# An Edge-Cloud Approach for Video Surveillance in Public Transport Vehicles

Idelkys Quintana, Luis Sequeira, *Member, IEEE* and Jose Ruiz-Mas

***Abstract*—A Video Surveillance System (VSS) is of primary importance in public transport hubs such as airports, train or bus stations but also inside the vehicle itself. In this paper, we present a heuristic architecture model for on-board video surveillance system based on Internet of Video Things (IoVT) devices which addresses the need for delivering smart video surveillance in public transport vehicles (e.g., buses) minimizing the impact on network performance. A proof-of-concept was implemented using a public Cloud Service Provider (CSP) and two Raspberry Pi as edge computing nodes. On the edge nodes, a Machine Learning (ML) application was deployed along with a network-efficient video streaming system. On the other hand, laboratory tests are included to understand the network traffic dynamic, furthermore results are enhanced with a set of simulations in order to analyze the performance of a video streaming application and $6$ different congestion control algorithms in terms of packet loss and delay.**

***Index Terms*—Edge Cloud, Internet of Things (IoT), IoVT, Video Surveillance, Peer-to-Peer (P2P) Streaming.**

## I. Introducción

En la última década, el uso de VSS se ha incrementado con la finalidad de reducir la tasa de criminalidad, el crimen organizado y el terrorismo. Estos sistemas son ampliamente utilizados en centros de transporte público como aeropuertos, estaciones de trenes y autobuses. Hoy en día, el reconocimiento de imágenes juega un papel muy importante en muchos VSSs ya que no solo permite realizar reconocimientos faciales, sino que además, ayuda a monitorizar, detectar e interpretar actividades. En la actualidad, muchos dispositivos IoT tienen suficiente capacidad para realizar cálculos relativamente complejos, por lo tanto, pueden ser adecuados para usarse como nodos de cómputo en despliegues *Edge-Cloud*. Las tecnologías *Edge-Cloud* se centran principalmente en ampliar las capacidades de la nube [1] y proveer servicios con una latencia más reducida entre el nodo que procesa la información y el usuario final [2]. Además, puede presentar una mejora significativa del ancho de banda, el consumo de energía, proveer cierto grado de privacidad y mejorar la disponibilidad de la aplicación, en caso de una falla en la nube [3].

Los VSSs transmiten una gran cantidad de tráfico de video a la red, lo cual se intensifica en grandes despliegues. Bajo estas condiciones, la concatenación de diferentes flujos de tráfico en dispositivos intermedios (o incluso en el mismo dispositivo) puede crear cuellos de botella que producen pérdida de paquetes y afectan negativamente la Quality of Service (QoS). Los futuros VSSs deben ser confiables, escalables, integrados con tecnologías emergentes como la $5^{th}$ Generation Mobile Network ($5G$) y aprovechar las ventajas del modelo Network Function Virtualization (NFV). En este artículo, se propone una arquitectura heurística para un VSS inteligente en vehículos de transporte público basado en dispositivos IoVT. Dicha arquitectura tiene en cuenta que los dispositivos IoVT pueden ejecutar y transmitir otro tipo de tráfico además del video (datos de sensores y aplicaciones en segundo plano). El VSS aborda la necesidad de integrar aplicaciones de videovigilancia inteligentes, como la detección de objetos mediante la inferencia utilizando ML, sin impactar negativamente la congestión de la red de acceso. El modelo tiene como objetivo gestionar dinámicamente diversas aplicaciones dentro de un conjunto de nodos distribuidos. La solución propuesta pretende minimizar la latencia a la hora de procesar datos, hacer un mejor uso del ancho de banda disponible mediante la transmisión de video únicamente en caso de ser necesario y minimizar las pérdidas durante períodos de congestión.

La novedad de esta propuesta radica en que el VSS transmite el tráfico de video de manera eficiente en términos de QoS y reduce el consumo de ancho de banda. Se corrobora la viabilidad de la misma utilizando productos comerciales al implementar una prueba de concepto con los elementos principales utilizando un CSP, habilitándose la integración con infraestructuras NFV y $5G$. Además, se muestran resultados de pruebas en entornos controlados de laboratorio y de una serie de simulaciones sobre la mejora esperada de la eficiencia de red. Las principales contribuciones del artículo son:

1) Una arquitectura heurística para un VSS aplicado a vehículos de transporte público, la cual permite gestionar diferentes aplicaciones en un conjunto distribuido de dispositivos IoT.
2) Una prueba de concepto que proporciona información sobre la viabilidad de la arquitectura propuesta.
3) Un análisis de parámetros de QoS que permite una evaluación del rendimiento esperado de red; el cual involucra el tráfico de una aplicación en tiempo real y $6$ algoritmos de control de congestión para Transport Control Protocol (TCP) (*Sack*, *Reno*, *Fack* y *Linux*, *Vegas*, *New Reno*).

El artículo se divide de la siguiente manera: la sección II presenta una descripción de sistemas IoVT en VSS. La arquitectura y su implementación se muestran en la sección III. Los resultados y su discusión se proporcionan en la sección IV. Finalmente, el artículo concluye en la sección V.

I. Quintana and J. Ruiz, Universidad de Zaragoza, Dpto. de Ingeniería Electrónica y Comunicaciones, Zaragoza, España, e-mail: {idelkysq, jruiz}@unizar.es

L. Sequeira, Innohub LTD, Londres, Reino Unido, email: luis.sequeira@innohub.uk

## II. Sistemas IoVT en Video Vigilancia

A pesar de la gran cantidad de trabajos de investigación y despliegues de sistemas IoT, no hay un marco único o protocolo a seguir sobre cómo implementar una arquitectura de dichos sistemas. Existe una tendencia de utilizar arquitecturas de tres, cuatro o hasta cinco capas [4] (sensores, red, servicios, aplicación y negocios) en función de la complejidad de los servicios que se desea proveer [5], [6]. En [7], se presenta una arquitectura de capas para la implementación de servicios IoT, el modelo consiste en un conjunto de herramientas de hardware y software que se utiliza para el despliegue de servicios. Mediante el uso de una interfaz gráfica, los usuarios pueden realizar despliegues de infraestructuras IoT. Sin embargo, dicha propuesta no contempla mecanismos para la integración con otras arquitecturas basadas en modelos NFV.

Por otro lado, en [8] se propone una una arquitectura multinivel que provee *Fog as a Service* para servicios IoT extremo a extremo. El modelo expuesto hace uso de tecnologías Software Defined Networking (SDN) para separar los planos de datos y control, lo cual es un punto a favor para poder interconectar con otros sistemas NFV, pero carece de un despliegue real por lo que su viabilidad no ha sido validada. En [9], se discuten las características de una plataforma distribuida que proporciona servicios de cómputo, almacenamiento y red, con la finalidad de mejorar la latencia al procesar ciertas tareas en dispositivos IoT. La arquitectura se enfoca en servicios para el cuidado de la salud y no contempla dispositivos IoVT, por lo que su uso para la implementación de un VSS u otros servicios de video no parece factible.

Un VSS estable y eficiente, basado en IoVT, necesita protocolos confiables a nivel de acceso y servicios. Los dispositivos IoT incluyen, entre otros, sensores visuales o cámaras que son utilizados por aplicaciones de diferentes índoles. Estas aplicaciones pueden transmitir información al mismo tiempo y utilizar diferentes tipos de protocolos para el acceso, enlace o transporte. IEEE proporciona una amplia gama de estándares, como IEEE 802.3 (Ethernet), IEEE 802.11 (Red de área local inalámbrica) e IEEE 802.15.4 (Red de área personal inalámbrica). IEEE 802.15.4 es un protocolo muy popular entre los sistemas IoT, el cual está diseñado para transmisión de datos de baja potencia y baja velocidad (máximo 250 $kbps$) [10]. Sin embargo, los dispositivos IoVT envían una gran cantidad de datos a la red y requieren velocidades de transmisión más altas. En este sentido, IEEE 802.3 e IEEE 802.11 serán una opción más eficiente en un VSS.

Parte del diseño de una arquitectura consiste en una adecuada selección de tecnologías y protocolos para su implementación. Para tal efecto, existe una amplia gama de protocolos de aplicación utilizados en los sistemas IoVT, entre los cuales se puede mencionar MQTT, XMPP, AMQP, DDS, CoAP [11] y HTTP [12]. La elección de los protocolos de transporte (es decir, User Datagram Protocol (UDP) o TCP) depende de cómo estas aplicaciones o servicios han sido implementados. Por ejemplo, CoAP y DDS utilizan UDP para el transporte de datos, mientras que el resto usa TCP. En este sentido, algunos de los algoritmos de control de congestión proporcionarán un mejor rendimiento, menos pérdida de paquetes o más retardo bajo determinas condiciones o niveles de congestión [13]. Otro aspecto que podría afectar la QoS, en términos de *throughput* y pérdida de paquetes, es la presencia de ráfagas de paquetes en el tráfico de aplicaciones, tales como *streaming* de videovigilancia y videoconferencia, produciendo una degradación de la calidad [14].

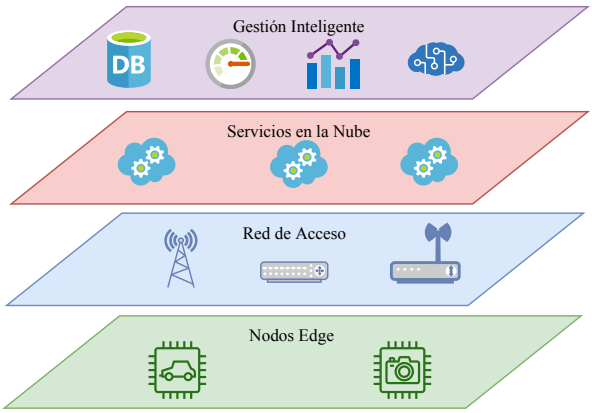


Fig. 1: Arquitectura general para un VSS utilizando dispositivos IoVT en vehículos de transporte público.

## III. Arquitectura Propuesta y su Implementación

En esta sección, se describe la arquitectura propuesta para un VSS basado en dispositivos IoVT. El modelo aborda la necesidad de proveer videovigilancia inteligente en vehículos de transporte público de una manera eficiente en términos de congestión de red. El objetivo es proporcionar gestión dinámica de aplicaciones dentro de un conjunto de nodos *Edge-Cloud* ubicados en una flota de vehículos de transporte público (por ejemplo, autobuses). Además, se implementa una prueba de concepto con los elementos principales para demostrar la viabilidad de la arquitectura propuesta.

### A. Arquitectura Propuesta

La Fig. 1 muestra el modelo general propuesto, en el cual, la capa *Nodos Edge* tiene la función de extender la capacidad de procesamiento de la nube al proveer cierta potencia de cómputo en los dispositivos IoT. Al ubicar dichos nodos cerca de la fuente de datos, se garantiza una reducción en la latencia, lo que permite una respuesta más rápida de la aplicación y una mejor experiencia del usuario final. Esta capa consta de un conjunto distribuido de dispositivos inteligentes IoT equipados con una cámara integrada. Cada nodo es responsable de establecer y mantener una comunicación segura desde el dispositivo a la capa *Servicios en la Nube*, así como de la ejecución de aplicaciones *serverless* y *containers* localmente, además de gestionar mensajes, datos y seguridad. La capa *Red de Acceso* proporciona conectividad (por ejemplo, $4G$, $5G$ o WiFi) para cada nodo.

La capa *Servicios en la Nube* garantiza el acceso a recursos ilimitados para tareas de cómputo más intensivas y una rica fuente de servicios adicionales gestionados en la nube. Dicha capa puede verse como un entorno de múltiples nubes, combinando nubes públicas y privadas. Esto favorece

una mayor redundancia y brinda la capacidad de encontrar el servicio óptimo en la nube para una necesidad comercial o técnica particular. También ofrece escalabilidad, ya que la capacidad y los recursos se pueden aumentar o disminuir según la demanda. Esta elasticidad brinda la posibilidad de ajustar los recursos en función de la demanda real. Esta característica es especialmente importante para las empresas cuyas demandas son altamente estacionales y enfrentan muchos picos de demanda, como suele pasar con los servicios de transporte público.

Siguiendo el paradigma NFV de European Telecommunications Standards Institute (ETSI), la capa *Gestión Inteligente* permite la coordinación de múltiples recursos en diferentes Virtual Infrastructure Managers (VIMs). Esta capa es responsable de la creación, ubicación y gestión del ciclo de vida de aplicaciones y servicios de red; también es capaz de interactuar con un controlador SDN. Esto proporciona una integración entre la nube e infraestructuras SDN. Por otro lado, organiza los despliegues de servicios en *Nodos Edge* utilizando la VIM correspondiente de la capa *Servicios en la Nube*. Los despliegues de servicios se pueden reorganizar en función de los datos que proveen las capas *Servicios en la Nube* y *Nodos Edge*.

Para la gestión de aplicaciones, la capa *Gestión Inteligente* recibe métricas de QoS (pérdida de paquetes, *throughput* y retardo) desde la capa *Nodos Edge*. Con esta información, se pueden instanciar las diferentes cadenas de microservicios de una manera eficiente en términos de utilización de ancho de banda de la *Red de Acceso*. Para ello, se establece la combinación óptima entre el *throughput* máximo que los nodos envían a la red y el algoritmo de control de congestión más adecuado según el protocolo de transporte utilizado.

### B. Implementación

En la Fig. 2, se presenta la implementación de la prueba de concepto para la arquitectura VSS propuesta utilizando dispositivos IoVT. Desde el punto de vista del diseño, esta propuesta se centra en la capa *Nodos Edge* y la interacción con la capa *Servicios en la Nube*. Esto se debe al hecho de que las capas *Red de Acceso* y *Servicios en la Nube* son transparentes desde el punto de vista de diseño, y por lo general, son seleccionadas en función de estrategias comerciales. El propósito de esta prueba de concepto es proporcionar algunas ideas sobre la viabilidad de la arquitectura propuesta y mostrar las decisiones de diseño adoptadas en este enfoque.

Existen ciertos requerimientos mínimos para que un dispositivo pueda ser utilizado en la capa *Nodos Edge*, los cuales dependen del CSP utilizado, en general se puede resaltar: soportar un sistema operativo Linux, un procesador con al menos $1GHz$, $128MB$ de memoria y la potencia requerida por las aplicaciones que se desean utilizar. Para los nodos de cómputo se utilizaron dos Raspberry Pi y la conexión a Internet se proporcionó a través de una interfaz de comunicación WiFi estándar como *Red de Acceso*. La capa *Gestión Inteligente* se basa en Open Source Mano (OSM) de ETSI. Se toma como punto de partida el trabajo desarrollado previamente en [15], donde se implementó un sistema de gestión y orquestación

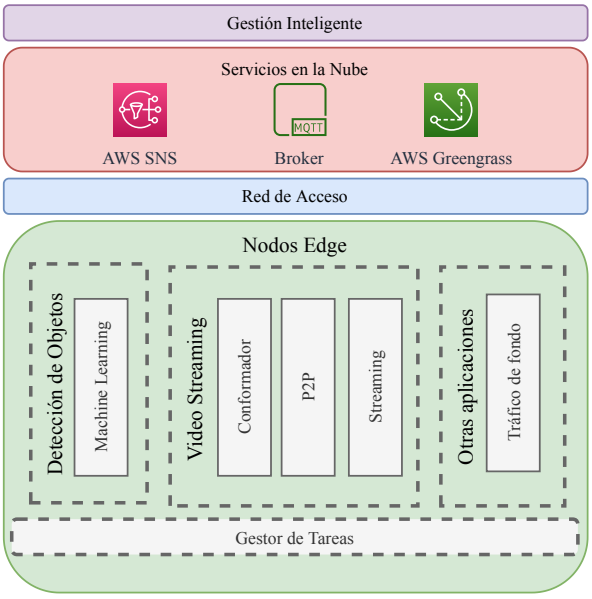


Fig. 2: Implementación de la prueba de concepto para un VSS basada en dispositivos IoVT utilizando como ejemplo dos aplicaciones: *Video Streaming* y *Detección de Objetos*.

de infraestructuras $5G$ para múltiples CSP que incluye una solución SDN (basada en *OpenDaylight*) para gestionar de forma dinámica flujos de tráfico. Para ello se desarrollaron dos *plugins*, uno de los cuales posibilita la comunicación con la nube privada de un fabricante de redes $5G$ permitiendo la conexión con redes de este tipo. El segundo *plugin* se implementa en la interfaz gráfica de OSM con el fin de monitorizar (latencia, *jitter*, *throughput*, ancho de banda y pérdida de paquetes) en tiempo real del estado de la red SDN y de los microservicios gestionados.

En la actualidad casi todos los CSPs ofrecen soporte para infraestructuras IoT, por lo que la selección óptima del mismo debería tener en cuenta, además de aspectos técnicos, estrategias de negocios y modelos de precios. En la capa *Servicios en la Nube*, se utilizó Amazon Web Services (AWS) como CSP, específicamente el servicio *Greengrass* que permite ejecutar *containers* y aplicaciones *serverless* en algunos dispositivos IoT y proporciona una comunicación segura entre el dispositivo y la nube. En la nube se implementó un *Broker* de mensajes basado en un modelo de publicación/suscripción para la entrega de mensajes, de modo que tanto los nodos como la capa de *Gestión Inteligente* puedan acceder a cualquier información enviada desde el *Edge-Cloud*. Para este fin, se usó Simple Notification Service (SNS) de AWS ya que se integra fácilmente con *Greengrass*.

Con el fin de probar el diseño de la arquitectura con un ejemplo de videovigilancia inteligente, se han implementado dos aplicaciones independientes (*Detección de Objetos* y *Video Streaming*) que pueden verse como una cadena de servicios. Por un lado, la *Detección de Objetos* es una aplicación *serverless* que captura imágenes de la cámara integrada y realiza tareas de inferencia ML utilizando el entorno MXNET. Se ha utilizado un modelo de ML preentrenado como ejemplo para facilitar la implementación de la prueba de concepto,

sin embargo, la eficiencia de dicho algoritmo está fuera del alcance de este trabajo.

Para proporcionar cierto nivel de privacidad en entornos de transporte público, como por ejemplo en autobuses, las imágenes son capturadas y procesadas cada $1s$ en los nodos y se eliminan una vez que se realiza la predicción correspondiente, de tal manera que no se envían imágenes a través de Internet ni se almacenan en la nube. Para cada predicción, la *Detección de Objetos* envía un mensaje al *Broker* a través de una conexión segura, dicho mensaje contiene la lista de objetos detectados en formato JavaScript Object Notation (JSON). El *Broker* de mensajes es un servicio administrado en la nube, donde los mensajes con objetos detectados se almacenan y comparten con la capa de *Gestión Inteligente*. Una vez que un objeto peligroso es detectado (por ejemplo, un arma), se ejecuta la aplicación *Video Streaming*. Esto permite hacer un uso eficiente del ancho de banda disponible, ya que se envía una pequeña cantidad de datos (un mensaje JSON) desde el dispositivo IoT, mientras que la transmisión de video se iniciará sólo en el caso que se detecte un objeto peligroso. A pesar de que un análisis de eficiencia energética está fuera del alcance de esta propuesta, al utilizar este mecanismo se garantiza una reducción en el consumo de energía, ya que a diferencia de otras arquitecturas [16], se transmite el video sólo en caso de ser necesario. También, es posible ejecutar el *Video Streaming* bajo demanda en caso que sea requerido.

Por otro lado, la aplicación *Video Streaming* es una cadena de servicios que consta de tres componentes: *Streaming*, *P2P* y *Conformador*. En tiempo real, el *Streaming* usa la cámara y transmite el video, el cual es utilizado por el *P2P* para crear una red P2P con otros nodos y la capa de *Gestión Inteligente* (esto se hace con el propósito de reducir el tráfico enviado por cada nodo). Antes de enviar el tráfico a través de la *Red de Acceso*, el *Conformador* permite realizar una generación de paquetes más eficiente en cuanto a la utilización del ancho de banda, limitando las ráfagas y por tanto reduciendo la pérdida de paquetes. La aplicación *Video Streaming*, que se activa cuando se detecta cierto objeto, consiste en una cadena de *containers*: VideoLAN Client (VLC) se usa para transmitir el video desde la cámara y SopCast se ejecuta como un cliente P2P. El *Conformador* no se ha implementado como parte de la prueba de concepto, en su lugar, se proporcionan resultados de simulación significativos usando Network Simulator (NS)[1] en la sección IV.

Para proveer una idea más clara, el objetivo del *Conformador* es minimizar el impacto causado por el tráfico a ráfagas en los *buffer* de los dispositivos de acceso y enviar a la capa *Gestión Inteligente* métricas de QoS para decidir qué parámetros optimizar en la generación de tráfico. Uno de los motivos por los cuales estas ráfagas se producen es debido a la concurrencia de diversos flujos de datos en la interfaz de los dispositivos de acceso. Los dispositivos IoVT producen diferentes flujos de datos, por ejemplo video, datos de sensores y datos de control. En este caso en particular, se transmiten los datos de las aplicaciones que se ejecutan en los nodos (*Detección de Objetos*, *Video Streaming* y *Otras aplicaciones*) y la comunicación entre el *Gestor de Tareas* y *Servicios en la Nube*, donde existe una confluencia de tráfico que además, involucra diferentes protocolos de transporte. Por ejemplo, la comunicación entre el *Gestor de Tareas* y la nube se transmite sobre HTTPS utilizando TCP, mientras que el *Streaming* y el P2P pueden utilizar una combinación de TCP y/o UDP.

La función principal del *Conformador* es limitar los picos de *throughput* que a veces aparecen cuando se envía tráfico de red. El objetivo es evitar la pérdida de paquetes en el *buffer* durante los períodos de congestión utilizando un algoritmo óptimo de control de congestión TCP [13] y alisado de tráfico [17] (un método de bajo consumo de recursos que un dispositivo IoT puede permitirse), a costa de introducir ciertos niveles de retardo, que sean tolerables por las aplicaciones. Estos retardos adicionales se han seleccionado para controlar la pérdida de paquetes causada por la llegada del tráfico en ráfagas y para mejorar la calidad de la comunicación. En este caso, la tasa de salida de cada paquete varía, pero el comportamiento promedio del tráfico sería el mismo.

## IV. Experimentos y Simulaciones

Para proporcionar un análisis sobre las ideas detrás del diseño, esta sección presenta resultados experimentales de laboratorio y un conjunto extenso de simulaciones. Dado que el video puede ser transmitido sobre TCP o UDP, se ha decidido proveer diversidad en los resultados, realizando las pruebas de la sección IV-A cuando se transmite video con TCP y las pruebas de las secciones IV-B y IV-C cuando se utiliza UDP. La sección IV-A permite comparar el efecto que introduce el retardo en el comportamiento del tráfico cuando una aplicación transmite datos a la nube para ser procesados, o cuando dicho procesamiento se ejecuta en un nodo *Edge-Cloud*. Esto se hace comparando el tiempo entre paquetes del tráfico de red, cuando el dispositivo IoVT envía una transmisión de video a diferentes ubicaciones en la nube. La sección IV-B proporciona una descripción del tráfico transmitido de una aplicación de video P2P, lo cual es útil para comprender y ajustar los parámetros en el *Conformador*. Por otro lado, en la sección IV-C se presenta un análisis con base en parámetros de QoS (es decir, retardo y pérdida de paquetes) de un conjunto de simulaciones realizadas en NS, las cuales describen el rendimiento esperado del *Conformador*. Para ello, se analizan dichos parámetros de QoS cuando la aplicación P2P comparte el enlace con otro tráfico de fondo TCP (como es usual en los dispositivos IoVT). Además, se presenta un análisis de 6 algoritmos de control de congestión para dicho tráfico de fondo TCP.

### A. El efecto del retardo en el video streaming

Tradicionalmente, la detección de objetos utilizando ML se realiza en un servidor externo, el cual puede estar en la nube. Esto requiere enviar información de video desde el nodo a la nube, donde se realizan predicciones utilizando técnicas de ML. Dicho envío de datos es afectado por el retardo entre el nodo que envía el video y el servidor donde se procesa. Con este experimento se quiere mostrar que no solamente el retardo se incrementa, sino que el comportamiento del tráfico de la aplicación cambia, deteriorando la calidad.

[1] https://www.nsnam.org

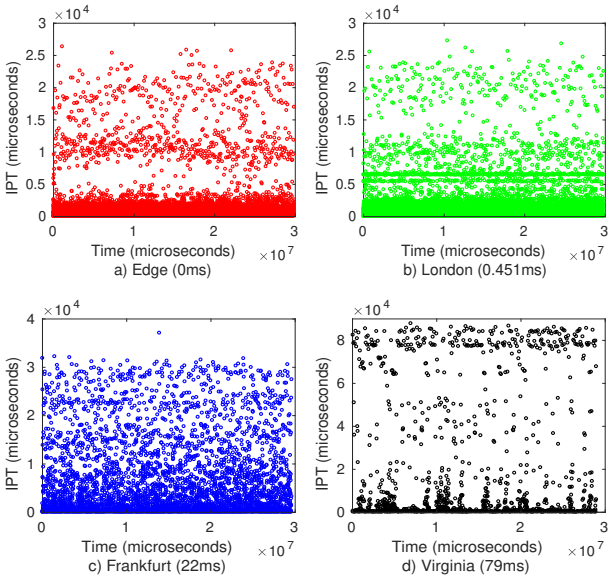


Fig. 3: Comparación del tiempo entre paquetes para diferentes ubicaciones de un servidor en la nube.

Para analizar el efecto que el retardo tiene en el tiempo entre paquetes de una transmisión de video se ha utilizado un Raspberry Pi como nodo *Edge-Cloud*, que envía tráfico de video a un servidor ubicado en una nube privada implementada en OpenStack (la misma utilizada para la capa de *Gestión Inteligente* [15]). De esta forma se mantienen las mismas condiciones que si se utilizara cualquier CSP, y por lo tanto, no se altera la arquitectura propuesta en la sección III. En cada prueba, el nodo envía una transmisión de video al servidor en formato MJPEG utilizando la librería de Python Video4Linux, la cual implementa por defecto TCP para el transporte de datos. Al mismo tiempo, se agrega un retardo a la tarjeta de red del servidor (usando la herramienta *traffic control* en Linux) para emular el retardo desde el nodo a la nube. Para obtener valores reales de retardo, se realiza *ping* a tres regiones (o centros de datos) de AWS (de forma similar que en [18]); los valores promedio de 100 repeticiones a cada localidad son: Londres ($0.451ms$), Frankfurt ($22ms$) y Virginia ($79ms$).

En la Fig. 3 se muestra la comparativa del tiempo entre paquetes cuando se transmite video a las 4 localidades mencionadas. El eje "Y" representa el tiempo entre paquetes, mientras que el eje "X" representa el tiempo en el que cada paquete ha sido transmitido. Como se observa, a medida que la ubicación cambia (el retardo emulado), no solo aumenta el tiempo entre paquetes (lo que es de esperar), sino que además, es evidente que en un mismo período hay menos paquetes. El aumento del retardo produce mayor dispersión en la generación de paquetes por parte de la aplicación, es decir hay más aleatoriedad en los tiempos de envío de paquetes, lo cual se aprecia mejor al comparar la Fig. 3a con la Fig. 3c. A pesar de que el retardo está en un rango tolerable para muchas aplicaciones, de $0.451\ ms$ en la Fig. 3b hasta $79\ ms$ en la Fig. 3d, se aprecia que el impacto que tiene el protocolo de transporte (en este caso TCP) en el tiempo entre paquetes influye en cómo el tráfico es enviado a la red. Por ejemplo, bajo posibles condiciones de congestión o pérdida de paquetes, TCP podría incrementar el retardo experimentado por la aplicación debido a sus mecanismos de control de congestión, pudiendo deteriorar la calidad. Está claro que el principal beneficio de *Edge-Cloud* es que las operaciones de procesamiento tienen menos retardo, ya que están más cerca de la fuente de información. Por otro lado, el comportamiento del tráfico de red de las aplicaciones TCP puede verse afectado negativamente (más allá del retardo adicional), lo que a su vez empeorará la QoS.

Desde el punto de vista de consumo de energía, al procesar la información de video en los nodos *Edge-Cloud*, no se transmite el tráfico a la nube y por lo tanto se reduce la energía consumida por la interfaz de red. Sin embargo, al realizar tareas de ML en los nodos *Edge-Cloud* se incrementa en cierta medida el consumo de energía debido al procesamiento requerido. Por lo tanto, sería interesante realizar un estudio detallado de la eficiencia energética de la arquitectura propuesta en este artículo, lo cual se propone para futuros trabajos.

### *B. Características del tráfico para un video streaming P2P*

Con esta prueba, se desea determinar el *throughput* que alcanza la transmisión de video y que tan fácil es encontrar ráfagas de paquetes durante una transmisión. En este caso, se utiliza un escenario controlado de laboratorio para obtener algunos detalles de la aplicación bajo condiciones con suficientes recursos computacionales y de energía, de manera que las posibles limitaciones de un dispositivo IoT no afecten la forma en que el tráfico de red es generado por la aplicación. Por este motivo, se obtuvo una traza real de tráfico, utilizando un ordenador convencional con un enlace a Internet de 100 $Mbps$. En dicho equipo el cliente P2P se comunicaba con otros 300 *peers* en Internet.

La Fig. 4 muestra la *Empirical Cumulative Distribution Function (ECDF)* del tiempo entre paquetes de la traza obtenida. Dado que el eje "X" es logarítmico debido al amplio rango de valores, se aprecia que el tráfico P2P presenta una gran dispersión en términos de tiempo entre paquetes y que la aplicación no presenta una tasa uniforme de envío de paquetes. Dicha dispersión se observa en la Fig. 4 dado que $\approx 30\%$ de los paquetes se envían con un tiempo entre paquetes $\leq 100\mu s$, otro $\approx 20\%$ entre $100\mu s$ y $1ms$, $\approx 40\%$ entre $1ms$ y $10ms$ y un $10\%$ superior a $10ms$, que podría llegar hasta $1s$. Además, al analizar el tráfico se detectaron muchas ráfagas, compuestas por paquetes con un tiempo entre ellos muy pequeños, de la Fig. 4 se observa que aproximadamente un $50\%$ tienen un tiempo $\leq 1ms$.

Bajo estas condiciones, puede producirse pérdida de paquetes y la QoS podría deteriorarse en dispositivos con más baja capacidad. La aplicación tiene un *throughput* medio de aproximadamente $1.84\ Mbps$. Por otro lado, existen algunos datos interesantes de resaltar, como que el transporte se lleva a cabo mediante UDP, además de que $\approx 50\%$ de los paquetes son pequeños con menos de 100 $bytes$ de longitud y $\approx 45\%$ son paquetes grandes, cercanos a 1300 $bytes$. Los paquetes grandes corresponden a la información de video intercambiada con otros *peers* en Internet. Cada paquete de video es confirmado por un paquete ACK de nivel de aplicación, con una

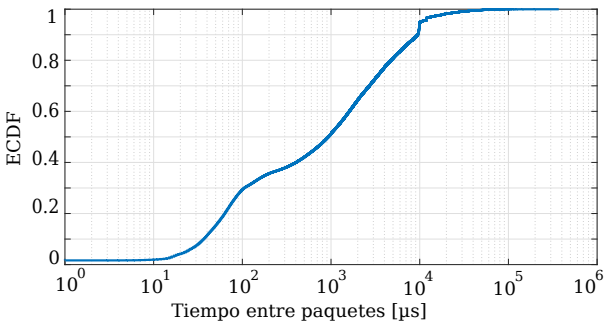


Fig. 4: ECDF del tiempo entre paquetes para una transmisión de video *streaming* P2P.

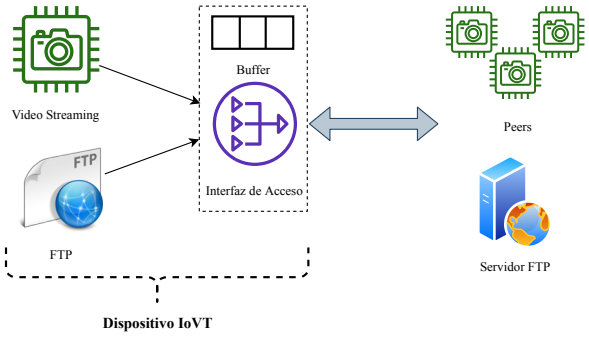


Fig. 5: Escenario de simulación utilizado para las pruebas, en el cual se transmiten dos tipos de tráfico diferentes.

carga útil de 28 $bytes$. Este hecho genera una gran cantidad de paquetes pequeños, y por lo tanto, una baja eficiencia en términos de uso de recursos de red. La capa de aplicación utiliza paquetes pequeños para administrar el estado de cada *peer*, pero también para monitorizar y controlar los paquetes de video a fin de reconstruir y reproducir adecuadamente el contenido.

### C. Resultados de simulación

Para las simulaciones se utiliza el tráfico de la captura anteriormente analizada, además se ha seleccionado un escenario representativo, donde un dispositivo IoT ejecuta la aplicación de transmisión de video junto con otros servicios o aplicaciones TCP en segundo plano. Lo que sería de esperar cuando el nodo IoT envíe información de control a la nube o datos de otros sensores. En la Fig. 5 se representa el escenario de simulación en el que un cliente P2P y otro servicio (File Transfer Protocol (FTP)) comparten el mismo enlace de acceso a Internet. La capacidad del enlace de acceso varía entre 2 y 5 $Mbps$. Capacidades de enlace inferiores a 1.84 $Mbps$ (que es el *throughput* promedio de la traza de tráfico) no se han considerado para evitar el deterioro del rendimiento del cliente P2P. Se han seleccionado múltiples niveles de alisado, que varían de 2 a 5 $Mbps$. Esto ayuda a evitar la congestión en los *buffer*, y por lo tanto, la pérdida de paquetes. Niveles de alisado inferiores al *throughput* medio (en este caso, 1.84 $Mbps$) no se deben usar para evitar el deterioro de la aplicación. Se ha comprobado que para este flujo de datos en particular, si se usan alisados que superan los 5 $Mbps$, el tráfico resultante presenta un comportamiento en términos de ráfagas y tiempos entre paquetes muy similares a cuando se alisa a 5 $Mbps$.

Las pruebas se repiten para diferentes capacidades del enlace, para cada nivel de alisado y para diferentes algoritmos de control de congestión (*Sack*, *Reno*, *Fack*, *Linux*, *Vegas* y *New Reno*) utilizados para el servicio TCP. Además, cada valor mostrado corresponde a la media de 100 iteraciones de cada simulación. En cada caso, se obtiene la pérdida de paquetes para las dos aplicaciones (P2P y FTP). Con la finalidad de realizar las simulaciones en condiciones similares a la prueba de concepto realizada en la sección III-B, se utilizan los parámetros de *buffer* descritos en [19] para una conexión WiFi, ya que son valores ampliamente utilizados en dispositivos comerciales: en el enlace ascendente se implementa un *buffer* de 50 paquetes con una política de gestión de tipo *drop-tail*, mientras que el enlace descendente está dimensionado con un *buffer* de 500 paquetes. Además, el propósito general de la arquitectura es tratar a la red de acceso de manera transparente como se menciona en la sección III, de tal manera que sea el *Conformador*, instruido por la capa *Gestión Inteligente*, el que adapte el tráfico según las condiciones del acceso.

*1) Retardo añadido:* Las técnicas de conformado de tráfico introducen ciertos niveles de retardo, por lo tanto, se debe alcanzar un balance entre los retardos añadidos y los niveles de alisado. Este nivel de retardo introducido dependerá de las características del tráfico de la aplicación y su tolerancia al retardo; en este caso, SopCast puede tolerar la adición de hasta $60s$ [17]. En caso de utilizar otra aplicación, dicha tolerancia al retardo se debe verificar ya que podría impactar el rendimiento y la QoS de diferentes maneras.

Con el objetivo de determinar el umbral de alisado apropiado para este tipo de tráfico, se obtiene el retardo añadido a cada paquete para cada nivel de alisado utilizado (de 2 a 5 $Mbps$). La Fig. 6 presenta el porcentaje de paquetes a los cuales se les ha añadido un determinado retardo. Cada barra representa un nivel de alisado y se han agrupado los paquetes dentro de ciertos umbrales para su análisis. Para el nivel más alto (2 $Mbps$), se introducen retardos intolerables por SopCast y cualquier otro servicio en tiempo real ($> 10s$), esto se debe a que el nivel de alisado está muy cerca del *throughput* medio que necesita la aplicación (1.84 $Mbps$). Sin embargo, a medida que disminuye el umbral de alisado (hacia valores cercanos a 5 $Mbps$), el valor de los retardos añadidos también disminuyen. Se puede observar una cantidad significativa de paquetes que no cambian su tiempo de envío ($0s$) para todos los niveles de alisado. Los retardos añadidos ocurren cuando los paquetes llegan en ráfagas. Siempre existe una disparidad entre el nivel de alisado, los retardos añadidos y la pérdida de paquetes. Por ejemplo, el retardo máximo introducido (en solo algunos paquetes) en la traza alisada a 3 $Mbps$, no supera los $20s$ (tolerable por SopCast) y $\approx 18.3\%$ de los paquetes no se ven afectados.

*2) Pérdida de paquetes:* En la Fig. 7 y la Fig. 8 se muestran las pérdidas de paquetes obtenidas para la aplicación P2P y el servicio TCP utilizado como tráfico de fondo, respectivamente. Los resultados se presentan para cada capacidad del enlace del canal (cada subfigura) y los diferentes algoritmos de control de

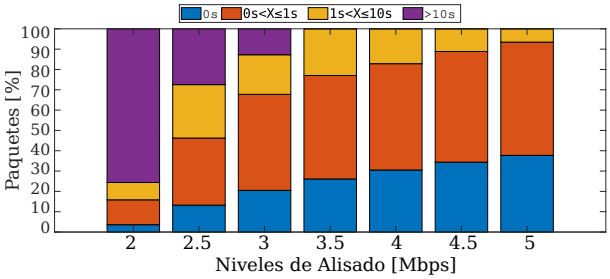


Fig. 6: Porcentaje de paquetes a los cuales se les ha introducido un determinado retardo, según cada nivel de alisado.

congestión para el tráfico de fondo TCP. Por ejemplo, la Fig. 7c muestra los resultados de pérdida de paquetes obtenida por la aplicación P2P cuando la capacidad del canal es de 3 $Mbps$. Donde se observa que *Sack* y *Reno* presentan valores similares de pérdida cuando se alisa a 4 $Mbps$.

Se puede notar que existe una diferencia en la pérdida de paquetes según la variante de control de congestión utilizada. Las pérdidas en el TCP dependen en gran medida de la implementación de los mecanismos de control de congestión. Además, no sólo el servicio FTP tiene pérdidas de paquetes, sino que también se afecta el tráfico de la aplicación P2P, que presenta los mayores niveles de pérdidas, lo cual se aprecia al comparar el nivel de porcentaje de pérdidas mostrado en la Fig. 7 con respecto a la Fig. 8 para una misma capacidad del enlace.

Por otro lado, se observa que existen algunas diferencias notables en cuanto a las pérdidas entre los diferentes algoritmos de control de congestión, por ejemplo, *Vegas* es el que menor cantidad de pérdidas presenta (tanto para SopCast como para el propio FTP). Esto se debe al comportamiento drástico de su control de flujo, ya que cuando detecta congestión en la transmisión detiene el envío de paquetes, lo que hace que no sea la variante más difundida en Internet. Por otra parte, *New Reno* presenta el peor valor de pérdidas. Esta variante puede detectar múltiples pérdidas de paquetes y admite múltiples retransmisiones pero la detección de una pérdida requiere demasiado tiempo y este retardo provoca la pérdida de gran cantidad de paquetes de información. El resto de las variantes (*Sack*, *Reno*, *Fack* y *Linux*) tienen un comportamiento similar, a medida que detectan congestión van disminuyendo el tamaño de su ventana y por tanto se van adaptando al entorno. En general, las pruebas realizadas muestran que el algoritmo de control de congestión del tráfico de fondo repercute en la QoS de otras aplicaciones que comparten el enlace, y por lo tanto, el uso del *Conformador* podría ser una implementación viable en sistemas VSS basados en dispositivos IoVT.

## V. Conclusiones

En este artículo, se presenta una arquitectura heurística de un VSS inteligente para vehículos de transporte público basado en dispositivos IoVT. Se implementó una prueba de concepto utilizando nodos *Edge-Cloud*, una aplicación ML que detecta objetos junto con una transmisión de video P2P. Los resultados muestran que la solución propuesta es interesante en dispositivos de bajas capacidades. Por otro lado, es claro que los algoritmos de control de congestión para TCP tienen un gran impacto, no sólo para las aplicaciones que los utilizan, pero también para el tráfico que comparte el mismo enlace. Como líneas futuras se propone analizar la eficiencia del consumo de energía y del algoritmo de detección de objetos.

## Agradecimientos

Este trabajo ha sido financiado por *European Social Fund*, Gobierno de Aragón (España) grupo de investigación T31 – 20R, Universidad de Zaragoza y Banco Santander.

## References

[1] S. Sansó, C. Guerrero, I. Lera, and C. Juiz, "A platform for lightweight deployment of iotapplications based on a function-as-a-servicemodel," *IEEE Latin America Transactions*, vol. 17, no. 07, pp. 1155–1162, 2019.

[2] O. Nassef, L. Sequeira, E. Salam, and T. Mahmoodi, "Building a lane merge coordination for connected vehicles using deep reinforcement learning," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based iot (feciot): Architecture, applications, and research issues," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4118–4149, 2018.

[5] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[6] P. Sethi and S. R. Sarangi, "Internet of things: architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, 2017.

[7] M. Kalverkamp and C. Gorldt, "Iot service development via adaptive interfaces: Improving utilization of cyber-physical systems by competence based user interfaces," in *2014 International Conference on Engineering, Technology and Innovation (ICE)*. IEEE, 2014, pp. 1–8.

[8] Y. Yang, "Fa2st: Fog as a service technology," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2017, pp. 708–708.

[9] Y. Shi, G. Ding, H. Wang, H. E. Roman, and S. Lu, "The fog computing service for healthcare," in *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech)*. IEEE, 2015, pp. 1–5.

[10] "Ieee approved draft standard for low-rate wireless networks," *IEEE P802.15.4-REVd/D06, March 2020*, pp. 1–945, 2020.

[11] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "Coap congestion control for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 154–160, 2016.

[12] T. Sultana and K. A. Wahid, "Choice of application layer protocols for next generation video surveillance using internet of video things," *IEEE Access*, vol. 7, pp. 41 607–41 624, 2019.

[13] J. Saldana, M. Suznjevic, L. Sequeira, J. Fernandez-Navajas, M. Matijasevic, and J. Ruiz-Mas, "The effect of tcp variants on the coexistence of mmorpg and best-effort traffic," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, 2012, pp. 1–5.

[14] L. Sequeira, J. Navajas, and J. Saldana, "The effect of the buffer size in qos for multimedia and bursty traffic: When an upgrade becomes a downgrade," *KSII Transactions on Internet and Information Systems*, vol. 8, pp. 3159–3176, 09 2014.

[15] I. Quintana-Ramirez, A. Tsiopoulos, M. A. Lema, F. Sardis, L. Sequeira, J. Arias, A. Raman, A. Azam, and M. Dohler, "The making of 5g: Building an end-to-end 5g-enabled system," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 88–96, 2018.

[16] A. Sammoud, A. Kumar, M. Bayoumi, and T. Elarabi, "Real-time streaming challenges in internet of video things (iovt)," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[17] I. Quintana, L. Sequeira, J. Fernandez, J. Ruiz, and J. Saldana, "Minimizing the impact of p2p-tv applications in access links." *IEEE Latin America Transactions*, vol. 17, no. 02, pp. 183–192, 2019.

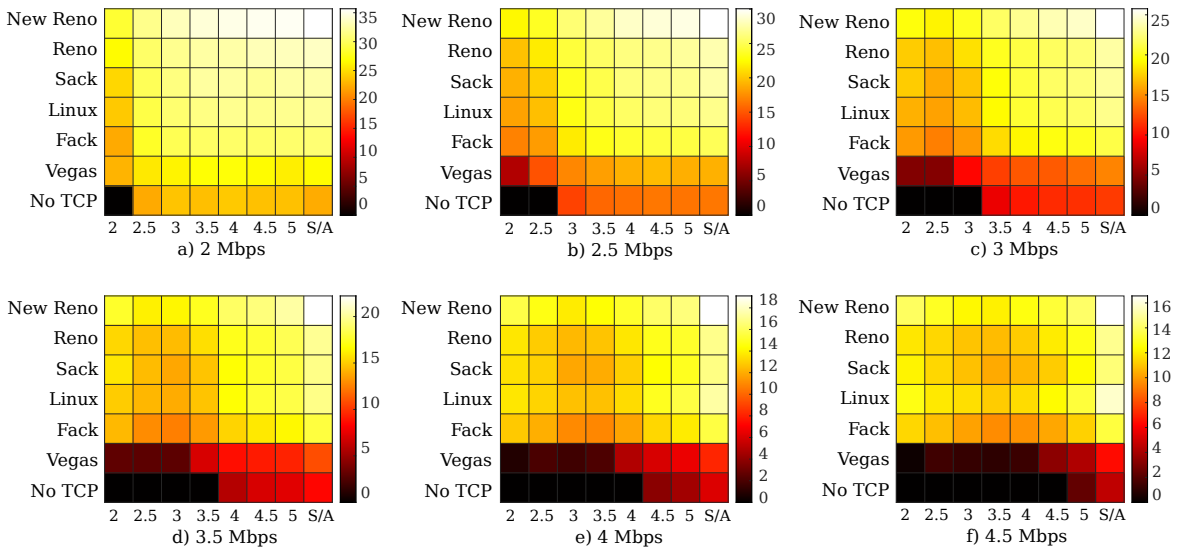


Fig. 7: Pérdida de paquetes UDP para la aplicación P2P cuando comparte el enlace con un tráfico de fondo TCP. Las pruebas se repiten para diferentes algoritmos de control de congestión en dicho tráfico de fondo.

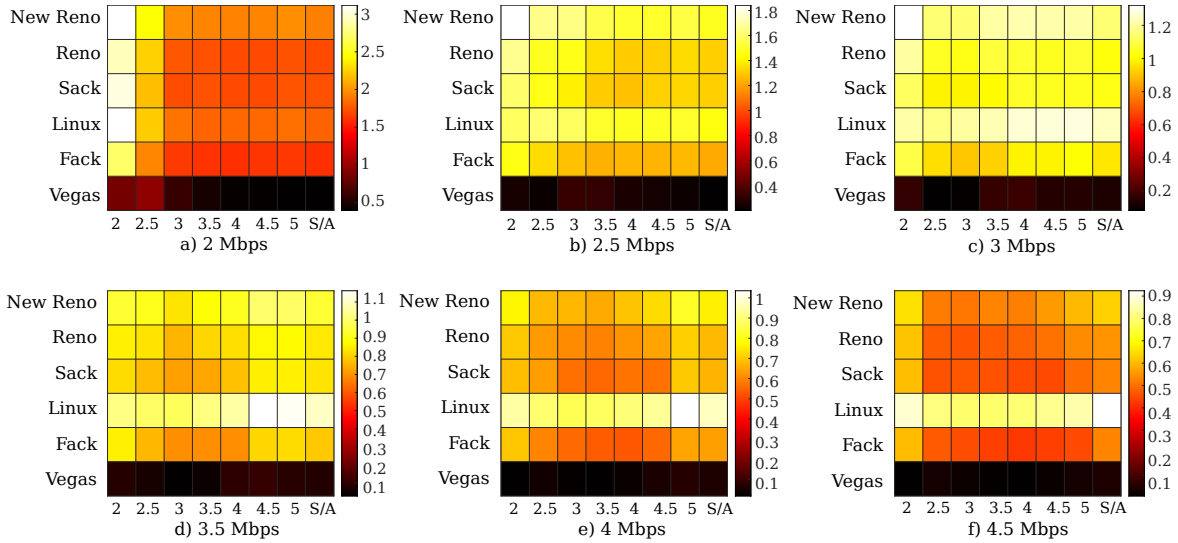


Fig. 8: Pérdida de paquetes para el tráfico de fondo TCP cuando comparte el enlace con una aplicación P2P (UDP). Las pruebas se repiten para diferentes algoritmos de control de congestión en dicho tráfico de fondo.

[18] P. J. Braun, S. Pandi, R. S. Schmoll, and F. H. P. Fitzek, "On the study and deployment of mobile edge cloud for tactile internet using a 5g gaming application," in *IEEE Annual Consumer Communications Networking Conference (CCNC)*, January 2017, pp. 154–159.

[19] L. Sequeira, J. L. de la Cruz, J. Ruiz-Mas, J. Saldana, J. Fernandez-Navajas, and J. Almodovar, "Building an sdn enterprise wlan based on virtual aps," *IEEE Communications Letters*, vol. 21, no. 2, pp. 374–377, 2016.

**Idelkys Quintana** received her M.Sc. in Telecommunications from Central University of Las Villas in 2010. She is a PhD. candidate at University of Zaragoza. She is currently working as cloud consultant for the private sector, focusing on the adoption of cloud technologies with high quality standards. Her research interests include network optimization and QoS/QoE in multimedia services and cloud technologies.

**Luis Sequeira** received his Ph.D. in Information Technologies at University of Zaragoza in 2015. He has been involved in R&D projects funded by European Commission H2020, UK Government EPSRC, Spanish Government and private sector, where he has been designing and implementing PoCs with industry emerging technologies. His research focuses on connected vehicles, cloud orchestration, software defined networking and QoS in real-time services.

**Jose Ruiz-Mas** received his Ph.D. in Telecommunications from the University of Zaragoza in 2001. Currently, he is Associate Professor at the University of Zaragoza. He has been the Co-Investigator of EU Research Projects, the Ministry of Science and Technology, the Sanitary Research Funds, and the Government of Aragon (Spain), Telefónica, Orange and Teltronic. His research activity lies in wireless networks, distributed multimedia system and QoS/QoE.