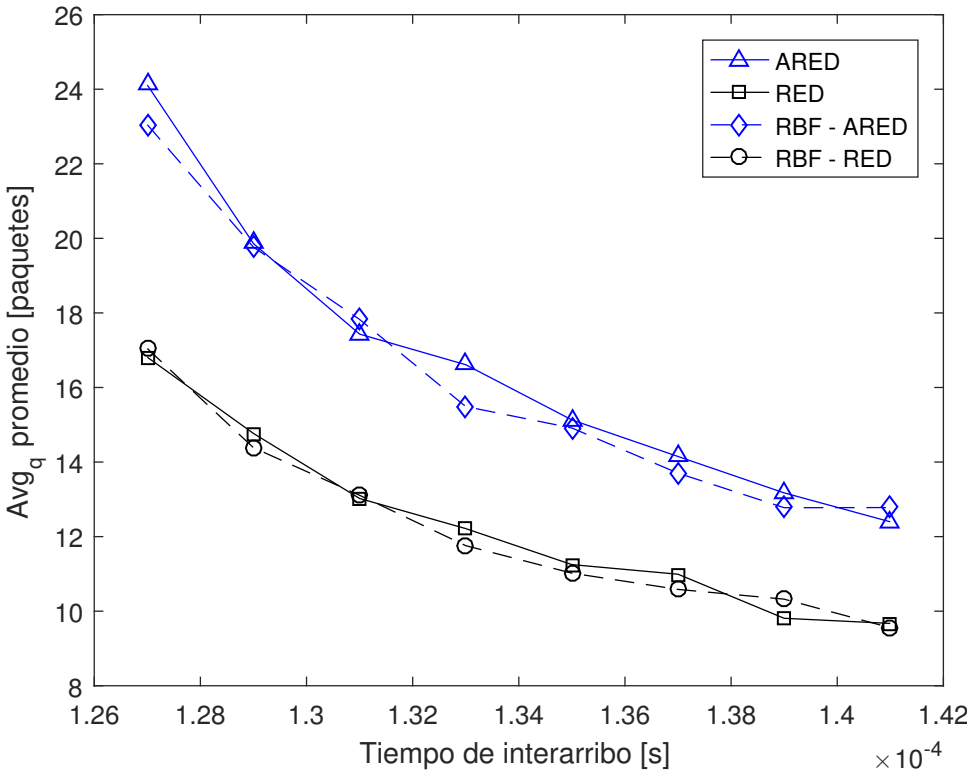
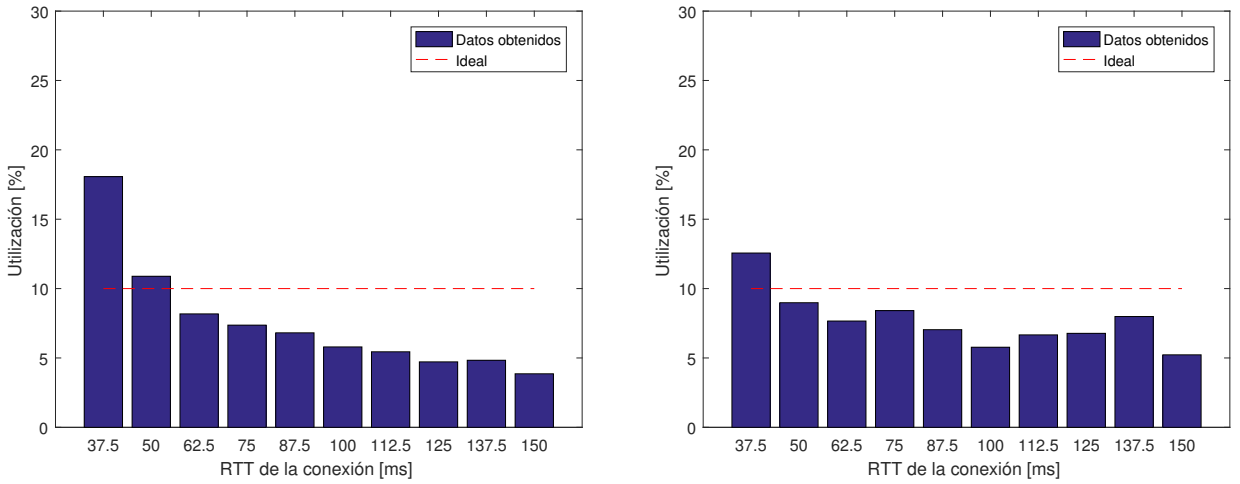


Figura 5.11: Tamaño promedio de la cola en función del tiempo de interarribo.

eje de las abscisas, y en el eje de las ordenadas se muestra el equivalente del throughput en términos de utilización del canal.

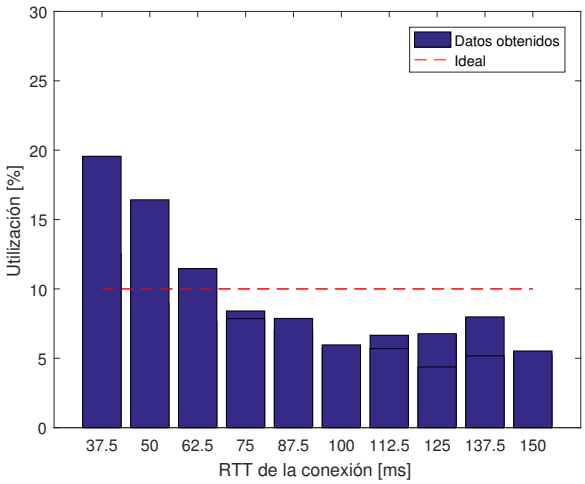


(a) Utilización promedio obtenida por cada conexión cuando se hace uso de RED.

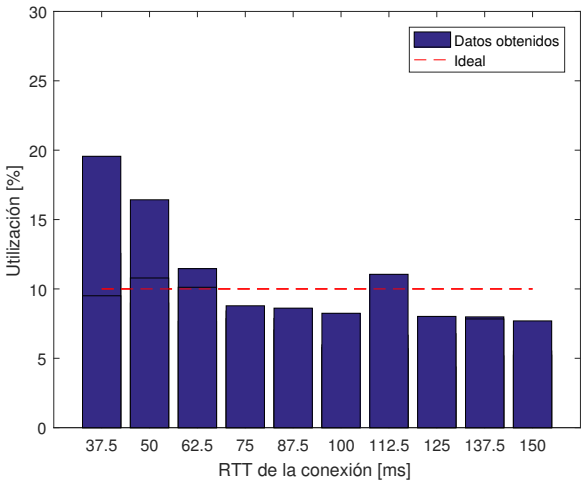
(b) Utilización promedio obtenida por cada conexión cuando se hace uso de RBF-RED.

Figura 5.12: Utilización promedio por conexión en una topología donde compiten 10 conexiones TCP, cuando se hace uso de RED y RBF-RED

Observando estos resultados lo primero que se puede observar es que en general se cumple que el throughput promedio decrece a medida que aumenta el valor del RTT de la conexión, tal como se esperaría en base al modelo de estado estacionario. De la misma forma, al observar



(a) Utilización promedio obtenida por cada conexión cuando se hace uso de ARED.



(b) Utilización promedio obtenida por cada conexión cuando se hace uso de RBF-ARED.

Figura 5.13: Utilización promedio por conexión en una topología donde compiten 10 conexiones TCP, cuando se hace uso de ARED y RBF-ARED.

lo que sucede luego de aplicar la propuesta se puede apreciar una clara mejora en cuanto a la nivelación relativamente homogénea de los throughputs. Por otra parte, visualmente pareciera ser que los throughputs son mayores en el caso de ARED, lo cual es explicable en base a la idea de que ARED logra ajustar su agresividad a un nivel menor que RED, logrando así una mejor utilización total del cuello de botella.

Con el fin de tener resultados que representen de mejor manera el comportamiento promedio, se realizan múltiples simulaciones con 10 semillas distintas. Luego de esto, en primer lugar lo que se obtiene es la curva de la utilización promedio por conexión en función de su RTT, lo cual se muestra en la Figura 5.14, donde se puede notar que existe una nivelación evidente cuando se hace uso de la propuesta, tanto en el caso de RED como en el de ARED. En particular, los resultados obtenidos con RBF-ARED se acercan mucho a la recta ideal. Por último, si se observa con atención el gráfico se puede apreciar que en general las curvas de datos que corresponden a ARED se ubican por encima de las de RED, lo cual nos muestra indirectamente que probablemente la utilización del canal que logra ARED y RBF-ARED es superior, y que es equivalente a afirmar que ARED opera de forma menos agresiva que RED.

5.2.2. Análisis de la utilización total

Con el fin de comprobar que existe una diferencia en la agresividad de ambos algoritmos es necesario observar los datos de utilización total promedio alcanzada por cada algoritmo. Al observar la Figura 5.15 resulta claro que efectivamente se logra una utilización mayor con ARED, y además se aprecia que la diferencia es significativa (mayor al 10 %). Esto tiene sentido si el valor de max_p en el tiempo se ubica por debajo de 0.1, punto en el cual la agresividad de RED y ARED se igualarían.

Lo anterior se comprueba en la Figura 5.16, donde se muestra la evolución de en el tiempo

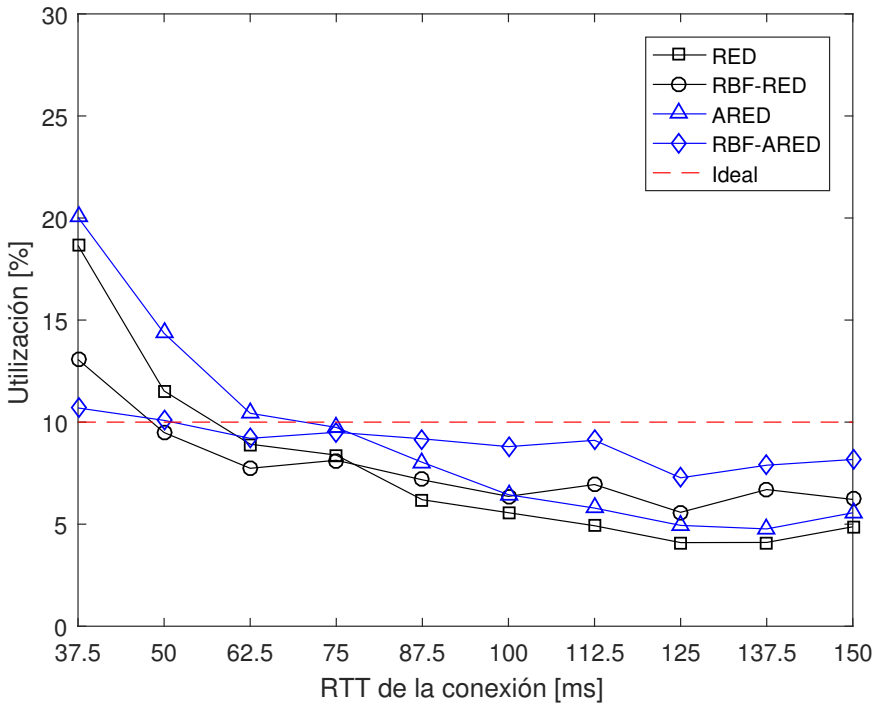


Figura 5.14: Utilización promedio obtenida por cada conexión (caracterizada por su RTT), en escenario donde compiten 10 conexiones TCP. El promedio se obtiene a partir de 10 simulaciones independientes.

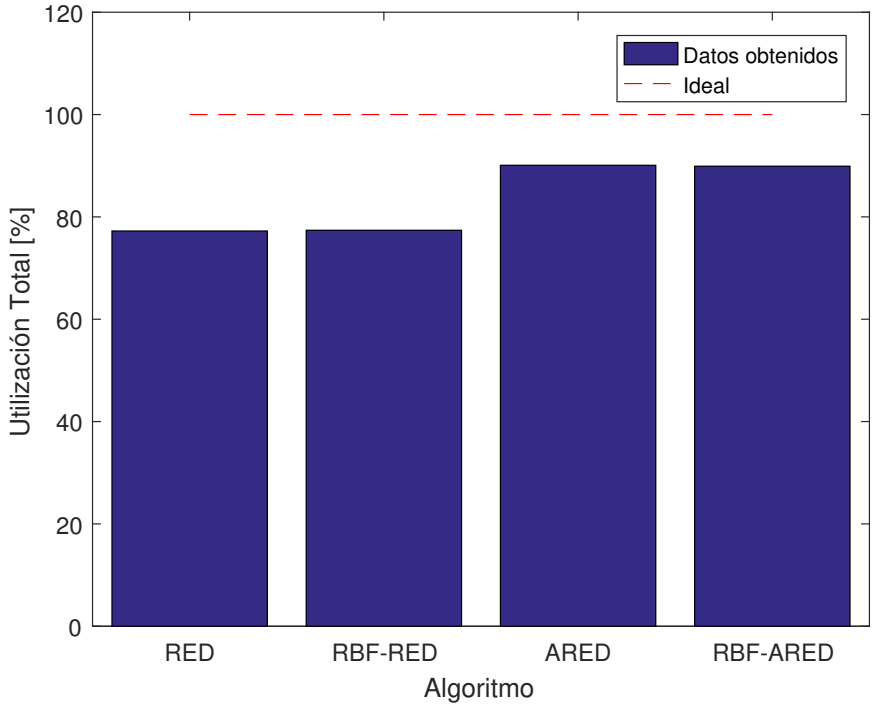


Figura 5.15: Utilización total promedio del canal por algoritmo, en escenario donde compiten 10 conexiones TCP.

de max_p y avg_q. Adicionalmente se entregan como referencia el límite inferior y superior determinados por ARED para determinar en qué momento se debe ajustar el valor de max_p

(ver Figura 2.7). En particular, la idea es aumentar este último (y por consiguiente la agresividad del algoritmo) si el tamaño de la cola supera el límite superior, y viceversa. En esta figura se aprecia que el valor de max_p no cambia mucho desde el instante inicial, y de hecho converge al valor mínimo impuesto por ARED. Esto se atribuye a que la agresividad del algoritmo mientras se utiliza el valor mínimo de max_p, es lo suficientemente alta como para mantenerse por debajo del rango objetivo del algoritmo.

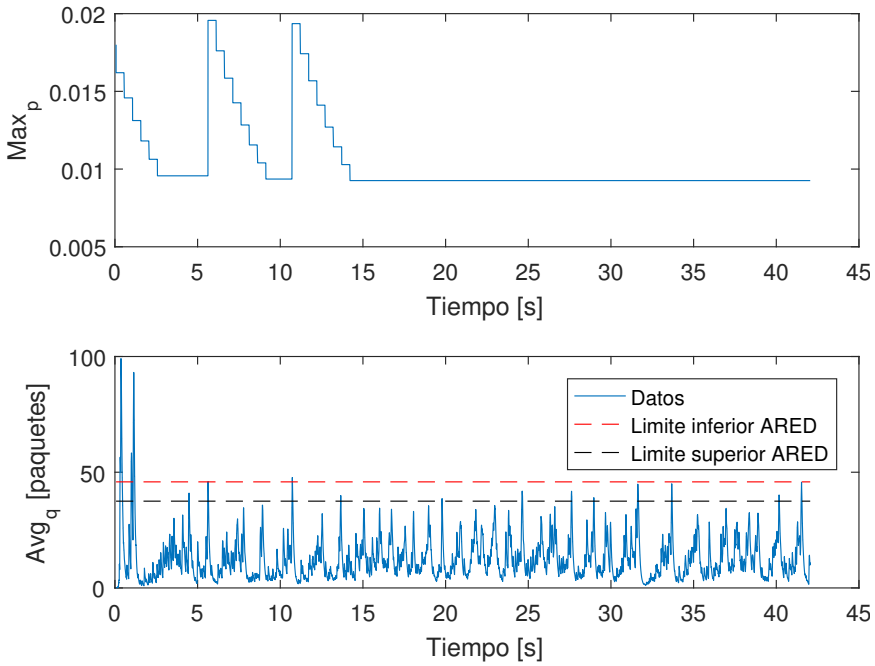


Figura 5.16: Evolución de max_p y avg_q en el tiempo, cuando se hace uso de RBF-ARED.

Un aspecto que se debe volver a destacar a partir de lo visto en los resultados de utilización total de este escenario es que no existen cambios significativos en la utilización luego de aplicar la propuesta. En efecto, si se compara la magnitud de esta diferencia con la que se genera debido a la diferencia de agresividad de los algoritmos se puede notar que la primera es aproximadamente 10 veces menor. Por otra parte, el efecto que se observa tampoco es consistente, pues en el caso de RED es un pequeño aumento, mientras que para ARED se observa una leve disminución.

5.2.3. Análisis de la justicia obtenida

Ahora bien, tal como se mencionó en la Sección 4.5.2, se utilizan los datos anteriores para calcular el Fairness Index, con el fin de tener una métrica que nos permita evaluar qué tan justo es el rendimiento de cada algoritmo. En este caso el caso ideal sería una división homogénea del canal (cada uno debería utilizar el 10 % de la capacidad idealmente), y en base a los resultados previos se sabe que la variante que más se acercó a esto fue RBF-ARED. Los resultados para el Fairness Index se presentan en la Figura 5.17, y en ella se corroborar un aumento significativo en el valor del índice tanto para el caso de RED como ARED cuando se hace uso de RBF-AQM.

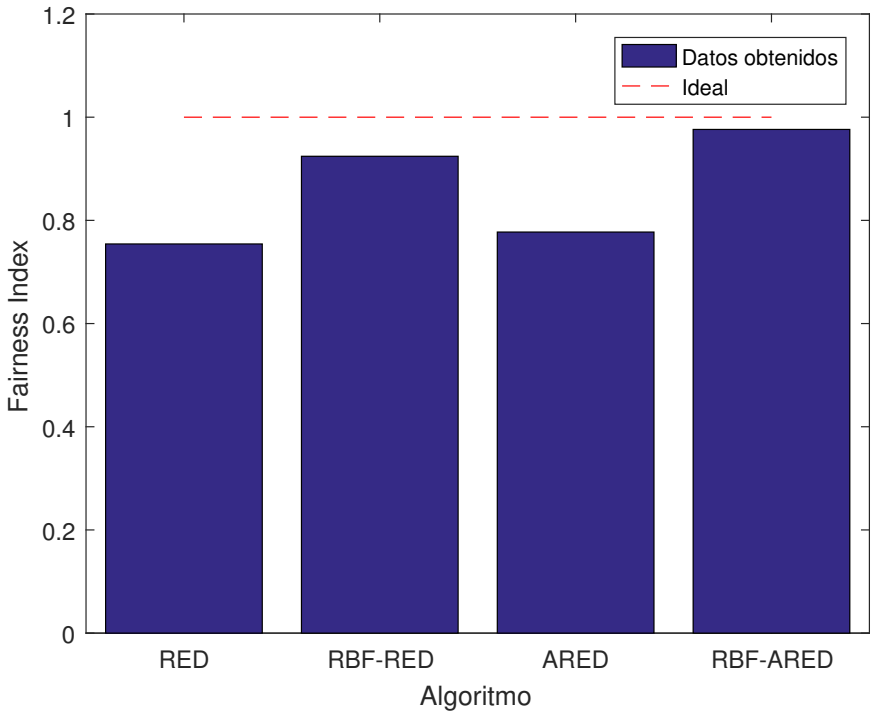


Figura 5.17: Fairness Index obtenido por cada algoritmo, en escenario donde compiten 10 conexiones TCP.

Adicionalmente, tal como indicaban los resultados de utilización promedio por conexión, se aprecia que ARED tiene una ventaja no despreciable con respecto a su nivel de justicia alcanzado. Una forma de explicar esto es considerando lo visto en la Figura 5.16, lo cual permite verificar que la agresividad de ARED es lo suficientemente buena como para mantener a avg_q cerca del valor objetivo de aproximadamente unos 42 paquetes (en base a parámetros definidos para operación de ARED, en particular dependiente de $target\_delay = 5$ [ms]). Lo anterior permite concluir que es probable que RED esté utilizando probabilidades de pérdidas demasiado altas para la magnitud de la congestión que se genera en este escenario, generando en instancias el descarte innecesario de paquetes de conexiones de latencia alta y bajo throughput. En el caso contrario, ARED tiene una mayor sensibilidad en su toma de decisiones en base a una agresividad más apropiada para las condiciones del escenario.

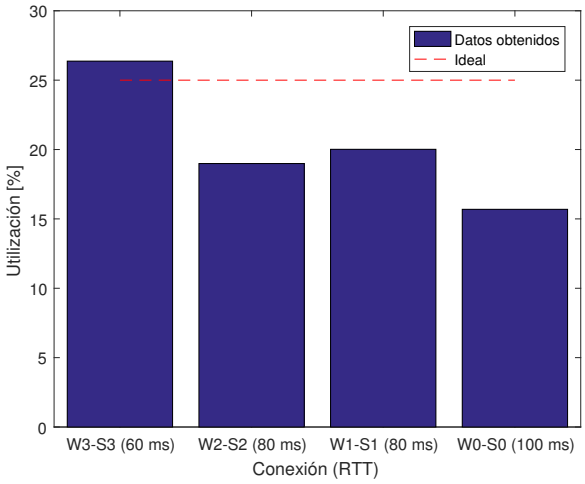
Un aspecto interesante es el contraste entre lo observado en este escenario y el primero, donde no se presentaba un comportamiento consistente de superioridad de un algoritmo respecto a otro. Esto se puede atribuir al hecho de que en dicho caso la diferencia entre las agresividades no era lo suficientemente grande. Un indicio de esto es una diferencia menor a nivel de utilización total, así como también tamaños promedio de la cola que tampoco presentaban diferencias mayores. Lo anterior además tiene sentido con las condiciones de dicho escenario, donde gran parte de la capacidad es dominada por tráfico que no responde frente a pérdidas, lo cual evidentemente eleva la magnitud de la congestión. Esto último también se corroboró mediante el análisis de la evolución del tamaño de la cola en función del tiempo de interarribo.

5.3. Tercer escenario: 4 conexiones TCP y múltiples routers

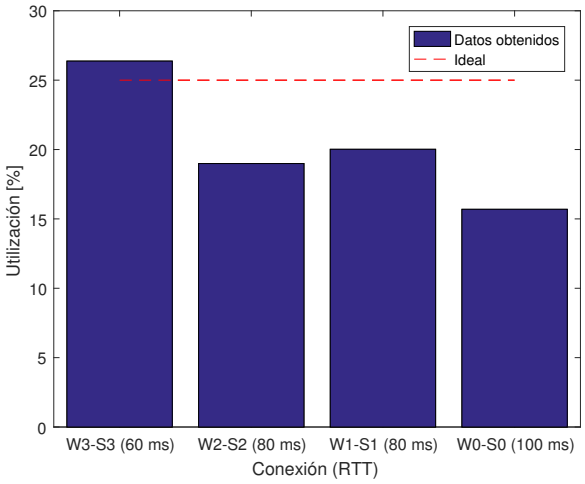
De la misma forma que para el caso del segundo escenario, el primer foco de atención corresponde a la utilización obtenida por cada una de las conexiones. Luego, lo primero que se hace es revisar los resultados de simulaciones individuales para hacerse una idea de lo que puede ocurrir, y luego se analizarán los datos obtenidos luego de hacer un promedio sobre 10 semillas.

5.3.1. Análisis de la utilización por conexión

En primer lugar, se presentan en la Figura 5.18 los resultados obtenidos al hacer uso de RED. En esta se puede apreciar que en general el cambio al aplicar la propuesta no es tan considerable, especialmente en las conexiones cuyo RTT es igual a 80 [ms]. No obstante, sí se puede apreciar una nivelación de la utilización obtenida por el otro par de conexiones. En el caso de los resultados de ARED, los que se muestran en la Figura 5.19, se puede ver que la nivelación de las utilizaciones resulta bastante más efectiva. Sin embargo, dado que estos resultados son de una simulación específica, y considerando también que la diferencia entre los RTTs no es tan alta, los throughputs alcanzados no se distancian tanto como en el escenario anterior.



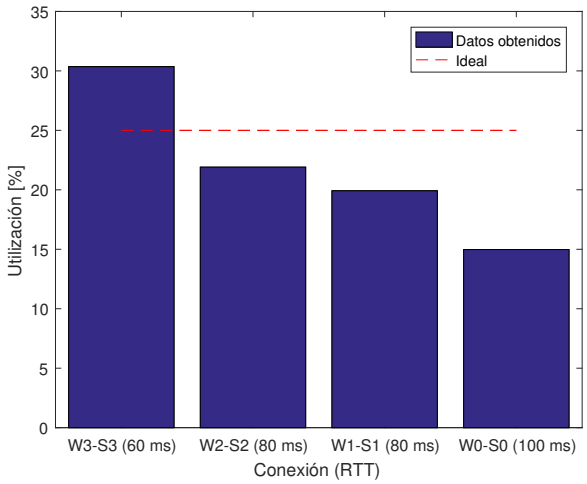
(a) Utilización promedio obtenida por cada conexión cuando se hace uso de RED.



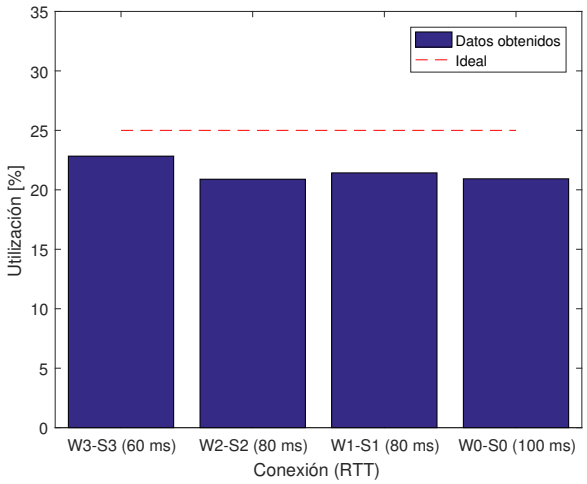
(b) Utilización promedio obtenida por cada conexión cuando se hace uso de RBF-RED.

Figura 5.18: Utilización promedio por conexión en una topología donde compiten 4 conexiones TCP, cuando se hace uso de RED y RBF-RED.

Los resultados obtenidos al promediar sobre 10 semillas se presentan en la Figura 5.20, donde se puede notar que existe un patrón practicamente idéntico que siguen las curvas de cada algoritmo. Este se caracteriza por una utilización que domina, perteneciente a la conexión de RTT = 60 [ms], seguido por una utilización menor y practicamente idéntica para ambas conexiones cuyo RTT es de 80 [ms], y finalmente se observa una caída hacia



(a) Utilización promedio obtenida por cada conexión cuando se hace uso de ARED.



(b) Utilización promedio obtenida por cada conexión cuando se hace uso de RBF-ARED.

Figura 5.19: Utilización promedio por conexión en una topología donde compiten 4 conexiones TCP, cuando se hace uso de ARED y RBF-ARED.

la menor de las utilizaciones, la cual es alcanzada por la conexión de RTT = 100 [ms]. La principal diferencia radica en la nivelación de los extremos que se da cuando se hace uso de la propuesta.

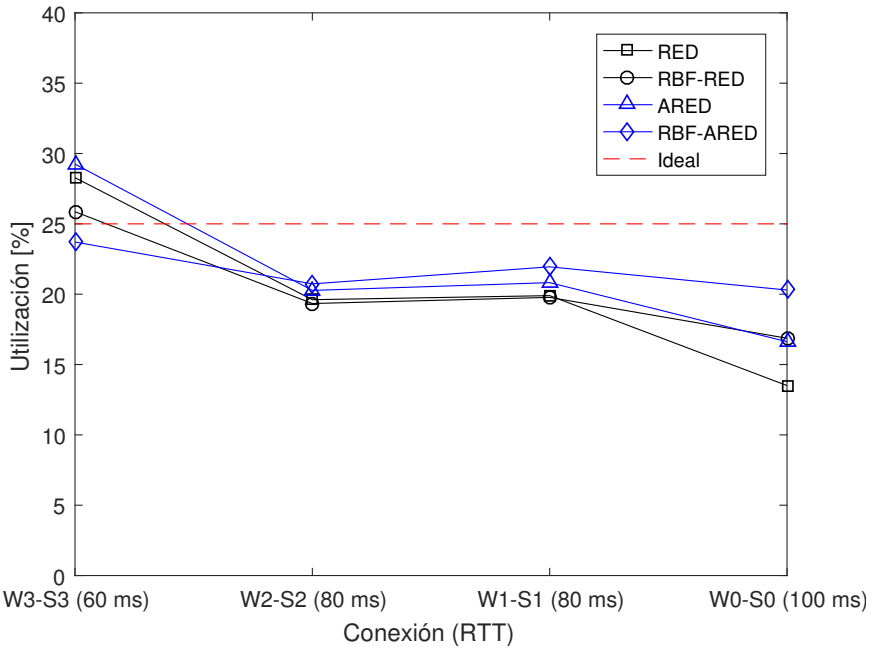


Figura 5.20: Utilización promedio obtenida por cada conexión (cuyo RTT se presenta entre paréntesis), en escenario donde compiten 4 conexiones TCP. El promedio se obtiene a partir de 10 simulaciones independientes.

No obstante, si se inspecciona con cuidado el gráfico se puede notar un caso particularmente interesante cuando se comparan ARED y RBF-RED. Esto, pues en general las utilizaciones son muy similares, con la única excepción de la utilización de la conexión de RTT = 60 [ms].

De tal forma, a pesar de que los resultados de RBF-RED presentan un mayor equilibrio (y por lo tanto probablemente muestren un mejor resultado de fairness), la utilización total será menor. Luego, este resulta un caso particular donde una agresividad alta, la cual genera problemas a nivel de utilización, es capaz de transformar un resultado con buena justicia en uno ineficiente, y de peor rendimiento que un algoritmo como ARED que no hace uso de la propuesta. Esto permite darle aun más relevancia al uso de una agresividad apropiada en un algoritmo de AQM, lo cual es alcanzable solo por propuestas adaptativas como ARED.

Adicionalmente, estos resultados permiten dar cuenta de que no existe una mayor diferencia en cuanto a la utilización que obtienen las conexiones del mismo RTT, lo cual es un indicio de que la congestión en $R3$ debe ser muy baja e infrecuente, ya que no muestra ningún efecto considerable en los resultados.

Finalmente, lo anterior permite probar un punto mencionado en la Sección 3.2, ya que aquí se muestra claramente que a pesar de que el factor de ajuste es unitario para la conexión de RTT más bajo, las pérdidas efectivas disminuyen, lo cual se demuestra por el aumento de la utilización. Lo mismo se puede apreciar de forma menos drástica en la evolución que se genera sobre las conexiones de RTT intermedio, donde las utilizaciones se mantienen constante (RED) o bien aumentan levemente (ARED), aún cuando el factor de ajuste de la propuesta es mayor a 1, aumentando las probabilidades de pérdidas del algoritmo original. De la misma manera, no parece que el hecho de que todos los RTTs estén a un mismo lado de la referencia (todos son menores o iguales a 100 [ms]) genere ningún tipo de complicación para lograr un buen fairness.

5.3.2. Análisis de la utilización total

Los resultados de utilización total presentados en la Figura 5.21 son muy similares a los encontrados en el segundo escenario. En efecto, nuevamente se aprecia una diferencia clara en el nivel de utilización que logran ARED y RED, y no se observa un efecto notable cuando se considera el uso de la propuesta.

Luego, basado en los resultados de utilización por conexión y lo observado en el escenario anterior, se considera que lo más probable es que los resultados de Fairness Index revelen una diferencia no menor entre lo que se puede lograr mediante RED y ARED.

5.3.3. Análisis de la justicia obtenida

Los resultados del Fairness Index obtenido se presentan en la Figura 5.22 se puede ver el rendimiento a nivel de fairness. Aquí es posible notar por una parte nuevamente que la propuesta logra mejorar la distribución equitativa de las utilizaciones. Además, al igual que en 5.2.3, se muestra que ARED logra niveles de justicia mejores a los que alcanza RED. De hecho, llama la atención nuevamente el caso mencionado en el análisis de la Figura 5.20, ya que al comparar ARED y RBF-RED se aprecia que la diferencia en fairness no es muy significativa, lo cual tiene sentido considerando que el nivel de utilización de 3 conexiones es

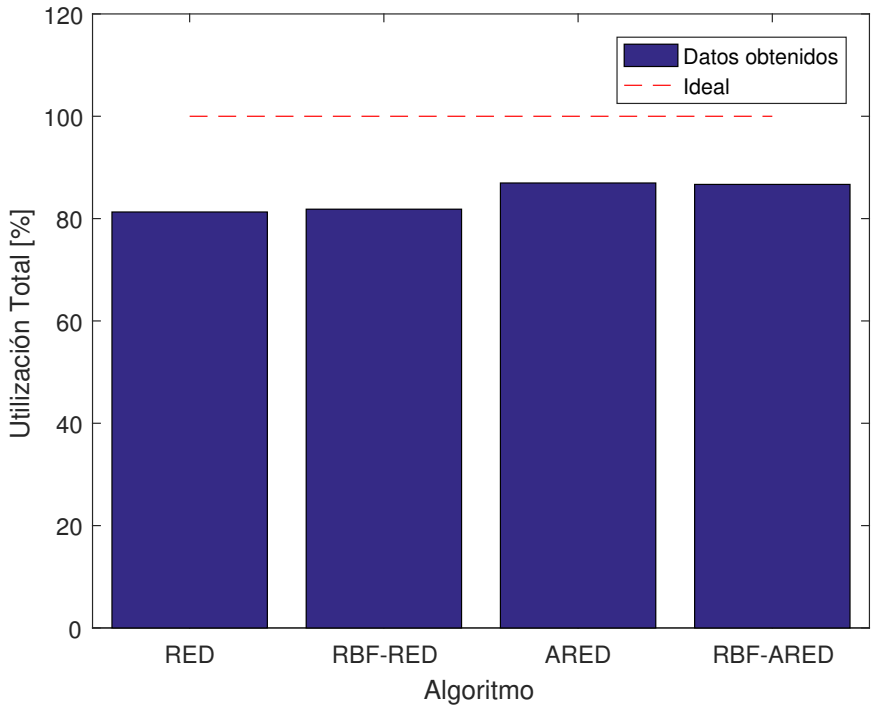


Figura 5.21: Utilización total promedio del canal por algoritmo, en escenario donde compiten 4 conexiones TCP.

muy similar cuando se hace uso de uno o el otro algoritmo.

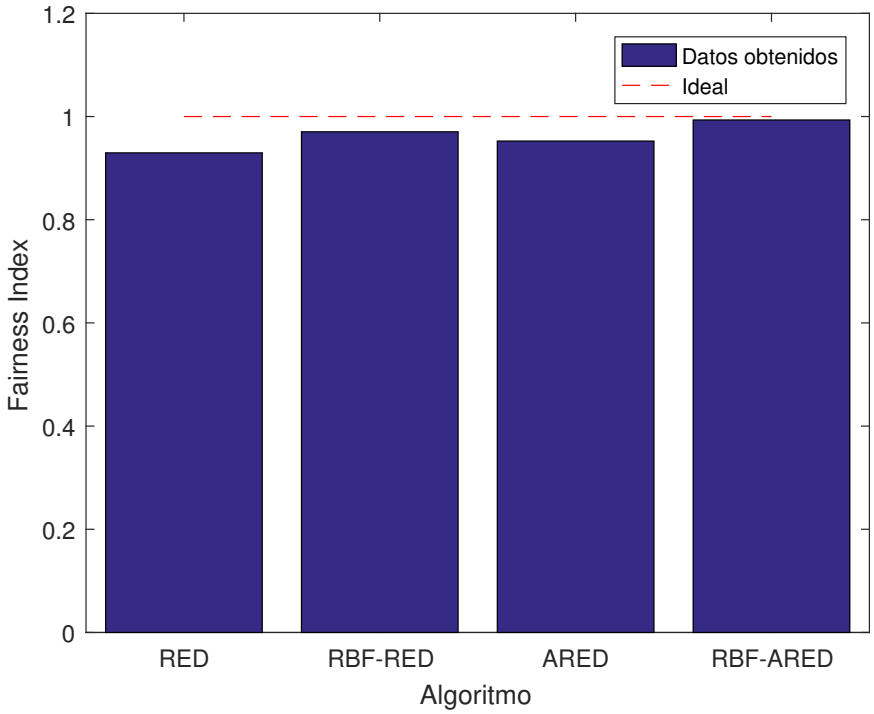


Figura 5.22: Fairness Index obtenido por cada algoritmo, en escenario donde compiten 4 conexiones TCP.

Un último detalle importante tiene que ver con los valores obtenidos para esta métrica,

los cuales son claramente mayores a los de la Figura 5.17. Se debe recalcar que esto tiene sentido, pues el sistema es naturalmente más justo debido a la cercanía de los RTTs de las conexiones que lo componen.

Capítulo 6

Conclusiones y Trabajo Futuro

La revisión del estado del arte permite comprobar que la actividad en el campo del control de congestión a nivel de TCP se ha mantenido relativamente alta en los últimos años. No obstante, existe una prevalencia clara de CUBIC como el protocolo dominante, el cual ha ganado suficiente aceptación como para volverse el protocolo por defecto en múltiples sistemas operativos, a pesar de ser un protocolo diseñado hace más de una década. Por otro lado, respecto a AQM la actividad a nivel de investigación también ha logrado mantenerse relativamente constante en las últimas dos décadas, aunque su magnitud es significativamente menor a la alcanzada al nivel de TCP. No obstante, en el último tiempo pareciera existir un nuevo interés por parte de la comunidad científica, lo cual se refleja en múltiples RFCs en los últimos cinco años y la formación de una comunidad en torno al fenómeno conocido como bufferbloat.

En cuanto al problema del RTT fairness, a nivel de capa de transporte este ha sido abordado parcialmente gracias a CUBIC. No obstante, este protocolo no tiene como objetivo mejorar el rendimiento que obtiene TCP tradicional en este aspecto, y se ha demostrado que el sesgo en contra de las conexiones de latencia alta sigue siendo significativo haciendo uso de él. Si bien existen otros protocolos de transporte como Hybla y H-TCP, los cuales pueden lograr un mejor rendimiento en este sentido, esta propiedad por si sola no contribuye significativamente a la posibilidad de reemplazar el uso de CUBIC en el corto plazo. Adicionalmente, cuando la congestión es lo suficientemente alta, el comportamiento de la mayoría de los protocolos de transporte diseñados para enlaces de alta velocidad converge a TCP tradicional, con el fin de mantener un buen nivel de TCP friendliness. Esto dificulta abordar la injusticia debido a distintos niveles de latencia, lo cual es propio del comportamiento de TCP tradicional. Considerando lo anterior, se considera que existe mucho más potencial en abordar el problema desde el punto de vista de AQM. Esto permitiría no solo complementar a la capa de transporte en la eliminación de este sesgo, sino que adicionalmente permite atacar el problema del bufferbloat, apuntando de tal forma a un Internet más justo y con menor latencia.

El algoritmo propuesto se basa en el uso de la latencia de las conexiones (caracterizada por su RTT) para proveer un trato más justo en situaciones donde compiten conexiones con distintos niveles de latencia. Actualmente existen distintas posturas y factores a considerar

respecto al tema del fairness, lo cual se debe en gran parte a posibles conflictos entre este objetivo y otros aspectos relevantes tales como la buena utilización de los recursos de la red. Esto ha derivado en múltiples criterios y métricas para la evaluación del fairness, y en el caso de este trabajo se adoptan una postura y métrica comúnmente utilizadas en la literatura. En particular, se asocia la justicia a una distribución equitativa de la capacidad del cuello de botella y se hace uso del Fairness Index propuesto por R.Jain.

Los resultados obtenidos mediante simulaciones muestran que el nivel de fairness alcanzado por RED y ARED en general es similar, lo cual tiene sentido considerando que la única diferencia está asociada al nivel de agresividad alcanzado por ambos. No obstante, los resultados indican que si existe una diferencia signficativa en este último aspecto, entonces se aprecia que el algoritmo que posee una agresividad menor (pero lo suficientemente alta como para controlar el tamaño de la cola en torno al objetivo) logra brindar una distribución más nivelada de los throughputs, lo cual se observó en los dos de los tres escenarios utilizados, donde ARED logró una mejor utilización total y un mejor Fairness Index. Por otra parte, se observa que el algoritmo propuesto es capaz de mejorar de forma significativa el nivel de fairness de forma consistente en todos los escenarios analizados, mejorando los resultados en este aspecto en al menos un 20 %. En particular, este efecto se observa independientemente de si el algoritmo utiliza como referencia a RED o ARED. Adicionalmente, se comprueba que los cambios a nivel de utilización total son menores al 0.25 % al hacer uso del algoritmo propuesto, demostrando que la agresividad promedio del algoritmo propuesto se mantiene relativamente constante con respecto al algoritmo de referencia, de forma independiente de la distribución de RTTs de las conexiones, la cual cambia en cada uno de los escenarios.

Respecto a la implementación del algoritmo se consideran múltiples alternativas para llevar el dato del RTT a los routers. Dos de ellas resultan muy atractivas, debido a la sencillez de su implementación y el hecho de que son altamente escalables. La primera de estas propone la inclusión de una nueva opción para la cabecera del paquete IP, la cual se utilizaría para transmitir el valor estimado del RTT desde la capa de transporte de los usuarios hasta los routers. Una desventaja de esta alternativa es la posibilidad de la alteración maliciosa del campo con el fin de obtener menos pérdidas, sin embargo esto es comparable a lo que se puede lograr haciendo uso de una versión de TCP modificada especialmente para obtener una mayor tasa de transmisión, y adicionalmente el efecto podría ser mitigado con la definición de cotas para las probabilidades de pérdidas. La segunda plantea el uso de un valor estimado del RTT en base a las direcciones IP de origen y destino. Si bien esta alternativa tiene la ventaja de no necesitar un cambio a la especificación actual del formato del paquete IP, el dato utilizado podría distanciarse de forma significativa del valor real. Esto último se puede deber a factores como el uso de VPNs, o bien una estimación deficiente por parte del ISP en función de las direcciones IPs.

Considerando que los resultados obtenidos por el algoritmo propuesto fueron favorables a nivel de fairness, que no se observaron efectos negativos a nivel de utilización total, y que existen alternativas escalables y sencillas para su implementación, se considera que el objetivo general de la tesis se pudo cumplir de forma exitosa y que sería valioso continuar la investigación de esta propuesta.

Respecto a un posible trabajo futuro, existen distintos aspectos abordables que cuales

permitirían generalizar la validez del algoritmo, tales como la exploración de escenarios con un mayor número de conexiones, un estudio del efecto ocasionado por el cambio de parámetros como RTT_{ref} y λ, la consideración de otros protocolos de referencia tales como CoDel, o incluso algún mecanismo que ajuste el exponente del factor que considera las latencias en función del nivel de congestión. Eso último podría podría contribuir a facilitar el acople con el comportamiento de protocolos como CUBIC, los que logran diferenciar sus ventanas en función de la latencia. Por otra parte, analizar qué sucede con el rendimiento del algoritmo cuando se tiene un conocimiento imperfecto del RTT sería útil para caracterizar la robustez del algoritmo propuesto, y evaluar la calidad de potenciales técnicas y alternativas de implementación.

Bibliografía

[1] J. Chawla and S. Kumari, "Performance evaluation of droptail and random early detection," *International Research Journal of Engineering and Technology (IRJET*, vol. 3, no. 5, pp. 2395–0072, 2016.

[2] W.-C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring red gateway," in *Proceedings of INFOCOM*, pp. 1320–1328, IEEE, 1999.

[3] S. Floyd, R. Gummadi, S. Shenker, *et al.*, "Adaptive red: An algorithm for increasing the robustness of red's active queue management," 2001.

[4] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White Paper*, 2018.

[5] C. Callegari, S. Giordano, M. Pagano, and T. Pepe, "A survey of congestion control mechanisms in linux tcp," in *International Conference on Distributed Computer and Communication Networks*, pp. 28–42, Springer, 2013.

[6] F. Baker and G. Fairhurst, "Ietf recommendations regarding active queue management," tech. rep., 2015.

[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, no. 4, pp. 397–413, 1993.

[8] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, *et al.*, "Recommendations on queue management and congestion avoidance in the internet," tech. rep., 1998.

[9] N. Kuhn, P. Natarajan, N. Khademi, and D. Ros, "Characterization guidelines for active queue management (aqm)," tech. rep., 2016.

[10] D. Lin and R. Morris, "Dynamics of random early detection," in *ACM SIGCOMM computer communication review*, vol. 27, pp. 127–137, ACM, 1997.

[11] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on networking*, vol. 6, no. 4, pp. 362–373, 1998.

[12] C. Zhang, J. Yin, Z. Cai, and W. Chen, "Rred: robust red algorithm to counter low-rate denial-of-service attacks," *IEEE Communications Letters*, vol. 14, no. 5, pp. 489–491,

2010.

[13] K. Zhou, K. L. Yeung, and V. O. Li, "Nonlinear red: A simple yet efficient active queue management scheme," *Computer Networks*, vol. 50, no. 18, pp. 3784–3794, 2006.

[14] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tüxen, M. Welzl, D. Damjanovic, *et al.*, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 619–639, 2016.

[15] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[16] "World data transfer record back in danish hands." `https://www.dtu.dk/english/News/2014/07/Verdensrekord-i-dataoverfoersel-paa-danske-haender-igen`. Accessed: 2019-12-11.

[17] "Demonstration of world record: 159 tb/s transmission over 1045 km with 3-mode fiber." `https://www.nict.go.jp/en/press/2018/04/13-1.html`. Accessed: 2019-12-11.

[18] "Nokia and tim break european record for long-distance data transmission over high-speed network - laying groundwork for 5g." `https://nokia.ly/2YYuZRd`. Accessed: 2019-12-11.

[19] "Caltech computer scientists develop fast protocol to speed up internet." `https://www.caltech.edu/about/news/caltech-computer-scientists-develop-fast-protocol-speed-internet-674`. Accessed: 2019-12-11.

[20] S. Floyd, "Rfc 5166: Metrics for the evaluation of congestion control mechanisms," 2008.

[21] S. Floyd, "Highspeed tcp for large congestion windows," 2003.

[22] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," *ACM SIGCOMM computer communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[23] D. Leith, "H-tcp protocol for high-speed long distance networks," in *Proc. International Workshop on Protocols for Fast Long-Distance Networks, Argonne, Illinois, USA, Feb. 2004*, 2004.

[24] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pp. 1–10, IEEE, 2017.

[25] K. T. J. Song, Q. Zhang, and M. Sridharan, "Compound tcp: A scalable and tcp-friendly congestion control for high-speed networks," *Proceedings of PFLDnet 2006*, 2006.

[26] M. Allman, V. Paxson, and E. Blanton, "Tcp congestion control," tech. rep., 2009.

[27] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM computer communication review*, vol. 18, pp. 314–329, ACM, 1988.

[28] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, and M. Roccetti, "Tcp libra: Derivation, analysis, and comparison with other rtt-fair tcps," *Computer Networks*, vol. 54, no. 14, pp. 2327–2344, 2010.

[29] D. Miras, M. Bateman, and S. Bhatti, "Fairness of high-speed tcp stacks," in *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*, pp. 84–92, IEEE, 2008.

[30] T. Kozu, Y. Akiyama, and S. Yamaguchi, "Improving rtt fairness on cubic tcp," *International Journal of Networking and Computing*, vol. 4, no. 2, pp. 291–306, 2014.

[31] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 2, pp. 133–145, 2000.

[32] W. Bao, V. W. Wong, and V. C. Leung, "A model for steady state throughput of tcp cubic," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pp. 1–6, IEEE, 2010.

[33] Y.-T. Li, D. Leith, and R. N. Shorten, "Experimental evaluation of tcp protocols for high-speed networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 15, no. 5, pp. 1109–1122, 2007.

[34] W.-c. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: A new class of active queue management algorithms," tech. rep., Technical Report CSE-TR-387-99, University of Michigan, 1999.

[35] G. Abbas, S. Manzoor, and M. Hussain, "A stateless fairness-driven active queue management scheme for efficient and fair bandwidth allocation in congested internet routers," *Telecommunication Systems*, vol. 67, no. 1, pp. 3–20, 2018.

[36] I. Cisco, "quality of service solutions configuration guide, release 12.2," *QC-79*, 2010.

[37] S. Floyd, K. Ramakrishnan, and D. L. Black, "The addition of explicit congestion notification (ecn) to ip," 2001.

[38] J. Chu, Y. Cheng, N. Dukkipati, and M. Mathis, "Increasing tcp's initial window," 2013.

[39] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. Addison Wesley, 2011.

[40] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The newreno modification to tcp's fast recovery algorithm," tech. rep., 2012.

[41] E. Blanton, K. Fall, and M. Allman, "A conservative selective acknowledgment (sack)-

based loss recovery algorithm for tcp," 2003.

[42] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, *TCP Vegas: New techniques for congestion detection and avoidance*, vol. 24. ACM, 1994.

[43] A. H. David and G. Armitage, "Improved coexistence and loss tolerance for delay based tcp congestion control," in *IEEE Local Computer Network Conference*, pp. 24–31, IEEE, 2010.

[44] G. Hasegawa, K. Kurata, and M. Murata, "Analysis and improvement of fairness between tcp reno and vegas for deployment of tcp vegas to the internet," in *Proceedings 2000 International Conference on Network Protocols*, pp. 177–186, IEEE, 2000.

[45] D. S. Miller, "Git commit that enabled cubic as the default algorithm for congestion control in linux kernel." `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=597811ec167fa01c926a0957a91d9e39baa30e64`, 2006. Accessed: 2019-08-19.

[46] S. Liu, T. Basar, and R. Srikant, "Tcp-illinois: a loss and delay-based congestion control algorithm for high-speed networks, valuetools' 06: Proceedings of the 1st international conference on performance evaluation methodologies and tools," *Pisa, Italy, October*, 2006.

[47] "Performance enhancements in the next generation tcp/ip stack." `https://docs.microsoft.com/en-us/previous-versions//bb878127(v=technet.10)`. Accessed: 2019-12-17.

[48] D. Havey, "Windows transport converges on two congestion providers: Cubic and ledbat." `https://techcommunity.microsoft.com/t5/Networking-Blog/Windows-Transport-converges-on-two-Congestion-Providers-Cubic/ba-p/339819`, 2018. Accessed: 2019-08-19.

[49] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," in *IEEE INFOCOM 2004*, vol. 4, pp. 2490–2501, IEEE, 2004.

[50] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 63–74, 2011.

[51] G. Marfia, C. Palazzi, G. Pau, M. Gerla, M. Sanadidi, and M. Roccetti, "Tcp libra: Exploring rtt-fairness for tcp," in *International Conference on Research in Networking*, pp. 1005–1013, Springer, 2007.

[52] K. Winstein and H. Balakrishnan, "Tcp ex machina: Computer-generated congestion control," 2013.

[53] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "{PCC}: Re-architecting congestion control for consistent high performance," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pp. 395–408, 2015.

[54] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "{PCC} vivace: Online-learning congestion control," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pp. 343–356, 2018.

[55] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, p. 50, 2016.

[56] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, *et al.*, "B4: Experience with a globally-deployed software defined wan," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 3–14, ACM, 2013.

[57] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "Tcp congestion avoidance algorithm identification," *IEEE/Acm Transactions On Networking*, vol. 22, no. 4, pp. 1311–1324, 2013.

[58] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A survey on recent advances in transport layer protocols," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3584–3608, 2019.

[59] "Pantheon Project Website." `https://pantheon.stanford.edu/`. Accessed: 2019-08-19.

[60] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, pp. 731–743, 2018.

[61] V. Jacobson, "Berkeley tcp evolution from 4.3-tahoe to 4.3-reno," *Proceedings of 18th IETF*, vol. 365, 1990.

[62] J. C. Hoe, *Start-up dynamics of TCP's congestion control and avoidance schemes*. PhD thesis, Massachusetts Institute of Technology, 1995.

[63] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "Tcp selective acknowledgment options," tech. rep., RFC 2018, 1996.

[64] Y. R. Yang and S. S. Lam, "General aimd congestion control," in *Proceedings 2000 International Conference on Network Protocols*, pp. 187–198, IEEE, 2000.

[65] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (bic) for fast long-distance networks," in *Ieee Infocom*, vol. 4, pp. 2514–2524, INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 2004.

[66] A. Baiocchi, A. P. Castellani, and F. Vacirca, "Yeah-tcp: yet another highspeed tcp," in *Proc. PFLDnet*, vol. 7, pp. 37–42, 2007.

[67] G. Appenzeller, I. Keslassy, and N. McKeown, *Sizing router buffers*, vol. 34. ACM, 2004.

[68] E. S. Hashem, "Analysis of random drop for gateway congestion control," tech. rep.,

MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1989.

[69] J. Postel, "Internet control message protocol," 1981.

[70] J. Nagle, "On packet switches with infinite storage," *IEEE transactions on communications*, vol. 35, no. 4, pp. 435–438, 1987.

[71] R. Jain, K. Ramakrishnan, and D.-M. Chiu, "Congestion avoidance in computer networks with a connectionless network layer," *arXiv preprint cs/9809094*, 1998.

[72] A. Mankin, "Random drop congestion control," in *ACM SIGCOMM Computer Communication Review*, vol. 20, pp. 1–7, ACM, 1990.

[73] V. Rosolen, O. Bonaventure, and G. Leduc, "A red discard strategy for atm networks and its performance evaluation with tcp/ip traffic," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 3, pp. 23–43, 1999.

[74] S. Floyd, "Red: Discussions of setting parameters, november 1997," *URL http://www. aciri. org/floyd/REDparameters. txt*, 1997.

[75] V. Paxson, "End-to-end internet packet dynamics," in *ACM SIGCOMM Computer Communication Review*, vol. 27, pp. 139–152, ACM, 1997.

[76] M. Talau, "Red queue disc ns3 model source code." `https://github.com/nsnam/ns-3-dev-git/blob/master/src/traffic-control/model/red-queue-disc.cc`, 2018. Accessed: 2019-08-19.

[77] T. J. Ott, T. Lakshman, and L. H. Wong, "Sred: stabilized red," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 3, pp. 1346–1355, IEEE, 1999.

[78] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Computer networks*, vol. 36, no. 2-3, pp. 203–235, 2001.

[79] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "Rem: Active queue management," *IEEE network*, vol. 15, no. 3, pp. 48–53, 2001.

[80] C. Zhang, J. Yin, and Z. Cai, "Rsfb: a resilient stochastic fair blue algorithm against spoofing ddos attacks," in *2009 9th International Symposium on Communications and Information Technology*, pp. 1566–1567, IEEE, 2009.

[81] R. Pan, B. Prabhakar, and K. Psounis, "Choke-a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 2, pp. 942–951, IEEE, 2000.

[82] A. Eshete and Y. Jiang, "Generalizing the choke flow protection," *Computer Networks*, vol. 57, no. 1, pp. 147–161, 2013.

[83] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "On designing improved controllers for aqm routers supporting tcp flows," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3, pp. 1726–1734, IEEE, 2001.

[84] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "Pie: A lightweight control scheme to address the bufferbloat problem," in *High Performance Switching and Routing (HPSR), 2013 IEEE 14th International Conference on*, pp. 148–155, IEEE, 2013.

[85] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional integral controller enhanced (pie): A lightweight control scheme to address the bufferbloat problem," tech. rep., 2017.

[86] G. White and R. Pan, "Active queue management (aqm) based on proportional integral controller enhanced pie) for data-over-cable service interface specifications (docsis) cable modems," tech. rep., 2017.

[87] "Bufferbloat Community Website." `https://www.bufferbloat.net/projects/`. Accessed: 2019-08-19.

[88] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Computing*, no. 3, p. 96, 2011.

[89] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[90] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled delay active queue management," tech. rep., 2018.

[91] D. Taht, J. Gettys, E. Dumazet, T. Hoeiland-Joergensen, E. Dumazet, P. McKenney, J. Gettys, T. Hoeiland-Joergensen, and P. McKenney, "The flow queue codel packet scheduler and active queue management algorithm," 2018.

[92] H. Adhari, E. P. Rathgeb, A. Singh, A. Könsgen, and C. Goerg, "Transport layer fairness revisited," in *2015 13th International Conference on Telecommunications (ConTEL)*, pp. 1–8, IEEE, 2015.

[93] A. Tang, J. Wang, and S. H. Low, "Counter-intuitive throughput behaviors in networks under end-to-end control," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 2, pp. 355–368, 2006.

[94] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[95] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," in *ATM Forum contribution*, vol. 99, 1999.

[96] E. L. Hahne, "Round robin scheduling for fair flow control in data communication networks.," tech. rep., MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND DECISION SYSTEMS, 1986.

[97] B. Suter, T. Lakshman, D. Stiliadis, and A. K. Choudhury, "Design considerations for supporting tcp with per-flow queueing," in *Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98*, vol. 1, pp. 299–306, IEEE, 1998.

[98] K. Bharath-Kumar and J. Jaffe, "A new approach to performance-oriented flow control," *IEEE Transactions on Communications*, vol. 29, no. 4, pp. 427–435, 1981.

[99] B. Radunovic and J.-Y. Le Boudec, "A unified framework for max-min and min-max fairness with applications," *IEEE/ACM Transactions on networking*, vol. 15, no. 5, pp. 1073–1083, 2007.

[100] F. Kelly, "Mathematical modelling of the internet," in *Mathematics unlimited—2001 and beyond*, pp. 685–702, Springer, 2001.

[101] S. Huaizhou, R. V. Prasad, E. Onur, and I. Niemegeers, "Fairness in wireless networks: Issues, measures and challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 5–24, 2013.

[102] R. Ware, M. K. Mukerjee, S. Seshan, and J. Sherry, "Beyond jain's fairness index: Setting the bar for the deployment of congestion control algorithms," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pp. 17–24, ACM, 2019.

[103] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.

[104] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of dctcp: stability, convergence, and fairness," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pp. 73–84, ACM, 2011.

[105] S. Molnár, B. Sonkoly, and T. A. Trinh, "A comprehensive tcp fairness analysis in high speed networks," *Computer Communications*, vol. 32, no. 13-14, pp. 1460–1484, 2009.

[106] C. Caini and R. Firrincieli, "Tcp hybla: a tcp enhancement for heterogeneous networks," *International journal of satellite communications and networking*, vol. 22, no. 5, pp. 547–566, 2004.

[107] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, *et al.*, "Bbr: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[108] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Rfc 2616: Hypertext transfer protocol–http/1.1," 1999.

[109] T. Lakshman and U. Madhow, "The performance of tcp/ip for networks with high bandwidth-delay products and random loss," *IEEE/ACM transactions on networking*, vol. 5, no. 3, pp. 336–350, 1997.

[110] M. Belshes, "More bandwidth doesn't matter (much)." `https://github.com/nsnam/ns-3-dev-git/blob/master/src/traffic-control/model/red-queue-disc.cc`, 2010. Accessed: 2019-11-25.

[111] M. Sargent, M. Allman, and V. Paxson, "Computing tcp's retransmission timer," *Computing*, 2011.

[112] G. Renker and G. Fairhurst, "Sender rtt estimate option for the datagram congestion control protocol (dccp)," *RFC 6323*, 2011.

[113] P. Moravek, D. Komosny, R. Burget, J. Sveda, T. Handl, and L. Jarosova, "Study and performance of localization methods in ip based networks: Vivaldi algorithm," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 351–367, 2011.

[114] R. Landa, R. G. Clegg, J. T. Araújo, E. Mykoniati, D. Griffin, and M. Rio, "Measuring the relationships between internet geography and rtt," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–7, IEEE, 2013.

[115] "Internet protocol version 4 (ipv4) parameters." `https://www.iana.org/assignments/ip-parameters/ip-parameters.xhtml`. Accessed: 2019-08-06.

[116] "Riverbed modeler: The fastest discrete event-simulation engine for analyzing and designing communication networks." `https://www.riverbed.com/products/steelcentral/steelcentral-riverbed-modeler.html`. Accessed: 2019-12-11.

[117] "Matlab product webpage." `https://www.mathworks.com/products/matlab.html`. Accessed: 2019-12-11.

[118] L. Zhang, S. Shenker, and D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffc," in *Proceedings of ACM Sigcomm*, vol. 91, pp. 133–147, 1991.

[119] T. R. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On improving the fairness of tcp congestion avoidance," in *IEEE GLOBECOM 1998 (Cat. NO. 98CH36250)*, vol. 1, pp. 539–544, IEEE, 1998.

Apéndice A

Efecto a nivel de QoE cuando se logra justicia

Sea un conjunto de N conexiones, donde el nivel de latencia de la conexión i se caracteriza por RTT_i. Se asume un sistema en que los throughputs de las conexiones siguen el modelo de [31], donde cada una de las conexiones experimenta una tasa de pérdidas constante e igual a p, y la diferencia entre los throughputs se debe a las diferencias en el valor del RTT. Luego, sea th_i el throughput de la conexión i bajo estas condiciones, este queda dado por la expresión de la Ecuación A.1, donde $W(p)$ corresponde a la ventana de congestión como función de la tasa de pérdidas.

$$th_i = \frac{W(p)}{RTT_i} \tag{A.1}$$

Considerar ahora el caso donde se logra justicia, donde, si se asume que utilización se mantiene constante, se tiene que el throughput alcanzado por todas las conexiones será igual al throughput promedio de aquellos alcanzados en el caso inicial. Luego el throughput promedio se puede obtener mediante la expresión de la Ecuación A.2

$$\bar{th} = \sum_{i=1}^{N} th_i \tag{A.2}$$

Cuando se habla sobre QoE, lo que interesa comparar es el cambio relativo entre el caso inicial, el cual es injusto, y el caso donde se obtiene justicia. Luego, sea $\Delta_i = \bar{th} - th_i$, entonces el cambio relativo para la conexión i queda dado por $\frac{\Delta_i}{th_i}$. Dicho esto, esta sección tiene como objetivo demostrar que la suma de estos cambios siempre es positiva, lo cual quiere decir que el QoE promedio no puede disminuir. Lo anterior se reduce a probar la Desigualdad A.3.

$$\sum_{i=1}^{N} \frac{\Delta_i}{th_i} \geq 0 \tag{A.3}$$

En primer lugar, es interesante notar que el cambio relativo $\frac{\Delta_i}{th_i}$ se puede describir como una función lineal de RTT_i:

$$\frac{\Delta_i}{th_i} = f(RTT_i) = mRTT_i + n \tag{A.4}$$

$$= \frac{\bar{th}}{th_i} - 1 \tag{A.5}$$

$$= \frac{\bar{th}}{W(p)} RTT_i - 1 \tag{A.6}$$

Haciendo uso de esta relación lineal, se desarrollará la suma de los cambios relativos para demostrar que esta no puede ser negativo:

$$\sum_{i=1}^{N} \frac{\Delta_i}{th_i} = \sum_{i=1}^{N} \Big(\frac{\bar{th}}{W(p)} RTT_i - 1 \Big) \tag{A.7}$$

$$= \sum_{i=1}^{N} \Big(\frac{\bar{th}}{W(p)} RTT_i \Big) - N \tag{A.8}$$

$$= \sum_{i=1}^{N} \Big(\frac{\bar{th}}{W(p)} RTT_i \Big) - N \tag{A.9}$$

$$= \frac{\bar{th}}{W(p)} \sum_{i=1}^{N} RTT_i - N \tag{A.10}$$

$$= \frac{1}{W(p)} \Big(\frac{1}{N} \sum_{j=1}^{N} \frac{W(p)}{RTT_j} \Big) \Big(\sum_{i=1}^{N} RTT_i \Big) - N \tag{A.11}$$

$$= \frac{1}{N} \left[\Big(\sum_{j=1}^{N} \frac{1}{RTT_j} \Big) \Big(\sum_{i=1}^{N} RTT_i \Big) \right] - N \tag{A.12}$$

$$= \frac{1}{N} \left[\frac{th_1}{th_1} + ... + \frac{th_N}{th_N} + \Big(\sum_{j=1}^{N} \sum_{i=1, i \neq j}^{N} \frac{RTT_i}{RTT_j} \Big) \right] - N \tag{A.13}$$

$$= \frac{1}{N} \left[N + \Big(\sum_{j=1}^{N} \sum_{i=1, i \neq j}^{N} \frac{RTT_i}{RTT_j} \Big) \right] - N \tag{A.14}$$

$$= \frac{1}{N} \left[N + \Big(\sum_{k \in K} P_k \Big) \right] - N \tag{A.15}$$

Para poder finalizar la demostración, es útil notar que el número total de elementos de la sumatoria en A.14 es igual al doble de $\binom{N}{2}$. Luego, el término P_k corresponde a uno de la forma $\left(\frac{RTT_i}{RTT_j} + \frac{RTT_j}{RTT_i} \right)$, y por lo tanto $|K| = \binom{N}{2}$. Dicho esto, se debe considerar que es posible demostrar la siguiente desigualdad:

$$\frac{x}{y} + \frac{y}{x} = \frac{x^2 + y^2}{xy} \geq 2 \qquad , \quad \forall x, y > 0 \tag{A.16}$$

Luego, haciendo uso de la Desigualdad A.16, es posible acotar cada uno de los elementos de la sumatoria en A.15:

$$\sum_{i=1}^{N} \frac{\Delta_i}{th_i} = \frac{1}{N}\left[N + \Big(\sum_{k \in K} P_k\Big)\right] - N \tag{A.17}$$

$$\geq \frac{1}{N}\left[N + \Big(\sum_{k \in K} 2\Big)\right] - N \tag{A.18}$$

$$= \frac{1}{N}\left[N + 2\binom{N}{2}\right] - N \tag{A.19}$$

$$= \frac{1}{N}\left[N + 2 \times \frac{1}{2}N(N-1)\right] - N \tag{A.20}$$

$$= \frac{1}{N}\left[N + N^2 - N\right] - N \tag{A.21}$$

$$= \frac{1}{N}\left[N^2\right] - N \tag{A.22}$$

$$= N - N = 0 \qquad \blacksquare \tag{A.23}$$

Es decir, se ha demostrado que, considerando los supuestos mencionados, la QoE promedio no puede ser negativa cuando se logra una distribución perfectamente equitativa de los throughputs. Lo anterior se logra de forma independiente de la distribución de los niveles de latencia de las conexiones consideradas, pues en el análisis no se hace uso de los valores particulares de los RTTs.

Apéndice B

Códigos utilizados y proyecto de Riverbed Modeler

Para tener acceso al código que es utilizado para implementar el algoritmo en el simulador, se debe ingresar al siguiente repositorio de github:

`https://github.com/FelFred/RBF-AQM/`

Si se desea obtener más información sobre la implementación en el simulador o bien los códigos utilizados para procesar la información, por favor contactar a la dirección de correo que se presenta a continuación:

`Email: felipe.fredes@ug.uchile.cl`