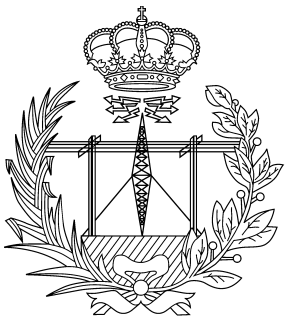
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Máster

ANÁLISIS DE POLÍTICAS DE SCHEDULING MULTI-FLUJO EN QUIC SOBRE CONECTIVIDAD LEO

(Assessment of QUIC multi-stream scheduling over LEO connectivity)

Para acceder al Título de

Máster Universitario en Ingeniería de Telecomunicación

Autor: Fátima Khan Blanco

Julio - 2023

Máster Universitario en Ingeniería de Telecomunicación

CALIFICACIÓN DEL TRABAJO FIN DE MÁSTER

Realizado por: Fátima Khan Blanco
Directores del TFM: Luis Francisco Diez Fernández, Ramón Agüero Calvo
Título: "ANÁLISIS DE POLÍTICAS DE SCHEDULING MULTI-FLUJO EN QUIC SOBRE CONECTIVIDAD LEO"
Title: "Assessment of QUIC multi-stream scheduling over LEO connectivity "

Presentado a examen el día: 17 de julio de 2023

para acceder al Título de

Máster Universitario en Ingeniería de Telecomunicación

Composición del tribunal:
Presidente (Apellidos, Nombre): Marta García Arranz
Secretario (Apellidos, Nombre): Jorge Lanza Calderón
Vocal (Apellidos, Nombre): José Ángel Irastorza Teja

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFM
(solo si es distinto del Secretario)

V^{o} B^{o} del Subdirector

Trabajo Fin de Máster N^{o}
(a asignar por Secretaría)

Índice general

Índice de figuras

Índice de tablas

Acrónimos

3GPP 3rd Generation Partnership Project.

ADMM Alternating Direction Method of Multipliers.

B5G Beyond 5G.

BBR Bottleneck Bandwidth and Round-trip propagation time.

BDP Bandwidth-Delay Product.

BP Back Pressure.

BS Base Station.

CECLS Cloud and Edge Computing LEO Satellites.

CU Central Unit.

DASH Dynamic Adaptive Streaming over HTTP.

DNN Deep Neural Network.

DS Deep-Shadowing.

DU Distributed Unit.

E2E End to End.

eMBB Enhanced Mobile Broadband.

FIFO First In, First Out.

FQ Fair Queuing.

GEO Geoestacionary Equatorial Orbit.

gnB Next Generation NoedB.

HAP High Altitude Platformn.

HOL Head-of-line.

HTTP Hypertext Transfer Protocol.

IoT Internet of Things.

IP Internet Protocol.

ISL Inter-Satellite Link.

ISM Industrial, Scientific, and Medical.

ITU International Telecommunication Union.

JSON JavaScript Object Notation.

LEO Low Earth Orbit.

LMS Land Mobile-Satellite.

LoS Line of Sight.

LSEC LEO Satellite Edge Computing.

MAC Medium Access Control.

MEC Mobile Edge Computing.

MEO Medium Earth Orbit.

mMTC Massive Machine-Type Communications.

MNO Mobile Network Operator.

MPQUIC Multi-Path QUIC.

MPTCP Multi-Path TCP.

MS Mid-Shadowing.

MSS Mobile-Satellite Service.

MTU Maximum Transmission Unit.

NG Next Generation.

NGSO Non-Geostationary Satellite Systems.

NOMA Non-Orthogonal Multiple Access.

NR New Radio.

NTN Non-Terrestrial Networks.

OEC Orbital Edge Computing.

OEC-TA OEC Task Allocation.

ONRM overhearing Non-Orthogonal Multiple Access (NOMA) repairing reliable multicast.

QoS Quality of Service.

RAN Radio Access Network.

RR Round-Robin.

RTO Retransmission Timeouts.

RTT Round-Trip Time.

SAN Satellite Access Network.

SCTP Stream Control Transmission Protocol.

SNO Satellite Network Operators.

STBP sojourn-time-based BP.

STECN Satellite-Terrestrial integrated Edge Computing Networks.

STK Satellite Tool Kit.

TAP Terminal Access Point.

TCP Transmission Control Protocol.

TFO TCP Fast Open.

TLS Transport Layer Security.

UAV Unmanned Aerial Vehicle.

UDP User Datagram Protocol.

UE User Equipment.

UML User-mode Linux.

URLLC Ultra-Reliable Low-Latency Communications.

WFQ Weighted Fair Queuing.

Resumen

Las Non-Terrestrial Networks (NTN) en general, y los satélites de baja órbita (Low Earth Orbit-LEO) en particular, desempeñarán un papel relevante en las próximas generaciones de comunicaciones móviles, las redes 5G y Beyond 5G (B5G), con mayor presencia que en los sistemas anteriores. A su vez, se están proponiendo y desarrollando protocolos en las capas superiores mejor adaptados a los requisitos de los nuevos servicios y aplicaciones, entre los que destaca QUIC, como alternativa a nivel de transporte. La presencia de este protocolo de transporte ha aumentado de manera notable desde que se propuso originalmente, pero su comportamiento sobre redes inalámbricas sigue siendo una cuestión abierta. En este trabajo se abordará el análisis y mejora del protocolo QUIC sobre redes no terrestres.

Como primer paso, se implementa un conjunto de funcionalidades en el simulador de red `ns-3` con el objetivo de emular escenarios LEO con diferentes topologías, utilizando modelos de capacidad de canal dinámicos definidos en la literatura. El enfoque propuesto de emulación permite analizar el comportamiento de protocolos y servicios reales ante diferentes situaciones de conectividad subyacente. Este entorno se ha utilizado para evaluar el comportamiento tanto del protocolo tradicional Transmission Control Protocol (TCP) como QUIC, bajo diversas configuraciones.

Además, se ha prestado especial atención a la capacidad multi-flujo de QUIC, que es una de las ventajas más relevantes que incluye este protocolo para mejorar su rendimiento. Para ello se propone un *shceduler* basado en el retardo para la gestión de los flujos de datos, cuyo comportamiento se ha analizado, con el fin de compararlo con el exhibido por las soluciones por defecto y otras alternativas. Se lleva a cabo una extensa campaña de experimentos usando el entorno de emulación mencionado, combinado con técnicas de virtualización, para integrar aplicaciones reales, y `ns-3`, que permite modelar, de forma realista, los enlaces NTN. Los resultados muestran que las políticas de *scheduling* propuestas distribuyen equitativamente el retardo entre los flujos, sin perjudicar a la tasa de envío de datos, incluso en situaciones de carga muy alta.

Palabras clave: QUIC, LEO, Scheduling, LMS, TCP.

Abstract

Non-Terrestrial Networks (NTN) in general, and Low Earth Orbit (LEO) satellites in particular, will play a key role in the next generations of mobile communications, 5G networks and Beyond 5G (B5G), being more relevant than in previous systems. In addition, new upper layers protocols, better adapted to novel applications and services, have recently arisen, being QUIC one of the most relevant solutions. Although the presence of QUIC has continuously increased since it was originally proposed, its performance over different wireless technologies is still an open research issue. In this context, this thesis addresses its performance analysis over non-terrestrial networks.

As a first step, a set of utilities is implemented in the `ns-3` network simulator, with the aim of deploying LEO scenarios with different topologies, using dynamic channel capacity models from the literature. The proposed emulation approach allows us to conduct systematic analyses of real protocols and services over different underlying channels. In this sense, it has been used to assess the behavior of both Transmission Control Protocol (TCP) and QUIC under different scenarios and setups.

Besides, we have paid special attention to QUIC multi-stream feature, which is one of its most relevant advantages and can be indeed exploited to improve its performance. In this sense, a delay-based scheduler has been proposed, based on dynamic queuing control, whose behavior has been extensively analysed, comparing it with that exhibited by legacy solutions. An extensive experiment campaign is carried out by exploiting the emulation environment along with virtualization techniques, to integrate real applications, and `ns-3`, to realistically model NTN links. The results show that the proposed scheduling policies fairly distribute the delay between streams, without jeopardizing the throughput, even under very high load situations.

Key words: QUIC, LEO, Scheduling, LMS, TCP.

1 Introducción

En la actualidad, se espera que la presencia de las redes Non-Terrestrial Networks (NTN) en las próximas tecnologías celulares, 5G y redes B5G, sea notablemente más relevante que en los sistemas anteriores. Así, se consideran elementos fundamentales de dichos despliegues de red, ya que permitirán brindar servicios de conectividad a áreas remotas (satélite), así como desplegar rápidamente recursos de comunicación en ciertos lugares, utilizando Unmanned Aerial Vehicle (UAV). En resumen, el impacto de agregar redes NTN a arquitecturas celulares más tradicionales resulta beneficioso en términos de fiabilidad, escalabilidad, cobertura y continuidad del servicio.

Por otro lado, estos beneficios vienen con varios desafíos de la mano. Por ejemplo, es bien conocido el bajo rendimiento exhibido por los protocolos de transporte tradicionales, más notoriamente TCP, sobre enlaces inalámbricos. Se espera que esto se vuelva más prominente con la llegada de los sistemas 5G y B5G, incluidos los que integren NTN. Además, los protocolos de transporte heredados no se comportan bien con los patrones de tráfico que caracterizan a los servicios que se están implantando en la actualidad, como es el caso del propio tráfico web.

Para subsanar las deficiencias mencionadas anteriormente, la comunidad científica se está esforzando en diseñar y desarrollar nuevos protocolos de transporte, que podrían superar algunas de las limitaciones que presenta TCP. Uno de los ejemplos más relevantes de tales esfuerzos es el protocolo QUIC, originalmente promovido por Google, que ha sido estandarizado recientemente. Entre sus ventajas podemos destacar que QUIC permite el *multi-streaming*, lo que evita el bloqueo por Head-of-line (HOL). En este sentido, cuando una pérdida afecta a un determinado flujo, el resto de flujos activos no se ven afectados.

Con el objetivo de aumentar la eficiencia del sistema, surgen algoritmos de *scheduling* para gestionar los recursos de transmisión, modulando el rendimiento y retardo de los *streams*. Durante una conexión QUIC, la capacidad está limitada en primer lugar por los algoritmos de control de flujo y congestión, como CUBIC, que establecen cuándo hay una oportunidad de transmisión. Seguidamente, el mecanismo de *scheduling* decide de qué flujo se seleccionarán datos para construir un paquete QUIC. Por defecto el protocolo implementa una estrategia RR, donde los segmentos de datos del paquete se asignan por igual a cada flujo, siempre siguiendo el mismo orden, por lo que se gestionan sin ningún tipo de prioridad. Por lo tanto, si todavía hubiera espacio disponible en el paquete QUIC actual, este se distribuiría de manera justa.

1.1. Objetivos

En este trabajo se abordará la evaluación de la combinación de QUIC y NTN y, en particular, se estudia el desempeño de este protocolo de transporte sobre comunicaciones LEO. Mediante el uso de una metodología novedosa de emulación, que permite imitar el comportamiento de los enlaces inalámbricos que caracterizan la conectividad subyacente, se puede extraer conclusiones sobre el rendimiento de dicho protocolo sobre estas redes. Primero se compara el rendimiento de QUIC con TCP y seguidamente, se dará paso al estudio del comportamiento de diferentes estrategias de *scheduling*, para determinar los beneficios potenciales que podrían brindar. Con esta finalidad, las contribuciones realizadas en el trabajo se pueden resumir en los siguientes puntos:

1. En primer lugar, se implementa una metodología de emulación de enlace basada en `ns-3`. Concretamente se ha modificado el comportamiento heredado del enlace *PointToPoint* en `ns-3` para adaptar sus características durante una ejecución, de acuerdo con diferentes eventos. Esto permite capturar el dinamismo de los enlaces subyacentes de NTN. Además, el marco propuesto puede implementar fácilmente topologías personalizadas, abarcando varios enlaces ISL, con diferentes características. Por ejemplo, se puede incluir tiempos de desconexión, que resultan característicos de este tipo de canales.

2. En segundo lugar, se utiliza dicha metodología, así como técnicas de virtualización, para evaluar el comportamiento del protocolo QUIC sobre escenarios de red satelital LEO. Para ello, se consideran diferentes configuraciones, cambiando las longitudes del búfer, el número de *streams*, la distribución de capacidad entre *streams*, y se comparará el rendimiento con el obtenido al usar TCP.

3. Por ultimo, se explota la capa de aplicación, mediante la configuración de múltiples *streams* que pretenden imitar el tráfico web, donde en una conexión es posible enviar diferentes objetos. En este sentido, se evalúa el comportamiento de cinco algoritmos de *scheduling*, además de la solución por defecto de QUIC, RR. En este sentido, se muestra el comportamiento de los algoritmos de *scheduling* Fair Queuing (FQ), WFQ y, además, de alternativas basadas en la ocupación de las colas, haciendo uso de la teoría de Lyapunov. Finalmente, se propone e implementa un nuevo *scheduler*, que aplica la mencionada teoría a la métrica de retardo en lugar de la ocupación de búferes. Finalmente, se analiza el rendimiento de todas las soluciones de *scheduling* en términos de ocupación del búfer, rendimiento y retardo.

1.2. Estructura de la memoria

A continuación, se presenta una descripción de la estructura del documento. El segundo capítulo aborda los antecedentes relacionados con el tema de estudio. Los dos capítulos siguientes se centran en el desarrollo llevado a cabo durante la realización de este trabajo, detallando los aspectos técnicos y metodológicos más relevantes. Posteriormente, se presentan las conclusiones derivadas de la investigación que se ha realizado, resumiendo los principales resultados. Por

último, se mencionan las posibles líneas futuras de investigación que podrían surgir a partir de este estudio.

Capitulo 2 - Antecedentes

Este capítulo se centra en la revisión de la literatura existente y en la descripción detallada de los tres elementos fundamentales que sustentan este trabajo: las redes no terrestres, el protocolo QUIC y el *scheduling*. Se lleva a cabo una exhaustiva recopilación de trabajos previos relacionados con el tema, con el objetivo de establecer el estado del arte en cada una de estas áreas. Asimismo, se proporciona una explicación en profundidad de los conceptos clave y los fundamentos teóricos relacionados. Esta revisión y descripción detallada sientan las bases para comprender plenamente el contexto en el que se enmarca este trabajo.

Capítulo 3 - Entorno de ns-3

Seguidamente, se procede con la descripción del diseño e implementación abordados, siguiendo el orden lógico del desarrollo. Se presenta el entorno `ns-3`, una plataforma de simulación de red ampliamente utilizada en el ámbito de las redes de comunicación. Se describe detalladamente la metodología utilizada y las herramientas desarrolladas para construir diferentes topologías que abarcan diversos escenarios dentro del ámbito de las redes no terrestres. Estas topologías se diseñan y configuran de acuerdo con los requisitos específicos de los experimentos, y permiten simular de manera realista las características y el comportamiento de las redes NTN. La explicación de la metodología y las herramientas proporciona una base sólida para la comprensión de la implementación y la reproducción de los experimentos en un entorno de simulación controlado.

Capítulo 4 - Plataforma y estrategias de scheduling

En este capítulo se aborda la integración del entorno de simulación `ns-3` con la aplicación basada en el protocolo QUIC, aprovechando la metodología descrita en el capítulo anterior. Se presentan los resultados obtenidos al utilizar diferentes algoritmos de *scheduling*, haciendo especial hincapié en aquellos que han sido desarrollados en el marco de este trabajo, basados en el retardo. Además, se proporciona una explicación matemática de dichos algoritmos y se detalla su implementación.

Capítulo 5 - Conclusiones y líneas futuras

Para concluir este trabajo, se presentan en este capítulo las conclusiones derivadas de la investigación llevada a cabo. Se ha logrado desarrollar una serie de utilidades que permiten simular diferentes topologías de redes de órbita terrestre baja (LEO) en el entorno de simulación `ns-3`. Además, se han destacado las ventajas de utilizar algoritmos de *scheduling* basados en retardo. Finalmente, se identifican posibles direcciones futuras de investigación.

2 Antecedentes

Este capitulo detalla el estado del arte relacionado, recogiendo aquellas investigaciones previas que preceden a este trabajo. Seguidamente, se dará paso a una breve explicación de las redes no terrestres y su papel en los sistemas de comunicaciones presentes y futuros. A continuación se describirá el protocolo QUIC, presentando sus funcionalidades, ventajas frente a alternativas más tradicionales, así como la justificación de su uso. Finalmente, se procede a realizar un breve repaso del papel de los algoritmos de *scheduling*, así como sus potenciales beneficios.

2.1. Redes no terrestres 5G y B5G

En los últimos años, se ha presenciado un incremento significativo de despliegues de redes 5G, gracias al papel desempeñado por los operadores de comunicaciones. Así, se han ampliado de manera notable, para ofrecer una amplia gama de servicios, ya que, hasta ahora, el enfoque principal de las redes 5G se centraba en brindar cobertura a los usuarios convencionales. Sin embargo, esta expansión abarca ahora la provisión de servicios para la Internet of Things (IoT) a gran escala, aplicaciones industriales y comunicaciones máquina a máquina.

Con el objetivo de garantizar la continuidad del servicio, la ubicuidad y la escalabilidad, se ha producido una evolución de la red 5G mediante la integración de infraestructuras no convencionales, como las redes no terrestres NTN. Estas redes ofrecen ventajas significativas al extender la cobertura de la red 5G a áreas remotas y de difícil acceso, en las que las infraestructuras terrestres son limitadas o inexistentes [3].

En el contexto del 5G y las generaciones futuras, las NTN sustentarán tres tipos de escenarios definidos por la International Telecommunication Union (ITU), caracterizados a su vez por tres tipos de servicios. En primer lugar, se destaca el Enhanced Mobile Broadband (eMBB), donde las NTN tienen como objetivo proporcionar conectividad de banda ancha en áreas desatendidas y en plataformas móviles, además de ofrecer mayor robustez mediante la integración de los sistemas terrestres y las NTN. También se está explorando la posibilidad de utilizar las NTN como canal para la entrega de contenidos de transmisión y multidifusión, aliviando así la carga de la red terrestre.

En segundo lugar, se encuentra el servicio Massive Machine-Type Communications (mMTC), donde las NTN admiten conectividad para servicios de IoT de área local. Por último, se encuentran los denominados servicios Ultra-Reliable Low-Latency Communications (URLLC), que plantean desafíos significativos para las NTN, debido a los retardos en la propagación satelital y a los estrictos requisitos de fiabilidad, disponibilidad y latencia que tienen aquellos [36].

No obstante, a pesar de los numerosos beneficios que ofrecen, es fundamental examinar de manera detallada las características clave y los desafíos asociados con la integración de estas redes. Las redes no terrestres presentan diversas características distintivas en comparación con la red terrestre convencional. Estas diferencias se deben principalmente a la altitud y al movimiento de los nodos que conforman estas redes.

Uno de los papeles más relevantes de las redes no terrestres y, por ende, una de sus principales motivaciones, seria la capacidad de garantizar comunicaciones resilientes y servicios continuos. Así, las estaciones no terrestres conformarían un apoyo a la red terrestre convencional, proporcionando servicios de cobertura inalámbrica de alta capacidad tanto en puntos de alta demanda, como en situaciones en las que la infraestructura convencional esté desbordada. Además, puede ser una alternativa de respaldo, ofreciendo redundancia a la red, cuando las estaciones terrestres estén fuera de servicio, o llegando a zonas remotas donde el acceso mediante una infraestructura convencional puede ser costosa, véase áreas rurales o zonas marítimas. Asimismo, estas redes son capaces de brindar servicios bajo demanda a los usuarios ubicados en el borde de la red. Otra aplicación relevante es la denominada MEC, que tiene como objetivo proporcionar capacidades de cómputo y almacenamiento cercanos a los usuarios finales.

Otro objetivo adicional sería la implementación de comunicaciones satélites superpuestas, permitiendo la interconexión de estaciones terrestres, brindando conectividad de acceso de alta capacidad a los dispositivos de tierra mediante la retransmisión a través de una red mallada superpuesta. Esta alternativa puede resultar más económica que la infraestructura de fibra óptica en ciertos escenarios.

Por otro lado, cabe hacer hincapié en la difusión ubicua del IoT, puesto que el número de dispositivos IoT conectados está experimentado un crecimiento significativo. El potencial del IoT abarca una amplia gama de aplicaciones, que van desde el transporte, sistemas inteligentes de estacionamiento, alumbrado inteligente, salud, edificios inteligentes, red eléctrica inteligente, dispositivos inteligentes. Las redes IoT pueden extenderse, diferenciándose entre corto y largo alcance, principalmente cubriendo aplicaciones de exterior.

La Figura 2.1 ilustra la integración de las redes terrestres con las tradicionales en las generaciones móviles actuales y futuras. Se intentan resaltar especialmente los casos de uso y las oportunidades actuales, al mismo tiempo que muestra las posibles aplicaciones de la tecnología no terrestre en el futuro.

Los casos de uso mencionados anteriormente son un claro indicador de que los sistemas 5G y posteriores dependerán cada vez más de componentes no terrestres para ofrecer sus servicios a nivel global. Esto se debe a sus capacidades únicas para extender la cobertura en áreas donde es imposible o poco rentable desplegar una infraestructura terrestre, así como a su papel complementario en la descarga de una parte importante del tráfico, especialmente en áreas altamente congestionadas. Sin embargo, las peculiaridades técnicas derivadas de la presencia del canal no terrestre, el cual difiere significativamente de uno terrestre, plantean diversos desafíos que deben resolverse y generan la necesidad de soluciones novedosas [2]. A continuación, se detallan algunas de las problemáticas más relevantes:

- **Retardo y pérdidas de propagación:** La altitud de los nodos puede causar un retardo extra en la comunicación, especialmente en el caso de los satélites Geoestacionary Equatorial

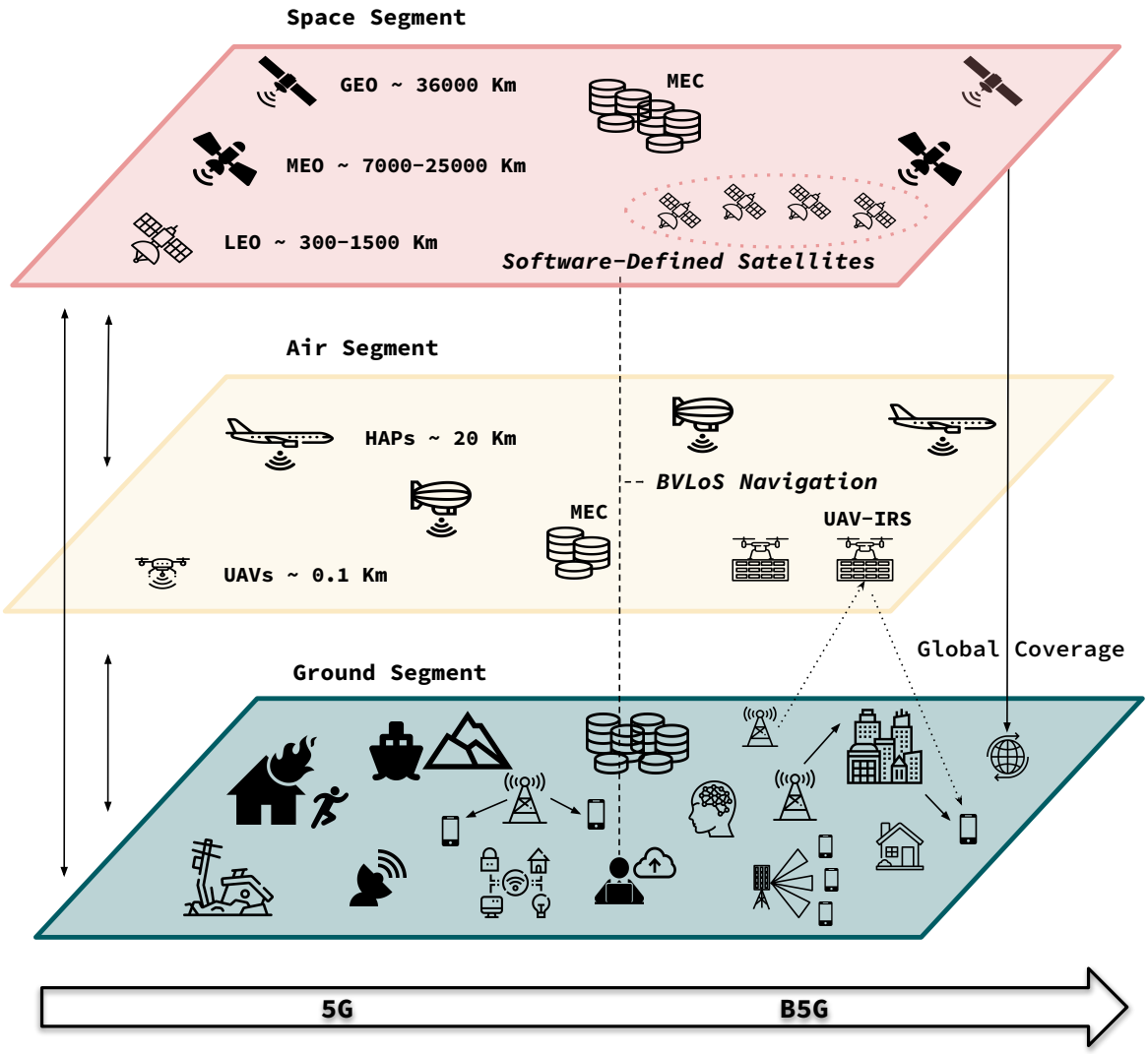


Figura 2.1: Convergencia de las redes terrestres y no terrestres, y servicios de interés.

Orbit (GEO), caracterizados por poseer mayor altura orbital como se aprecia en la Figura 2.1, llegando a Round-Trip Time (RTT) de 270 ms [2]. Este efecto puede crear un cuello de botella para servicios con estrictos requisitos de latencia. Además, los retardos de propagación grandes pueden impactar en los protocolos de capas superiores, en sus mecanismos de retransmisión.

- **Efecto Doppler:** El desplazamiento de los nodos no terrestres puede resultar en efectos Doppler adicionales en las comunicaciones, que se ve agravado especialmente en el caso de las constelaciones de Órbita Terrestre Baja (LEO, por sus siglas en inglés), donde se producen grandes cambios en frecuencia. Además, esta característica tiene un impacto significativo en los sistemas de comunicación utilizados en trenes de alta velocidad. Genera así un desafío importante que debe abordarse para mitigar tanto los efectos Doppler como el fenómeno de multi-camino.

- **Cobertura, rendimiento y traspasos:** Cuanto mayor sea la altitud de los nodos, mayor

será el tamaño de la huella del haz. Para un satélite GEO, la cobertura puede alcanzar los 3500 km, mientras que en los satélites Medium Earth Orbit (MEO) y LEO, también conocidos como Non-Geostationary Satellite Systems (NGSO), llega a los 1000 km. Aunque el valor específico depende de la altitud exacta y de los ángulos de elevación admitidos entre los usuarios de tierra y el satélite. Los satélites NGSO permiten proporcionar una cobertura variable en el tiempo y en el espacio, mientras que la cobertura de los satélites GEO es bastante estática. Por ello, se requieren frecuentes procedimientos de traspaso para encaminar el tráfico de un satélite NGSO a otro y garantizar así la continuidad de los servicios a los usuarios. Por otro lado, aunque la cobertura de los High Altitude Platformn (HAP) y los UAV es menor, pueden proporcionar enlaces de mayor *throughput* a los usuarios terrestres de forma más flexible. Además, pueden desplegarse rápidamente en zonas específicas o según de la demanda en el momento.

- **Despliegue:** Los UAVs y HAPs ofrecen una solución barata, rápida, flexible y temporal, en comparación con los satélites, cuyo despliegue es de largo plazo. Las Base Station (BS) aéreas pueden ser desplegadas para una operación durante horas, mientras que los satélites requieren de meses hasta su puesta en marcha. Sin embargo, mientras que en el caso de UAVs y HAPs su uso puede dedicarse a cubrir fallos o aumentos de la congestión, los satélites proporcionan cobertura más extensa.
- **Entorno:** El estudio del clima es crucial para las comunicaciones por satélite, pues se trata de un entorno hostil afectado por radiación, así como infraestructura o materia. Además, cabe resaltar que el enlace entre la estación terrena y el primer satélite, o enlace LMS, ha sido estudiado extensamente y se han tomado medidas con el fin de modelar su comportamiento, recogiendo efectos climáticos.

Por consiguiente, se abre un amplio abanico de posibilidades para abordar las problemáticas mencionadas. Como se ha indicado previamente, en los sistemas vía satélite suele ser significativamente mayor el retardo que el tiempo de procesamiento, lo cual resulta ser un factor determinante en el diseño del sistema. Muchos aspectos de las redes no terrestres, como las capas de protocolo, los mecanismos de retransmisión y los tiempos de respuesta en la asignación de recursos, se ven afectados por los elevados retardos de propagación. Además, alcanzar la tasa máxima del canal de manera rápida y eficiente en redes donde la variabilidad del canal puede ser alta supone un gran desafío. Es por ello, que el control de congestión se puede ver gravemente afectado por el empleo de protocolos tradicionales (por ejemplo, TCP).

2.1.1. Bandas de frecuencias

El 5G está extendiendo los límites de las comunicaciones inalámbricas, permitiendo el despliegue de casos de uso que requieren velocidades exigentes, así como baja latencia y elevada fiabilidad. Como parte de la estandarización de las tecnologías 5G celulares, el 3rd Generation Partnership Project (3GPP) ha desarrollado una nueva tecnología de acceso llamada 5G New Radio (NR), diseñada para convertirse en estándar en el marco de las redes 5G.

El objetivo de las tecnologías 5G se extiende mas allá de ofrecer servicios de banda ancha, permitiendo avances claves que habilitan la posibilidad de un gran espectro de aplicaciones

nuevas. 5G NR comprende desde bandas de frecuencia baja y media por debajo de los 6 GHz, así como también bandas de frecuencia alta, sobrepasando los 24 GHz, definido como FR2/mmWave (ondas milimétricas). Por tanto, el rango de frecuencias de los sistemas 5G abarca el espectro celular previo así como adicionalmente también aquellos por debajo de 6 GHz y superiores.

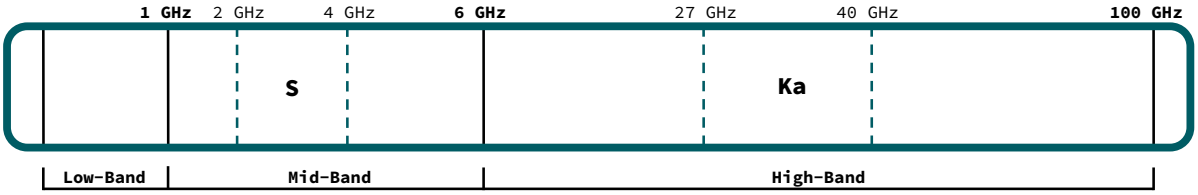


Figura 2.2: Bandas de frecuencia.

El termino de ondas milimétricas se refiere a la parte específica del espectro de radio frecuencia con longitud de onda corta (por ejemplo, el rango de 27 GHz hasta 40 GHz - Banda Ka). Su empleo en 5G se verá incrementado, debido a una serie de factores como son su baja utilización hasta el momento o la capacidad de proporcionar tasas de transferencia elevadas. Sin embargo, dicho espectro requiere de línea de vista entre los terminales y las antenas. Cualquier obstáculo u obstrucción pasiva, como las señales de carretera frente a las celdas, los árboles o los edificios, así como los objetos en movimiento, como los automóviles, tienen el potencial de degradar o bloquear una señal 5G FR2. A continuación, se describen las características más relevantes de dos bandas principales en este ámbito, en primer lugar, la Banda S perteneciendo al espectro medio así como la Banda Ka, que se enmarca en la categoría de ondas milimétricas. Ambas son utilizadas en las redes de acceso basadas en satélites LEO.

Banda S

La Banda S del Mobile-Satellite Service (MSS) ha sido armonizada a nivel mundial por la ITU en el rango de frecuencias de 1980 a 2010 MHz para las transmisiones desde dispositivos móviles hacia satélites, y de 2170 a 2200 MHz para las transmisiones desde satélites hacia dispositivos móviles [39]. Esta banda se utiliza principalmente para facilitar las comunicaciones móviles, como las tecnologías IMT-2000 / UMTS, la comunicación móvil vía satélite y los sistemas de radar meteorológico y marítimo. También se incluye en esta banda la frecuencia de 2,4 GHz, conocida como Industrial, Scientific, and Medical (ISM), que es ampliamente utilizada por tecnologías como Bluetooth, Wi-Fi, Zigbee y otras [30].

Tabla 2.1: Características de la Banda S.

Banda de frecuencia	Rango de frecuencia	Longitud de onda
S	2-4 GHz	7.5-15 cm

En comparación con las bandas superiores, la Banda S del MSS muestra una menor vulnerabilidad a condiciones adversas como la lluvia. Sin embargo, a cambio de esta mayor resistencia, presenta un rendimiento inferior y un espectro de frecuencia más estrecho. Además, requiere antenas de mayor tamaño en comparación con las bandas superiores.

Banda Ka

En la actualidad, la banda Ka se utiliza ampliamente por parte de los operadores de redes satelitales Satellite Network Operators (SNO) para proporcionar servicios de banda ancha a los clientes [39]. Si se lleva a cabo una coordinación adecuada, es posible anticipar el uso compartido de las siguientes bandas de frecuencia para servicios satelitales fijos, tanto por parte de los operadores de redes móviles Mobile Network Operator (MNO) como por los SNO, incluso si no se están considerando en las especificaciones del 3GPP. Estas bandas de frecuencia específicas, que se utilizarán en este trabajo, se sitúan en el rango de 17,7 a 20,2 GHz para las transmisiones descendentes y en el rango de 29,5 a 30 GHz para las transmisiones ascendentes [30].

Tabla 2.2: Características de la Banda Ka.

Banda de frecuencia	Rango de frecuencia	Longitud de onda
Ka	27-40 GHz	5.0-11.3 mm

La banda Ka posibilita la transmisión de datos a velocidades elevadas, y permite el uso de antenas más compactas, lo que a su vez reduce los costes asociados. No obstante, una de sus desventajas es que los efectos de atenuación provocados por factores externos son más significativos, lo que puede resultar en niveles más altos de pérdida de señal.

2.1.2. Arquitectura de la red no terrestre

Como se mencionó anteriormente, la arquitectura 5G permite la combinación de diversos elementos para lograr la interconexión entre elementos terrestres y no terrestres, así como establecer enlaces de alimentación, comunicación entre satélites mediante ISL y un enlace de servicio, como se aprecia en la Figura 2.3.

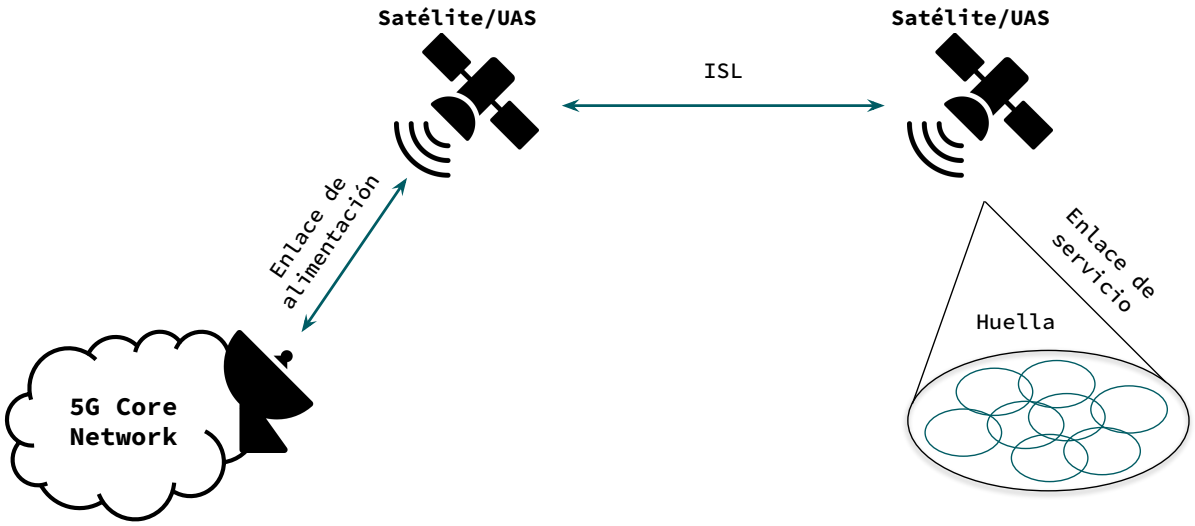


Figura 2.3: Arquitectura de una red no terrestre.

En la Figura 2.3 se ilustra la arquitectura de las redes no terrestres con carga útil transparente. La red central 5G se conecta a un Next Generation NoedB (gnB) mediante la interfaz Next

Generation (NG). El gnB está situado en tierra y se conecta a una pasarela NTN a través del enlace de alimentación.

Dependiendo de la posición de las funcionalidades del gnB, es posible distinguir tres casos de uso básicos: arquitectura transparente, regenerativa y en *split*. En el último caso, las funcionalidades atribuidas al gnB están divididas en la Distributed Unit (DU) y en Central Unit (CU), donde la DU se encuentra localizada en un elemento no terrestre mientras que la CU se despliega en tierra [11].

- **Carga útil transparente:** En este caso, el satélite actúa como un repetidor analógico de radiofrecuencia, tanto para los enlaces de alimentación como para los enlaces de servicio. En relación con el 5G, el satélite simplemente repite la interfaz radio 5G NR-Uu (datos de usuario) desde el enlace de alimentación hasta el de servicio. En resumen, el satélite se comporta como un amplificador de señal, sin alterar o procesar la información transmitida.

- **Carga útil regenerativa:** En este caso, el satélite regenera la señal proveniente de la Tierra. La interfaz NR-Uu opera en el enlace de servicio, mientras que las interfaces de gestión operan sobre una interfaz de radio satelital en el enlace de alimentación. En otras palabras, la estación base gNB (5G) se despliega en altura, es decir, en el satélite. El satélite procesa y regenera la señal recibida desde la Tierra antes de transmitirla hacia los enlaces de servicio.

2.2. QUIC

Durante décadas User Datagram Protocol (UDP) y TCP han sido los únicos protocolos de transporte con gran adopción. Han surgido variantes o funcionalidades añadidas a TCP, pero muchas de ellas tuvieron una implantación limitada por numerosos problemas debido a la propia implementación del protocolo, pudiendo destacar Stream Control Transmission Protocol (SCTP) [40] o Multi-Path TCP (MPTCP) [10].

Por otro lado, QUIC fue propuesto por Google en el año 2012 como un protocolo confiable sobre UDP. Entre las ventajas de QUIC se puede destacar la multiplexación dentro de la propia implementación, resolviendo la ineficiencia por parte de TCP debido a la problemática del bloqueo por HOL. Además permite llevar a cabo establecimientos de re-conexión de 0-RTT de sobrecarga.

Una característica a destacar entre los protocolos de nivel de aplicación, como es Hypertext Transfer Protocol (HTTP) y los servicios actuales, es la necesidad de multiplexar objetos a través de *streams*. En particular, TCP no lleva embebida dicha funcionalidad, si no que es implementada a nivel de aplicación, como se observa en la Figura 2.4. Por otro lado, TCP garantiza que la llegada de los paquetes sea en orden, donde los que llegan fuera de orden, debido a aspectos como la pérdida o variaciones del canal, deben ser almacenados en una cola hasta que el siguiente segmento en la secuencia correcta sea recibido. Asimismo, como TCP no es consciente de la multiplexación a nivel de aplicación, dicha cualidad puede afectar negativamente a los distintos flujos, acarreando el bloqueo por HOL. Sin embargo, como se ha mencionado, QUIC contempla en sí mismo la multiplexación, permitiendo la entrega ordenada a nivel de *stream* frente a nivel de conexión, resolviendo la problemática de TCP.

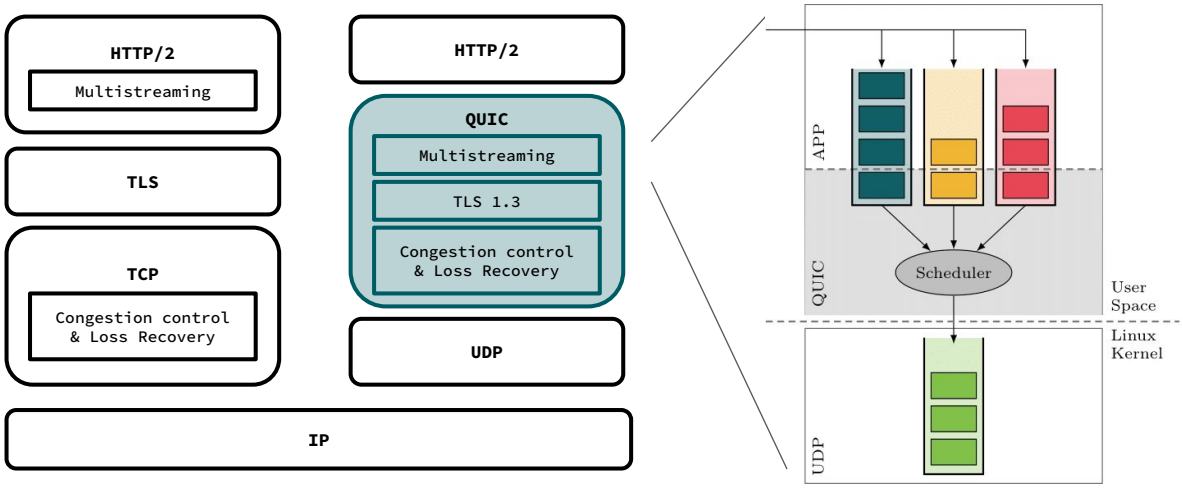
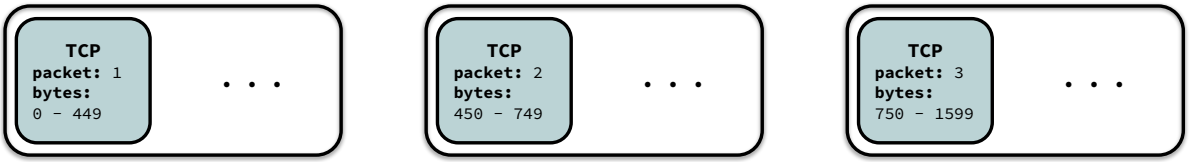
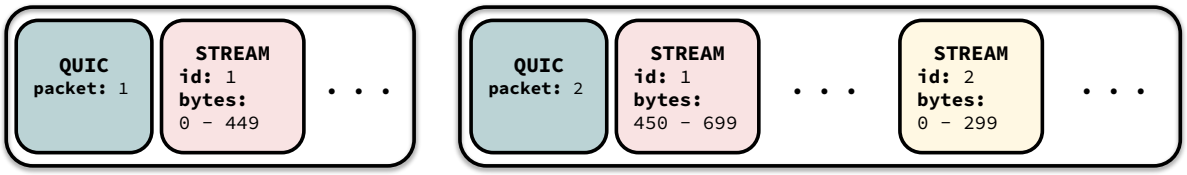


Figura 2.4: Comparativa de los protocolos de transporte, QUIC y TCP, así como de las funcionalidades que incorporan.



(a) Protocolo TCP



(b) Protocolo QUIC

Figura 2.5: Seguimiento de flujos de bytes TCP vs. QUIC.

La multiplexación de diferentes *streams* se consigue mediante el intercalado de diferentes *stream-frames* dentro de un paquete QUIC. Aunque, el incluir varios *frames* en un paquete puede inducir en un bloqueo de diferentes *streams*, se busca un acuerdo de compromiso entre la utilización de la capacidad de datos del paquete en su totalidad y el HOL, tal y como se recoge en la Sección 13 del RFC 9000 [20]:

"*A single QUIC packet can include multiple STREAM frames from one or more streams. One of the benefits of QUIC is avoidance of head-of-line blocking across multiple streams. When a packet loss occurs, only streams with data in that packet are blocked waiting for a retransmission to be received, while other streams can continue making progress. Note that when data from multiple streams is included in a single QUIC packet, loss of that packet blocks all those streams from making progress. Implementations are advised to include as few streams as necessary in outgoing packets without losing transmission efficiency to underfilled packets.*". De este modo, el reconocimiento de QUIC es a nivel de rango de bytes de los distintos *frames* de datos que puede incluir un paquete, como se observa en la Figura 2.5.

Otra ventaja relevante a destacar es el proceso de establecimiento de la conexión. TCP

requiere del conocido *3-way handshake*, con una sobrecarga de 1-RTT. Además, al no implementar seguridad, en muchos casos se requiere el establecimiento de la sesión Transport Layer Security (TLS), que induce otro RTT de retardo antes de comenzar con el envío de datos. En cambio, QUIC incluye en el propio protocolo mecanismos de seguridad integrados, TLS, teniendo una ventaja de 1-RTT frente a las conexiones cifradas de TCP mediante el uso de TLS. Además, si se considera que el cliente puede recordar información de que inicialmente ha estado conectado, se podría restablecer una conexión sin retardo (0-RTT). Aunque esta funcionalidad también se puede usar en TCP mediante la opción TCP Fast Open (TFO), ésta no está habilitada por defecto.

En resumen, dependiendo del contexto, se podría decir que QUIC posee una ventaja de 1-RTT o 2-RTT frente a TCP, lo cual puede significar una gran diferencia cuando se trabaja con cantidades pequeñas de información o cuando la red subyacente presenta grandes retardos, como se refleja en la Figura 2.6.

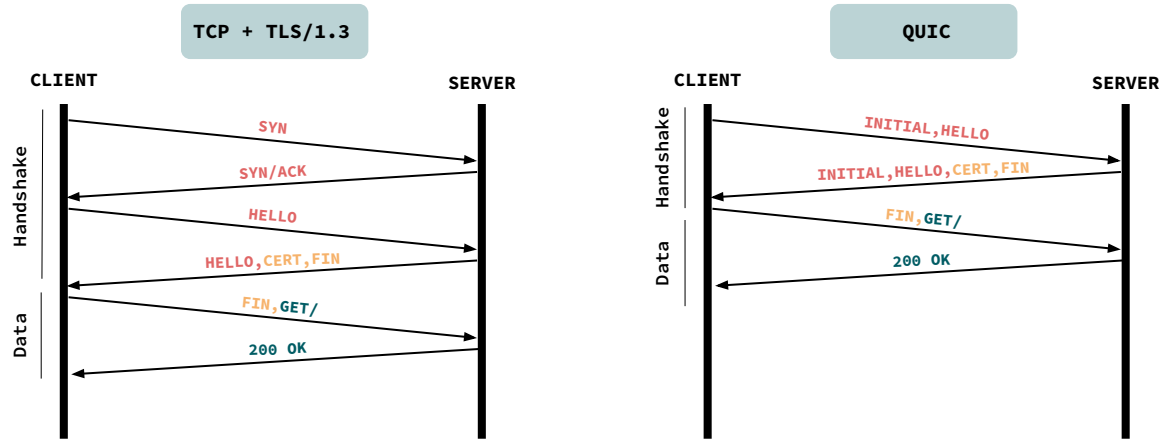


Figura 2.6: *Handshake* en TCP y QUIC.

2.2.1. Estructura del paquete QUIC

Después de presentar los rasgos más relevantes de QUIC frente a protocolos anteriores, se describe del paquete de datos, puesto que será importante conocerlo de cara a su posterior utilización en las soluciones de *scheduling*, donde se requiere considerar su contenido y características.

Como se ha mencionado anteriormente, el protocolo QUIC utiliza unidades de paquetes para transportar sus datos. Dentro del paquete QUIC, se pueden identificar tanto su propia cabecera como la de los `Stream frames`; como se mencionó previamente, el propio paquete puede contener datos de distintos flujos. En la Figura 2.7 se presenta la cabecera de cada trama de un `stream` [20], que tiene los siguientes campos:

1. El campo `Type` en el paquete QUIC indica el tipo de `Stream frame` que se va a enviar. Los valores posibles van desde `0x08` hasta `0x0f (0b00001XXX)`, donde los tres bits menos significativos determinan los siguientes campos.

   El bit `OFF` (0x04) se utiliza para indicar la presencia del campo `Offset`. Si está activo (valor 1), significa que el campo `Offset` está presente; de lo contrario, implica que estos son

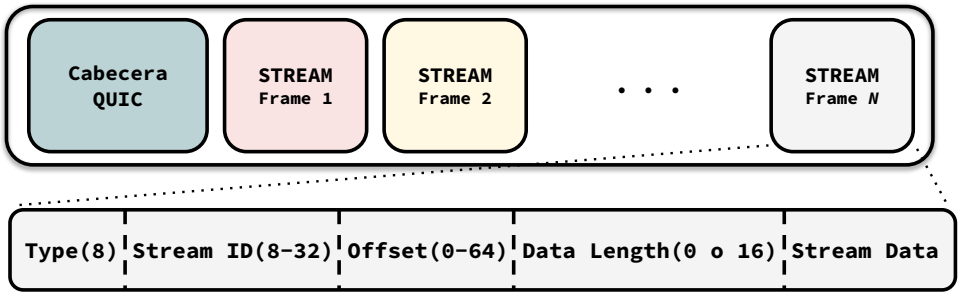


Figura 2.7: Composición del paquete QUIC y cabecera de `Stream Frames`.

los primeros bytes del flujo actual. Por otro lado, el bit `LEN` (0x02) se utiliza para indicar la presencia del campo `Data Length`. Si está establecido en 1, significa que el campo `Data Length` está presente; de lo contrario, implica que el campo de `Stream Data` se extiende hasta el final del paquete QUIC. Esto puede indicar dos casos: (1) solo hay información de este flujo a enviar, si es el primer *frame*; o (2) no existen más tramas después de esta, si es la última. Asimismo, el bit `FIN` (0x01) indica que esta trama pertenece al final del `stream`. En consecuencia, el tamaño final del flujo se determina sumando los valores de los campos `Offset` y `Data Length`.

2. El campo `Stream ID` indica el identificador del flujo. Este identificador no sólo indica el `stream` actual, sino que también proporciona información adicional, como el origen (cliente o servidor) y el tipo de flujo. Su formato es un número entero de longitud variable, es decir, mediante los dos bits más significativos del primer byte se codifica el número de bytes (es decir, la longitud del `Stream ID`) a usar, en logaritmo base 2, codificándose en los bits restantes dicho número entero. Por otro lado, otro rasgo a destacar es que los flujos pueden ser unidireccionales o bidireccionales, dependiendo de la dirección de los datos. Si sólo se transportan datos del emisor al receptor, se considera un flujo unidireccional; si se transportan datos en ambas direcciones, se considera un flujo bidireccional. Para indicar el papel del extremo transmisor, los dos bits menos significativos de este campo se utilizan para indicar si es el cliente o el servidor, y si es un flujo unidireccional o bidireccional.

3. El campo `Offset` en una trama especifica el desplazamiento en bytes dentro del flujo de datos. Indica la cantidad de bytes enviados previamente en dicho flujo. Para la primera trama del flujo, este campo se omite, ya que su valor sería 0. El propósito de este campo es mantener un seguimiento de la cantidad de datos enviados en un flujo en particular. El formato de este campo es un número entero de longitud variable.

4. El campo `Data Length` en una trama especifica la longitud del campo de datos (`Stream Data`) en la trama actual. Si esta trama es la única, o la última, en un paquete, este campo se omite, ya que su valor se puede calcular utilizando otros elementos de la cabecera. Su propósito es indicar la cantidad de datos que se encuentran en el campo correspondiente de la trama actual. Su formato es un número entero de longitud variable.

5. El campo `Stream Data` en una trama contiene la información (bytes) perteneciente al flujo que se va a enviar. Este campo almacena los datos reales que se transmitirán en el flujo.

Dependiendo de la implementación o del protocolo concreto, estos datos pueden tener diferentes formatos o estructuras. Su propósito es transportar los datos del flujo en la trama.

2.2.2. Control de congestión

El control de congestión es una de las funcionalidades más importante de la capa de transporte para permitir una utilización justa y elevada de la red, formada por enlaces que pueden estar compartidos por varios flujos. Es decir, el control de congestión comprende un conjunto de mecanismos capaces de detectar y corregir los problemas que surgen cuando la red no puede cursar todo el trafico ofrecido.

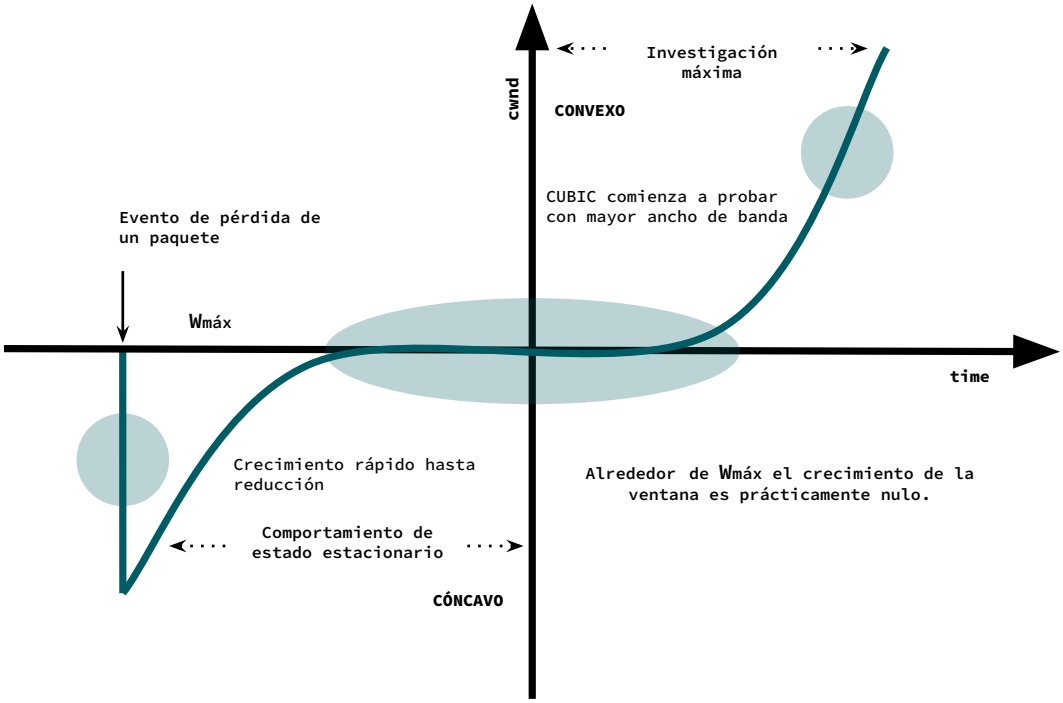


Figura 2.8: Explicación de los estados del algoritmo de congestión CUBIC.

De esta manera, si la demanda exigida excede en un momento concreto los límites posibles por parte del recurso, el redimiendo de la red se verá deteriorado, produciéndose episodios de congestión. Existen varias implementaciones en la actualidad para mitigar su efecto, enfocadas en la detección de pérdidas (reactivo), en el retardo (proactivo) y aquellos basados tanto en las pérdidas como en el ancho de banda (híbridos).

El protocolo QUIC proporciona una variedad de señales genéricas que permiten soportar diferentes algoritmos de control de congestión desde el lado del emisor [19]. A continuación se va a describir de forma genérica el control de congestión CUBIC, ya que es el usado en la implementación QUIC utilizada en el trabajo, y cuyo comportamiento condiciona por tanto los resultados obtenidos.

El algoritmo de congestión CUBIC [35] ha sido ampliamente empleado en el kernel de Linux, extendiéndose su uso a diferentes implementaciones. Surge con el principal objetivo de aprovechar el ancho de banda disponible, mediante el aumento rápido de la ventana de congestión al inicio

de la conexión. Además, su adaptación dinámica permite que se ajuste eficientemente a las variaciones de la red. CUBIC determina el valor del tamaño de la ventana de congestión a partir de una función cubica del tiempo desde la ultima detección de congestión (última pérdida de paquete), Ecuación 2.1.

$$W_{cubic}(t) = C(t-K)^3 + W_{max} \tag{2.1}$$

Como se observa en la expresión, C es el parámetro CUBIC, t es el tiempo transcurrido desde la última reducción de la ventana, y W_{max} es el tamaño de la ventana de congestión del último evento de pérdida que ha ocurrido. K es el periodo de tiempo que la función toma para incrementar W_{cubic} a W_{max} y se obtiene mediante el empleo de la siguiente Ecuación 2.2:

$$K = \sqrt[3]{\frac{W_{max} \cdot \beta}{C}} \tag{2.2}$$

donde β es la constante de disminución de la ventana tras un evento de pérdida (también conocido como *multiplication decrease factor*). Como resultado, la ventana de congestión presenta la variación que ilustra la Figura 2.9 (`cwnd`), reduciendo su magnitud de manera proporcional al factor β ($W_{CUBIC} = W_{\max} \cdot \beta$).

En la Figura 2.8 se observa cómo es el comportamiento de la ventana del algoritmo CUBIC. Se puede apreciar como $W_{\max}$ delimita dos comportamientos distinguibles: el primero de estado estacionario *Steady State Behaviour* y el segundo de ellos, de sondeo máximo, o también conocido como *Max Probing*. El primer estado sigue una forma cóncava, y su duración se prolonga desde que se produjo la primera reducción de la ventana de congestión (`cwnd`) hasta que se alcanza el valor máximo $W_{\max}$. Es decir, tras reducirse la ventana a un valor mínimo, se produce un crecimiento abrupto hasta llegar a valores cercanos a $W_{\max}$, donde el crecimiento es lento, debido a los episodios de pérdidas producidos con anterioridad.

Posteriormente, cuando se alcanza el valor $t = K$, es cuando CUBIC alcanza el valor $W_{\max}$, cambiando a estado *Max Probing*. Tras alcanzarse el valor $W_{\max}$, la ventana de congestión sigue una forma convexa. Cerca de $W_{\max}$ el crecimiento es lento, y una vez sobrepasado el punto de inflexión, se produce un crecimiento rápido hasta que se vuelva a dar un suceso de pérdida de un paquete.

A modo de ejemplo, la Figura 2.9 ilustra una posible evolución de la ventana de congestión usando CUBIC. Inicialmente se observa la fase de *Slow Start*, en la cual se trata de evitar la congestión inicial de la red. Posteriormente se observa una función cóncava debido a una pérdida de un paquete, hasta que la ventana llega al valor de $W_{\max}$. En una tercera fase, se observan los perfiles cóncavos y convexos, donde este último se corresponde con la contribución de incremento rápido de la ventana de congestión, después de mantenerse constante en torno $W_{\max}$.

2.3. Estrategias de *scheduling*

Las estrategias de *scheduling* pueden juegan un papel significativo en el rendimiento y la velocidad de las comunicaciones. Principalmente su uso se sitúa en el nivel Medium Access Control (MAC) en redes móviles. Asimismo, se usan estrategias de scheduling para la gestión de colas de tráfico (por ejemplo en *switches*) que comparten un único recurso de comunicaciones

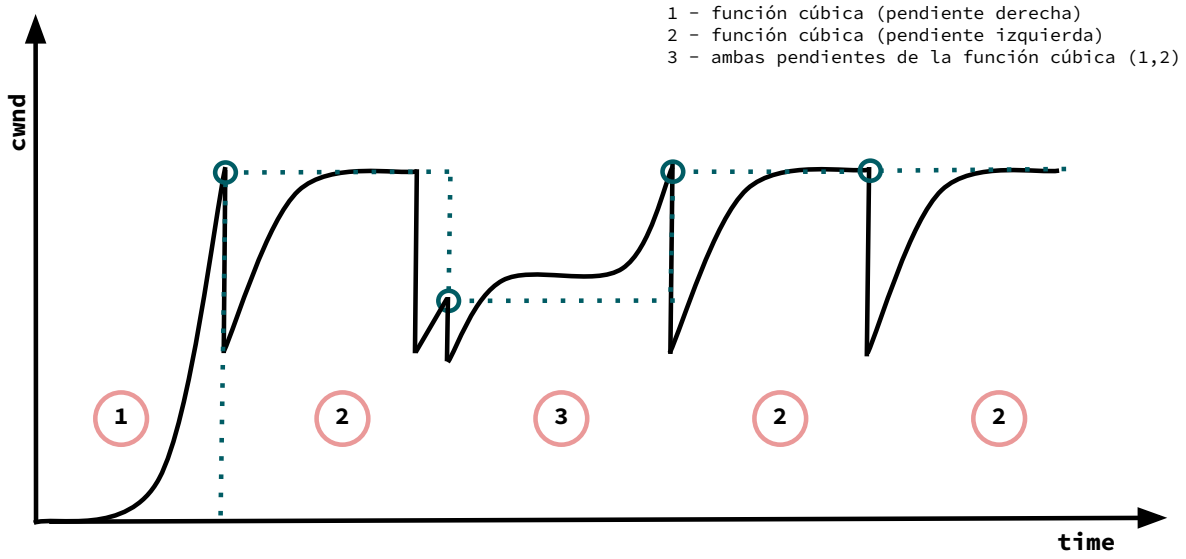


Figura 2.9: Evolución de ventana de congestión CUBIC.

(un interfaz). En el caso de QUIC, el control de congestión arbitra el recurso de comunicaciones compartido por los diferentes *streams*, mediante el crecimiento y decrecimiento de la ventana, por lo que se trata de otro ámbito de aplicación de técnicas de *sheduling*. A continuación, se describen los *schedulers* más tradicionales, entre los que se incluyen el usado por la implementación de QUIC utilizada en este trabajo. También se describe un *scheduler* desarrollado en un trabajo previo [17] y que se utilizará para comparar el propuesto en el marco de la investigación descrita en este documento.

2.3.1. Round Robin

En primer lugar, se detalla el comportamiento de la implementación por defecto de QUIC, RR, cuyo aplicación se extiende desde la asignación de tiempos a diferentes servicios o procesos, así como a la planificación de paquetes. La implementación de QUIC, enfocada en la asignación de capacidades a diferentes *stream* que contienen paquetes, difiere de la metodología tradicional de RR en que en el caso de que haya espacio de datos sobrante se vuelve a repartir entre los *streams*. Asimismo, se considera una estrategia First In, First Out (FIFO) en las colas de cada *stream*, siendo el primer paquete que entra el primero que sale.

Su comportamiento se reduce a la asignación de capacidad del paquete QUIC en un orden secuencial a los diferentes flujos que intervienen. Es decir, se asigna cierta capacidad, y se sigue un orden circular, sin prioridad. Esto puede dar lugar a ineficiencias en el caso de que los *streams* tengan capacidades descompensadas; además la ausencia de prioridades podría no ser adecuado cuando un flujo posee información relevante para la conexión o la comunicación.

2.3.2. Fair Queuing

FQ pertenece a la familia de algoritmos de *scheduling* usada en ámbitos de procesos así como de redes. Su diseño se enfoca en conseguir equidad entre los *streams* al compartir un recurso

limitado. Esta estrategia asigna una fracción de recursos igual a cada uno de los flujos. La ventaja frente a otras estrategias es el hecho de que un flujo con mayor tasa binaria o generación de paquetes, no ocuparía más de la parte justa asignada.

Formalmente, con una capacidad de recursos C, siendo los recursos los bytes de datos en el paquete QUIC a compartir por cualquier número activo de *streams* N, se concede a cada uno de los flujos un reparto de C/N. En el caso de que uno de los *streams* demande menos capacidad que la asignada, es decir $r_i < C/N$, el restante es repartido nuevamente por los *streams* que continúen teniendo datos para enviar.

Weight Fair Queueing

WFQ surge como una extensión natural de su predecesor FQ. Mientras que FQ comparte la capacidad de los recursos en partes iguales, WFQ permite la pre-asignación de pesos a cada uno de los flujos, obteniendo la fracción de capacidad de los recursos que sera dada a cada uno de *streams*, $c_i = \frac{w_i \cdot C}{\sum_{\forall i} w_i}$.

Una problemática que surge es cómo asignar el valor de dichos pesos, w_i. Una opción razonable es que cada flujo tenga como peso el correspondiente a su taa, esto es: $w_i = \frac{Rate_i}{\sum_{\forall i} Rate_i}$. Sin embargo, esto requiere conocer de antemano las tasas, y no permite adaptación a cambios.

2.3.3. Back-Pressure

Teniendo en cuenta la estabilidad de las colas para cualquier carga de tráfico y capacidad, se evalúa un algoritmo de *scheduling* BP, el cual está basado en la teoría de Lyapunov. En este caso su operación se basa en un sistema de colas, donde $Q_k(t)$ corresponde con la ocupación de las colas de los *streams* en un momento t, mientras que la decisión de *scheduling* se define como $\alpha_k(t)$.

La capacidad de transmisión de la conexión en cualquier instante de tiempo se modela mediante la variable aleatoria $w(t)$, cuyo valor depende de las condiciones del canal y el control de congestión. Por ello, cada cola se actualiza como se indica en la expresión 2.3:

$$Q_k(t+1) = max[Q_k(t) - b_k(t), 0] + a_k(t) \tag{2.3}$$

donde $a_k(t)$ y $b_k(t)$ son las variables de llegada y salida, respectivamente. Además, cabe indicar que $b_k(t)$ se corresponde con cada valor de decisión de la variable $\alpha_k(t)$, mientras que $a_k(t)$ es una variable aleatoria sobre la cual no se tiene control ninguno. En cada instante, se ha de tomar una decisión de *scheduling* de un conjunto $\mathcal{A}$ de opciones, con el objetivo último de estabilizar las colas, asegurando que la capacidad de transmisión no supere la del canal, $\sum_k \alpha_k(t) \leq \omega(t)\ \forall t$. Se ha demostrado [32] que esto se consigue tomando en cada *slot* de tiempo una decisión que optimiza el siguiente problema 2.4:

$$\begin{aligned} \max_{\alpha(t)} \quad & \sum_{k=1}^{N} Q_k(t) \cdot b_k(t) \\ \textbf{s.t.} \quad & \alpha \in A \end{aligned} \tag{2.4}$$

En resumen, este problema se reduce en asignar prioridad a aquella cola con mayor ocupación, hasta alcanzar la capacidad de transmisión del paquete correspondiente de QUIC.

2.4. Trabajos relacionados

En los últimos años ha aumentado notablemente la atención prestada a las comunicaciones NTN, en general, y LEO, en particular. Una de las principales razones es el reciente interés en explotarlas como tecnología habilitadora para redes 5G y B5G [1, 21]. En este sentido, Di *et al.* [7] propusieron una arquitectura de red para soportar Satellite Access Network (SAN) utilizando constelaciones LEO. Además de discutir la arquitectura propuesta, los autores enumeran las soluciones de gestión de red habilitadas por la adopción de LEO, así como sus aplicaciones potenciales. En línea con estos trabajos, Liu *et al.* introducen en [28] una SAN, y prestan especial atención a las nuevas líneas de investigación relacionadas con este paradigma de comunicación. Más allá de las perspectivas de la arquitectura correspondiente, existen trabajos que analizan la convergencia técnica de las redes terrestres y no terrestres. En este sentido, Leyva-Mayorga *et al.* proporcionan en [27] una visión general de la integración de la constelación LEO dentro de la red de acceso 5G/B5G, y se centran en caracterizar los enlaces físicos de la SAN resultante.

Los organismos de normalización también han empezado a considerar los satélites como habilitadores de las redes B5G. Así, los autores de [18] analizan los esfuerzos actuales en la integración de NTN y tecnologías 5G. El documento describe en primer lugar los actuales enfoques de estandarización del 3GPP, para discutir después los principales retos: (1) opciones de arquitectura; (2) técnicas de gestión de red; y (3) modificaciones necesarias en los terminales de los usuarios. Del mismo modo, Darwish *et al.* estudian en [6] cómo definen los organismos de normalización la red de acceso radioeléctrico basada en LEO, prestando especial atención a las normas 3GPP NR. También hay trabajos que proponen soluciones técnicas para redes de acceso basadas en LEO. Los autores de [26] proponen una solución de gestión de enlaces ascendentes para servicios mMTC-NB-IoT soportados por constelaciones LEO. La solución propuesta no requiere modificaciones en los dispositivos NB-IoT y presenta técnicas para reducir el desplazamiento Doppler diferencial.

Como puede verse, los trabajos mencionados sientan las bases de cómo pueden utilizarse las constelaciones LEO para la provisión de servicios en 5G/B5G. En este contexto, un segundo grupo de trabajos se centran en las capacidades de SAN en general, y LEO en particular, para permitir servicios de borde/nube asistidos por satélite, que es un escenario de aplicación de interés en este trabajo.

Xie *et al.* ofrecen en [46] una visión general de Satellite-Terrestrial integrated Edge Computing Networks (STECN), que combina las redes satelitales-terrestres con la computación de borde para mejorar la Quality of Service (QoS). El documento analiza los principios de diseño, las funcionalidades clave, así como algunos retos de este enfoque. En la misma línea, los autores de [45] proponen la combinación de MEC con LEO, promoviendo la denominada LEO Satellite Edge Computing (LSEC), e investigan la asignación de recursos y la descarga de computación en la red LSEC. Los autores plantean un problema de optimización no convexo que considera conjuntamente la asignación de recursos y de computación, y proponen un conjunto de transformaciones del problema original para convertirlo en convexo.

Basándose en la motivación de proporcionar servicios a los usuarios de tierra, Zhang *et al.* se centran en [50] en explotar Orbital Edge Computing (OEC), desarrollando un algoritmo para asignar recursos OEC Task Allocation (OEC-TA) que muestra un buen rendimiento en

términos de coste computacional. En el análisis de rendimiento de su algoritmo los resultados indican ser mejores tanto en retardo medio como en consumo energético, llegando a mejoras de hasta un 10 % y un 16,5 %, respectivamente. Otros autores como Zhou *et al.* se centran en el streaming multimedia basado en multicast en la próxima generación de redes inalámbricas y móviles. También analizan la utilización de redes jerárquicas no terrestres compuestas por satélites LEO, plataformas de gran altitud HAP y equipos de usuario User Equipment (UE). Sin embargo, los protocolos multidifusión existentes para NTN provocan un gran retardo y *jitter* debido a la pérdida de paquetes, constituyendo el principal objetivo de este trabajo. Para ello proponen un protocolo multidifusión fiable, overhearing NOMA repairing reliable multicast (ONRM), para reducir el retardo.

De forma similar, Tang *et al.* introducen en [41] un esquema de descarga para redes Cloud and Edge Computing LEO Satellites (CECLS) cuyo objetivo es minimizar el consumo total de energía de los usuarios de tierra. El problema binario original se transforma en uno más manejable, utilizando un método de relajación de variables binarias, que luego se resuelve explotando el Alternating Direction Method of Multipliers (ADMM). Por otro lado, He *et al.* abordan en [14] el edge-computing asistido por LEO. En particular, consideran una constelación de satélites LEO con capacidades de edge-computing, y un satélite GEO para mejorar las comunicaciones subyacentes. El trabajo propone una estrategia de selección de servidores, explotando la teoría de colas para estimar la carga computacional en los nodos de borde.

Otros trabajos asumen una integración total de los satélites y las redes terrestres existentes. Este es el caso de [42], que considera una topología con estaciones base y pequeñas células basadas en LEO, así como enlaces de backhaul terrestres y por satélite. Sobre dicha topología, los autores formulan un problema de optimización para minimizar el consumo de energía de toda la red, manteniendo las restricciones de QoS. El problema de optimización entero mixto resultante se descompone y resuelve utilizando una Deep Neural Network (DNN) y una aproximación convexa sucesiva, que contemplan tanto variables enteras como continuas.

Aunque estos trabajos comparten el marco de aplicación con el aquí se presenta, el objetivo es diferente y complementario. La mayoría de ellos persiguen proporcionar inteligencia de red para mejorar la integración con satélites LEO. Por otro lado, en el trabajo que se describe en este documento se busca la evaluación del rendimiento de los protocolos de transporte existentes (TCP) y futuros (QUIC) sobre estas redes. Así pues, se mencionan seguidamente diversos trabajos que han estudiado el comportamiento de QUIC, y cómo se comporta en comparación con TCP.

Por ejemplo, Shreedhar *et al.* evalúan en [38] el rendimiento de QUIC para cargas de trabajo de web, almacenamiento en la nube y vídeo en entornos no controlados. Sin embargo, los autores no prestan mucha atención a la red de acceso subyacente. De hecho, el documento menciona los enlaces residenciales como segmento de acceso, sin especificar si suponen conexiones cableadas o inalámbricas. Del mismo modo, Qian *et al.* también estudian en [34] el rendimiento de QUIC. En este caso, el trabajo se centra en la interacción con redes de acceso LTE-A. Los autores estudian el comportamiento de diferentes algoritmos de control de congestión, y amplían el funcionamiento básico de QUIC para dar soporte *multi-home*, desarrollando un prototipo que permite comunicaciones duales WiFi y LTE-A. Más cerca del escenario de computación en el borde/nube, Dizdarevi'c y Jukan evalúan el rendimiento de QUIC para el continuo IoT-borde-

nube en [8]. Los autores realizan la evaluación en dos escenarios diferentes: (1) IoT-edge, y (2) IoT-edge-cloud. La evaluación, que utiliza dispositivos RaspBerryPi, se centra en la arquitectura de la nube, mientras que se presta poca atención a la red de acceso. De hecho, el documento no detalla la tecnología de conectividad de acceso concreta que se utiliza para los dispositivos IoT, y se centra en la arquitectura de nube implementada sobre Ethernet.

Como se puede observar, existe un vacío en la evaluación de protocolos de transporte en general, y de QUIC en particular, sobre SAN y, más concretamente, sobre escenarios que integran redes terrestres y no terrestres, existiendo pocos trabajos que hayan abordado esta evaluación.

En el pasado, Tsonuda *et al.* [43] analizaron el rendimiento de SCTP sobre redes LEO utilizando multi-stream adaptativo. Además, la característica multi-homing de SCTP también ha sido explotada sobre redes LEO en [51] para mejorar los traspasos. Centrándose en el protocolo QUIC, Yang *et al.* abordan en [49] la interacción entre redes LEO y QUIC, pero desde una perspectiva diferente. Se centran en Multi-Path QUIC (MPQUIC) y proponen un modelo de rendimiento para LEO (traspaso, interrupción, etc.). Basándose en ese modelo, el artículo propone un nuevo algoritmo de "inicio rápido" (opuesto al conocido *slow-start*) para el control de la congestión. Como puede verse, el objetivo de este trabajo es diferente al descrito en este documento, ya que se pretende realizar un análisis del rendimiento de las implementaciones existentes en la actualidad, y extensiones relativas a la funcionalidad *multi-stream*.

Asimismo, Martin y Khademi analizan en [29] la idoneidad del control de congestión Bottleneck Bandwidth and Round-trip propagation time (BBR) para QUIC sobre redes GEO. Los autores exploran el potencial de los satélites GEO de alto rendimiento, convertidos en parte integrante de las redes celulares de quinta y sexta generación, y cómo el creciente despliegue del protocolo QUIC en el espacio de usuario, y el control de congestión BBR, pueden afectar a las rutas de Internet que atraviesan enlaces por satélite, debido a que el rendimiento del transporte depende de la elección del control de congestión y de la implementación de QUIC.

El trabajo más cercano es el realizado por Yang, Li y Wu [48]. Modelan el canal inalámbrico con Matlab Satellite Tool Kit (STK) y utilizan las trazas correspondientes para analizar el rendimiento de QUIC sobre constelaciones LEO. También utilizan la emulación sobre dispositivos reales, pero emplean enlaces cableados de alta capacidad, cuyas características (retardo y tasa de pérdida) son modificadas estadísticamente en base al análisis anterior. Este trabajo se distingue del previamente mencionado porque tiene un enfoque de evaluación bastante diferente, ya que [48]: (1) no tiene en cuenta la longitud de los búferes, por lo que no considera el retardo autoinfligido; (2) se supone que las pérdidas en las capas inferiores se deben a las fluctuaciones de la capacidad del canal, lo que refleja el hecho de que una reducción de la SINR conllevaría una reducción de la capacidad, mientras que en el modelo que se presentará posteriormente también se contemplan las posibles pérdidas debidas a la saturación de los búferes; (3) se centra en archivos pequeños (páginas web de unas decenas de `kB`), mientras que en este trabajo se analiza el rendimiento de la transferencia de archivos grandes; (4) al tener en cuenta la capacidad del canal, se puede analizar el impacto de la tasa de aplicación.

Por último, cabe mencionar que ninguno de los trabajos mencionados ha abordado realmente el análisis de las técnicas de *scheduling multi-stream*. En un enfoque distinto, Cui *et al.* proponen en [5] la integración y adaptación de Dynamic Adaptive Streaming over HTTP (DASH) en el

protocolo QUIC, soportando de esta manera la multiplexación de varios *streams*. Esto permite mejorar la eficiencia de las comunicaciones, y reducir el retardo en aplicaciones como el *streaming* de vídeo. En el trabajo se describe la integración de DASH sobre QUIC, así como la evaluación sobre `ns-3`, realizando una comparativa con la versión tradicional sobre TCP.

Como se puede observar, el trabajo presentado en este documento es diferente al mencionado, ya que en este último no se realiza una comparativa de diferentes estrategias de *scheduling* ni se llevan experimentos en entornos satelitales LEO.

Tal y como se ha mencionado anteriormente, no existen trabajos previos que propongan políticas de *scheduling* que exploten la característica multi-streaming de QUIC. De hecho, existen muy pocos trabajos que hayan evaluado realmente el rendimiento de esta característica. Por otro lado, algunas investigaciones han estudiado diferentes enfoques y mecanismos basados en una métrica basada en el retardo, introduciendo diferentes estrategias para gestionarlo.

Aunque no están especialmente pensados para QUIC, merece la pena mencionar algunos trabajos que han propuesto esquemas de scheduling que basan su estrategia en el retardo. Por ejemplo, Hai *et al.* proponen en [13] un algoritmo de encaminamiento BP óptimo basado en retardo llamado sojourn-time-based BP (STBP), que utiliza una métrica que considera el tiempo de permanencia acumulado para calcular el peso de cada decisión. Sus resultados, que se basan en un estudio de simulación, demuestran que el esquema propuesto mejora el retardo extremo a extremo al tiempo que garantiza la optimización del rendimiento. En el caso de la propuesta que se describe posteriormente, se utiliza el control dinámico de colas y la teoría de Lyapunov para proponer un planificador basado en el retardo, que sólo utiliza información local (de los búferes internos).

Por otro lado, existen pocos trabajos que hayan abordado las técnicas de scheduling, pero en el ámbito de las comunicaciones *multipath*, es decir, no como mecanismo para gestionar los múltiples *streams* que una conexión QUIC puede crear. Se puede destacar [31], donde los autores proponen un planificador que reduce el retardo extremo a extremo, manteniendo un alto rendimiento de la red. Se analiza formalmente las propiedades del esquema propuesto, cuyo rendimiento es evaluado tanto sobre MPTCP como sobre MPQUIC .Además, Viernickel *et al.* introdujeron en [44] una política de programación stream-to-path para MPQUIC, capaz de minimizar el efecto HOL y de reducir el tiempo necesario para establecer subflujos. Asimismo, se proponen en [37] otros esquema de asignación de *streams* a caminos en MPQUIC, en base a una caracterización de los flujos.

Teniendo en cuenta la revisión de la literatura, el trabajo presentado en este documento complementa el estado del arte mediante la evaluación de QUIC en entornos satelitales, y con la propuesta y evaluación de un *scheduler* a nivel de *stream* basado en retardo.

3 Entorno de ns-3

En este capítulo se presenta la implementación realizada en el entorno de simulación `ns-3`, junto con las funcionalidades desarrolladas para la creación de escenarios que incluyan canales satelitales y diversas topologías. Además, se describe detalladamente la metodología empleada para automatizar este proceso. Posteriormente, se presentan los resultados obtenidos de una campaña de validación que respalda el correcto funcionamiento de la implementación.

3.1. Entorno `ns-3`

Con el objetivo de emular topologías Radio Access Network (RAN) compuestas por constelaciones de satélites LEO, cuyos canales mmWave son dinámicos, se ha optado por el empleo del software `ns-3`, consiguiendo un entorno de simulación ligero para analizar el rendimiento de protocolos de capas superiores. Su elección se basa en la aplicabilidad concreta que se le va a dar, esto es el análisis del comportamiento de protocolos de transporte. Normalmente, los entornos mmWave son analizados a nivel de capas inferiores (MAC y PHY), los cuales se caracterizan por tener una gran precisión temporal, sin embargo, no es de interés en este caso, debido a que ese nivel de precisión puede generar un cuello de botella en simulaciones de tiempo real.

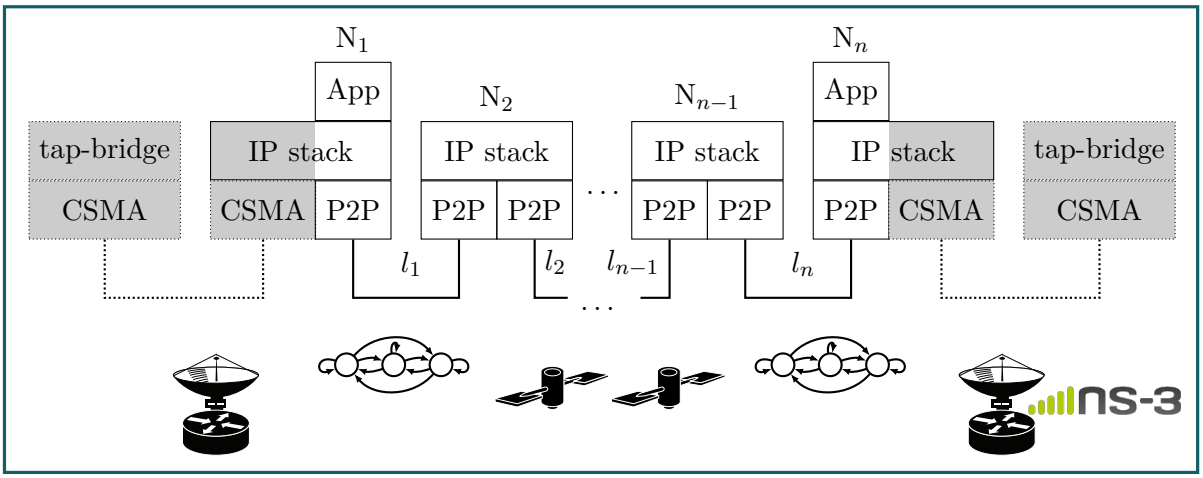


Figura 3.1: Diagrama del banco de pruebas de evaluación que integra NS-3, contenedores Docker y el modelo LMS.

Por otro lado, convencionalmente se emplean emuladores de enlace como Pantheon [47], que integra la herramienta MahiMahi [33], siendo capaz de imitar el comportamiento de mecanismos de control de congestión, así como configurar las características del enlace. Al mismo tiempo, se ha

demostrado que el comportamiento de los enlaces inalámbricos puede modelarse adecuadamente mediante cadenas de Markov, asumiendo diferentes estados de transmisión [9, 4, 15].

Con todo, la solución propuesta busca evaluar el rendimiento de protocolos de capa alta mediante enlaces simples, que se combinan para crear topologías más complejas, abarcando múltiples enlaces con diferentes características, a través del desarrollo de una herramienta de simulación eficiente y flexible, apoyada en `ns-3`, y que integra cadenas de Markov para reflejar el dinamismo de los canales subyacentes. La versión modificada de `ns-3` con los modelos propuestos, así como las utilidades de automatización que se han desarrollado se encuentran disponibles públicamente en el repositorio Github [1].

3.1.1. Adaptación clase `point-to-point`

La implementación llevada a cabo se aprovecha principalmente en la funcionalidad proporcionada por el enlace punto a punto, implementado por la clase `PointToPointNetDevice` de `ns-3`. Este dispositivo está compuesto por un `búfer` que recibe tráfico de la pila de protocolos y, a su vez, por una interfaz caracterizada por una determinada velocidad de transmisión y probabilidad de error.

Por tanto, el simulador utiliza originalmente la velocidad de transmisión y la longitud de los paquetes para programar las transmisiones a través de la interfaz. En su comportamiento más habitual, la velocidad de transmisión suele fijarse al principio de la simulación, permaneciendo estable a partir de entonces.

Por otro lado, la clase `PointToPointNetDevice` tiene un método público (`SetDataRate`) que permite cambiar la tasa durante la simulación, Figura 3.2. Sin embargo, se observaron algunos problemas al utilizarlo en casos extremos. En concreto, cuando la velocidad se fija a 0 (por ejemplo, si se pretende emular una desconexión temporal) para ser posteriormente incrementada, el dispositivo no era capaz de reiniciar la transmisión de los paquetes que estuvieran en cola. Para superar esta limitación, cada vez que se produce un cambio de velocidad o de estado se llama a la función `SetDataRate`, paso 1; en este momento se comienza a implementar la lógica interna, comprobando en primer lugar si el valor anterior era 0, si así fuera se comprueba la ocupación del `búfer`, paso 2. Posteriormente, si hubiera algún paquete en espera, paso 2, se iniciaría su transmisión (para lo que se llama a la función `TransmitStart`), paso 3, de forma que el dispositivo recupera su funcionamiento normal. En cualquier caso, cabe destacar que el cambio de velocidad de transmisión no es preventivo, por lo que los eventos de transmisión de paquetes no se reprogramaran ante un cambio de velocidad.

Integración del modelo

Para el modelado del canal, el trabajo se ha centrado en la integración de modelos basados en Cadenas de Markov, habituales para emular el comportamiento de enlaces inalámbricos. A modo de ejemplo, la Figura 3.3a ilustra una cadena que modela un canal con 3 estados. En ella se considera que los enlaces pueden experimentar diferentes condiciones, debidas al desvanecimiento y a la presencia de obstáculos, que a su vez repercutirían en la velocidad de

[1] https://github.com/tlmat-unican/Lightweight-ns-3-link-simulation

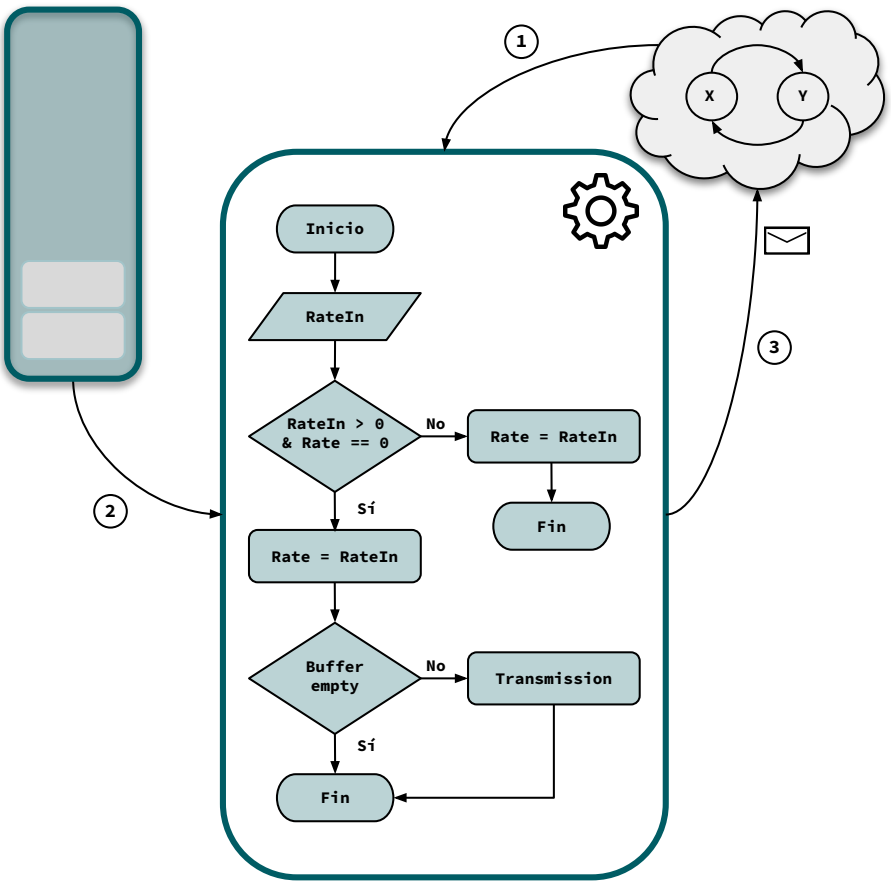


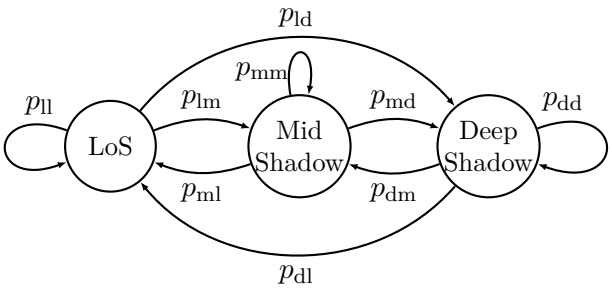
Figura 3.2: Diagrama de la lógica implementada en el método `SetDataRate` de la clase `PointToPointNetDevice`.

transmisión. Por lo tanto, teniendo en cuenta estas características ambientales, el canal transita entre los siguientes estados: Line of Sight (LoS), con condiciones ideales; Mid-Shadowing (MS), en el que las condiciones de conectividad empeoran; y Deep-Shadowing (DS), en el que la capacidad de transmisión se ve gravemente afectada. En cada estado, la capacidad de transmisión se considera constante. Del mismo modo, pueden utilizarse cadenas de Markov con 2 estados para modelar enlaces con velocidad constante e interrupciones, como puede verse en la Figura 3.3b, de modo que la capacidad de transmisión de uno de los estados sea igual a cero.

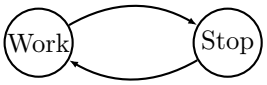
Para cada enlace del escenario, se implementa una cadena de Markov independiente para modificar la tasa de transmisión en una conexión punto a punto. De esta forma, cuando un enlace cambia a un nuevo estado, el simulador llama a la función `SetDataRate`, cuya lógica se observa en la Figura 3.2, del dispositivo punto a punto correspondiente, y programa un evento para salir del estado según la distribución de tiempos de permanencia. Este proceso se repite hasta el final de la simulación, independientemente del enlace y de si se están utilizando en ese momento, a diferencia de los emuladores de canal basados en trazas.

3.1.2. Definición del escenario

En esta sección se muestran el conjunto de utilidades que se han desarrollado para crear una topología personalizada basada en un escenario `ns-3` genérico. El escenario se alimenta con un



(a) Cadena de Markov de tres estados para el canal LMS.



(b) Cadena de Markov de dos estados para emular desconexiones temporales.

Figura 3.3: Cadenas de Markov que han sido integradas.

archivo de configuración JavaScript Object Notation (JSON), que describe completamente la topología de la red. En particular, define el número de enlaces del escenario, así como su tipo y características. Se contempla la posibilidad de abarcar desde enlaces más simples hasta aquellos más complejos. Para ello, se han incorporado enlaces que adoptan los modelos de cadenas de Markov de 3 y 2 estados anteriormente mencionados, así como enlaces ideales con configuraciones estáticas. En el caso de los enlaces basados en cadenas de Markov, se definen además subtipos, que representan configuraciones de diferentes bandas de frecuencia, utilizando datos de la bibliografía [9].

Por simplicidad, las configuraciones particulares de los modelos (matrices de transición, tiempos medios de permanencia, entre otros) se implementan en código `C++`, mediante el empleo de mapas y estructuras, mientras que el fichero de configuración indica el modelo que hay que cargar para cada enlace.

En todos los casos, el fichero de configuración también establece el valor de los parámetros heredados del enlace punto a punto y del dispositivo, y que no dependen del modelo adoptado: tasa de error, retardo, tamaño de Maximum Transmission Unit (MTU) y tamaño del `búfer`.

Por otro lado, la implementación permite inyectar tráfico de fondo TCP en cada enlace de forma independiente. El flujo de tráfico de fondo se implementa mediante una aplicación `ON-OFF`, y el fichero de configuración define sus principales atributos: tasa de tráfico, tamaño del paquete, número de bytes, y tiempos en los estados `ON` y `OFF`.

Por último, se instancian un cliente y un servidor para generar el flujo de tráfico que se analiza. Este flujo atraviesa la topología definida y se generan registros para seguir la transmisión y recepción de paquetes, la ventana de congestión, la ocupación de los búferes y la variación de la capacidad del canal.

La Figura 3.4 representa el diagrama de flujo de la implementación, es decir, los pasos a seguir con el fin de desplegar la topología. En este caso, con fines ilustrativos, se asume una simplificación de un solo enlace. Así, cada condición sería precedida por un bucle, que comprobaría cada uno de los enlaces (en este caso el único que hay). Como se aprecia, se despliegan en primer lugar los enlaces, comprobando previamente si son ideales, circunstancia en la que se emplearía el modelo

PointToPoint tradicional de ns-3. En caso de que sea la configuración por defecto de ns-3, así como el retardo nulo y no haya trafico de fondo, sólo se despliega aquel enlace con menor tasa, ya que será el que actúe de cuello de botella.

Seguidamente, se despliegan los enlace extremos de capacidad infinita así como los tap-bridges que posibilitan la conexión con los *host* reales de los extremos, en los que se encuentran las aplicaciones, en caso de que se haya activado su *flag* mediante línea de comandos. En caso contrario, se procede a la instalación de la aplicación ON-OFF en los extremos de la topología. Seguidamente, se llevan a cabo una serie de comprobaciones que añaden funcionalidades adicionales a la topología: en primer lugar, si la *flag* del tráfico de fondo está activada se procede a la instalación de las aplicaciones en los nodos extremos del enlace, donde tienen el rol de fuente y sumidero, respectivamente. La emulación de enlaces LMS, Figura 3.3a, se lleva a cabo por medio de la variación de la tasa de enlace mediante cadenas de Markov, y los métodos de la clase PointToPointNetDevice mencionados previamente. Finalmente, se procede a comprobar la existencia de algún enlace con la propiedad de desconexiones activada; en caso afirmativo, se procede con la misma metodología que en el caso de los enlaces LMS, emulando de esta manera interrupciones. Una diferencia clave con respecto al modelo previo es que en el estado de interrupción, la tasa de transmisión es nula, Figura 3.3b.

Concretamente, para implementar la funcionalidad de la máquina de eventos, representada en la Figura 3.5, se requiere de varios argumentos de entrada: un puntero a un objeto PointToPointNetDevice, un número de interfaz, una matriz de probabilidades de transición, una matriz de factores de capacidad, un valor inicial de capacidad (coincidiendo con el estado LoS), que es el estado con mayor tasa binaria. También se incluye un índice para identificar el enlace en cuestión al que afecta. A continuación, se realiza una serie de operaciones y cálculos mediante estos argumentos, Figura 3.5.

En primer lugar, se actualiza el estado actual mediante la función swap(), que selecciona un nuevo estado basado en las probabilidades definidas en la matriz M. Posteriormente, se calcula el tiempo de permanencia en el estado actual mediante la función time, que utiliza el estado actual y el índice del enlace para obtener el tiempo de permanencia en el estado correspondiente, el cual es modelado con una variable aleatoria exponencial negativa.

A continuación, se calcula el ancho de banda, basado en el factor de capacidad correspondiente al estado actual. El dispositivo de red se configura con este ancho de banda utilizando el método SetDataRate. Por último, se programa la próxima llamada a la función que implementa la lógica de la máquina de eventos en cuestión después de un cierto período de tiempo determinado. Este proceso se repite hasta que la transferencia de los datos ha finalizado, como se indica en la Figura 3.5, cuando la simulación termina.

Seguidamente, se explican tres casos de uso que pueden ser desplegados mediante la implementación llevada a cabo a través del archivo JSON mencionado anteriormente. Para ello se muestra la topología empleada, así como los modelos utilizados de ns-3. Es importante destacar que todas las topologías comparten un rasgo común: la implementación de enlaces PointToPoint como redes independientes, en las cuales cada nodo posee dos interfaces conectadas a sus redes vecinas. Mencionar que se puede llevar a cabo mediante *host* reales así como con el propio ns-3, instalando aplicaciones en los extremos de la topología.

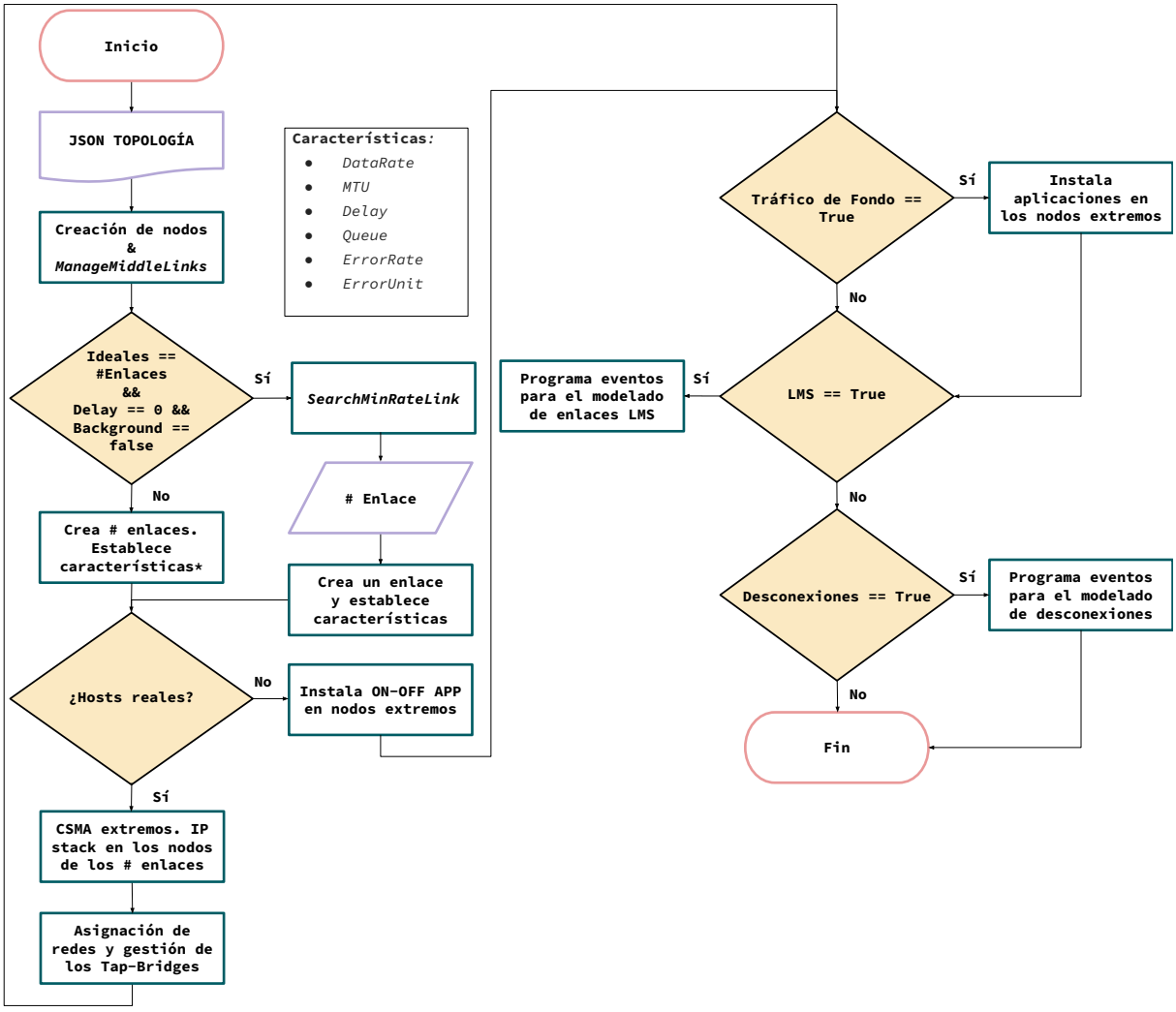


Figura 3.4: Visión conceptual de la implementación para el despliegue de diferentes topologías. Ejemplo desarrollado para solo un enlace.

En primer lugar, en la Figura 3.6 ilustra una topología donde hay un enlace LMS que conecta una estación terrena con un primer satélite LEO, lo cual encaja con un escenario típico de MEC. Para la explicación, se emplea la arquitectura donde se emplean *hosts* reales. Los extremos que generan datos se encuentran en contenedores `Docker`, conectados mediante `tap-bridges` al escenario `ns-3`, para lo que se emplea enlaces `CSMA` con capacidad elevada, de tal manera que sean transparentes a los enlaces de interés. Al tratarse de un escenario MEC se emplea un único enlace LMS, implementado mediante la metodología representada en la Figura 3.5.

Como segundo ejemplo, la Figura 3.7 presenta un enlace ISL donde hay interrupciones que podrían aparecer por distintas causas, como es la ausencia de línea de vista o por condiciones del entorno; como se ha explicado, estas interrupciones han sido modeladas mediante una cadena de Markov de dos estados, donde el estado de interrupción presenta capacidad nula. Se incorporan además dos enlaces LMS en los extremos, que conectan una estación base terrena con un satélite y viceversa. A modo de ejemplo, en el Anexo A.3 se indica la configuración para desplegar esta topología.

En ultimo lugar, como se aprecia en la Figura 3.8 se puede llevar a cabo un escenario donde

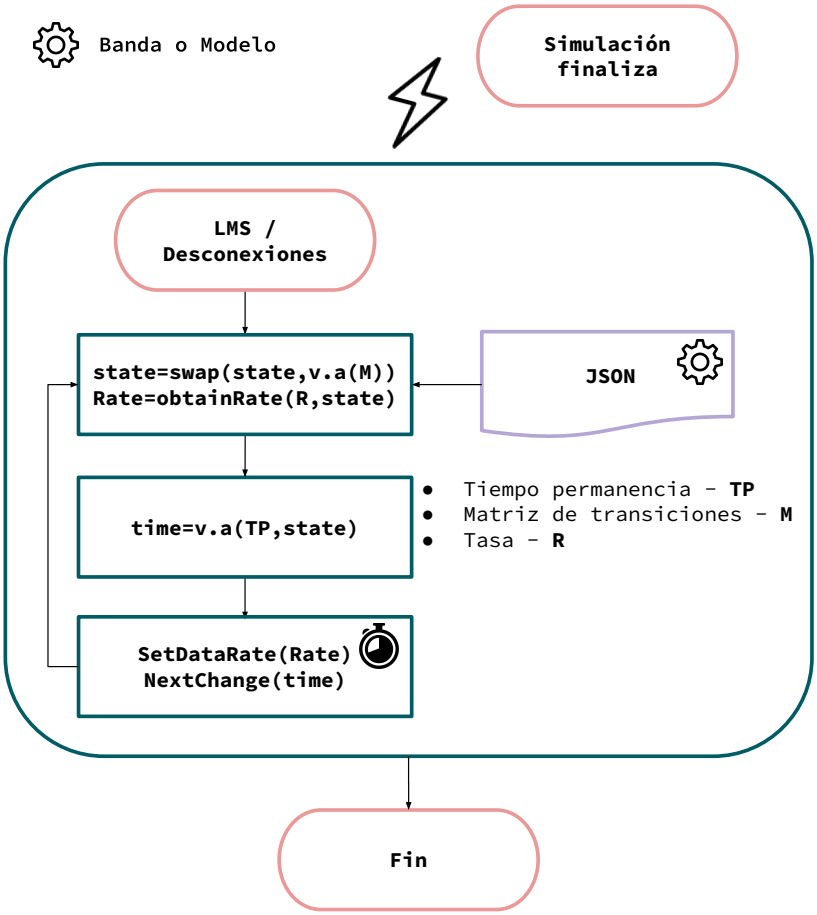


Figura 3.5: Lógica implementada de la función que modela la máquina de eventos encargada de la variación del canal.

el enlace ISL, a diferencia del anterior, es compartido. Esto se consigue instalando aplicaciones en los nodos extremos del enlace afectado, donde uno de ellos funciona como fuente y otro como sumidero, de tal menara que se envía una cantidad de datos fijada en el archivo de configuración mediante el protocolo TCP a una tasa concreta, se adjunta archivo de configuración en el Anexo A.4.

Como se puede observar, estos tres escenarios son ejemplos de utilización de la plataforma para emular entornos satelitales LEO, y para poder centrar la capacidad de análisis en las capas altas. Además, es importante destacar que el usuario puede reducir e incrementar la complejidad del escenario, a través del archivo de configuración.

3.2. Validación de la plataforma

En esta sección, se valida la metodología propuesta, así como el correcto funcionamiento de las utilidades implementadas sobre varios escenarios de comunicaciones LEO, cuyo esquema genérico se muestra en la Figura 3.1. Como puede observarse, el escenario emula dos enlaces LMS, que conectan estaciones terrestres y satélites. Además, se pueden añadir varios enlaces ISL, con diferentes características, véase interrupciones o tráfico de fondo. En los nodos que emulan las estaciones terrenas, se envía tráfico TCP, utilizando CUBIC como algoritmo de control de

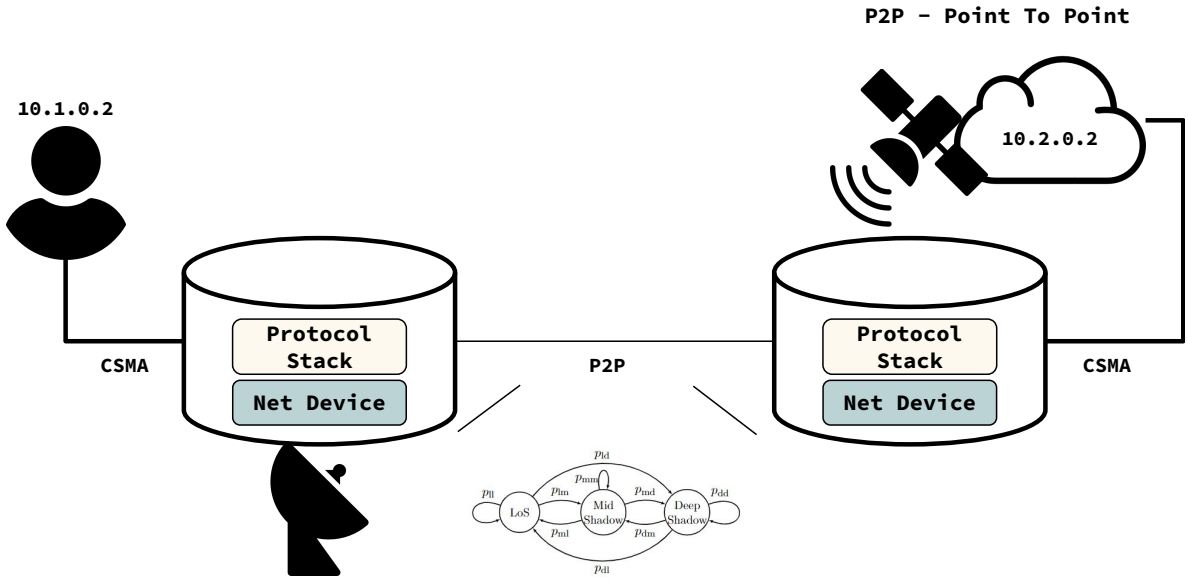


Figura 3.6: Topología de escenario con MEC en constelación LEO.

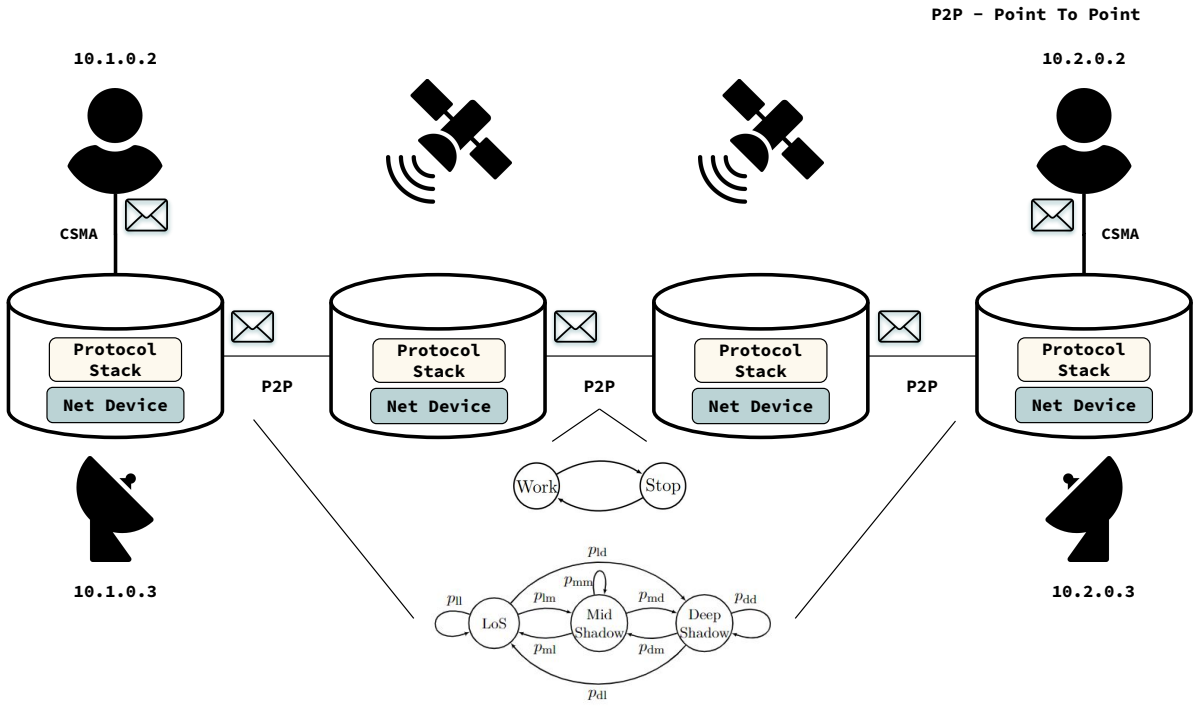


Figura 3.7: Topología de escenario global con tráfico de fondo en enlace intermedio con ISL.

congestión [12], con una tasa constante que se encuentra por debajo de la tasa promedio de los enlaces de acceso.

En primer lugar, se muestran los resultados obtenidos sobre un único enlace sencillo, controlado para validar el correcto comportamiento de la metodología propuesta. Posteriormente, se amplía la evaluación con una configuración más realista, abarcando tanto canales LMS como enlaces ISL, con tráfico de fondo y desconexiones. En la Tabla 3.1 se indican los parámetros de configuración utilizados en los experimentos llevados a cabo. A su vez, las matrices de transición que modelan en canal LMS se recogen, con más detalle, en la Tabla 4.4.

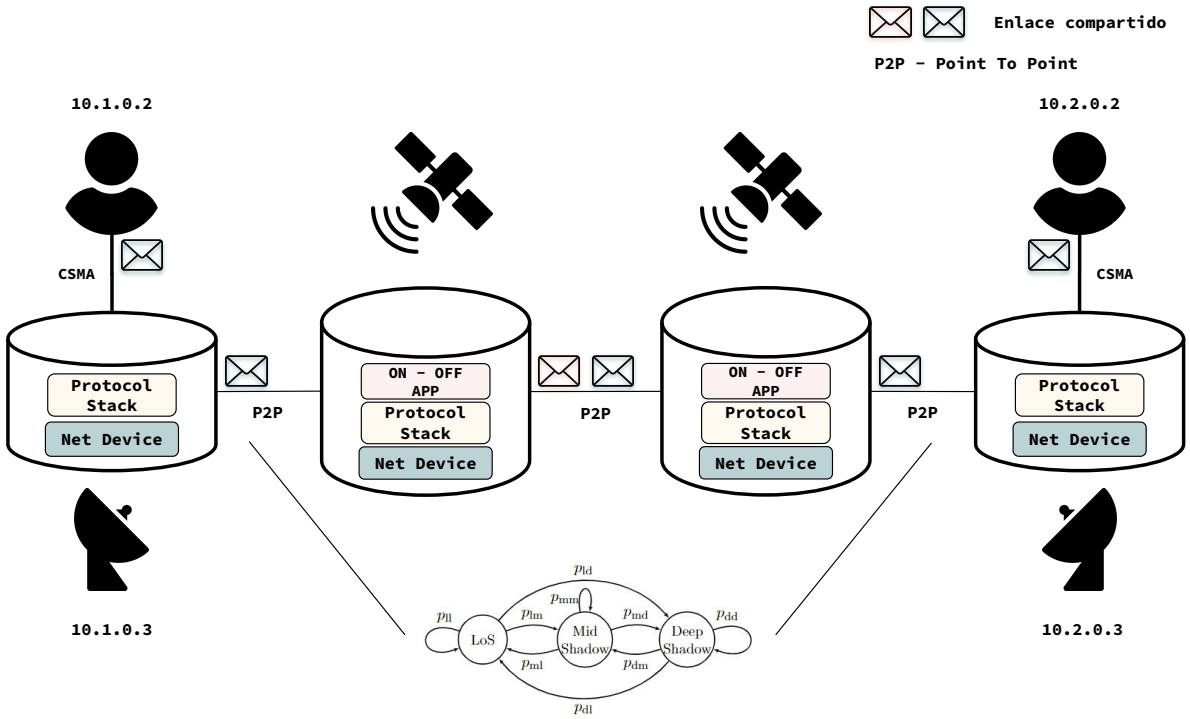


Figura 3.8: Topología de escenario global con tráfico de fondo en enlace intermedio con ISL.

3.2.1. Modelo sintético

Este escenario contempla un único enlace modelado como una cadena de Markov de 3 estados, Figura 3.3a, y una capacidad de transmisión en cada estado definida en la Tabla 3.1. La configuración emula el canal LMS definido en [16], pero con tiempos de permanencia constantes de 5 segundos, con el objetivo de hacerlo más predecible y poder analizar mejor el comportamiento de TCP y por ende, su mecanismo de control de congestión. Los valores de capacidad de transmisión se definen para emular las condiciones de los canales LoS, MS y DS, tal y como se menciona en la Sección 3.1.1.

En la Figura 3.9 se muestra la evolución temporal de la conexión, utilizando diferentes colores de fondo para reflejar cada estado. En primer lugar, la Figura 3.9a muestra la evolución de la ocupación del `búfer` (eje izquierdo) y de la ventana de congestión, `cwnd` (eje derecho). Como puede observarse, el `búfer` se satura durante los periodos MS y DS, y se vacía durante las condiciones LoS, con una tasa binaria mayor. En cuanto a la ventana de congestión, crece de forma constante durante los estados MS y LoS, función cúbica, y se reduce de manera brusca al principio de las fases DS, cuando se producen pérdidas debidas a la saturación del `búfer`, consecuencia de la baja capacidad del propio canal en ese estado.

La Figura 3.9b ilustra, para la misma simulación, la evolución del `búfer` y del retardo, definido como la diferencia entre el instante de transmisión de un paquete y su correcta recepción en el destino. En este caso, se puede observar que el retardo aumenta notablemente en la transición de los estados MS a DS. Además, se ve que cuando el canal está en la situación de DS se reduce bruscamente, como consecuencia del agotamiento de la ventana de congestión. Finalmente, el retardo alcanza su valor mínimo durante las condiciones LoS. Este comportamiento es coherente con el funcionamiento de TCP CUBIC, aunque se obtendrían patrones diferentes para distintos

Tabla 3.1: Configuración de los escenarios

Enlace LMS	
Modelo	Cadena de 3 estados Sintético y Realista [16]
Capacidad media del enlace	≈ 45 `Mbps`
Capacidad de los estados	$[80, 40, 16]$ `Mbps`
Capacidad de la cola	0,5· `BDP`
Retardo base	10 `ms`
Enlace ISL	
Modelo	Cadena de 2 estados Tasa constante
Capacidad de los estados	2 estados: $[0, 80]$ `Mbps` Tasa constante: 50 `Mbps`
Tasa del tráfico de fondo	$[5, 20]$ `Mbps`
Tiempo medio de interrupción	$[0, 3]$ `s`
Capacidad de la cola	Sin límite (∞)
Retardo base	10 `ms`
Aplicación	
Tasa de los datos de aplicación	40 `Mbps`
Tamaño del fichero	300 `MB`

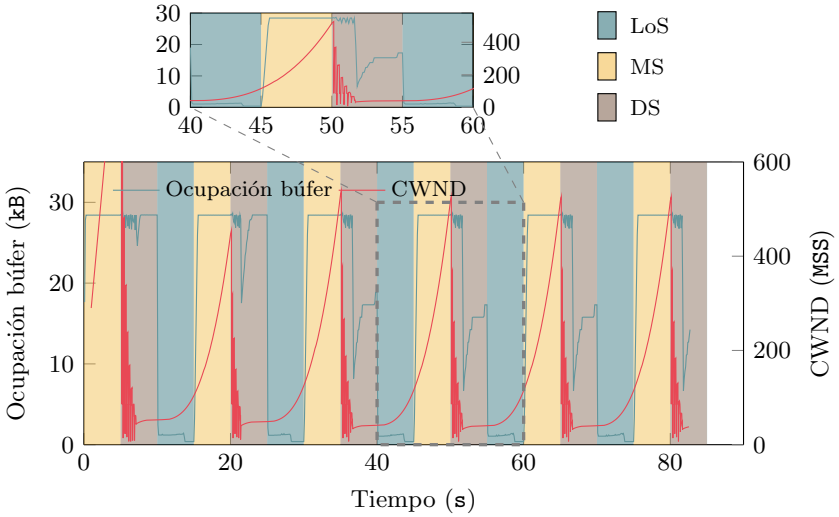
tamaños de `búfer` (se recuerda que su capacidad se ha limitado a medio Bandwidth-Delay Product (BDP), producto de la capacidad media del enlace y dos veces el retardo base RTT), según los valores recogidos en la Tabla 3.1), permitiendo observar así las pérdidas por saturación.

3.2.2. Análisis basado en comunicaciones LEO

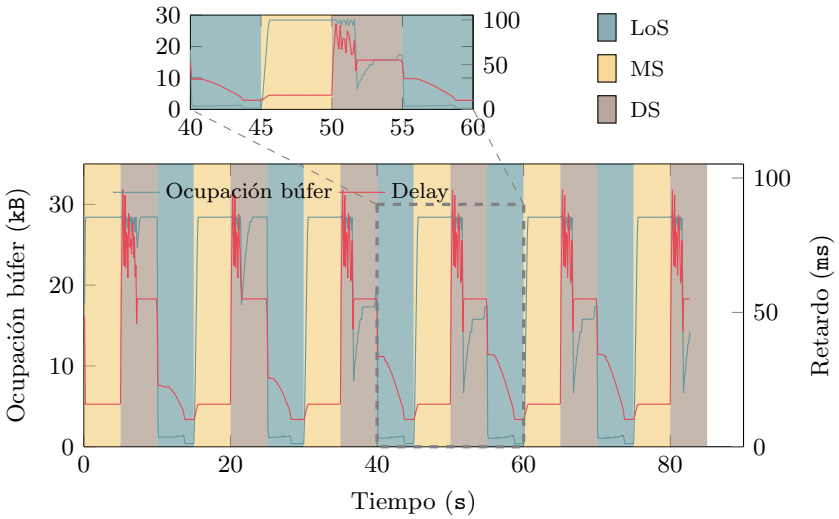
A continuación, se extiende la validación del entorno utilizando un modelo de canal realista para el enlace LMS. El modelo es de nuevo una cadena de Markov de 3 estados, con tiempos de permanencia distribuidos exponencialmente, y transiciones con probabilidades definidas en [16]. En primer lugar, se analiza el comportamiento de un único enlace LMS, para abordar posteriormente las comunicaciones extremo a extremo, abarcando dos enlaces LMS, ascendente y descendente, y enlaces entre satélites con tráfico de fondo e interrupciones.

La Figura 3.10 muestra los resultados que se presentaron anteriormente para el canal LMS 'sintético', en este caso para una configuración realista, topología Figura 3.6. De nuevo, los colores de fondo indican el estado del canal. Como puede observarse, existe una variación muy rápida de sus condiciones, cambiando de manera continua entre cualquier par de estados, lo que lo vuelve muy impredecible.

En la Figura 3.10a se muestra la evolución del `búfer` y la ventana de congestión a lo largo del tiempo. En este caso, la ventana de congestión apenas aumenta, debido a las continuas variaciones de capacidad, que dificultan la adaptación de la tasa de transmisión. Esto se evidencia en el detalle de la gráfica, que se hace en un intervalo de la conexión, donde se observa que el `búfer` está vacío incluso en situación DS, debido al pequeño tamaño de la ventana de congestión.



(a) Búfer vs. Ventana de congestión

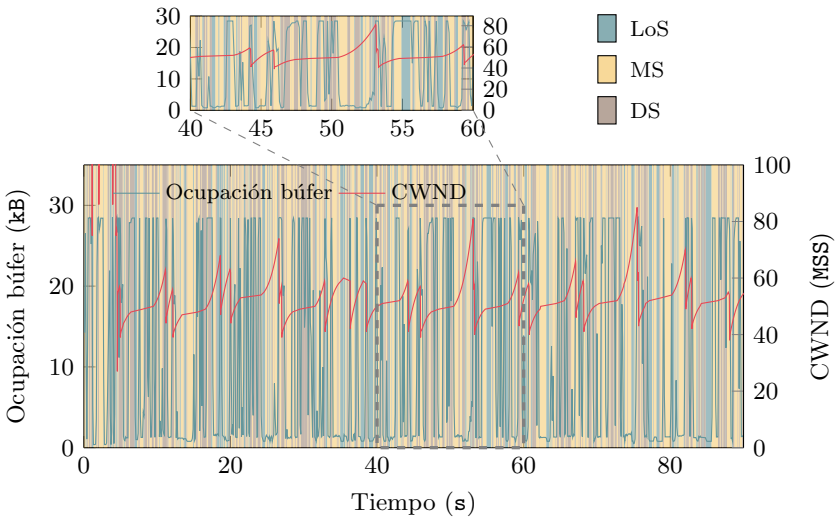


(b) Búfer vs. Retardo

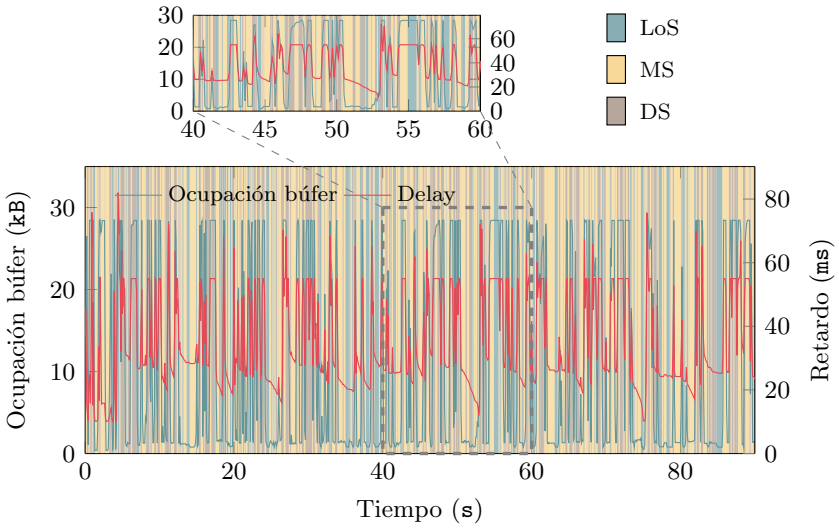
Figura 3.9: Creación de un único canal de 3 estados con un tiempo constante de permanencia de 5 segundos.

Asimismo, la Figura 3.10 muestra que el retardo del tráfico sigue la tendencia de ocupación del `búfer`. Dado que el control de congestión no es capaz de aprovechar toda la capacidad del canal, se envía menos tráfico, y el retardo es relativamente inferior al observado en la Figura 3.9b para el canal LMS sintético. De nuevo, los resultados son coherentes con el funcionamiento del mecanismo de control de congestión empleado por TCP CUBIC.

A continuación, se analiza el comportamiento extremo a extremo en un escenario con dos enlaces LMS, ascendente y descendente, y un ISL que los conecta, según la topología que se muestra en la Figura 3.7. El enlace ISL se modela como una cadena de Markov de 2 estados con tiempos de permanencia distribuidos exponencialmente y definidos como se ve en Tabla 3.1. El periodo activo se caracteriza con una tasa de transmisión de 80 `Mbps` (mayor que la capacidad media de los enlaces LMS realistas) y un tiempo de permanencia medio de 5 `s`; y el estado inactivo, que refleja las situaciones de interrupción temporal del enlace, tiene una tasa nula. Para el análisis, se va incrementando la duración del estado inactivo, en el rango de 0 a 3 segundos.



(a) Búfer vs. Ventana de congestión.

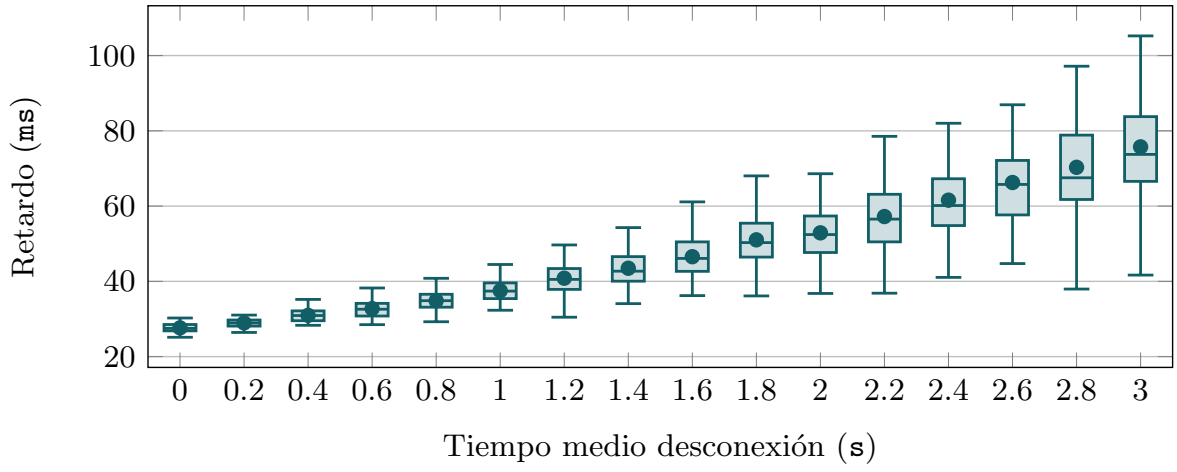


(b) Búfer vs. Retardo.

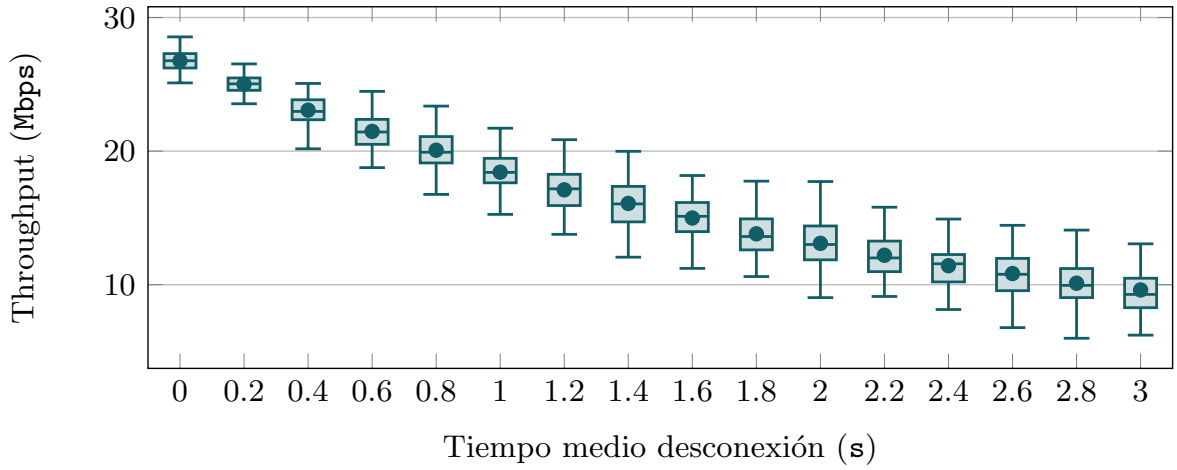
Figura 3.10: Realización de un único canal LMS y capacidad de búfer infinita

Para cada valor del tiempo medio de interrupción, se ejecutan 100 simulaciones independientes, en cada una de las cuales se envía un fichero de 300 MB. El experimento se detiene cuando la aplicación recibe todos los bytes.

La Figura 3.11 muestra la distribución del retardo medio extremo a extremo y el rendimiento observado, a medida que se va incrementando el tiempo medio de interrupción en el enlace ISL. Como se podía prever, en la Figura 3.11a se puede observar que el retardo aumenta de forma constante, y también lo hace su dispersión. A su vez, la Figura 3.11b muestra que el rendimiento disminuye como consecuencia tanto de la reducción de la capacidad debida a los tiempos de interrupción, como de la reacción del control de congestión ante dichas desconexiones temporales. Se destaca que esta situación se da incluso en el caso en el que el enlace ISL no presenta desconexiones, y no supone un cuello de botella a la topología global, debido a que en los extremos se sitúan sendos enlaces LMS, caracterizados por su gran variabilidad. A su vez, habría que resaltar que el tiempo de permanencia en cada estado se modela como una variable aleatoria exponencial negativa.



(a) Distribución del retardo medio.

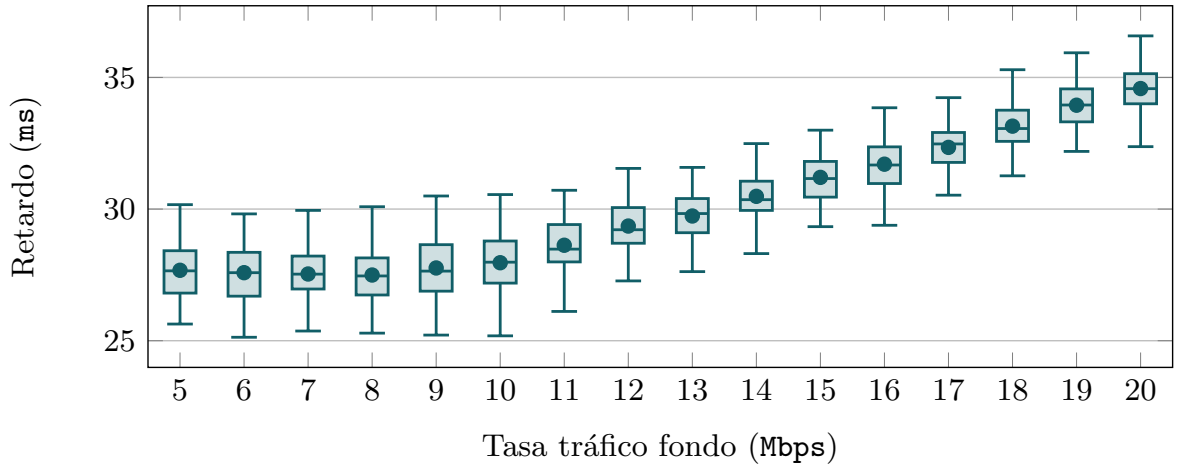


(b) Distribución del rendimiento medio.

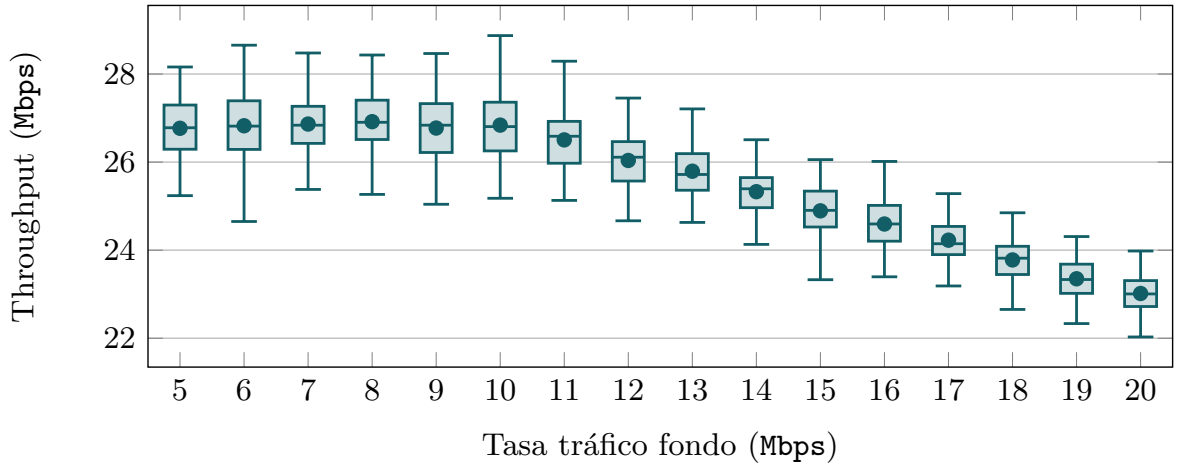
Figura 3.11: Impacto de las desconexiones en el enlace ISL para las comunicaciones extremo a extremo. Los resultados se obtienen a partir de 100 simulaciones para cada valor de tiempo medio de desconexión.

Por último, se ha configurado el enlace ISL con tráfico de fondo TCP. Para simplificar la evaluación de los resultados obtenidos, en esta configuración no se tienen en cuenta las interrupciones. Por lo tanto, el enlace ISL se configura con una capacidad constante (funcionamiento normal del enlace punto a punto) de 50 `Mbps`. Se realiza un barrido del tráfico de fondo en el enlace ISL de 5 a 20 `Mbps`, realizando 100 ejecuciones independientes para cada configuración. De nuevo, cada ejecución conlleva la transmisión de 300 `MB` y el experimento se detiene cuando se reciben todos los bytes. La Figura 3.12 muestra las distribuciones del retardo medio extremo a extremo y el rendimiento a medida que se incrementa el tráfico de fondo.

En primer lugar, en la Figura 3.12a se observa que el retardo extremo a extremo se mantiene bastante estable hasta que la tasa de tráfico de fondo alcanza los 10 `Mbps`. A partir de ese punto, el retardo aumenta constantemente, mostrando una variabilidad que no cambia de manera apreciable, debido a que el enlace ISL presenta una capacidad contante y el tráfico que se inyecta de fondo también. Mientras tanto, en la Figura 3.12b se muestra un comportamiento diferente para el rendimiento, que se mantiene bastante estable, en torno a los 27 `Mbps`, hasta que la



(a) Distribución del retardo medio.



(b) Distribución del rendimiento medio.

Figura 3.12: Impacto del tráfico de fondo en el enlace ISL sobre las comunicaciones de extremo a extremo. Los resultados se obtienen a partir de 100 simulaciones para cada valor de tráfico de fondo.

tasa de tráfico de fondo, en el enlace intermedio, alcanza los 10 `Mbps`, disminuyendo a partir de ese punto. Cabría esperar que el caudal del flujo extremo a extremo con bajo tráfico de fondo alcanzara un valor más cercano al promedio de los enlaces LMS (45 `Mbps`). Sin embargo, es importante señalar que la falta de adaptación de la ventana de congestión tiene un gran impacto en el rendimiento, como se observó en el análisis previo, y esto es aún más acusado en presencia de dos enlaces LMS en los extremos de la conexión (enlace ascendente y enlace descendente), cuyo comportamiento es además completamente independiente.

Como se ha podido ver, la metodología propuesta permite acometer la evaluación de protocolos de capas altas sobre enlaces inalámbricos únicos, y topologías más complejas, con un enfoque ligero, evitando tiempos de simulación elevados. Así, aunque estas evaluaciones pueden realizarse utilizando modelos de simulación más precisos de las capas bajas (PHY, MAC) la complejidad subyacente aumentaría sustancialmente, y en diferentes circunstancias, su impacto en los protocolos de transporte podría no ser relevante. Además, la metodología propuesta permite la interacción con tráfico real, utilizando `ns-3` TAP, como se describirá en más detalle en el Capítulo 4. Por el contrario, las simulaciones con modelos muy precisos de la conectividad

subyacente podrían no adaptarse a una ejecución en tiempo real (dependiendo de diferentes factores: complejidad del modelo, el hardware y el escenario), lo que dificultaría la interacción con aplicaciones e implementaciones de protocolos de transporte reales.

4 Plataforma y estrategias de *Scheduling*

En este capítulo se aborda la integración del entorno de simulación `ns-3`, presentado previamente, con aplicaciones reales, utilizando diversos protocolos. En primer lugar, se describirán las implementaciones de las aplicaciones en lenguaje `GO`, prestanto especial atención a la implementación que usa QUIC (`quic-go`). A continuación, se proporcionará una visión general de la implementación, así como de las funcionalidades más relevantes que han sido consideradas. Finalmente, se describirán las estrategias de *scheduling* propuestas, detallando el modelo matemático en el que se basan, y realizando un análisis de su rendimiento, en el que se compara su comportamiento con el de soluciones tradicionales.

4.1. Entorno Global

En esta sección se describen los niveles de aplicación y transporte que se contemplan en el entorno, así como sus capacidades e interacción con `ns-3`. La plataforma que se usa para evaluar el rendimiento se representa en la Figura 4.1. Combina una implementación real del protocolo QUIC (usando virtualización y contenedores Docker) con la simulación de la conectividad subyacente, empleando la herramienta *ns-3*.

En primer lugar, como se muestra en la Figura 4.1 en color verde, se ha implementado el canal a nivel de enlace entre las aplicaciones. Facilita así la configuración del escenario mediante la incorporación de enlaces, así como la definición de sus características, creando diferentes topologías mediante el empleo de un archivo de configuración JSON. Además, habilita la posibilidad de emplear aplicaciones externas con tráfico real (simulado) o bien mediante la opción propia del entorno `ns-3`, que se vio con más detalle en la Sección 3.1.

Por otro lado, en amarillo se muestran las aplicaciones desarrolladas en el lenguaje de programación `GO`, ubicadas en contenedores Docker, cuyas características se detallarán en profundidad en la Sección 4.2. Cada contenedor consta de una aplicación (cliente o servidor), de tal forma que un cliente inicia la conexión con un servidor y envía tráfico.

Destacar que las dos capas (verde y amarillo, como se observa en la Figura 3.1 y 4.2) cuentan con archivos de configuración independientes, el primero de ellos (color verde en la Figura 4.2) define la topología y la características de los enlaces a atravesar, mientras que el otro (en amarillo) caracteriza el tráfico envido por las aplicaciones, así como la estrategia de *scheduling* que se ha escogido.

Con el fin de automatizar el despliegue de la plataforma se ha empleado el lenguaje de programación `Python`, orquestando distintas funcionalidades requeridas para la construcción del

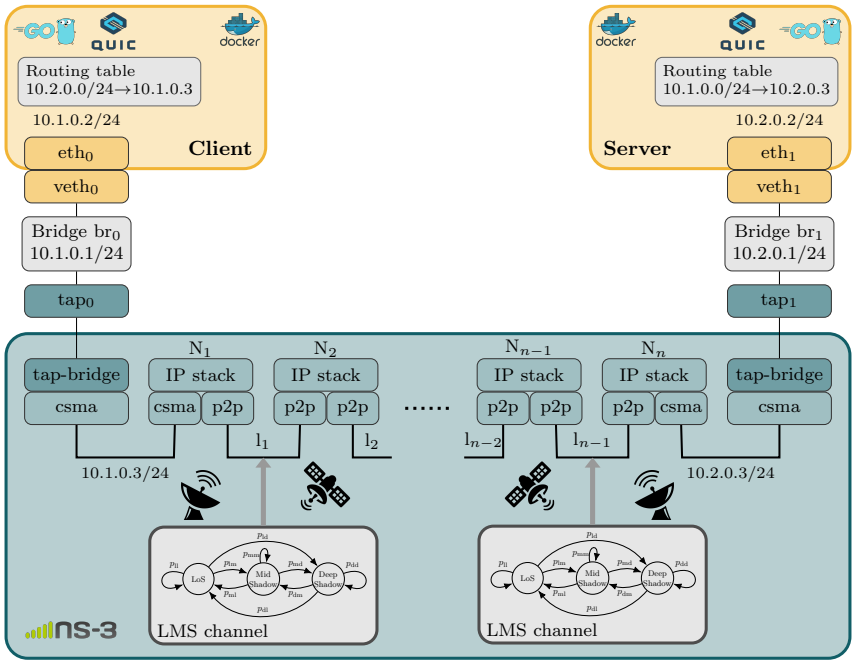


Figura 4.1: Diagrama de la evaluación de la plataforma, integrando `ns-3`, contenedores docker, y los modelos LMS así como ISL

escenario, la comunicación de las distintas partes que lo conforman, así como la gestión de los resultados, como se puede ver en la Figura 4.2. Por otro lado, en el propio *script* de Python se definen las características de la simulación que competen al nivel de aplicación, y el tipo de escenario a desplegar (especificando el nombre del archivo JSON), así como el número de simulaciones a realizar, entre otros, dependiendo del experimento que se quiera llevar a cabo. Finalmente, mencionar que la simulación tanto a nivel de aplicación como a nivel de enlace genera trazas, facilitando de esta manera el estudio de diferentes estrategias.

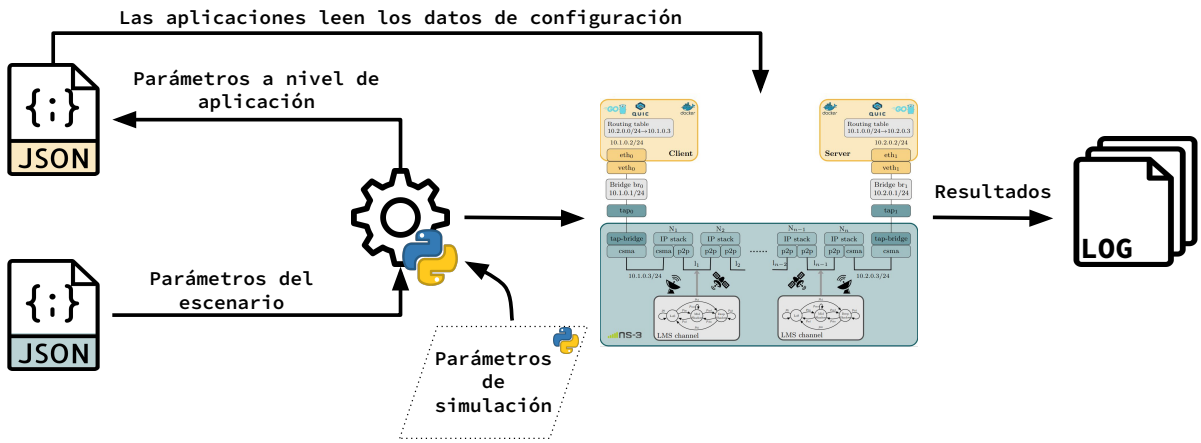


Figura 4.2: Diagrama de la metodología seguida para la definición de un escenario, construcción y simulación.

4.2. Implementación `quic-go`

En la actualidad existen numerosas implementaciones de QUIC, algunas de ellas con un enfoque centrado en proveer las características generales del protocolo, mientras que otras permiten integrar actualizaciones. En este trabajo se opta por la implementación desarrollada en el lenguaje `GO`, concretamente la versión `0.15.1`. Dicha implementación soporta de HTTP3 y la ultima versión, además, ha sido implementada con el objetivo de conseguir un rendimiento alto y eficiente.

Asimismo `GO`, también conocido como Golang y desarrollado por Google, incorpora una serie de ventajas importantes, como su sencillez, así como con su velocidad, comparable a la de otras alternativas, como C. Otra característica a desacatar es que la mayoría de los lenguajes anteriores están escritos para ejecutarse en un solo procesador, mientras que `GO` permite aprovechar características multiproceso. En definitiva, `GO` proporciona un gran equilibrio entre velocidad y sencillez. A continuación se citan los aspectos más relevantes de este lenguaje, que lo hacen atractivo para su uso:

- **Compilación rápida**: es un motivo fundamental para su propio desarrollo, con objetivo de aumentar su eficiencia. `GO` evita ralentizar el compilador al eludir algunas de sus metodologías. Mediante su gestión de dependencias, consigue mayor eficiencia y velocidad, operando en todos los módulos. Además permite la compilación cruzada, lo que le brinda la posibilidad de compilar un programa para diferentes sistemas operativos y arquitecturas desde una sola máquina. Este aspecto es especialmente útil cuando se desarrollan aplicaciones multiplataforma, o se necesita distribuir binarios para diferentes sistemas.

- **Concurrente y escalable**: ofrece características integradas para la programación concurrente, como *goroutines* y canales. Estas características permiten la ejecución concurrente de tareas, y facilitan el diseño de sistemas escalables y eficientes en términos del uso de recursos.

- **Eficiencia y rendimiento**: `GO` está diseñado para ser un lenguaje de programación eficiente y de alto rendimiento. Utiliza una recolección de código no necesario concurrente y una compilación estática, lo que le permite ejecutar aplicaciones rápidas y consumir poca memoria. Además, se trata de un lenguaje de tipado estático, capaz de realizar comprobaciones en tiempo de compilación, evitando errores en ejecución y tiempo de depuración.

- **Biblioteca estándar y comunidad activa**: este lenguaje cuenta además con una biblioteca estándar muy completa y bien diseñada, que ofrece una amplia gama de funcionalidades, desde operaciones básicas hasta protocolos de red, codificación/decodificación de datos, manipulación de archivos, entre otros. Esto evita la necesidad de depender en exceso de bibliotecas externas. Por otro lado cuenta con una comunidad de desarrolladores muy activa, brindando una gran cantidad de recursos, herramientas disponibles así como actualizaciones.

Estas son solo algunas de las ventajas del lenguaje de programación `GO`. Su combinación de rendimiento, sencillez, concurrencia y escalabilidad lo convierte en una opción atractiva para una

amplia gama de aplicaciones y proyectos. Concretamente, en este trabajo se ha empleado con el fin de desarrollar las aplicaciones cliente y servidor. A continuación, se procede a detallar las características más relevantes de las mismas, así como las modificaciones llevadas a cabo en la implementación `quic-go` para añadir *schedulers* a nivel de *stream*.

En primer lugar, la aplicación del lado del cliente tiene el propósito de iniciar la comunicación y la transmisión de datos. La implementación se ha ajustado para permitir la configuración de diversos parámetros como: el número de *streams*, sesiones, la cantidad de datos enviados por cada flujo y el tipo de tráfico. Por otro lado, para la gestión del *multi-streaming* se puede configurar el *scheduler* que se utilizará, junto con los parámetros necesarios para su funcionamiento. Con respecto al servidor, su funcionalidad se encarga de la recepción de los datos, así como del cierre de la conexión, una vez que ha finalizado la transmisión y se han recibido correctamente todos los datos. Resaltar que las aplicaciones tienen la capacidad de establecer comunicaciones usando los protocolos QUIC o TCP. En el caso de TCP la configuración relativa a los *streams* se ignora.

Tabla 4.1: Archivos de código más relevantes que componen el proyecto y su funcionalidad.

	Archivo	Funcionalidad
UC	`client.go`	Inicia la conexión, capacidad de enviar trafico TCP o QUIC.
	`server.go`	Recibe la información y cierra la conexión.
	`conf_scheduler.json` ⚙	Define el scheduler así como las características necesarias que requiere.
	`conf.json` ⚙	Define el tráfico que intercambian las aplicaciones.
	`uc-buffer.go` 📦	Registro del estado de las colas y almacenamiento.
`quic-go` 📦	`stream.go`	Define la estructura `stream` que implementa la interfaz `Stream`.
	`packet_packer.go`	Define la estructura `packetPacker` que se encarga de empaquetar los datos en paquetes QUIC.
	`framer.go`	Encapsula datos de aplicación en QUIC `frames` de acuerdo a una estrategia concreta, por defecto RR.
	`session.go`	Define la estructura `session` que representa una sesión QUIC. Gestiona las conexiones a nivel de `stream`.

En la Tabla 4.1 se enumeran los archivos de código más relevantes y su funcionalidad. Además de los archivos mencionados, servidor y cliente, `uc-buffer.go` se encarga de la gestión de colas. Los archivos mencionados conforman componentes externos a la implementación. Para separar la gestión de datos del cliente y del servidor, se ha integrado el archivo dedicado a la gestión de colas dentro del paquete `quic-go`, importado así en ambos componentes (servidor y cliente).

La inclusión de la gestión de colas dentro del paquete `quic-go` conlleva diversos beneficios en términos de organización y modularidad del código. Esta estructura permite un mejor mantenimiento del sistema, y le confiere una mayor capacidad de ampliación, al mantener una estructura clara y coherente en la implementación. De esta manera, se importa el paquete

`quic-go` tanto en el lado del cliente como del servidor, cada uno de los cuales se ejecuta en un contenedor. Esto permite que sea únicamente necesario modificar los archivos de configuración, utilizados por las aplicaciones.

4.2.1. Implementación

A nivel de aplicación, el escenario creado consta de un cliente que envía datos al servidor, ubicado en el otro extremo de la red, como se aprecia en la Figura 4.4. El envío de datos se configura por medio de un archivo JSON, donde se definen características como el protocolo a emplear (QUIC o TCP), la dirección Internet Protocol (IP) con la cual establecer una conexión, el tamaño de *slot* (utilizado en la generación de paquetes), la cantidad de datos a enviar, la tasa binaria, así como el tipo de tráfico a emplear. Así, la lógica que recae en el cliente permite enviar tráfico con distintas distribuciones: uniforme, constante, Poisson y lognormal. En caso de que el protocolo escogido sea QUIC, se da la posibilidad de configurar el número de sesiones que se pueden abrir y por ende, modificar las características previamente mencionadas a nivel de *stream*.

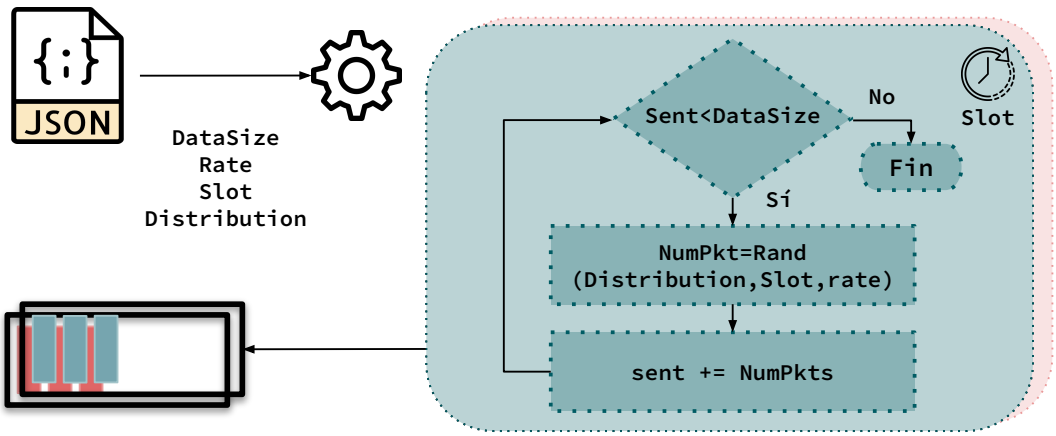


Figura 4.3: Generación de trafico a nivel de aplicación.

Como se ha mencionado anteriormente, una de las funcionalidades de la metodología propuesta es la posibilidad de generar tráfico siguiendo ciertas distribuciones mencionadas, utilizando el método `Rand`, que incluye el paquete `distuv` de `GO`. A partir del mismo, y en función de la tasa binaria indicada, el tamaño de *slot*, así como el tamaño de paquete, se determina el número eventos, es decir, paquetes que se transmitirán en una ranura de tiempo, como se observa en la Ecuación 4.1.

$$N_{pkts} = \frac{\bar{R}[Mbps]}{8 \cdot Pkt_{size}} \cdot T_{slot}[ms] \tag{4.1}$$

Este proceso se repite hasta enviar el tamaño total del fichero, y de manera periódica (duración del *slot*, fijado en el archivo de configuración). Además, se ejecuta para cada `stream` de manera independiente, a través del empleo de *goroutines*. A su vez, como se observa en la Figura 4.3 los datos se almacenan en un `búfer` gestionado por el archivo `uc-buffer.go`.

En lo que se refiere al lado del cliente, éste es en cierta manera configurable. En una primera instancia se procede al establecimiento de la conexión, que el cliente inicia a un servidor remoto. Seguidamente, se procede a la creación de las sesiones que se requieran y el número de *streams* internos a cada sesión (indicado en el archivo de configuración). Es relevante mencionar que la conexión gestiona aspectos a nivel de transporte, mientras que la sesión maneja las interacciones a nivel de aplicación, así como los *streams*. Además, la sesión emplea la conexión subyacente para el envío y recepción de datos.

Posteriormente, se procede al envío de datos, como se puede ver en la Figura 4.4, a través de la creación de una *goroutine* encargada de la generación de trafico por cada *stream*, según la distribución indicada. Dichos datos son almacenados en un `búfer` externo de la propia implementación. Por otro lado, otra *goroutione* comprueba dicha cola y procede a escribir en el *stream*, mediante el método `stream.Write()`; al dividir estas dos funcionalidades en dos procesos distintos se evita el bloqueo del *stream* y, en consecuencia, de toda la conexión. Este proceso se repite hasta enviar todos los datos indicados, siendo cada uno de los flujos independiente del resto.

Por otro lado, internamente, teniendo en cuenta el flujo de los datos para la construcción de un paquete, se siguen los siguientes procesos. En primer lugar, como se indicó previamente, se usa el método `Write()` para escribir en el *stream*; en segundo lugar, se procede al envío de datos, a través de la interfaz *framer*, encargándose de seleccionar los *frames* de los distintos *streams* que componen un paquete QUIC. Estos datos se devuelven a la función de la interfaz *packetPacker*, que se encarga de componer los paquetes. Finalmente, la información se pasa a nivel de sesión, que se encargará de realizar el envío de los paquetes.

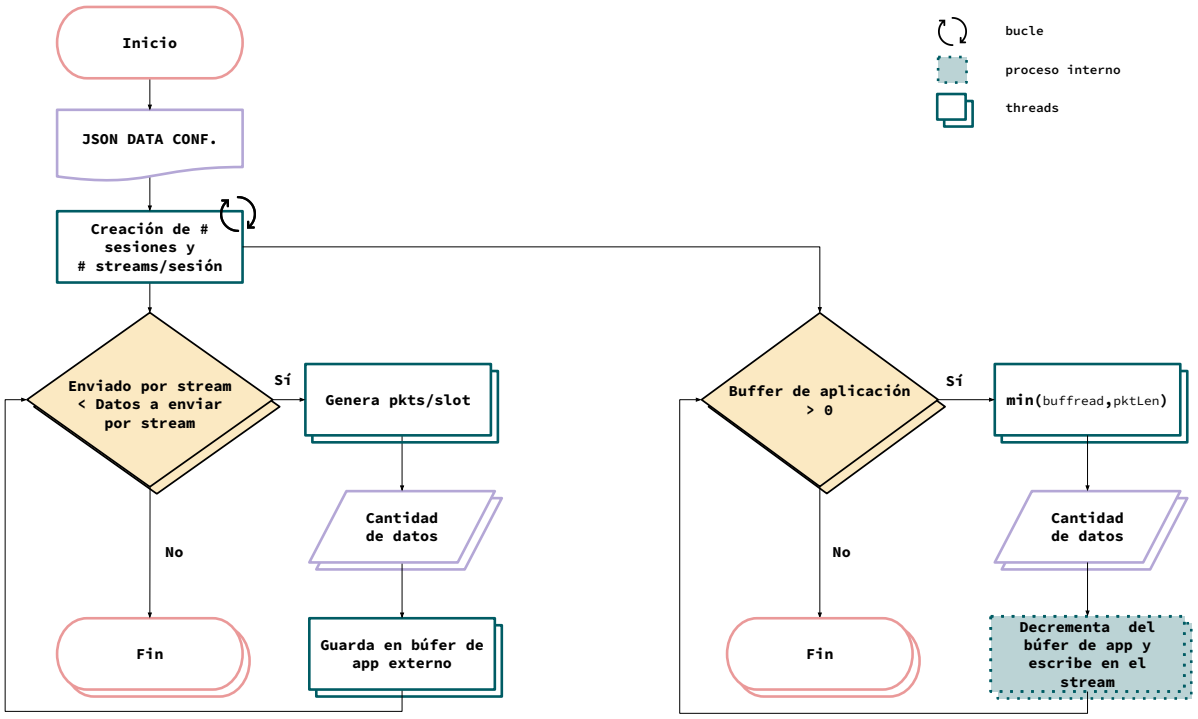


Figura 4.4: Diagrama de procesos que intervienen en le envió de daros en el lado del cliente.

Haciendo hincapié en la interfaz *framer*, se puede observar que sigue una metodología

común, independiente del algoritmo aplicado. En primer lugar, comprueba el estado de las colas, guardándolo. A continuación, esa información se pasa a la lógica del algoritmo en cuestión, que ordena según los requisitos en los que se basa, o establece la capacidad seleccionada de cada uno de los flujos, hasta alcanzar la cantidad máxima de datos permitida por el propio paquete. En la Sección 4.3 se detalla la implementación del *scheduler* propuesto.

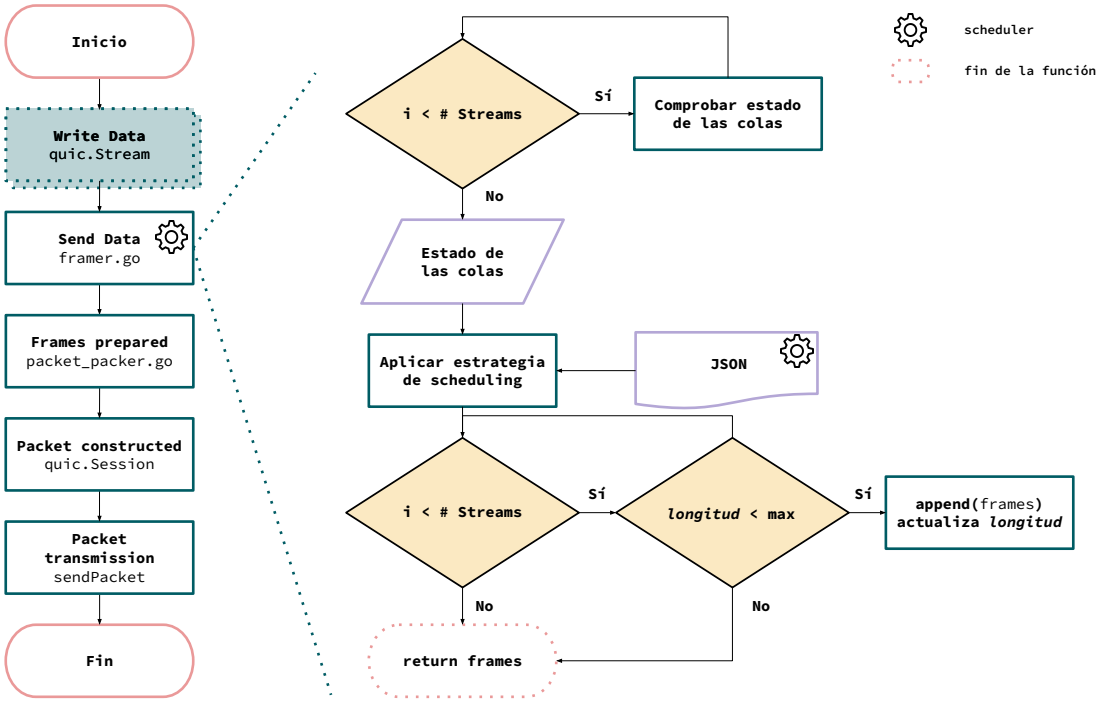


Figura 4.5: Diagrama de flujo con los elementos principales que intervienen en la composición de paquete.

4.2.2. Otras herramientas

A continuación, se explican las distintas herramientas Docker, UML-Utilities y iproute2, utilizadas para el despligue de la plataforma y su la integración de las aplicaciones con `ns-3`. Estas herramientas son utilizadas para facilitar la configuración de redes virtuales, y simular entornos de red complejos y realistas. Docker se utiliza para empaquetar y distribuir aplicaciones, las utilidades User-mode Linux (UML) para crear instancias de UML y configurar las interfaces de red virtuales; por otro lado, iproute2 se emplea para manipular la configuración de red dentro de esas instancias.

Docker

En primer lugar, Docker es una plataforma de desarrollo, transporte e instalación de aplicaciones de código abierto, que permite separar las aplicaciones de la infraestructura subyacente, facilitando un despliegue de software de manera más rápida.

Docker proporciona la capacidad de empaquetar y ejecutar aplicaciones en entornos aislados llamados contenedores. Esto resulta especialmente útil para simular diferentes tipos de redes

utilizando software, ya que puede ser complicado o costoso implementar dispositivos reales. Cuando el hardware no es un factor relevante, pero las funciones esenciales sí lo son, las herramientas software se convierten en una alternativa de interés.

En el contexto de este trabajo, se despliegan dos contenedores Docker, para el cliente y el servidor. Cada uno de ellos se basa en una imagen mínima llamada *alpine*. Además, para el despliegue de las aplicaciones se ha instalado el entorno `GO` para su correcta ejecución, así como otras herramientas relevantes. Estas configuraciones personalizadas se definen mediante el uso de un archivo llamado *Dockerfile*, en el que se describen las características específicas.

Además, para automatizar el proceso, se ha utilizado *Docker Compose*, que permite crear y gestionar dos contenedores en dos subredes independientes, adaptando las tablas de rutas para que puedan comunicarse entre sí. A su vez, también se crean *birdges* para facilitar la conectividad a través de ellos, lo que permite la integración con `ns-3`.

Tabla 4.2: Tabla de rutas de los contenedores Docker.

Dispositivo	Red	Máscara	Puerta de enlace
Cliente - sender	10.1.0.0	24	10.1.0.3
Servidor - server	10.2.0.0	24	10.2.0.3

Concretamente, el Cliente/*sender* se configura para que se reinicie siempre. Dentro del contenedor se ejecuta un comando para la configuración de la tabla de rutas y, posteriormente, se ejecuta otro comando con la finalidad de que permanezca activo. Además, se asigna una dirección estática al contenedor de la red *leftnet* (10.1.0.2).

Por otro lado, se definen dos redes con características similares. Siguiendo la explicación previa, *leftnet* es una red de tipo puente (*bridge*), a la cual se la asigna un rango de direcciones IP. Asimismo, se configuran opciones adicionales de red, como habilitar el enmascaramiento IP y asignar nombres a los puentes de red. De manera similar, este procedimiento se completa para el lado del servidor. Para más información véase en el Anexo A.1.

UML-utilities

Es una tecnología que permite ejecutar múltiples instancias de sistemas operativos Linux en un único sistema físico. Las utilidades UML son un conjunto de herramientas que facilitan la configuración y gestión de las instancias UML, configurar las interfaces de red y administrar los recursos del sistema. Concretamente se emplea el comando `tuncl`, el cual crea dispositivos virtuales de red que se emplean para establecer una conexión entre una instancia *uml* y el sistema operativo host. El *tuncl* se puede considerar como una interfaz virtual de red, que permite la comunicación entre el entorno virtual de UML y el entorno físico del sistema operativo host.

El término *tuncl* se deriva de la combinación de las palabras *tun* y *clone*. *tun* se refiere a los dispositivos de red de capa 3 en el kernel de Linux, mientras que *clone* indica que se trata de una interfaz virtual que se clona del sistema operativo host hacia el entorno virtual de UML.

La principal funcionalidad de *tuncl* es proporcionar una forma de enviar y recibir paquetes de red entre una instancia de UML y el sistema operativo host, permitiendo así la comunicación

tanto interna como externa. Al configurar *tuncl*, se puede asignar una dirección IP y configurar otros parámetros de red, lo que permite que UML se comunique con otros dispositivos o redes.

En este trabajo el comando *tuncl* se ha utilizado para generar los interfaces virtuales ($\texttt{tap}_0$ y $\texttt{tap}_1$, véase Figura 4.1) que permiten la interacción de aplicaciones reales con `ns-3`.

Iproute2

Iproute2 es una suite de utilidades de línea de comando que permiten administrar y configurar diversas funcionalidades de red en sistemas Linux. Entre sus características se encuentra la capacidad de crear y gestionar interfaces de red, incluyendo interfaces Terminal Access Point (TAP), que son útiles para la simulación y configuración de redes virtuales.

Concretamente se ha empleado para la gestión de los TAPs, habilitando el modo promiscuo, que permite que una interfaz de red capture y procese todos los paquetes que llegan a la red, incluidos aquellos que no están destinados específicamente a la MAC de dicha interfaz. Por otro lado, se emplea la opción de activar, con el fin de poner en marcha la interfaz de red, haciendo que esté en un estado operativo. Una vez que la interfaz TAP esté activada, podrá participar en la comunicación, y enviar y recibir paquetes de datos.

4.3. Delay Back-Pressure

En la Sección 2.3 se presentó la definición de los algoritmos de *scheduling* o planificación, los cuales se caracterizan por plantear un conjunto de reglas y procedimientos que establecen, en este contexto, el orden en el que se asigna la capacidad disponible en el paquete de QUIC a los diferentes flujos. En último lugar, se analizó un algoritmo de *scheduling* cuyo método de control de colas está basado en el criterio de Lyapunov, basado en la ocupación. A continuación, se aprovecha esta técnica para controlar el retardo. El criterio de Lyapunov se basa en la teoría de estabilidad de sistemas dinámicos, y en este trabajo se plantea un enfoque para diseñar algoritmos de control, con el objetivo de minimizar el retardo en la cola.

Estos algoritmos se basan en última instancia en la asignación de prioridad, lo que implica que en cada decisión de asignación de capacidad de datos a los *streams* se tiene en cuenta el estado de las colas, determinándose un orden de asignación. Como modificación del algoritmo BP, se introduce una nueva métrica de retardo, llamada tiempo de permanencia.

Tabla 4.3: Símbolos y definiciones

Símbolo	**Definición**
	Variables aleatrias
$a_k(t)$	Llegadas del stream k en el slot t (`bytes`)
$b_k(t)$	Salidas del stream k en el slot t (`bytes`)
$w(t)$	Capacidad de envío en la ranura t
$Q_k(t)$	Valor de la cola de la aplicación k en el slot t
$\hat{Q}_k(t)$	Retardo acumulado basado en el tiempo de permanencia de los paquetes en cada transmisión k
$\alpha_k(t)$	Decisión variable

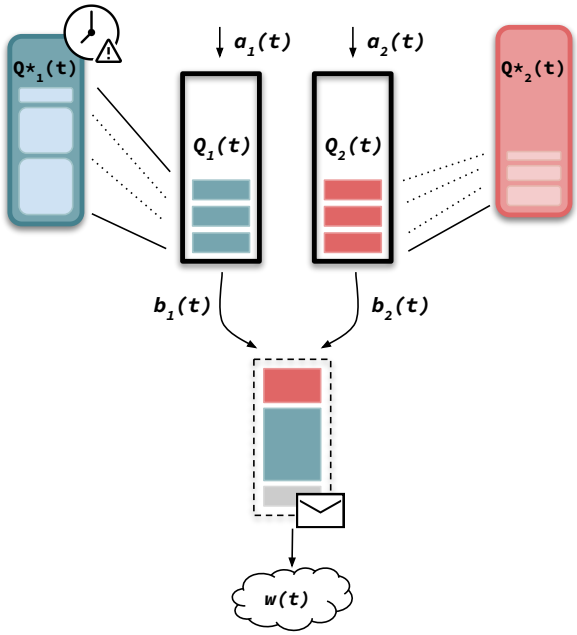


Figura 4.6: Diagrama del modelo del sistema propuesto.

En primer lugar, el algoritmo de *scheduling* es a nivel de *stream* y considera que cada uno de los flujos cuenta con una cola $Q_k(t)$ en la que se almacena su información (bytes), así como el instante en el que llegaron. A continuación, con el fin de obtener el tiempo de permanencia de cada paquete, se hace un seguimiento de sus tiempos de llegada. Para cada evento de paquete $a_k(t) \in \boldsymbol{Q}(t)$, se define $d(a_k(t))$ como el tiempo en el cual un paquete $a_k(t)$ llega. Por tanto, el acumulado del tiempo de llegadas de todos los paquetes en $Q_k(t)$ se obtiene como:

$$D\left(Q_k(t)\right) = \sum_{p \in Q_k(t)} d(p)$$

Como se observa en la Figura 4.6, se considera una cola virtual en la que los paquetes se almacenan antes de ser enviados, y se monitoriza el tiempo de permanencia de cada uno de ellos. Es entonces, en cada *slot* (t), cuando se toma una decisión, actualizando la ocupación de la cola en consecuencia como sigue:

$$\hat{Q}_k(t) = \sum_{p \in Q_k(t)} h_k(t,p) = tQ_k(t) - D\left(Q_k(t)\right)$$

Se obtiene así el tiempo de permanencia acumulado para cada *stream* k, donde $h_k(t,p) = t - d(p)$ representa el tiempo de permanencia de p, cuando este todavía no ha sido eliminado de la cola. Es importante resaltar que $Q_k(t)$ se corresponde con el número de paquetes en cola del flujo k. Siendo $b_k(t)$ el número de salidas del flujo k, se puede denotar el acumulado del tiempo de llegada de esos paquetes como $D\left(b_k(t)\right)$, quedando la suma de los tiempos de permanencia de cada paquete que sale de la cola definido como:

$$\hat{b}_k(t) = \sum_{p \in b_k(t)} h_k(t,p) = tb_k(t) - D\left(b_k(t)\right)$$

Con la misma metodología se puede obtener el tiempo de permanencia de los paquetes que llegan, $a_k(t)$. Así, en cada *slot*, la cola se actualiza según la expresión:

$$\hat{Q}_k(t+1) = [\hat{Q}_k(t) - \hat{b}_k(t)] + \hat{a}_k(t)$$

Como se vio en el *scheduler* BP, cada decisión $\alpha(t)$ se elige entre un conjunto de posibles opciones, $\mathcal{A}$, asegurando no exceder la capacidad de transmisión $\sum_k \alpha_k(t) \leq \omega(t)\ \forall t$. Por tanto, el *scheduler* basado en tiempo de permanencia puede derivarse del siguiente problema de *max-weight*:

$$\begin{aligned} \max_{\alpha(t)} \quad & \sum_{m=1}^{N} \hat{Q}_m(t) \cdot b_m(t) \\ \textbf{s.t.} \quad & \alpha \in A \end{aligned}$$

En la práctica, su implementación se traduce en asignar a los *streams* con mayor retardo acumulado, mayor prioridad. Por ello, se reordenan en orden descendente de prioridad, vaciándolos hasta alcanzar la capacidad de transmisión.

A continuación, se detalla la implementación del *scheduler*. En primer lugar, ha sido necesario gestionar las colas de manera que se añada una métrica virtual basada en el tiempo, asignando a cada bloque de bytes alojado en el `búfer` externo a `quic-go` un *timestamp* del instante de llegada.

Seguidamente, a la hora de introducir o sacar datos en el/del `búfer` ha sido necesario gestionar el orden de la estructura de almacenamiento empleada (mapa), debido a la propia naturaleza de éstos en `GO`, donde se realiza el *tracking* con una estrategia FIFO. Además, si la cantidad de bytes que hay en la cola es superior a la que se va sacar se mantendría la diferencia (actualizando correspondientemente su valor), y se realiza una copia del tiempo asignado a esos datos, para la correcta gestión del tiempo de permanencia y su medida.

Por otro lado, para tomar la decisión, el *scheduler* considera el tiempo de permanencia de los datos en las colas, para lo que lleva a cabo la lectura y sumatorio de los instantes de llegada; también tiene en cuenta la cantidad paquetes (bloques) de bytes n. Así, considerando el tiempo actual t_{actual}, se multiplica por el número de bloques de bytes que entraron en la cola, n. Posteriormente, a partir de la diferencia de este valor con el sumatorio de los instantes de llegada T_{in}, se obtiene el tiempo de permanencia t_{sojourn} de la cola del *stream* en cuestión.

$$t_{\text{sojourn}}[stream] = n \cdot t_{\text{actual}} - T_{\text{in}}$$

Finalmente, esta métrica se compara entre cada uno de los *streams* activos, asignando mayor prioridad a aquellos con mayor tiempo en cola. Teniendo en cuenta la capacidad de datos del paquete QUIC, se procede a realizar su conformado, con la cantidad de datos pertinente, asignada a cada *stream*.

Max-Delay

Como simplificación de la anterior estrategia, en este caso se prioriza la cola con mayor retardo, para lo que se tiene en consideración que las colas a nivel de aplicación siguen una estrategia FIFO. Por tanto, se selecciona según el retardo del primer paquete en salir.

A diferencia del *scheduler* anterior, el enfoque de Max-Delay para la implementación solo requiere la lectura del primer registro de tiempo (*timestamp*) asociado a la cola del flujo en

cuestión, considerando que estos siguen una estrategia FIFO, consiguiendo así simplificar el seguimiento (*tracking*) de los datos. Por tanto, al leer únicamente el primer *timestamp* en la cola de los *streams*, se puede identificar rápidamente el que tiene mayor retardo y darle prioridad en el envío, lo que ayuda a reducir el tiempo de espera y mejorará la eficiencia de la transmisión de datos, disminuyendo además la carga computacional.

4.4. Evaluación del rendimiento

En esta sección se discuten los resultados obtenidos en la evaluación de TCP y QUIC, incluyendo las distintas estrategias de *scheduling* que se han presentado, sobre escenarios LEO. Sus principales parámetros de configuración se resumen en la Tabla 4.4. Como puede observarse, se utilizan dos bandas diferentes: *Ka* y *S*, cada una de ellas caracterizada por la matriz de probabilidades de transición, el tiempo medio de permanencia en cada estado y las tasas de transmisión correspondientes. La velocidad de transmisión máxima, que se corresponde a la situación de línea de visión directa, es de 4 y 80 `Mbps` para las bandas *S* y *Ka*, respectivamente. Además, las capacidades de *mid-shadowing* y *deep-shadowing* se fijan en el 50 % y el 20 % de dicha capacidad máxima.

A continuación se comparará, en primer lugar, el rendimiento de QUIC y TCP sobre enlaces LMS, utilizando las configuraciones por defecto. Seguidamente, el análisis se centrará en el protocolo QUIC, analizando el comportamiento de los *schedulers* descritos en la Sección 4.2 sobre enlaces LMS, considerando flujos de tráfico asimétricos. Esto se ampliará analizando el impacto de las desconexiones ISL sobre un escenario extremo a extremo.

Tabla 4.4: Configuración del escenario

Banda *Ka* Parámetros LMS de [9, Table XVII]	
Tasa LMS	$[80, 40, 16]$ `Mbps`
Tasa de enlace	45,33 `Mbps`
Matriz de transición LMS	$\mathcal{P} = \begin{pmatrix} 0 & 0{,}93156 & 0{,}068437 \\ 0{,}34526 & 0 & 0{,}65474 \\ 0{,}070012 & 0{,}92999 & 0 \end{pmatrix}$
Tiempo de permanencia LMS	$[0{,}2530, 0{,}7299, 0{,}1666]$ `s`
δ	100 `ms`
Banda *S* Parámetros LMS de [9, Table XVII]	
Tasa LMS	$[4, 2, 0{,}8]$ `Mbps`
Tasa de enlace	2,32 `Mbps`
Matriz de transición LMS	$\mathcal{P} = \begin{pmatrix} 0 & 0{,}94076 & 0{,}059243 \\ 0{,}77084 & 0 & 0{,}22916 \\ 0{,}49418 & 0{,}50582 & 0 \end{pmatrix}$
Tiempo de permanencia LMS	$[0{,}5485, 0{,}4992, 0{,}3529]$ `s`
δ	100 `ms`
Aplicación y búfer	
Tamaño del búfer	$[7, 15, \infty]$ `Packets`
Tamaño del paquete	1000 `Bytes`

4.4.1. Rendimiento de TCP y QUIC sobre un único enlace LMS

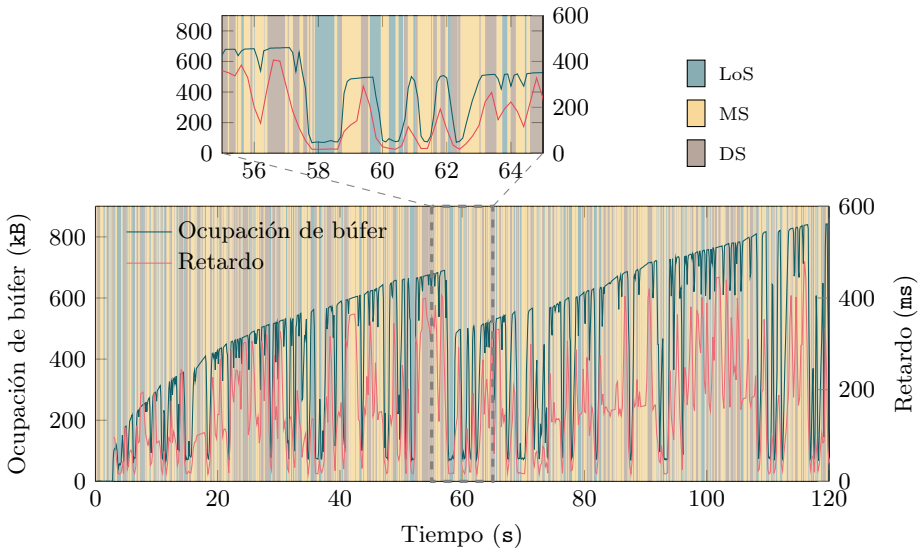
Antes de analizar el rendimiento medio de TCP y QUIC en enlaces LMS, la Figura 4.7 ilustra la variabilidad mostrada por un canal LMS concreto, y su impacto en la ocupación del búfer, la ventana de congestión y el retardo. En este experimento se emplean aplicaciones reales, que generan tráfico sintético. Para este análisis se ha tenido en cuenta el *búfer* del dispositivo `PointToPoint`, véase la Figura 3.1, y su ocupación indica, por tanto, la adaptación del control de congestión a la los cambios del canal; en este caso se ha empleado, como algoritmo de control de congestión, CUBIC. Los resultados se obtienen a partir de una única transmisión TCP, y corresponde al envío de un fichero, con un flujo y tasa de tráfico de aplicación constante, sobre la banda *Ka* y capacidad de `búfer` infinita. Los colores de fondo de la Figura 4.7 representan el estado del canal, y las líneas la evolución instantánea de los distintos parámetros.

Como se puede observar en la Figura 4.7a, el tamaño del `búfer` aumenta durante la conexión, con variaciones lentas que están correladas con la ventana de congestión, y variaciones rápidas que dependen del estado particular del canal. Al mismo tiempo, se puede apreciar cómo el retardo de los paquetes sigue la misma tendencia que la ocupación del *búfer*, mostrando grandes variaciones (*jitter*) cuando el canal cambia entre estados. Concretamente, se puede apreciar en el detalle de la Figura 4.7a cómo al aumentar el *búfer* el retardo extremo a extremo sube, pues se incrementa a su vez el tiempo de espera de los paquetes; además, si se produce una transición a un estado con mayor capacidad, el retardo se reduce. Resaltar que el tamaño del *búfer* no está limitado y cuenta con capacidad infinita, pero su magnitud se corresponde con el indicado por la ventana de congestión, donde el mecanismo utilizado indica la cantidad máxima de datos que se puede transmitir.

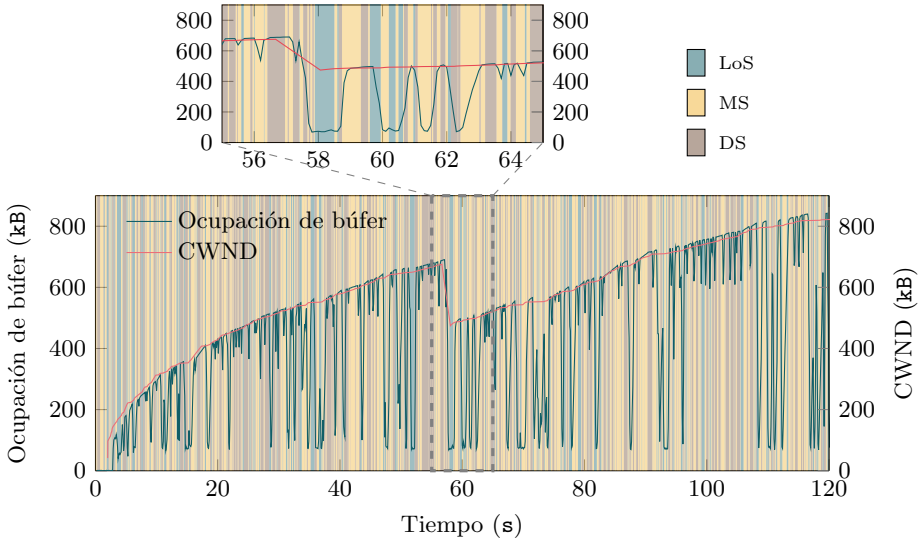
Por otro lado, siguiendo lo mencionado previamente, la Figura 4.7b muestra la evolución de la ventana de congestión en el mismo experimento. Como se puede observar, la ventana de congestión aumenta a la vez que lo hace la ocupación del *búfer*, hasta que se produce un evento de expiración de temporizador de retransmisión, también conocido como Retransmission Timeouts (RTO), momento en el que la ventana de congestión disminuye de manera brusca su valor, al igual que la ocupación del *búfer*. En resumen, la Figura 4.7 muestra que el enlace LMS complica la adaptación de los protocolos de transporte a la capacidad de transmisión real, lo cual tiene un fuerte impacto en el rendimiento de la aplicación, especialmente en el retardo y el *jitter*.

Seguidamente, se realiza una comparativa del rendimiento obtenido por QUIC y TCP en diferentes enlaces LMS para varias configuraciones, modificándose el tamaño de *búfer* de los dispositivos, al igual que la banda de frecuencia utilizada. Es relevante señalar que ambos protocolos utilizan CUBIC como mecanismo de control de congestión, por lo que se podría esperar un impacto similar en ambos. La Figura 4.8 muestra el retardo de los paquetes y el rendimiento de la aplicación obtenidos en ambas bandas, *Ka* y *S*, para diferentes tamaños de `búfer`. Los resultados se promedian en base a 30 ejecuciones independientes de un experimento que establece transmisiones activas durante 60 segundos. En todos los casos, la aplicación genera tráfico a una tasa igual a la capacidad promedio del enlace, siguiendo una distribución de *Poisson*. Cuando se utiliza QUIC, se representan los resultados obtenidos para diferente número de flujos activos (*stream*), utilizando el planificador predeterminado (RR).

Como era de esperar, se observa que el tamaño del *búfer* tiene un fuerte impacto en el retardo.



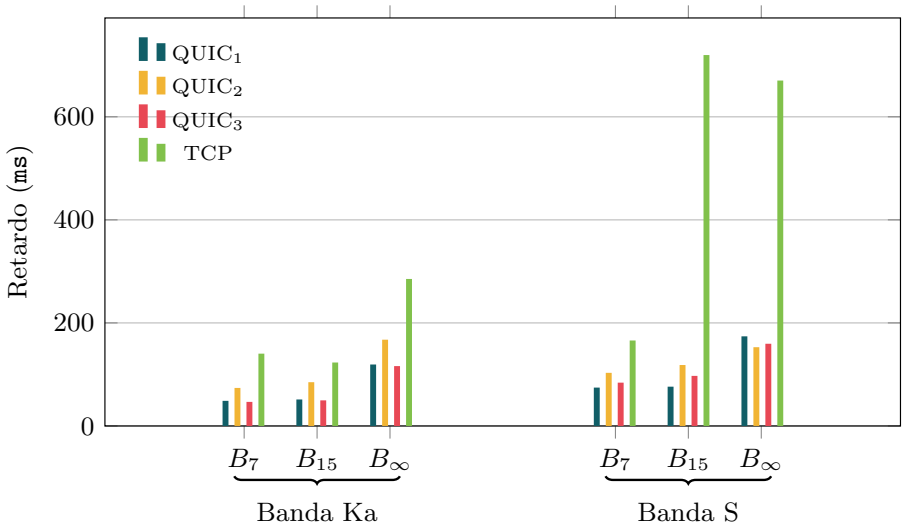
(a) Ocupación de búfer vs. Retardo



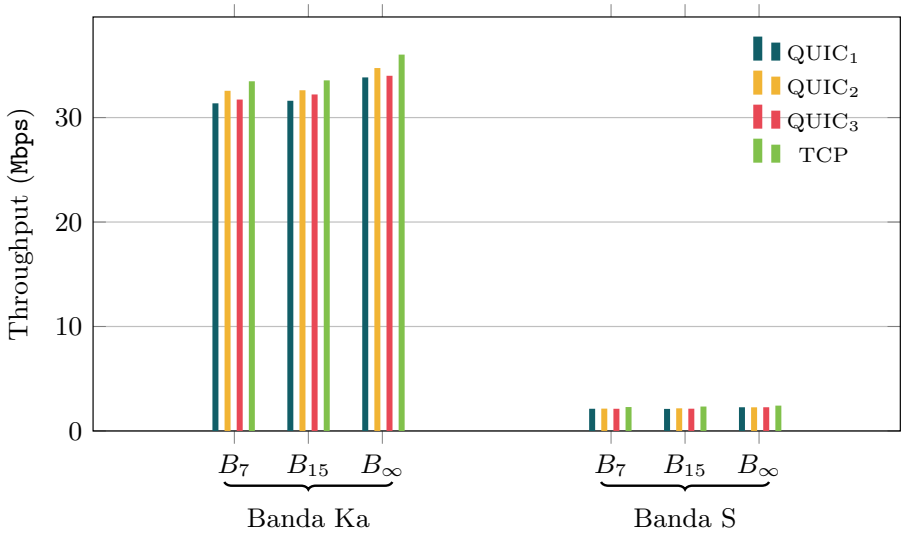
(b) Ocupación de búfer vs. Ventana de congestión

Figura 4.7: Ejemplo de la evolución del tamaño del búfer, retardo y ventana de congestión sobre un canal LMS con capacidad de búfer infinita.

Sin embargo, los resultados muestran que el rendimiento de TCP se ve más afectado, con un retardo muy alto cuando el *búfer* es infinito. Dado que ambos protocolos utilizan el mismo control de congestión, esta diferencia se debe a las mejoras en la detección de congestión que se incorporan en QUIC [19]. Por otro lado, la tasa promedio es bastante estable, independientemente del protocolo o tamaño del *búfer*. En cuanto a la capacidad de múltiples flujos, los resultados demuestran que su uso tiene poco impacto en el rendimiento del protocolo. Esta comparación indica que QUIC es una solución de transporte más adecuada para canales altamente variables, como son los enlaces LMS.



(a) Retardo medio



(b) Throughput medio

Figura 4.8: Comparación del rendimiento promedio de los protocolos TCP frente a QUIC sobre un único enlace LMS. Resultados promedios obtenidos mediante la realización de 30 ejecuciones donde el tráfico es generado durante 1 minuto a una tasa igual a la capacidad promedio del canal.

4.4.2. Rendimiento de los schedulers sobre un enlace LMS

Seguidamente, se analiza el comportamiento de las diferentes estrategias de *scheduling*, partiendo de las soluciones de referencia tradicionales y el algoritmo *Back-pressure*. En primer lugar, la Figura 4.9 muestra la evolución del *búfer* de la aplicación a lo largo de un solo experimento, con una duración de 60 segundos. A nivel de aplicación la tasa se fija al 80 % de la capacidad del enlace LMS, es decir, una carga bastante alta, sin llegar a saturar, y se monitoriza la evolución del `búfer` para dos flujos de tráfico, así como el tiempo acumulado de permanencia en el mismo. En color verde se muestra el comportamiento del primer flujo que tiene el doble de tasa que el otro, y en amarillo el de menor tasa. Vale la pena destacar que el tiempo acumulado se calcula como la suma del tiempo de permanencia de todos los bytes que esperan en el *búfer* de un flujo en un momento dado, cuando la estrategia de *scheduling* toma la decisión. Dado que los flujos

Tabla 4.5: Configuración de tráfico a nivel de aplicación.

Configuración común	
Capacidad promedio del canal (Ka)	45 *Mbps*
Tamaño del paquete	1000 *Bytes*
Distribución del tráfico	Poisson
Tiempo para la generación de paquetes	60 **s**
Escenario 1 (50 %)	
Tasas a nivel de aplicación	$Stream_1$: 15 *Mbps*; $Stream_2$: 7,5 *Mbps*
Escenario 2 (80 %)	
Tasas a nivel de aplicación	$Stream_1$: 27 *Mbps*; $Stream_2$: 15 *Mbps*
Escenario 3 (100 %)	
Tasas a nivel de aplicación	$Stream_1$: 30 *Mbps*; $Stream_2$: 15 *Mbps*

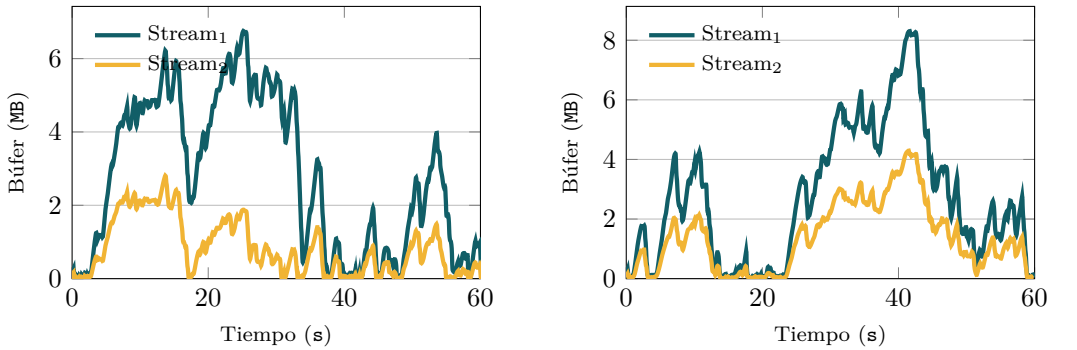
tienen diferentes características se compararán los algoritmos propuestos con el de *Back-pressure* basado en ocupación y WFQ, cuyos pesos están ajustados a la tasa media de tráfico de cada flujo.

Se puede apreciar cómo los *schedulers* basados en BP y en retardo son capaces de adaptar su funcionamiento al desequilibrio del tráfico. BP iguala la ocupación del *búfer*. Por el contrario, esto no se observa en las dos estrategias basadas en retardo. Mientras las dos políticas de *scheduling* basadas en retardo son capaces de armonizar el tiempo de espera para los dos flujos, independientemente de que usen tasas diferentes; por el contrario, BP no logra hacerlo.

El comportamiento de la solución de referencia, WFQ, es acorde a lo que se preveía, debido a que se configuró para otorgar el doble de capacidad al flujo más exigente. Por lo tanto, se puede observar cómo las políticas de *scheduling*, que no necesitan ser preconfiguradas (véase BP o aquellos basados en retardo), son capaces de ofrecer un buen rendimiento, al adaptar su funcionamiento a las condiciones cambiantes de forma dinámica.

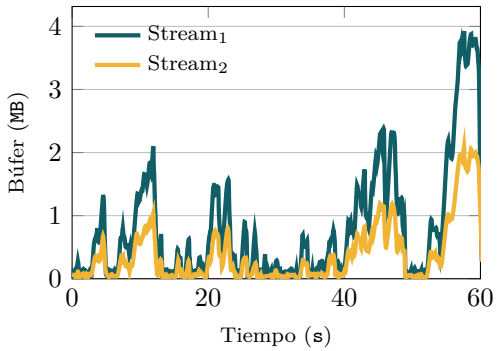
En la Figura 4.10 se muestra el comportamiento de todos los *schedulers*, estudiando el retardo promedio y el *throughput*. Para su representación se ha obtenido el promedio los resultados de 30 experimentos independientes, cada uno con una duración de 60 segundos. Se considera un escenario de tráfico bajo (la tasa de aplicación se fija en el 50 % de la capacidad del enlace LMS) y una situación de saturación, en la que la tasa de aplicación es igual a la capacidad media del enlace LMS. Como se puede observar, los *schedulers* estáticos, RR y FQ, muestran un comportamiento bastante desequilibrado de los dos flujos, más relevante en el escenario de saturación. La implementación por defecto de RR muestra una distribución ligeramente más equitativa, ya que es capaz de aprovechar algo mejor la capacidad de transmisión. Además, la estrategia WFQ es capaz de igualar el retardo en la configuración de baja carga, pero su comportamiento empeora considerablemente cuando aumenta la carga. Por otro lado, el algoritmo BP logra igualar hasta cierto punto el retardo de ambos flujos, pero induce un retardo mayor en el flujo dos, ya que su objetivo es armonizar la ocupación de ambas colas. Por último, las soluciones propuestas basadas en retardo muestran el comportamiento esperado, ya que se iguala en ambos escenarios. Es especialmente relevante el hecho de que, incluso en situaciones de carga extremadamente alta (saturación), las dos estrategias propuestas son capaces de equiparar el retardo en ambos flujos.

También es importante evaluar si los beneficios observados en términos de retardo no obstaculizan el rendimiento de la comunicación. Para ello, la Figura 4.11 muestra el rendimiento

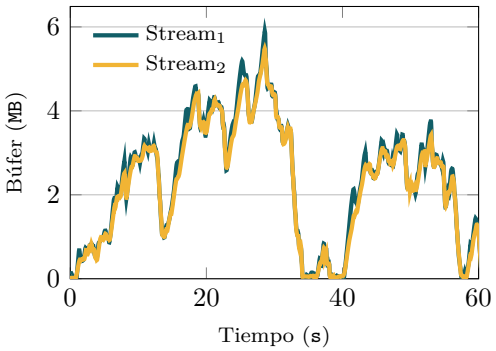


(a) Búfer (WFQ)

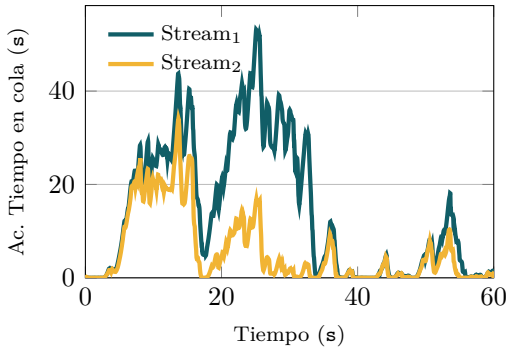
(b) Búfer (Delay)



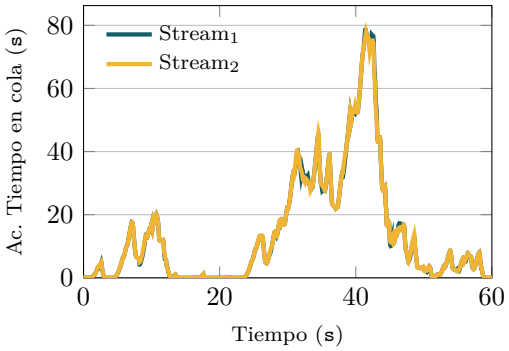
(c) Búfer (MaxDelay)



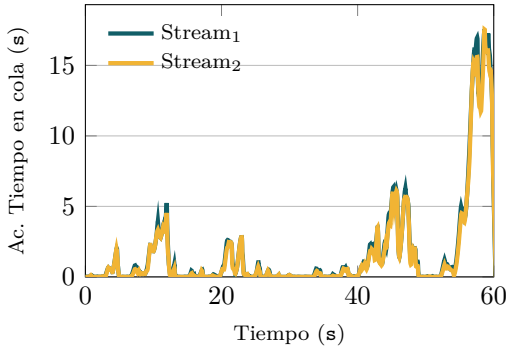
(d) Búfer (BP)



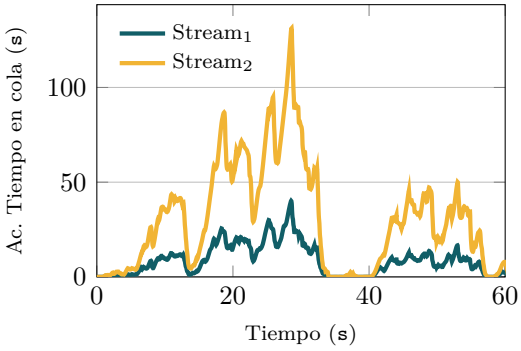
(e) Tiempo de espera (WFQ)



(f) Tiempo de espera (Delay)

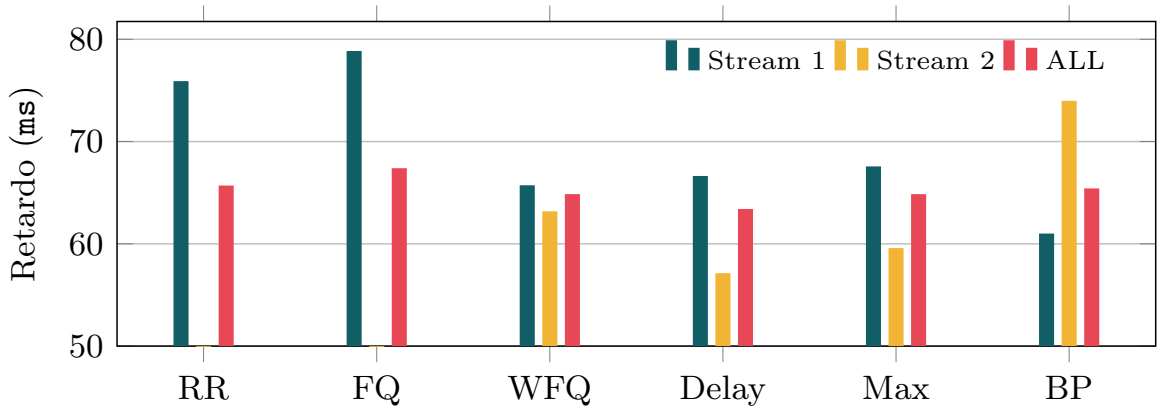


(g) Tiempo de espera (MaxDelay)

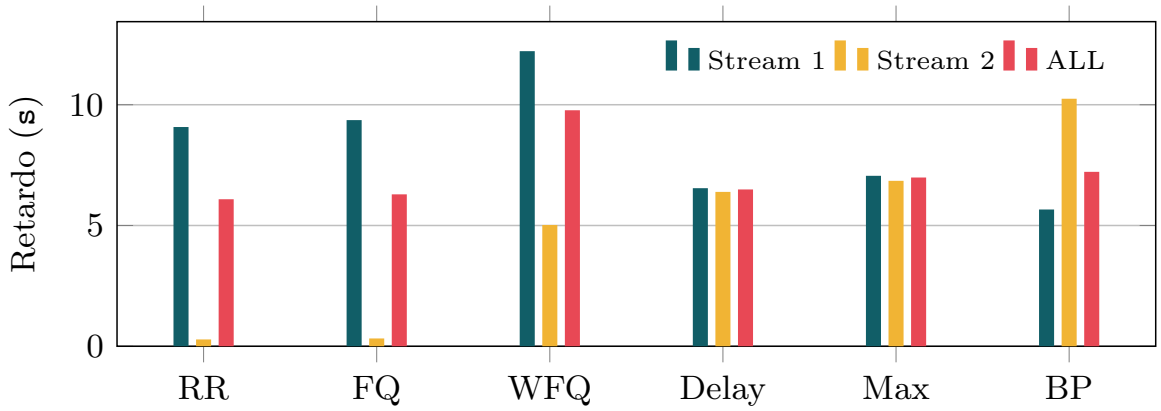


(h) Tiempo de espera (BP)

Figura 4.9: Evolución del búfer de cola de diferentes flujos a lo largo del tiempo utilizando diferentes algoritmos de *scheduling* y tasas de envío binarias en un enlace LMS, en un escenario de alta carga.



(a) Escenario 1 - Tasa de tráfico a 50 %



(b) Escenario 2 - Tasa de tráfico a 100 %

Figura 4.10: Retardo sobre un enlace LMS

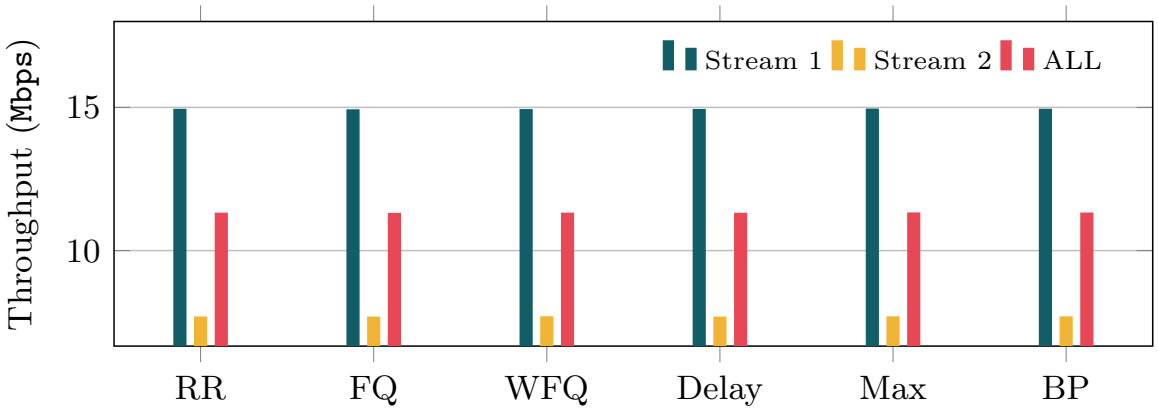
observado para los dos flujos (y el total) utilizando los 6 *schedulers*.

En una primera configuración, con carga baja, 50 % de la capacidad del canal, el impacto de los *schedulers* propuestos es casi insignificante, ya que el rendimiento es prácticamente igual al obtenido con los esquemas de referencia. En el escenario de saturación, se observa una ligera reducción en el rendimiento del segundo flujo en comparación con RR y FQ. A su vez, los resultados también ponen de manifiesto que los algoritmos propuestos, que no requieren ser preconfigurados, ofrecen mayores rendimientos que el tradicional WFQ.

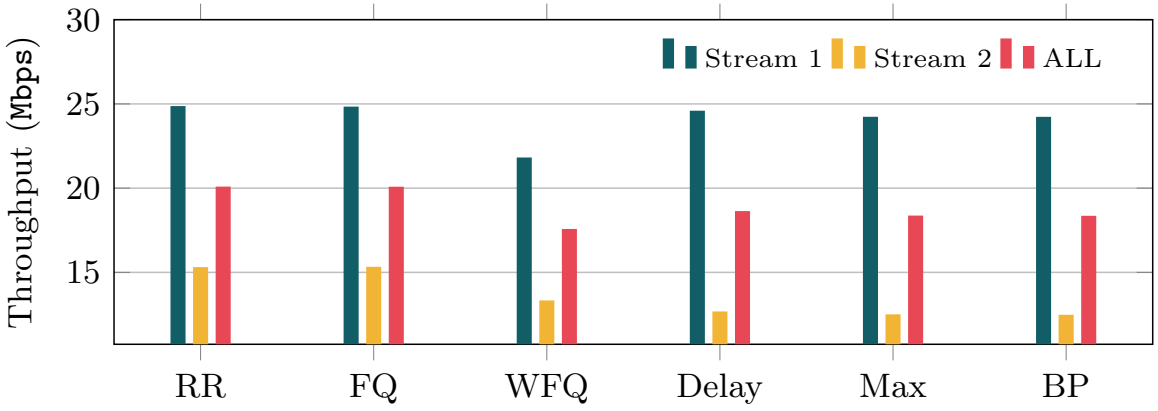
4.4.3. Escenario E2E

En esta sección se amplía el análisis a un escenario E2E, que abarca dos enlaces LMS y un ISL, que introduce desconexiones temporales. Como se mencionó en la Sección 4.4, el ISL se modela con una cadena de Markov de dos estados, representada en la Figura 3.7. Siguiendo la misma metodología que en el experimento anterior, se configuran 2 flujos no balanceados en QUIC, de manera que la tasa del primero es el doble que la del segundo. A su vez se procede a realizar un barrido de los valores de tiempo de desconexión promedio, reduciendo así la capacidad total de la red. En este caso, debido a esta reducción la tasa de tráfico agregado se establece en el 50 % de la capacidad promedio de los enlaces LMS, Tabla 4.5.

Se utiliza la misma configuración a nivel de aplicación que en el escenario anterior (dos flujos,



(a) Escenario 1 - Tasa de tráfico a 50 %



(b) Escenario 2 - Tasa de tráfico a 100 %

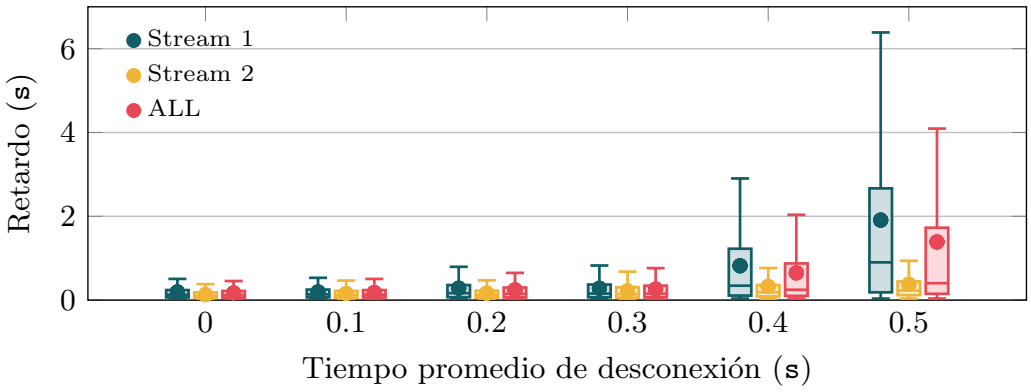
Figura 4.11: Rendimiento sobre un único enlace LMS

uno con el doble de la tasa que el otro), pero sólo se emplea la carga de tráfico al 50 %, Tabla 4.5, ya que las desconexiones en el ISL podrían reducir la capacidad E2E general.

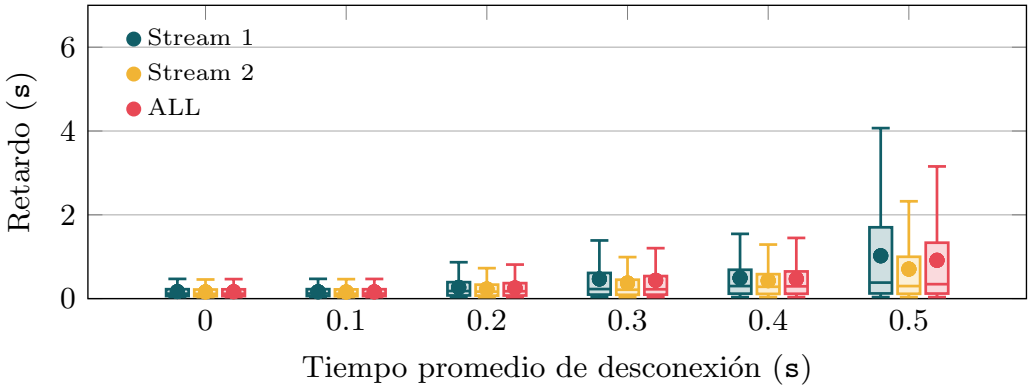
El ISL va cambiando de estado durante la simulación, en el activo tiene una tasa de transmisión de 80 `Mbps`, con un tiempo de permanencia medio de 5 segundos, y periodos de desconexión cuya duración se va modificando de 0 a 0,5 segundos; en estos intervalos el enlace intermedio pasa a un estado inactivo. En ambos casos, el tiempo de permanencia se modela con una distribución exponencial negativa. Se llevan a cabo 5 experimentos independientes, con una duración de 300 segundos para cada configuración, para asegurar que se produce un número suficiente de transiciones entre estados.

La Figura 4.12 muestra el retardo promedio extremo a extremo de ambos flujos y el total, en color rojo, para las cuatro estrategias de *scheduling*: RR, WFQ, BP y Delay, a medida que se incrementa el tiempo promedio de desconexión en el enlace ISL. Se puede observar un aumento constante del retardo, así como de su dispersión. Cabe destacar que, al utilizar el algoritmo BP, el segundo flujo experimenta un retardo adicional debido a la armonización de las colas. Por otro lado, el *scheduler* basado en retardo consigue equilibrar el comportamiento de los dos flujos, sincronizando los retardos en todas las configuraciones.

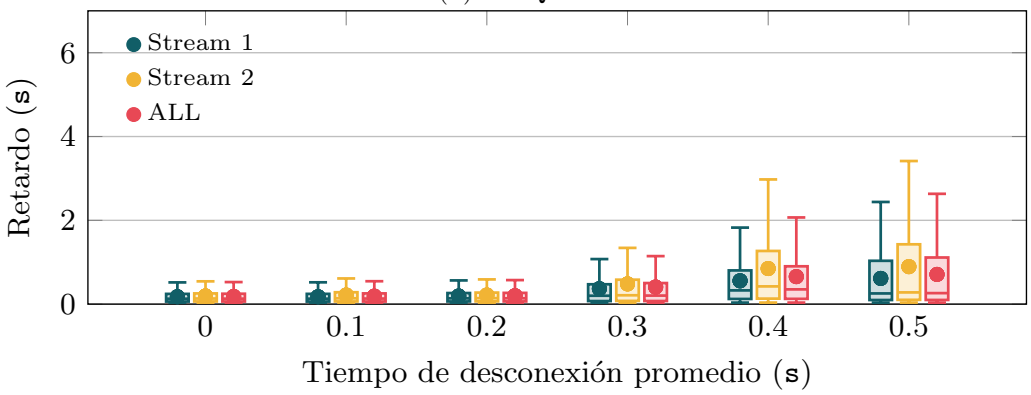
Estos resultados también evidencian el buen desempeño de las soluciones de *scheduling* propuestas. Mientras que el rendimiento se ve afectado negativamente por los periodos de



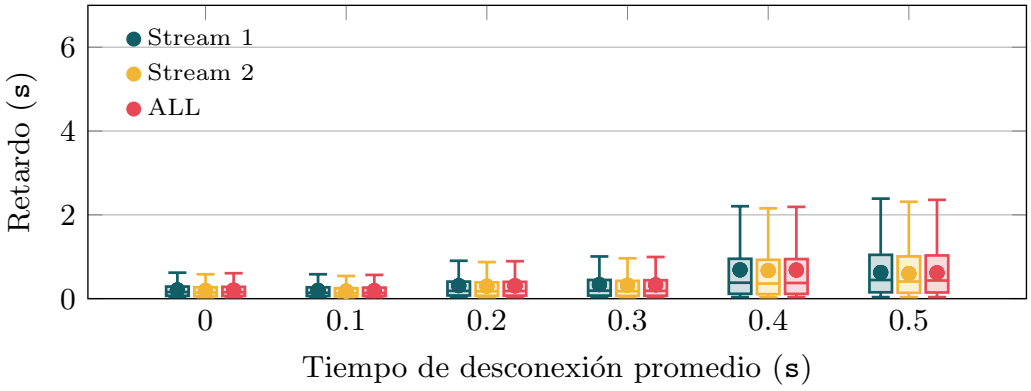
(a) RR



(b) WFQ



(c) BP



(d) Delay

Figura 4.12: Distribución del retardo promedio de RR, WFQ, BP y Delay sobre un escenario E2E con interrupciones en el enlace intermedio.

desconexión en las dos soluciones de referencia (RR y WFQ), tanto BP como las estrategias de *scheduling* basadas en retardo son capaces, no solo de igualar el retardo, sino también de reducirlo. Además, la capacidad de adaptación a las condiciones cambiantes del propio canal permiten reducir la variabilidad del retardo medido, lo que a su vez disminuye la fluctuación percibida en la aplicación receptora o *jitter*.

5 Conclusiones y líneas futuras

En este capítulo se concluye el documento, presentando las conclusiones derivadas del análisis de los experimentos llevados a cabo sobre diversos escenarios en una red satelital. También se identifican las líneas de trabajo que surgen a raíz de la investigación llevada a cabo.

5.1. Conclusiones

La aparición de nuevas tecnologías inalámbricas y topologías de red de acceso ha generado la necesidad de evaluar el comportamiento de los protocolos de capas superiores, tanto los tradicionales como aquellas alternativas más recientes. Si bien los emuladores de enlace pueden utilizarse para este tipo de análisis, centrados por ejemplo en los mecanismos de control de congestión, no están diseñados para reflejar la dinámica de las tecnologías de acceso, y a veces se limitan a escenarios más específicos. Por otro lado, implementar procedimientos detallados de capa baja en los simuladores de red podría no tener un impacto significativo en el rendimiento de las soluciones de capas superiores, incrementando además de manera considerable el tiempo de simulación, y limitando la interacción con aplicaciones reales.

Como complemento a los enfoques de análisis existentes, en este trabajo se describe un esquema de simulación ligero que combina el simulador `ns-3` con modelos teóricos de los canales subyacentes. Esta metodología, que ha sido puesta a disposición de la comunidad científica través de un repositorio público, se basa en una modificación de la implementación original de la clase `PointToPointDevice` en `ns-3`, y en varias utilidades que simplifican la definición de escenarios con diferentes topologías y configuraciones.

Utilizando este enfoque, se ha creado una red LEO y se han utilizado modelos de capacidad basados en cadenas de Markov para reflejar el comportamiento de los dos tipos de enlace que la componen. Seguidamente, se ha procedido a la validación de la metodología propuesta mediante el análisis del rendimiento del protocolo TCP en diferentes configuraciones. Inicialmente, se validó el correcto funcionamiento utilizando un canal sintético, y posteriormente se amplió el estudio utilizando modelos más realistas de enlaces LEO propuestos en la literatura, y que se adaptaron para integrarse en la herramienta desarrollada. El enfoque propuesto permite desplegar diferentes topologías y escenarios, desde enlaces individuales hasta redes LEO extremo a extremo que abarcan múltiples enlaces con tráfico de fondo y desconexiones.

Por otro lado, como segundo objetivo de este trabajo se propuso evaluar el rendimiento de protocolos de transporte (TCP y QUIC) sobre redes LEO. Como se ha mencionado, se trata de uno de los primeros trabajos en afrontar un análisis con este objetivo. Para ello se ha propuesto

una metodología novedosa que combina técnicas de virtualización, implementaciones reales de protocolos, y el entorno de simulación de red `ns-3`, para realizar experimentos sistemáticos con un modelo realista de la conectividad subyacente, sin necesidad de asumir una carga computacional elevada. Los resultados demuestran que QUIC mejora el comportamiento de TCP sobre canales con alta variabilidad.

Además, se han caracterizado minuciosamente distintas estrategias de *scheduling* que explotan la capacidad multi-streaming de QUIC. Se ha visto que el algoritmo utilizado por defecto en la implementación `quic-go` es subóptimo con flujos de tráfico no equilibrados. La estrategia WFQ puede mantener la velocidad de transmisión sin afectar significativamente al retardo medio, pero no es capaz de adaptarse a situaciones de tráfico no equilibrado. Por otro lado, el esquema basado en el algoritmo *Back-pressure*, es capaz de estabilizar las longitudes de cola de los flujos activos, utilizando únicamente el estado de los búferes. Se proponen además dos nuevas estrategias en este trabajo, cuya finalidad es mejorar el retardo percibido. La primera de ellas se basa en el tiempo total de permanencia de los paquetes en la cola. Por otro lado, con la finalidad de reducir la complejidad, se diseña una estrategia en función del retardo máximo inducido por cada flujo. Ambas estrategias muestran un rendimiento similar. Sin embargo, la primera alternativa ofrece un retardo ligeramente inferior, sin perjudicar el rendimiento. Por otro lado, la principal ventaja de la segunda estrategia, que es reducir la carga de información como consecuencia de la observación del estado de las colas, presenta un redimiendo ligeramente inferior respecto al anterior.

Seguidamente, se ha aumentado la complejidad de los escenarios, considerando una comunicación E2E, con un ISL que introduce periodos con interrupciones de conectividad, mostrando que su comportamiento se mantiene al aumentar el tiempo medio de desconexión. En este escenario, se puede observar cómo el *scheduler* basado en retardo es capaz de igualar el comportamiento de los dos flujos, y de reducir tanto su valor medio como su variabilidad, incluso para periodos de desconexión más largos.

En definitiva, la metodología propuesta permite aumentar la complejidad de la conectividad subyacente, introduciendo tiempos de desconexión en enlaces ISL o tráfico de fondo, como se ha descrito en los últimos experimentos que se han descrito en la sección de validación, así como en el análisis del rendimiento de los *schedulers*. Se ha observado en este caso que las estrategias basadas en el estado de las colas y en la métrica de retardo son capaces de equilibrar el comportamiento de los dos flujos.

Basado en el análisis y los resultados que se han presentado en este documento, se han generado cuatro artículos de investigación:

- Fátima Khan et al. *Realistic Assessment of Transport Protocols Performance over LEO-based Communications*. Enviado a la revista Computer Networks, pendiente de revisión. 2023

- Fátima Khan et al. *Estrategias de scheduling sobre QUIC en entornos NTN*. Enviado a las Jornadas de Ingeniería Telemática (JITEL), pendiente de revisión. 2023

- Fátima Khan et al. *Metodología para el análisis del comportamiento de protocolos de transporte sobre canales variantes: Aplicación en escenarios NTN*. Enviado a las Jornadas de Ingeniería Telemática (JITEL), pendiente de revisión. 2023

- Fátima Khan et al. *Exploiting QUIC multi-streaming over NTN: Delay-based scheduling policies*. Enviado a 2023 IEEE Global Communications Conference: Mobile and Wireless Networks, pendiente de revisión. 2023

El artículo [25], enviado a Computer Networks, ha pasado a la segunda ronda de revisión tras haber recibido una decisión *minor revision*. Además, también se han enviado dos contribuciones [22], [24] a las Jornadas de Ingeniería Telemática (JITEL), organizadas por la Sociedad Científica de Ingeniería Telemática (SCITEL), actualmente en proceso de revisión. Finalmente, el artículo [23] se ha enviado al IEEE Global Communications Conference (GLOBECOM) y se encuentra en proceso de revisión.

5.2. Líneas futuras

Como trabajo futuro, se prevé aprovechar la funcionalidad existente para ampliar el análisis e incluir otras características importantes. Por ejemplo, se explorará la integración de mecanismos de control de congestión y el soporte a comunicaciones *multipath* en la implementación de QUIC. Al hacerlo, será posible aplicar estrategias de *scheduling* tanto a nivel de gestión de colas de flujos como a nivel de selección de caminos, y escoger combinaciones óptimas de múltiples estrategias, como ilustra la Figura 5.1.

La incorporación de otros mecanismos de control de congestión, como BBR, permitirá mejorar la capacidad del protocolo QUIC para adaptarse a situaciones de congestión en la red. Se investigarán diferentes mecanismos para garantizar un comportamiento eficiente y equitativo de QUIC bajo condiciones de alta carga de tráfico, sobre entornos variables.

Asimismo, la integración de *multipath* en la propia implementación de QUIC, habilitando la utilización simultánea de múltiples caminos de comunicación, permitirá un mejor aprovechamiento de los recursos disponibles, así como una mayor tolerancia a fallos y una mejora en la capacidad de respuesta de la red. Igualmente, se explorarán estrategias de *scheduling* para gestionar de manera óptima las colas de los flujos y la selección de caminos, en función de los requisitos de los datos y las condiciones de la red.

La integración de estas características en QUIC permitirá mejorar el desempeño y las capacidades del protocolo en diferentes entornos de red. Además, permitirá proponer mejoras específicas para adaptar QUIC a escenarios reales, y garantizar un rendimiento óptimo en diferentes contextos.

A su vez, con el fin de validar el correcto funcionamiento de las estrategias escogidas en cada uno de los ámbitos, será necesaria la creación de diversos escenarios para comparar el comportamiento de las diferentes soluciones y determinar las combinaciones que ofrezcan un mejor rendimiento. Surge por tanto una serie de líneas futuras para ampliar la metodología de análisis que se ha diseñado alrededor de `ns-3`; entre ellas se planea mejorar la operación de la implementación propuesta, incluyendo la posibilidad de configurar los modelos de canal mediante archivos externos. Esto permitirá personalizar el comportamiento de los enlaces, incluso los que sean del mismo tipo, con tiempos de permanencia, probabilidades de transición y capacidades diferenciadas. Por otro lado, se dará la opción de escoger el tipo de protocolo que genera tráfico

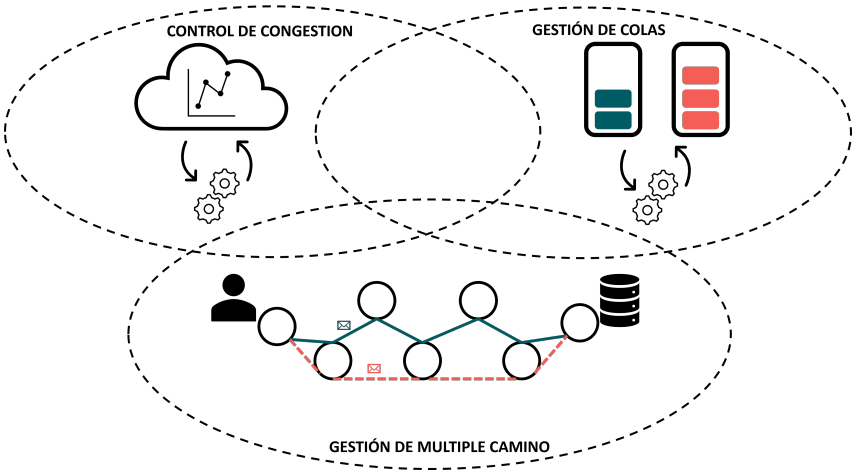


Figura 5.1: Líneas futuras de investigación que surgen a partir de trabajo.

de fondo en los enlaces compartidos, así como el despliegue de topologías con varios caminos hacia el mismo destino.

Bibliografía

[1] Ian F. Akyildiz, Ahan Kak y Shuai Nie. "6G and Beyond: The Future of Wireless Communications Systems". En: *IEEE Access* 8 (2020), págs. 133995-134030. DOI: 10.1109/ACCESS.2020.3010896.

[2] M. Mahdi Azari et al. "Evolution of Non-Terrestrial Networks From 5G to 6G: A Survey". En: *IEEE Communications Surveys & Tutorials* 24.4 (2022), págs. 2633-2672. DOI: 10.1109/COMST.2022.3199901.

[3] Anushka Bishen. *5G And Non-Terrestrial Networks*. 5G Americas. 24 de ene. de 2022. URL: https://www.5gamericas.org/5g-and-non-terrestrial-networks/ (visitado 02-07-2023).

[4] A. Chen, C. Chang e Y. Yao. "Performance evaluation of ARQ operations with OBP and inter-satellite links: delay performance". En: *IEEE 54th Vehicular Technology Conference Proceedings (VTC Fall 2001)*. Vol. 4. 2001, 2346-2350 vol.4. DOI: 10.1109/VTC.2001.957168.

[5] Changjiang Cui et al. "DASH+: Download Multiple Video Segments with Stream Multiplexing of QUIC". En: *2022 Tenth International Conference on Advanced Cloud and Big Data (CBD)*. 2022, págs. 66-72. DOI: 10.1109/CBD58033.2022.00021.

[6] Tasneem Darwish et al. "LEO Satellites in 5G and Beyond Networks: A Review From a Standardization Perspective". En: *IEEE Access* 10 (2022), págs. 35040-35060. DOI: 10.1109/ACCESS.2022.3162243.

[7] Boya Di et al. "Ultra-Dense LEO: Integration of Satellite Access Networks into 5G and Beyond". En: *IEEE Wireless Communications* 26.2 (2019), págs. 62-69. DOI: 10.1109/MWC.2019.1800301.

[8] Jasenka Dizdarević y Admela Jukan. "Experimental Benchmarking of HTTP/QUIC Protocol in IoT Cloud/Edge Continuum". En: *ICC 2021 - IEEE International Conference on Communications*. 2021, págs. 1-6. DOI: 10.1109/ICC42927.2021.9500675.

[9] F. P. Fontan et al. "Statistical modeling of the LMS channel". En: *IEEE Transactions on Vehicular Technology* 50.6 (2001), págs. 1549-1567. DOI: 10.1109/25.966585.

[10] Alan Ford et al. *TCP Extensions for Multipath Operation with Multiple Addresses*. Request for Comments RFC 8684. Num Pages: 68. Internet Engineering Task Force, mar. de 2020. DOI: 10.17487/RFC8684. URL: https://datatracker.ietf.org/doc/rfc8684 (visitado 07-07-2023).

[11] Romeo Giuliano y Eros Innocenti. “Machine Learning Techniques for Non-Terrestrial Networks”. En: *Electronics* 12 (ene. de 2023), pág. 652. DOI: 10.3390/electronics12030652.

[12] Sangtae Ha, Injong Rhee y Lisong Xu. “CUBIC: a new TCP-friendly high-speed TCP variant”. En: *ACM SIGOPS operating systems review* 42.5 (2008), págs. 64-74.

[13] Long Hai et al. “Delay-Optimal Back-Pressure Routing Algorithm for Multihop Wireless Networks”. En: *IEEE Transactions on Vehicular Technology* 67.3 (2018), págs. 2617-2630. DOI: 10.1109/TVT.2017.2770183.

[14] Meilin He et al. “A Novel Edge Computing Server Selection Strategy of LEO Constellation Broadband Network”. En: *2020 IEEE World Congress on Services (SERVICES)*. 2020, págs. 275-280. DOI: 10.1109/SERVICES48979.2020.00061.

[15] R. Hermenier, C. Kissling y A. Donner. “A delay model for satellite constellation networks with inter-satellite links”. En: *2009 International Workshop on Satellite and Space Communications*. 2009, págs. 3-7. DOI: 10.1109/IWSSC.2009.5286440.

[16] Cristina Hervella et al. “Realistic Assessment of Transport Protocols Performance over LEO-Based Communications”. En: PE-WASUN ’22. Montreal, Quebec, Canada: Association for Computing Machinery, 2022, págs. 91-98. ISBN: 9781450394833. DOI: 10.1145/3551663.3558680. URL: https://doi.org/10.1145/3551663.3558680.

[17] Cristina Aurora Hervella Baturone. “Análisis del rendimiento del protocolo QUIC en comunicaciones satelitales”. Tesis de mtría. Sep. de 2022. URL: https://hdl.handle.net/10902/26133.

[18] Mohsen Hosseinian et al. “Review of 5G NTN Standards Development and Technical Challenges for Satellite Integration With the 5G Network”. En: *IEEE Aerospace and Electronic Systems Magazine* 36.8 (2021), págs. 22-31. DOI: 10.1109/MAES.2021.3072690.

[19] Jana Iyengar y Ian Swett. *QUIC Loss Detection and Congestion Control*. RFC 9002. Mayo de 2021. DOI: 10.17487/RFC9002. URL: https://www.rfc-editor.org/info/rfc9002.

[20] Jana Iyengar y Martin Thomson. *QUIC: A UDP-Based Multiplexed and Secure Transport*. RFC 9000. Mayo de 2021. DOI: 10.17487/RFC9000. URL: https://rfc-editor.org/rfc/rfc9000.txt.

[21] Wei Jiang et al. “The Road Towards 6G: A Comprehensive Survey”. En: *IEEE Open Journal of the Communications Society* 2 (2021), págs. 334-366. DOI: 10.1109/OJCOMS.2021.3057679.

[22] Fátima Khan et al. *Estrategias de scheduling sobre QUIC en entornos NTN*. Enviado a las Jornadas de Ingeniería Telemática (JITEL), pendiente de revisión. 2023.

[23] Fátima Khan et al. *Exploiting QUIC multi-streaming over NTN: Delay-based scheduling policies*. Enviado a 2023 IEEE Global Communications Conference: Mobile and Wireless Networks, pendiente de revisión. 2023.

[24] Fátima Khan et al. *Metodología para el análisis del comportamiento de protocolos de transporte sobre canales variantes: Aplicación en escenarios NTN*. Enviado a las Jornadas de Ingeniería Telemática (JITEL), pendiente de revisión. 2023.

[25] Fátima Khan et al. *Realistic Assessment of Transport Protocols Performance over LEO-based Communications*. Enviado a la revista Computer Networks, pendiente de revisión. 2023.

[26] Oltjon Kodheli et al. "An Uplink UE Group-Based Scheduling Technique for 5G mMTC Systems Over LEO Satellite". En: *IEEE Access* 7 (2019), págs. 67413-67427. DOI: 10.1109/ACCESS.2019.2918581.

[27] Israel Leyva-Mayorga et al. "LEO Small-Satellite Constellations for 5G and Beyond-5G Communications". En: *IEEE Access* 8 (2020), págs. 184955-184964. DOI: 10.1109/ACCESS.2020.3029620.

[28] Shicong Liu et al. "LEO Satellite Constellations for 5G and Beyond: How Will They Reshape Vertical Domains?" En: *IEEE Communications Magazine* 59.7 (2021), págs. 30-36. DOI: 10.1109/MCOM.001.2001081.

[29] Aitor Martin y Naeem Khademi. "On the Suitability of BBR Congestion Control for QUIC over GEO SATCOM Networks". En: *Proceedings of the Workshop on Applied Networking Research*. ANRW '22. Philadelphia, Pennsylvania: Association for Computing Machinery, 2022. ISBN: 9781450394444. DOI: 10.1145/3547115.3547194. URL: https://doi.org/10.1145/3547115.3547194.

[30] *Microwave Frequency Bands*. URL: https://www.everythingrf.com/tech-resources/frequency-bands (visitado 23-06-2023).

[31] Michał Morawski y Przemysław Ignaciuk. "A Synchronizing Scheduler for Reduced Protocol Delay in Multipath Transmission". En: *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. 2022, págs. 553-558. DOI: 10.1109/ICARCV57592.2022.10004360.

[32] Michael J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Synthesis Lectures on Learning, Networks, and Algorithms. Cham: Springer International Publishing, 2010. DOI: 10.1007/978-3-031-79995-2. URL: https://link.springer.com/10.1007/978-3-031-79995-2 (visitado 02-06-2023).

[33] Ravi Netravali et al. "Mahimahi: A Lightweight Toolkit for Reproducible Web Measurement". En: *SIGCOMM Comput. Commun. Rev.* 44.4 (ago. de 2014), págs. 129-130. ISSN: 0146-4833. DOI: 10.1145/2740070.2631455. URL: https://doi.org/10.1145/2740070.2631455.

[34] Peng Qian, Ning Wang y Rahim Tafazolli. "Achieving Robust Mobile Web Content Delivery Performance Based on Multiple Coordinated QUIC Connections". En: *IEEE Access* 6 (2018), págs. 11313-11328. DOI: 10.1109/ACCESS.2018.2804222.

[35] Injong Rhee et al. *CUBIC for Fast Long-Distance Networks*. Request for Comments RFC 8312. Num Pages: 18. Internet Engineering Task Force, feb. de 2018. DOI: 10.17487/RFC8312. URL: https://datatracker.ietf.org/doc/rfc8312 (visitado 02-07-2023).

[36] Federica Rinaldi et al. "Non-Terrestrial Networks in 5G and Beyond: A Survey". En: *IEEE Access* 8 (2020), págs. 165178-165200. DOI: 10.1109/ACCESS.2020.3022981.

[37] Xiang Shi et al. "PStream: Priority-Based Stream Scheduling for Heterogeneous Paths in Multipath-QUIC". En: *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. 2020, págs. 1-8. DOI: 10.1109/ICCCN49398.2020.9209682.

[38] Tanya Shreedhar et al. "Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads". En: *IEEE Transactions on Network and Service Management* 19.2 (2022), págs. 1366-1381. DOI: 10.1109/TNSM.2021.3134562.

[39] Stephan Jaeckel, Leszek Raschkowski (FhG-HHI),Junhyeong Kim, Gosan Noh (ETRI), Marjorie Thary, Jean-Michel Houssin (TAS), Ji-In Kim (KTSat), Nicolas Cassiau (CEA). "Spectrum usage analysis and channel model". En: *Annalen der Physik* D3.1.H2020-EUK-815323/5G-ALLSTAR/D3.1 (2019), págs. 1-82. DOI: https://5g-allstar.eu/wp-content/uploads/2019/07/815323_Deliverable_3.1_Spectrum-usage-analysis-and-channel-model.pdf.

[40] Randall R. Stewart. *Stream Control Transmission Protocol*. Request for Comments RFC 4960. Num Pages: 152. Internet Engineering Task Force, sep. de 2007. DOI: 10.17487/RFC4960. URL: https://datatracker.ietf.org/doc/rfc4960 (visitado 07-07-2023).

[41] Qingqing Tang et al. "Computation Offloading in LEO Satellite Networks With Hybrid Cloud and Edge Computing". En: *IEEE Internet of Things Journal* 8.11 (2021), págs. 9164-9176. DOI: 10.1109/JIOT.2021.3056569.

[42] Zhixuan Tang et al. "Leveraging LEO Assisted Cloud-Edge Collaboration for Energy Efficient Computation Offloading". En: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, págs. 1-6. DOI: 10.1109/GLOBECOM46510.2021.9685309.

[43] Hiroshi Tsunoda et al. "Performance Evaluation of SCTP wth Adaptive Multistreamiing over LEO Satellite Networks". En: *2007 International Workshop on Satellite and Space Communications*. 2007, págs. 150-154. DOI: 10.1109/IWSSC.2007.4409407.

[44] Tobias Viernickel et al. "Multipath QUIC: A Deployable Multipath Transport Protocol". En: *2018 IEEE International Conference on Communications (ICC)*. 2018, págs. 1-7. DOI: 10.1109/ICC.2018.8422951.

[45] Kaixiang Wei et al. "Resource Scheduling and Offloading Strategy Based on LEO Satellite Edge Computing". En: *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. 2021, págs. 1-6. DOI: 10.1109/VTC2021-Fall52928.2021.9625072.

[46] Renchao Xie et al. “Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues”. En: *IEEE Network* 34.3 (2020), págs. 224-231. DOI: 10.1109/MNET.011.1900369.

[47] Francis Y Yan et al. “Pantheon: the training ground for Internet congestion-control research”. En: *2018 {USENIX} Annual Technical Conference ({USENIX} {ATC} 18)*. 2018, págs. 731-743.

[48] Siyu Yang, Hewu Li y Qian Wu. “Performance Analysis of QUIC Protocol in Integrated Satellites and Terrestrial Networks”. En: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 2018, págs. 1425-1430. DOI: 10.1109/IWCMC.2018.8450388.

[49] Wenjun Yang et al. “MM-QUIC: Mobility-aware Multipath QUIC for Satellite Networks”. En: *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. 2021, págs. 608-615. DOI: 10.1109/MSN53354.2021.00093.

[50] Yuru Zhang et al. “Aerial Edge Computing on Orbit: A Task Offloading and Allocation Scheme”. En: *IEEE Transactions on Network Science and Engineering* 10.1 (2023), págs. 275-285. DOI: 10.1109/TNSE.2022.3207214.

[51] Zhu Zhang, Qing Guo y Zihe Gao. “A Prediction Based SCTP Handover Scheme for IP/LEO Satellite Network”. En: *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*. 2010, págs. 1-4. DOI: 10.1109/WICOM.2010.5601455.

A Anexo

Fichero A.1: Docker Compose para el despliegue de host reales.

```
version: "3.1"
services:
  sender:
    build: .
    image: ldiez/ns3docker
    privileged: true
    container_name: sender
    command: ash -c "python3 /home/AdaptRouteTable.py -d 10.2.0.0/24 -g 10.1.0.3 -i
        ↪ eth0 && tail -F anything"
    restart: always
    networks:
      leftnet:
        ipv4_address: 10.1.0.2
    volumes:
      - ./:/home/

  receiver:
    build: .
    image: ldiez/ns3docker
    privileged: true
    container_name: receiver
    command: ash -c "python3 /home/AdaptRouteTable.py -d 10.1.0.0/24 -g 10.2.0.3 -i
        ↪ eth0 && tail -F anything"
    restart: always
    networks:
      rightnet:
        ipv4_address: 10.2.0.2
    volumes:
      - ./:/home/

networks:
  leftnet:
    driver: bridge
    driver_opts:
      com.docker.network.bridge.enable_ip_masquerade: "true"
      com.docker.network.bridge.name: br-left
```

```
    ipam:
      config:
        - subnet: 10.1.0.0/24
  rightnet:
    driver: bridge
    driver_opts:
      com.docker.network.bridge.enable_ip_masquerade: "true"
      com.docker.network.bridge.name: br-right
    ipam:
      config:
        - subnet: 10.2.0.0/24
```

Fichero A.2: Archivo de configuración JSON para el despliegue de un único enlace LMS.

```
{
    "NumLinks": 1,
    "Links": [
        {
            "Link-type": "LMS",
            "Band": "Ka",
            "DataRate": 60,
            "Delay": 10,
            "ErrorRate": 0,
            "Mode": "OTROS",
            "MTU": 1460,
            "DropTailQueue": 0,
            "Background": false
        }
    ]
}
```

Fichero A.3: Archivo de configuración JSON para el despliegue de una topología con desconexiones.

```
{
  "NumLinks": 3,
  "Links": [
      {
          "Link-type": "LMS",
          "Band": "Ka",
          "DataRate": 3,
          "Delay": 0,
          "ErrorRate": 0,
          "Mode": "OTROS",
          "MTU": 1460,
          "DropTailQueue": 30000,
          "Background": false
      },

```

```
    {
      "Link-type": "Disconnection",
      "Band": "I",
      "DataRate": 50,
      "Delay": 10,
      "ErrorRate": 0,
      "Mode": "OTROS",
      "MTU": 1460,
      "DropTailQueue": 0,
      "Background": false
    },

    {
      "Link-type": "LMS",
      "Band": "Ka",
      "DataRate": 3,
      "Delay": 0,
      "ErrorRate": 0,
      "Mode": "OTROS",
      "MTU": 1460,
      "DropTailQueue": 0,
      "Background": false
    }
  ]
}
```

Fichero A.4: Archivo de configuración JSON para el despliegue de una topología con enlace intermedio compartido.

```
{
    "NumLinks": 3,
    "Links": [
        {
            "Link-type": "LMS",
            "Band": "Ka",
            "DataRate": 3,
            "Delay": 0,
            "ErrorRate": 0,
            "Mode": "OTROS",
            "MTU": 1460,
            "DropTailQueue": 90000,
            "Background": false
        },
        {
            "Link-type": "Ideal",
            "DataRate": 50,
            "Delay": 10,
            "ErrorRate": 0,
            "Mode": "OTROS",
```

```
            "MTU": 1460,
            "DropTailQueue": 90000,
            "Background": true,
            "BackFeatures": {
                "DataRate": 20,
                "PktSize": 1460,
                "Max_bytes": 0,
                "OnTime": 1,
                "OffTime": 0
            }
        },
        {
            "Link-type": "LMS",
            "Band": "Ka",
            "DataRate": 3,
            "Delay": 0,
            "ErrorRate": 0,
            "Mode": "OTROS",
            "MTU": 1460,
            "DropTailQueue": 90000,
            "Background": false
        }
    ]
}
```