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

TRANSMISIÓN DE VIDEO UTILIZANDO TCP JUNTO A SVC

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

FELIPE IGNACIO POBLETE SPATE

PROFESOR GUÍA:
CLAUDIO ESTEVEZ MONTERO

MIEMBROS DE LA COMISIÓN:
CÉSAR AZURDIA MEZA
ELIECER PEÑA ANCAVIL

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: FELIPE IGNACIO POBLETE SPATE
FECHA: 2019
PROF. GUÍA: CLAUDIO ESTEVEZ MONTERO

TRANSMISIÓN DE VIDEO UTILIZANDO TCP JUNTO A SVC

Es notable como hoy en día, el contenido visual multimedia, mejora su calidad a pasos agigantados y al mismo tiempo aumenta su impacto sobre el espacio que ocupa en la red. Por tanto es absolutamente necesario que las tecnologías y protocolos de red se adapten de tal manera de permitir que los usuarios sigan disfrutando de una experiencia visual fluida y de mejor calidad.

El *Transmission Control Protocol* (TCP), usado hoy en día para transmitir paquetes a través de la red, es más bien genérico y no se adapta a situaciones particulares como la transmisión de video. Es por esto que surge la necesidad de crear un protocolo de tal manera de hacerlo más óptimo en estas situaciones. En este contexto, se experimenta con un algoritmo de TCP nuevo que busca generar una mejor y más estable calidad de imagen a lo largo del tiempo sin congestionar más la red. Esto pues el tamaño de la *ventana de congestión* del protocolo TCP normal es muy volátil a lo largo del tiempo. Por otra parte, existen distintas tecnologías de compresión de video que podrían verse beneficiadas por un protocolo TCP más óptimo. Una de estas es *Scalable Video Coding* (SVC), la cual tiene la capacidad de codificar varias calidades de un video en un mismo *bitstream*. De este se puede derivar un trozo más pequeño omitiendo secciones para así reducir el ancho de banda necesario para que el video sea transmitido de manera fluida. Este *bitstream* puede quedar tan pequeño como se necesite para mantener una reproducción de video constante a costa de perder fidelidad visual. Este tipo de compresión beneficia a transmisiones que buscan mantener una perfecta fluidez, como por ejemplo el servicio de *streaming* que ofrecen plataformas como *Twitch* y *YouTube*.

Se utilizó un *algoritmo de deadline* en Matrix Laboratory (MATLAB) que simula una transmisión con SVC, el cual genera un valor de tiempo máximo de llegada para cada *frame* del video. A este se le adjuntó el nuevo protocolo TCP para transmitir un archivo de video de resolución (352×288) de 150 frames de largo que muestra objetos en movimiento.

Las comparaciones de este nuevo protocolo con el tradicional muestran como el nuevo algoritmo es justo y no sobre congestiona la red. Además se observa mayor estabilidad sobre el *algoritmo de deadline* y tiempos de llegada mas cortos cuando la conexión sufre muchas pérdidas. Tanto el PSNR promedio como de cada *frame* del video, para distintos escenarios, resultó siempre ser mayor o igual cuando se utiliza el nuevo protocolo respecto del tradicional.

Se concluyó que en condiciones normales la diferencia en calidad que observará el usuario no es siempre perceptible, pero sí se disminuye la frecuencia de los artefactos cuando existen grandes pérdidas de paquetes o alta latencia.

A mi mamá Raquel, mis primos, abuelos
y amigos de la Universidad.

Agradecimientos

Agradezco de sobremanera el esfuerzo que a puesto mi madre Raquel en ayudarme a recorrer de forma mas amena mi camino por la universidad. Me he convertido en la persona que hoy soy, gracias a las increíbles experiencias que ella me ha dado la oportunidad de vivir.

A mi abuela Mirna, por no solo forzarme a engordar los fin de semana en Malloco, sino también por la paciencia que me tuvo al teléfono cuando le pedía que me enseñara a cocinar.

A mi abuelo Jaime, por siempre contar con su buena voluntad cuando necesité algún favor y por las tardes que compartimos trabajando arreglando cosas, las que me permitieron darme cuenta lo mucho que tenemos en común.

A mi prima Valentina, que ha vivido toda la vida conmigo como una hermana, por ser la mejor compañera de departamento que alguien pudiese tener.

A mi primo Nicolás, al considero un hermano desde pequeños, porque por mucho que pasen los años, siempre ha estado ahí para mi.

A mi tío Jaime, por el cariño, las risas y los buenos ratos que pasamos en familia.

A mis amigos eléctricos, por esas tardes jugando Rocket League y también esos fin de semana estudiando a más no poder, siempre entregándonos todo el apoyo posible. Agradesco especialmente a Fredes por ser el MVP cuando se trata de hacerme gastar mi tiempo en ocio.

A mi amigo Rafael, que desde el inicio siempre ha sido un gran compañero con el que comparto la mayoría de las cosas que me apasionan.

A mi profesor guía Claudio, por el apoyo en cada parte de este trabajo y las largas conversaciones sobre nuestros intereses en cada reunión.

Finalmente a mi jefe de carrera Andrés, por siempre tener una increíble disposición para ayudarme con mis infinitos problemas a lo largo de mi carrera.

Familia y amigos, gracias por todo en estos largos 6 años. Jamás olvidaré los buenos ratos que pasé y siempre intentaré aprender de los malos, para que sigamos manteniendo la conexión y el cariño que hoy nos une.

Sinceramente, gracias... por todo.

Tabla de Contenido

Índice de Ilustraciones

Acrónimo	**Significado**
ACK	Acknowledge
DASH	Dynamic Adaptive Streaming over HTTP
GOP	Group of Pictures
JSVM	Joint Scalable Video
NewTCP	Nuevo Protocolo TCP
OldTCP	Protocolo TCP Tradicional
PSNR	Peak Signal to Noise Ratio
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Tabla 1: Lista de Acrónimos

Capítulo 1

Introducción

1.1. Motivación

La mayor cantidad de datos que fluyen a través de la red hoy en día pertenecen a la categoría de multimedia. El consumo de este contenido a crecido a pasos agigantados debido a su popularidad y espacio que ocupa en el mercado. De hecho, en el presente año, Netflix, YouTube y otras compañías que ofrecen servicios similares ocupan más de la mitad del tráfico total de la red en los Estados Unidos [17].

Las tecnologías y esfuerzos que se han puesto en mejorar la calidad del contenido visual es de gran envergadura en todos los países del mundo. No mucho tiempo atrás la resolución 1080p (FullHD) se convirtió en un mínimo estándar para los televisores y monitores. Sin embargo hoy en día la resolución 2160p (4K), 16 veces mayor a 1080p, a ocupado la mayoría del mercado de televisores y contenido cinematográfico. A esto se le agregan tecnologías implementadas recientemente como HDR (High Dinamic Range). Todo esto no hace más que aumentar drásticamente el tamaño del contenido visual en la internet y así mismo el ancho de banda quw utilizan.

Es por lo anterior, que surge la necesidad de crear sistemas o tecnologías que amortigüen estos cambios radicales en el tamaño de los archivos multimedia y mejoren su transmisión a través de la red, para que los usuarios que consumen este tipo de contenido puedan mantener una experiencia fluida, es decir, que no se genere ningún *trade-off* a causa del aumento en la calidad de imagen.

Para adecuarse a estos cambios, las tecnologías de transmisión de datos y compresión de video han continuado evolucionando. Hoy en día vemos como modernos sistemas operativos (como Windows y Linux) han migrado su protocolo de comunicación a *TCP Cubic* para este fin. Por otro lado, se ha mejorado la famosa compresión de video *Moving Picture Experts Group* (MPEG) y creado nuevas tecnologías como SVC [19]. El presente trabajo se enfocará en el efecto que tiene el protocolo TCP sobre la tecnología de SVC.

1.2. Hipótesis de Investigación

Un flujo de paquetes de tamaño constante permite que todo video, transmitido en tiempo real a través de la red, perciba una disminución en la variación de su PSNR.

1.3. Objetivos del Trabajo de Memoria

A continuación se muestran los objetivos generales y específicos del presente trabajo.

1.3.1. Objetivos Generales

Diseñar un nuevo algoritmo de ventana de congestión de TCP más adecuado para la transmisión de contenido visual.

- Estabilizar la calidad de imagen que observa el usuario.
- Mantenerse alejado del *deadline* para reducir las perdidas de frames importantes.

1.3.2. Objetivos Específicos

- El nuevo algoritmo de ventana de congestión debe ser justo y no congestionar más la red (mismo valor promedio a lo largo de tiempo).
- Justificar el uso del nuevo protocolo comparando su efecto sobre SVC con TCP tradicional.
- Obtener el PSNR a lo largo de cada *frame* de un video para distintos escenarios.

Capítulo 2

Conceptos Básicos y Revisión Bibliográfica

En esta sección se dará contexto al conocimiento necesario para lograr entender correctamente todos los temas que se abordarán en el presente trabajo.

2.1. TCP/IP

El *Transmission Control Protocol and Internet Protocol* (TCP/IP) es un modelo jerárquico que construye y ordena los paquetes que se mueven en la red. Un paquete de información que viaja a través de la internet se puede mostrar como un acoplamiento de distintos *bits* de información los cuales sirven de indicadores para que los datos que queremos enviar (que se encuentran en medio de estos) lleguen a su destino correctamente.

El acoplamiento de información se divide en varias capas. En la Figura 2.1 se muestra como la información que queremos enviar llamada *Application Data* se convierte en un paquete más grande después de haber agregado distintos indicadores para su correcto transporte. La capa de interés, en este caso, es la de transporte, pues en esta es donde el control de congestión hace efecto. Además es en esta capa donde la compresión SVC y el *algoritmo de deadline* trabajan juntos para decidir la calidad final que tendrá el video.

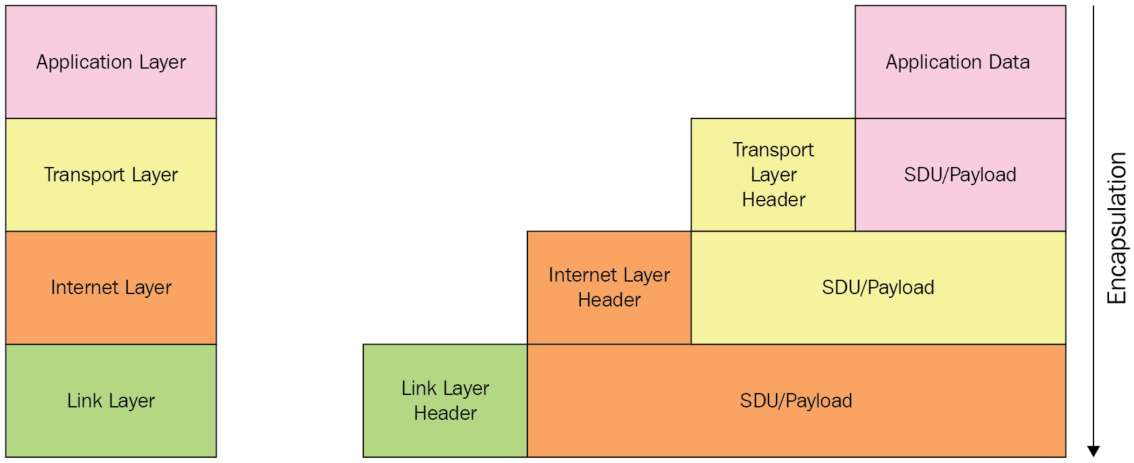


Figura 2.1: Modelo OSI de Capas TCP/IP [21].

2.1.1. Ventana de Congestión

La ventana de congestión es el proceso que ocurre en la capa de transporte del modelo TCP/IP, que indica principalmente cuantos paquetes se pueden enviar en un mismo instante de tiempo al receptor. De manera simple, el valor de ventana de congestión comienza normalmente con un valor de 4 y crece de uno en uno cada vez que los paquetes llegan correctamente al receptor. En el instante en donde alguno de estos paquetes se pierde o llega con error, se divide el valor de la ventana de congestión en 2 aproximando al entero más bajo [1].

Por otro lado, se han realizado ciertas adiciones a este algoritmo de forma de adaptarse rápidamente al estado en que se encuentre la red. Algunos de estos nuevos mecanismos se pueden observar en la Figura 2.2. Dos se los más importantes son los siguientes:

- *Slow Start*: La ventana crece de manera cuadrática al comienzo de la transmisión hasta que ocurra alguna pérdida de un paquete.
- *Congestion Avoidance*: Como se explicó anteriormente, cada vez que un paquete se pierde la ventana se divide por 2 de manera de evitar congestionar la red.

Se define la pérdida de un paquete cuando el cliente recibe una menor cantidad de estos en comparación al tamaño de su ventana de congestión.También es posible identificar una pérdida cuando se recibe un ACK (proveniente de la palabra en inglés *Acknowledge*) de un paquete más reciente siendo que aún no se ha confirmado la llegada de alguno que se envió antes que este.

Por otro lado, existen otros mecanismos como el *Fast Retransmit* y *Fast Recovery*, los cuales en conjunto a lo anterior, abarcan la mayoría de la conexiones TCP a nivel global. Vale la pena mencionar que existen otras alternativas a este algoritmo tales como TCP Reno, TCP Vegas y TCP Cubic, siendo esta última usada por algunos de los sistemas operativos más modernos como Linux y Windows [18].

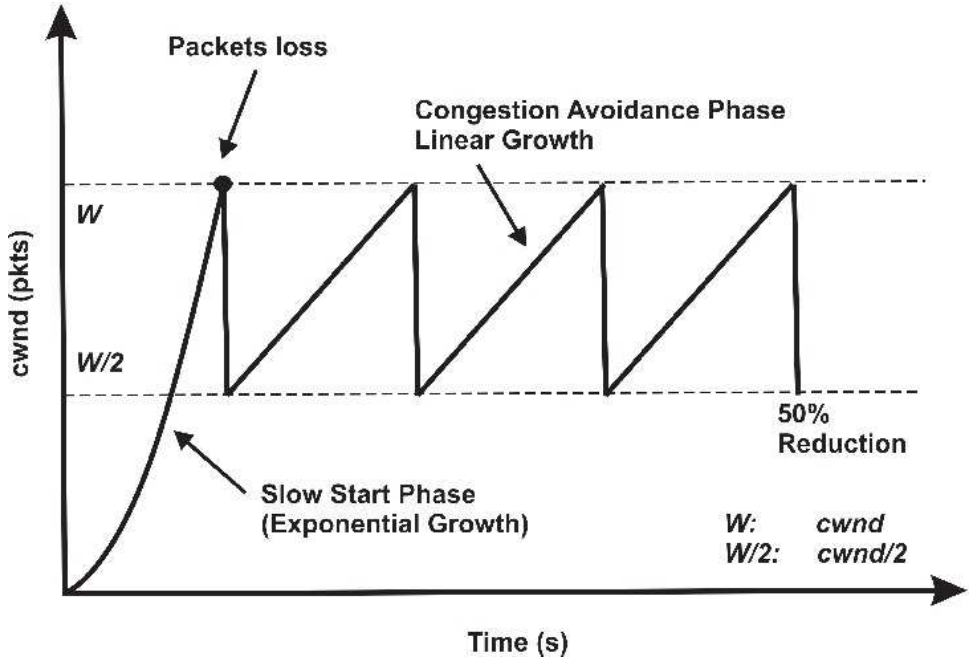


Figura 2.2: Comportamiento de la Ventana de Congestión de TCP [16].

2.2. Transmisión de Video

Cuando se transmite un video por internet, especialmente en situaciones donde se necesita ver el contenido de manera fluida (campeonatos deportivos, *streaming*, noticias en directo, etc.) se necesita no solo un óptima forma de transportar la información si no también alguna manera de comprimir estos datos para generar menos dependencia del estado de la red. Hoy en día este óptimo estándar considera a lo menos 30 imágenes por segundo en una transmisión [13]. Inicialmente, se buscaba comprimir lo más posible el archivo de video con pérdidas de calidad de imagen mínimas, sin embargo es limitado cuanto se puede comprimir un video sin comenzar a perder información importante de este. Adicionalmente, todos los usuarios tendrán un ancho de banda distinto y variante en el tiempo por lo que es absolutamente necesario tener disponibles varias calidades del mismo video.

2.3. Compresión de Video y SVC

El objetivo principal de comprimir un video consiste en eliminar todos los datos redundantes de información que contenga. Para visualizar de mejor manera este proceso podemos usar como ejemplo lo siguiente: Se tienen dos imágenes consecutivas que desean enviar y sucede que en parte de la segunda imagen se muestra exactamente el mismo objeto que en la primera y solo cambia el *background*. Sabiendo esto, podemos simplemente enviar el *background* de la segunda imagen y solo indicar que existe el mismo objeto que en el primera imagen, esto permite un ahorro enorme en la información que se debe enviar, pues es algo que ocurre de manera muy frecuente. Además también es posible comprimir aún más estas imágenes si poseen alguna información que sea imperceptible al ojo humano, como por ejemplo dos

colores uno al lado de otro que se diferencian levemente en su tonalidad.

Otras técnicas más complejas consisten en predecir o extrapolar información de cada *frame* (*Inter Picture Prediction*) o también compensar información perdida entre *frames* cuando se tienen los datos de la imagen posterior y anterior a la que se desea arreglar (*Motion Compensation/Estimation*) [8].

Existen muchos formatos de codificación que utilizan diferentes algoritmos para comprimir un video, una de las más famosas es H.264 [5]. De hecho, SVC es una variante de esta última, la cual permite codificar variadas calidades dentro de un mismo *bitstream* de tal manera que no sea necesario enviar distintas calidades del video sino que se envía una única información que contiene todo sin ocupar mayor ancho de banda. De esta forma cada usuario decodifica esta información de la forma que le sea más conveniente para que el video mantenga la fluidez sin importar que dispositivo o estado de red tenga. En la Figura 2.3 se muestra un esquema simple del funcionamiento de SVC.

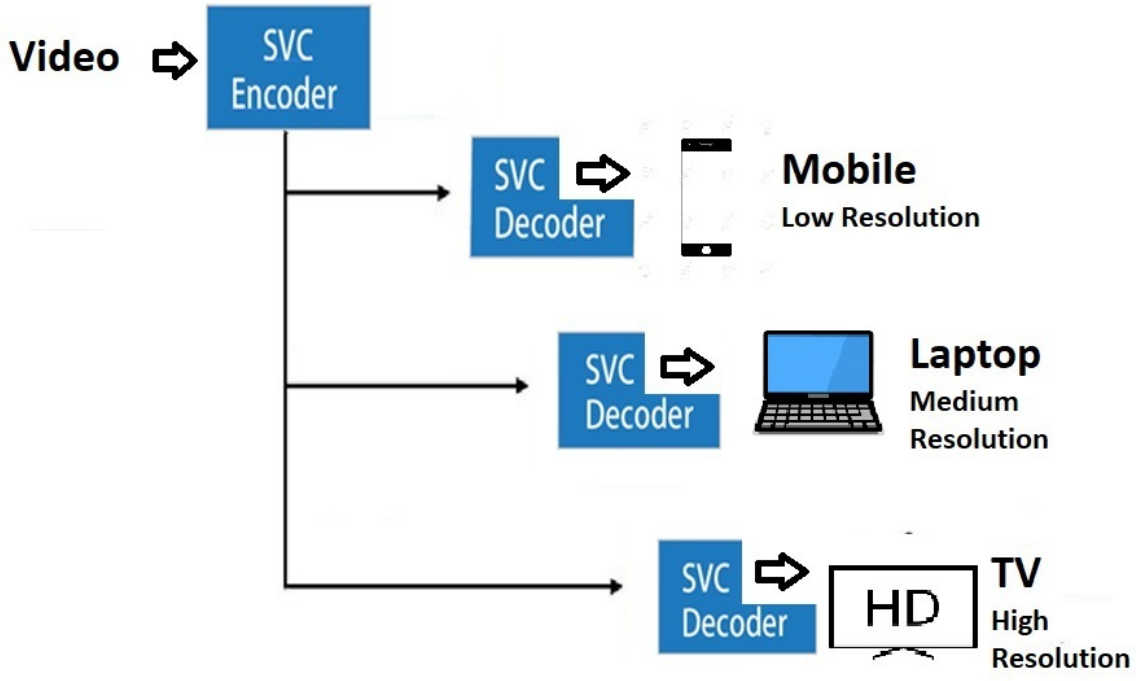


Figura 2.3: Scalable Video Coding (Adaptado de [3]).

Como se observa en el esquema anterior, cada dispositivo puede obtener una distinta calidad y cantidad de frames por segundo según requiera. Esto solo contando con un único *stream* de datos lo que permite una transmisión más fácil y no requiere que el servidor contenga múltiples calidades del mismo video pre-codificadas (algo que Netflix y YouTube hacen hoy en día) ahorrando así una importante cantidad de espacio.

2.4. Group of Pictures (GOP)

El GOP es una secuencia definida de *frames* (imágenes que componen un video). Esta secuencia se compone de distintos tipos de *frames* los cuales están codificados de tal manera de reducir el tamaño total que tendría una secuencia normal de imágenes aprovechándose de las similitudes que poseen varios frames consecutivos (ideal para este trabajo). El tamaño de estas secuencia es de 7 *frames* normalmente, repitiéndose a lo largo de todo el video.

A continuación se muestra un GOP junto a la cantidad de información relativa que posee cada *frame* respecto de los demás.

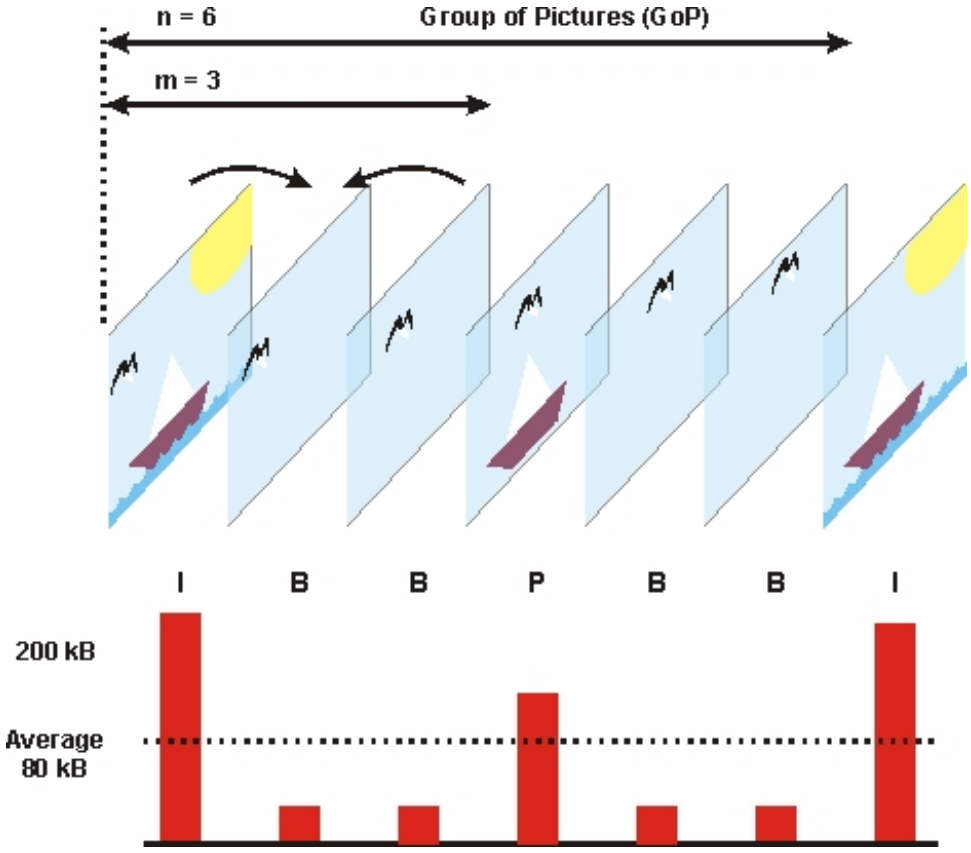


Figura 2.4: Group of Pictures.

- I-frame (*Intra frame*): Este es el que contiene más información por lo que es el más importante de todos. Funciona como base para generar los B *y* P utilizando predicción.
- P-frame (*Predictive frame*): Se construye a partir de la información de la diferencia entre los *frames* previos.
- B-frame (*Bipredictive frame*): Contiene información de la diferencia de los *frames* previos y posteriores.

Solo los *frames* B y P necesitan de los I para decodificarse, dado que estos solo contienen información parcial sobre la imagen. Es aquí donde se usa la técnica de *Motion Compensation/Estimation*. Es importante mencionar que en el video final los *frames* B y P, al decodificarse, se verán completos y no parcialmente como en la Figura 2.4.

2.4.1. Codificación YUV

La codificación YUV consiste en un técnica de procesamiento de *frames* acorde a un espacio de colores que tiene en cuenta la percepción humana, lo que permite usar un ancho de banda menor para las componentes que diferencian el color [14]. Por tanto se logra que los errores de transmisión o las imperfecciones en la compresión se oculten más fácilmente que usando RGB (Red, Blue, Green). La siguiente figura muestra como un *frame* de un video en formato YUV se divide en sus componentes de colores, comparándolas con RGB.

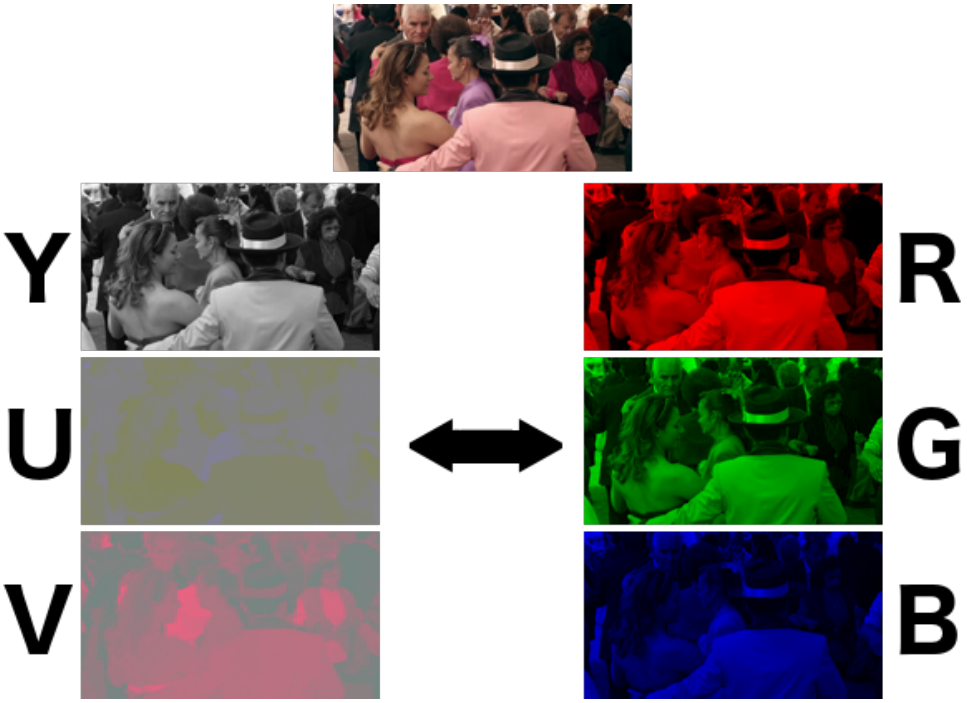


Figura 2.5: Codificación YUV vs RGB.

De la figura anterior se logra ver fácilmente como cada *frame* en la codificación RGB posee información importante sobre la imagen, al contrario de YUV donde solo el *frame* Y, que contiene la luminiscencia de la imagen, posee la información importante. Es por esto que para el presente trabajo se utilizará un video en formato YUV, debido a que cualquier pérdida u omisión de información de los frames U y V solo disminuirá levemente la calidad final del video.

2.5. Peak Signal to Noise Ratio

El *Peak Signal to Noise Ratio* (PSNR), es una forma de medir y comparar la calidad de compresión de imágenes [10]. En términos simples, corresponde a una medida de que tan diferentes son 2 imágenes entre si. En general se expresa en términos de decibeles (dB) debido a su comportamiento no lineal. Se representa matemáticamente como:

$$PSNR = 10 \times \log \frac{MaxErr^2 \times w \times h}{\sum_{j=0}^{h} \sum_{\mathrm{i}=0}^{w} (x_{\mathrm{i}j} - y_{\mathrm{i}j})^2} [\mathrm{d}B]. \tag{2.1}$$

MaxErr es el valor máximo en decimales que puede tomar cada pixel (255). h y w son el alto y ancho del video respectivamente y por último x e y son los valores que toma cada pixel del *frame* original y el reconstruido. Dado que se trabajará con un video en formato YUV, se puede calcular el PSNR para cada codificación de color (Y, U y V) por separado. Puesto que el PSNR-Y correspondería al valor más relevante, dada la alta sensibilidad del ojo humano a la luminiscencia, se utilizará como referencia para cuantificar y comparar ambos algoritmos de TCP.

2.6. Algoritmo de Deadline

Este algoritmo, adaptado especialmente a la transmisión de video por la red usando SVC, consiste en generar un tiempo limite (*deadline*) en la cual la información del video debe llegar al usuario para que mantenga su fluidez [7]. Cuando los paquetes de información están llegando cerca del *deadline*, se comienzan a omitir los menos relevantes (*frames* B y P) de tal manera de alejarse de este *threshold* con pérdida de calidad mínima. Si se llega a traspasar este límite, entonces también se comienzan a omitir *frames* del tipo I generando, inevitablemente, espacios en blanco para estos instantes de tiempo en el video.

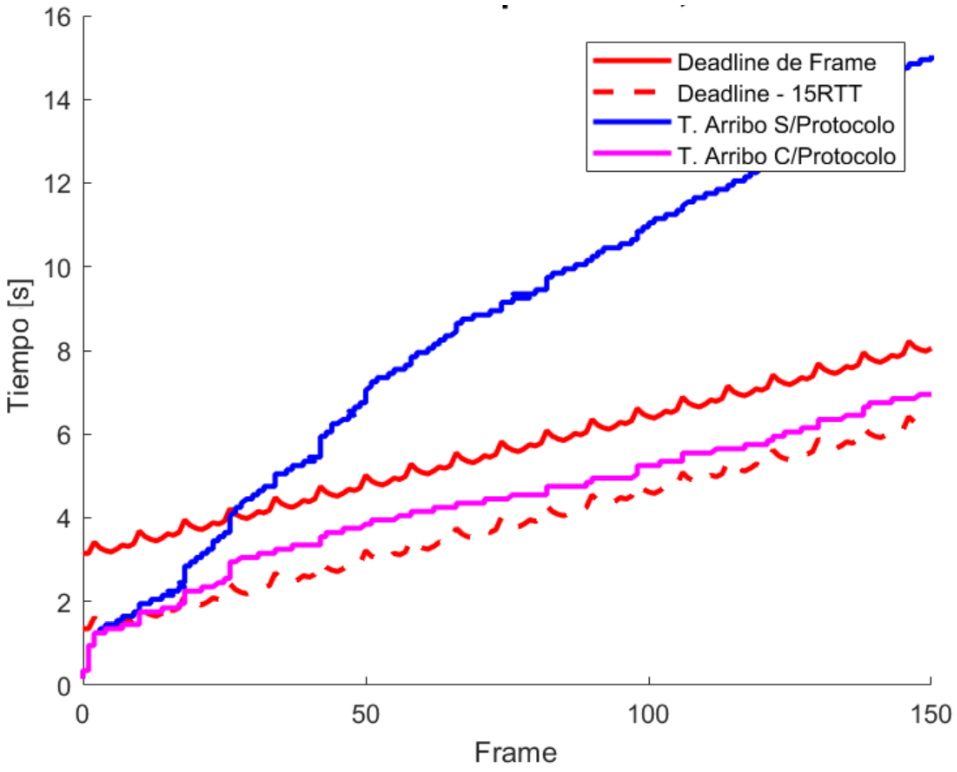


Figura 2.6: Algoritmo de Deadline.

La Figura 2.6 muestra un ejemplo de este algoritmo en funcionamiento. La curva azul representa los tiempos que demora cada *frame* en llegar en condiciones normales sin el uso de este algoritmo. El video dura 5 segundos, por lo que habría que esperar 16 segundos para verlo de manera fluida. La curva rosada muestra el efecto del algoritmo de deadline, es decir, se comienzan a omitir paquetes de tal forma de no tener tiempo de espera. La linea roja es el *deadline*, es decir, el tiempo máximo que se puede demorar el *frame* en mostrarse sin que el video deba parar, y por último, la curva punteada roja representa el límite donde el algoritmo comienza efectivamente a omitir paquetes, es decir, bajo este último límite no existe omisión de paquetes, por lo que la calidad del video se mantiene en su máximo.

2.7. Estado del Arte

2.7.1. Congestión en TCP

A lo largo de los años se han desarrollado múltiples variaciones del protocolo TCP, algunas de ellas completamente distintas del original. Por ejemplo se tiene FAST TCP [12] y TCP Vegas [20] los cuales se enfocan en detectar congestión a partir de los retrasos que sufren los paquetes al ser enviados. Por otro lado existe TCP Reno [11] que posee un sistema de retransmisión con *fast recovery*. Aún más importantes, se han creado protocolos como TCP BIC [9] y Cubic [2], los cuales están optimizados para conexiones con alta latencia pero con mucho gasto de ancho de banda.

En un contexto donde el tiempo de llegada de los paquetes es de suma importancia (i.e tener baja latencia es beneficioso), es de interés observar el comportamiento de TCP Cubic sobre SVC, de tal manera de mostrar cuanto puede amortiguar el efecto de la latencia.

Por otro lado, existe otro tipo de protocolo de red de menor confiabilidad llamado *User Datagram Protocol* (UDP). Se diferencia principalmente con TCP por el hecho de que no requiere de confirmación por parte del cliente cada vez que llegan los paquetes, es decir, es un protocolo orientado a la transmisión en una sola dirección. Esto permite menos retrasos y mayor *throughput*, sin embargo en caso de perderse información importante, como un *frame* del tipo *I*, no se solicita retransmisión por lo que el usuario verá frecuentemente *frames* en blanco al momento de perder paquetes ya sea por problemas en la red o por congestión, inclusive si la mayor cantidad del tiempo la conexión es estable con un alto ancho de banda. Desde UDP nacen otros protocolos tales como *Real-Time Protocol*, que permite transmitir video en tiempo real, aunque sus principales desventajas es el alto uso de ancho de banda, ïnjustaço-existencia con TCP (importante variable para este trabajo), y además es bloqueado fácilmente por distintos *firewalls* a lo largo de la red [4].

2.7.2. Transmisión de Video

Al día de hoy, dependiendo de la plataforma, existe una variedad de formas de transmitir archivos multimedia a través de internet. Dada la enorme cantidad de tráfico que tanto YouTube como Netflix producen, es lógico pensar que estas plataformas están utilizando los protocolos de transmisión más modernos y populares en el mundo. Como se mencionó anteriormente, estas empresas usan una técnica que consiste en codificar previamente un mismo video en distintas calidades. Dependiendo del estado de la conexión del usuario, los clientes elijen que calidad desean recibir, sin embargo esto no asegura que no se produzcan cortes en la reproducción del video dado que existe un *delay* al decidir y cambiar la calidad, además de estar cerrados a una limitada y pequeña cantidad de calidades diferentes.

El protocolo recién mencionado es llamado *Dynamic Adaptive Streaming over HTTP* (DASH), que se encuentra representado en la siguiente figura:

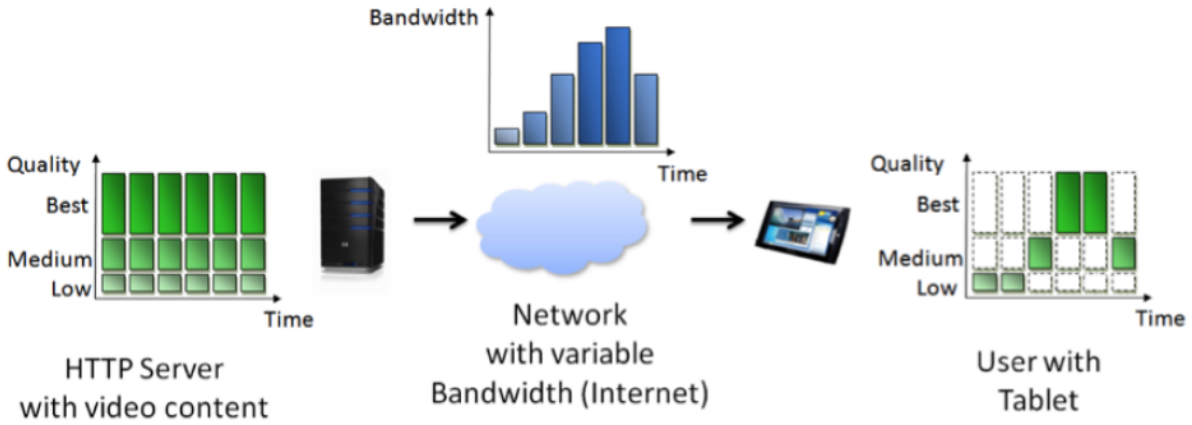


Figura 2.7: Transmisión DASH [22].

Capítulo 3

Implementación

Primero que todo, se utilizará **MatLab 2017b** para crear un nuevo algoritmo de control de congestión de TCP que tendrá por enfoque tener un valor de ventana estable y constante a lo largo del tiempo, con el requerimiento de ser justo (o *fair*) al co-existir con el protocolo tradicional de congestión. Se busca esta estabilidad, con el fin que se traduzca en una calidad de imagen menos variable en la transmisión de video, y por tanto, que se vea menos afectada por cambios en el estado de la red que se produzcan por pequeños lapsos de tiempo (menor a 1 segundo).

Posteriormente se analizará el efecto de este protocolo conectado al *algoritmo de deadline*. Para esto es necesario tener instaladas las librerías de JSVM (Joint Scalable Video Model), el cual es usado para la codificación y decodificación final del video.

3.1. Nuevo Algoritmo de Control de Congestión

El algoritmo nuevo consiste, fundamentalmente, en un formato similar a como funciona normalmente TCP, con la diferencia que el nuevo, al que llamaremos **NewTCP**, posee un valor de ventana de congestión de referencia (con valores decimales) y una real (valores enteros). La ventana de referencia es tal que aumenta de forma muy lenta, pero también disminuye poco a poco. Es importante destacar que disminuye de tal forma que toma el promedio entre el valor de ventana decimal más grande y el más pequeño que existió anteriormente, y a este promedio se le multiplica una constante Alfa (entre 0 y 1) que finalmente reduce este valor.

En la práctica el valor de la ventana de referencia se muestra como una versión diminuta de TCP tradicional (que llamaremos **OldTCP**) que oscila alrededor de su promedio. Este tipo de ventana, por si sola, es una versión más estable que **OldTCP**, sin embargo es posible estabilizar más aún el algoritmo utilizando esta ventana solo como referencia. Para esto se crea una segunda ventana que finalmente será la real, la cual solo actualiza su valor cuando se traspasa un cierto umbral. El funcionamiento general de este algoritmo está representado en la Figura 3.1.

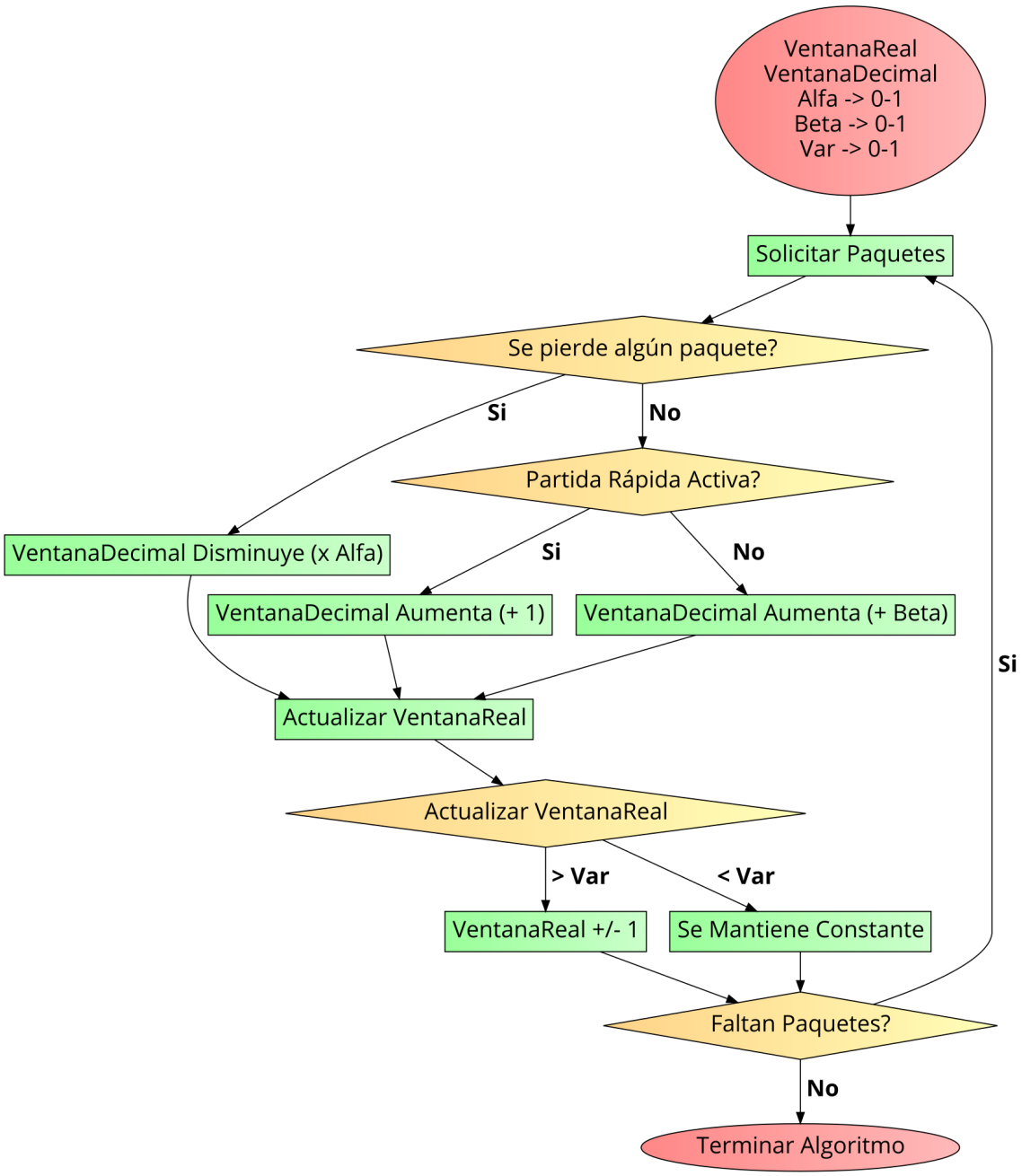


Figura 3.1: Diagrama de Funcionamiento del Nuevo Protocolo TCP.

Es importante destacar que los valores de los factores Alfa, Beta y Var son arbitrarios y pueden modificarse para variar el crecimiento, valor promedio y estabilidad de la ventana de congestión.

En primera instancia, es posible variar los parámetros del algoritmo para obtener cualquiera sea el efecto deseado, sin embargo es de vital importancia tener en cuenta que este control de congestión estará compitiendo con TCP tradicional (u otros protocolos) en la realidad, por lo que es imprescindible ajustar el comportamiento de este nuevo algoritmo para que co-exista de manera justa con otros protocolos.

En la Figura 3.2 es posible apreciar el comportamiento del nuevo protocolo contrastado con el tradicional para una probabilidad de pérdida de paquetes $p = 0,01$ inicial, cambiando su valor a $p = 0{,}05$ a partir de la ventana número 320, de forma de además observar lo que ocurre cuando cambian las condiciones de red.

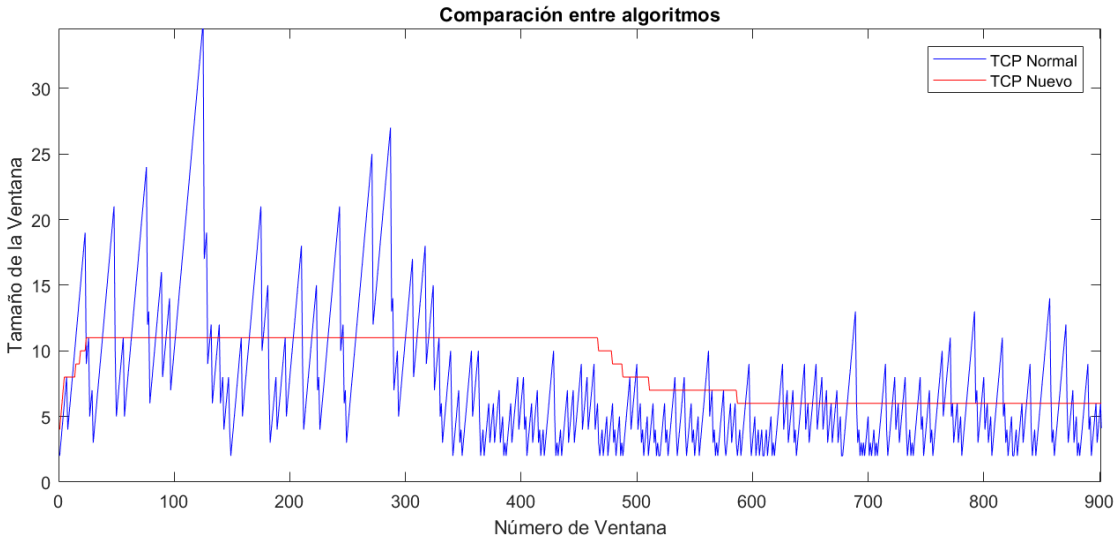


Figura 3.2: Nuevo Algoritmo de Control de Congestión.

Se observa como el nuevo protocolo tiende a converger al promedio de TCP tradicional y lo estable que se mantiene al cambiar el estado de la conexión.

En la Figura 3.3 se muestra en mayor detalle la adición del algoritmo de partida rápida, ya que sin este, toma alrededor de 200 paquetes converger al promedio y estabilizarle, por lo que para sesiones cortas el nuevo protocolo se vería en gran desventaja. Es posible ver esto en la Figura anterior (3.2) en donde al momento de cambiar el estado de la conexión el algoritmo se demora un tiempo importante en converger a un nuevo promedio.

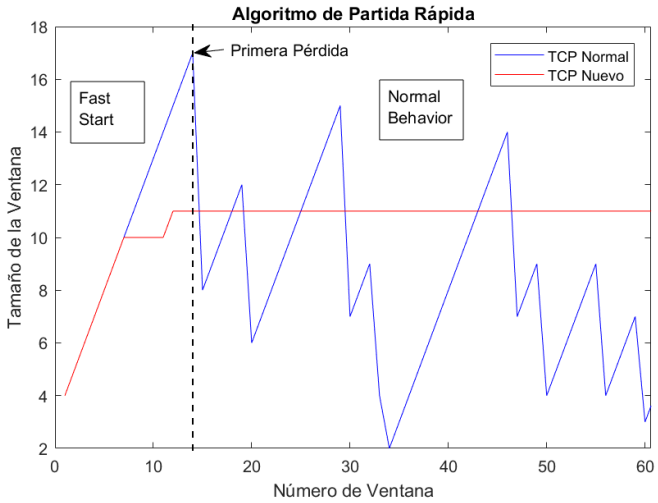


Figura 3.3: Algoritmo de Partida Rápida.

3.2. Fairness Test

De forma de cumplir con el requerimiento de co-existencia con otros algoritmos para no sobre congestionar la red, se creará un *Fairness Test* el cual consiste en simular el nuevo protocolo en una situación real donde este deba competir contra otro usuario que utilice **OldTCP**. Cabe notar, que tanto el trabajo hecho en la sección 3.2 y 3.3, debe realizarse cada vez que se necesite ajustar el protocolo a un distinto TCP, puesto que es muy poco probable que se utilice el mismo de forma definitiva en el tiempo.

Por otra parte, el *Fairness Test* consistirá, primero, en asignar un valor arbitrario máximo de paquetes soportados por el servidor (50 para este caso). Ambos algoritmos aumentarán su ventana hasta que se pierda un paquete. La pérdida de un paquete estará dada por el servidor en el momento en que la congestión de ambos usuarios supere el máximo soportado. Dependiendo del valor actual de ventana de ambos protocolos se genera una probabilidad, por ejemplo, si el protocolo 1 tiene ventana de tamaño 30 y el protocolo 2 una de tamaño 20 al alcanzar el tope, entonces el protocolo 1 tendrá un 60 % de probabilidades de ser el que pierda el paquete y el protocolo 2 un 40 %. Al final de la simulación se obtiene el promedio de probabilidades de ambos protocolos. Mientras más cercanos estos resultados son a 50 %, más justos entre si son los algoritmos.

En la Figura 3.4 se muestran los resultados del *Fairness Test* para un valor arbitrario de $\alpha = 0{,}985$. Cabe notar que Var y Beta solo afectan la estabilidad del algoritmo. Es claro por qué el primero lo hace, sin embargo el segundo requiere de una explicación. En términos simples, Beta multiplica a un valor que se vuelve más pequeño a medida que la ventana crece, por lo que el valor del término completo es poco dependiente de este, es decir, se puede considerar a Beta como algo similar al efecto de un *offset*. Por tanto solo se necesita buscar un valor lo más preciso posible para Alfa.

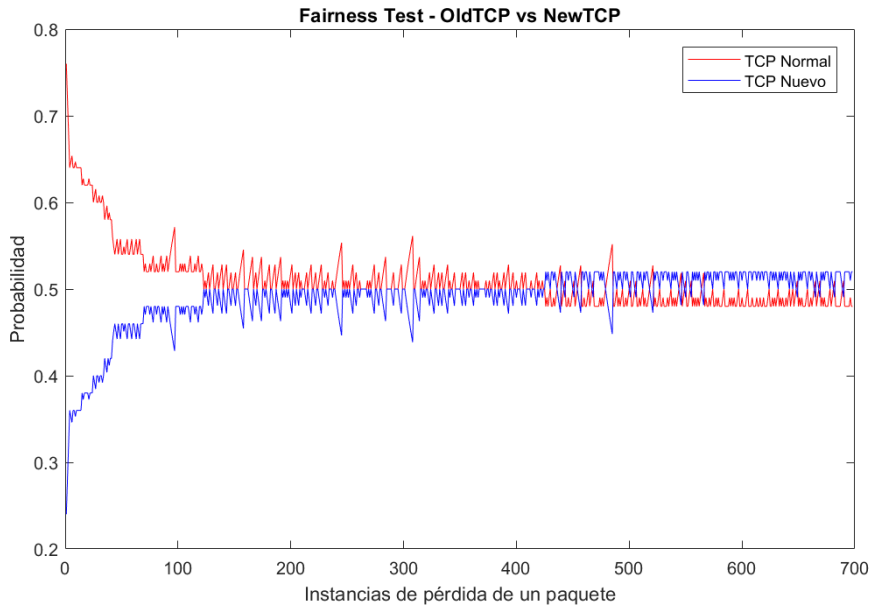


Figura 3.4: Prueba Número 1 - Fairness Test.

Para el test anterior el promedio de las probabilidades de pérdida del nuevo protocolo es de 53,58 % y en TCP tradicional es de 46,42 %. Para este valor arbitrario de Alfa se tiene que el nuevo protocolo es bastante justo, sin embargo aún es posible obtener mejores resultados.

3.3. Proceso de Ajuste del Protocolo

Utilizando el mismo test de la sección anterior, se creó un algoritmo que simula esta situación múltiples veces, disminuyendo en 0.000001 el valor de Alfa (también llamado "factor de reducción") cada vez. De esta forma se busca obtener un valor que nos acerque a un promedio de probabilidad lo más cercano posible a 50 % para que el algoritmo sea más justo. Los resultados de esta simulación se muestran el la siguiente figura:

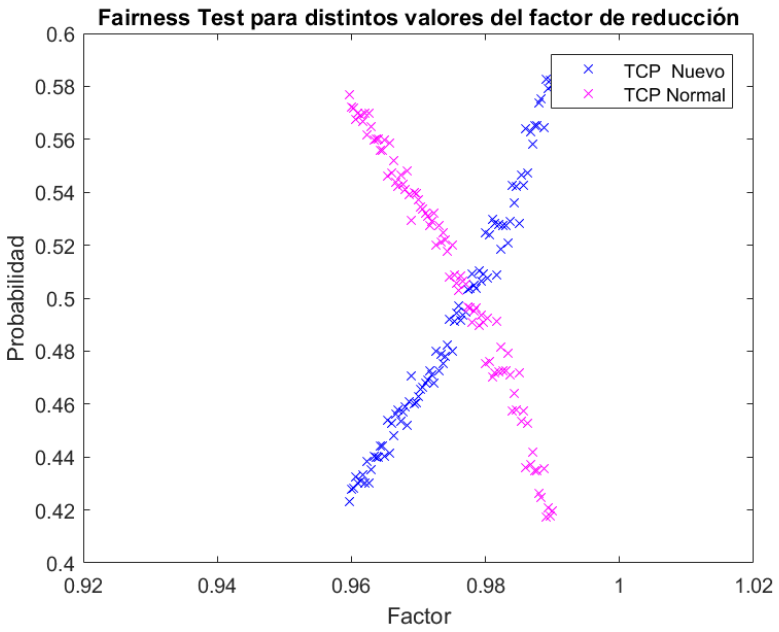


Figura 3.5: Fairness Test Para Distintos Valores de Alfa.

Cada dato de la Figura 3.5 representa el promedio de probabilidades que resultó para esa iteración del *Fairness Test*. Para buscar el mejor Alfa se debería ajustar una curva y obtener el valor de Alfa que resulte en 0.5, sin embargo es posible notar en la figura anterior, que no existe en la realidad un valor de 'Alfa' con el que se pueda obtener un resultado exacto de 0.5. Por tanto, existen múltiples valores de este que nos pueden acercar tanto por el lado derecho como el izquierdo a 50 %.

Con el fin de obtener un resultado aproximado y definitivo para 'Alfa' se realizará un ajuste cuadrático de ambas curvas, en donde la intersección de estas nos permitirá obtener un valor para 'Alfa'.

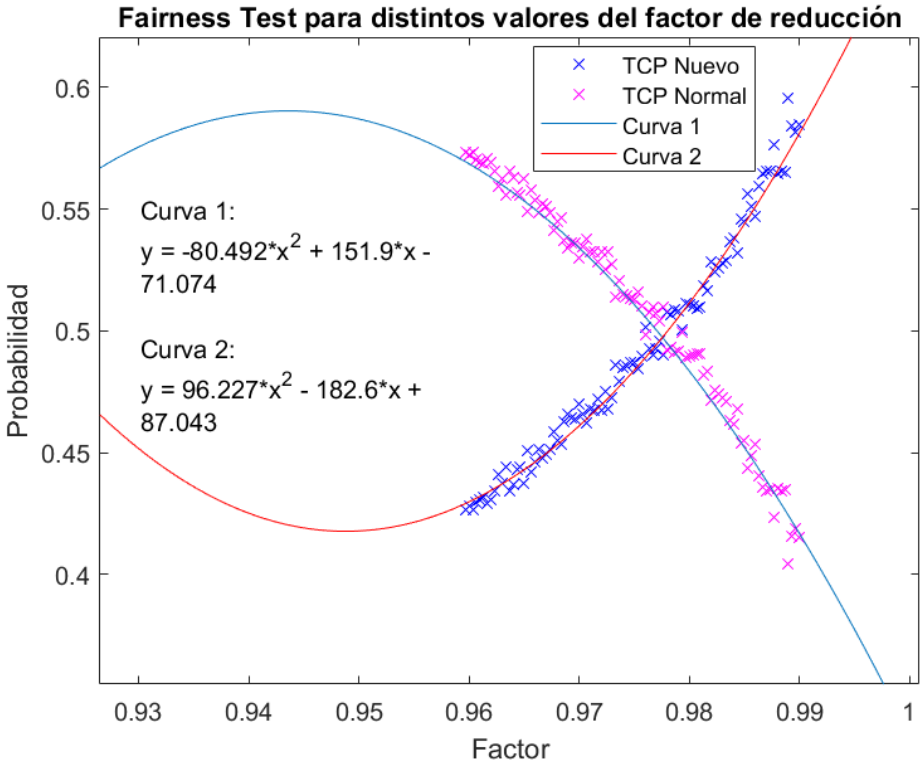


Figura 3.6: Curvas de Ajuste Para Alfa.

Resolvemos el siguiente sistema de ecuaciones proveniente de las curvas de la Figura 3.6:

$$Y = -80{,}492x^2 + 151{,}9x - 71{,}074 \tag{3.1}$$

$$Y = 96{,}227x^2 - 182{,}6x + 87{,}043 \tag{3.2}$$

Resulta entonces que un posible valor de 'Alfa' es:

$$\alpha = 0{,}97743$$

3.3.1. Pruebas del Protocolo Definitivo

A continuación se realizará un *Fairness Test* para 3 escenarios distintos: el nuevo protocolo compitiendo contra si mismo, contra TCP tradicional y por último TCP tradicional compitiendo contra sí mismo. De esta forma se puede observar más de cerca lo que ocurriría en la red en la realidad. Adicionalmente se agregará a cada *test* un histograma que muestre la diferencia de los valores de probabilidad de perder un paquete, para así observar la distribución de los datos.

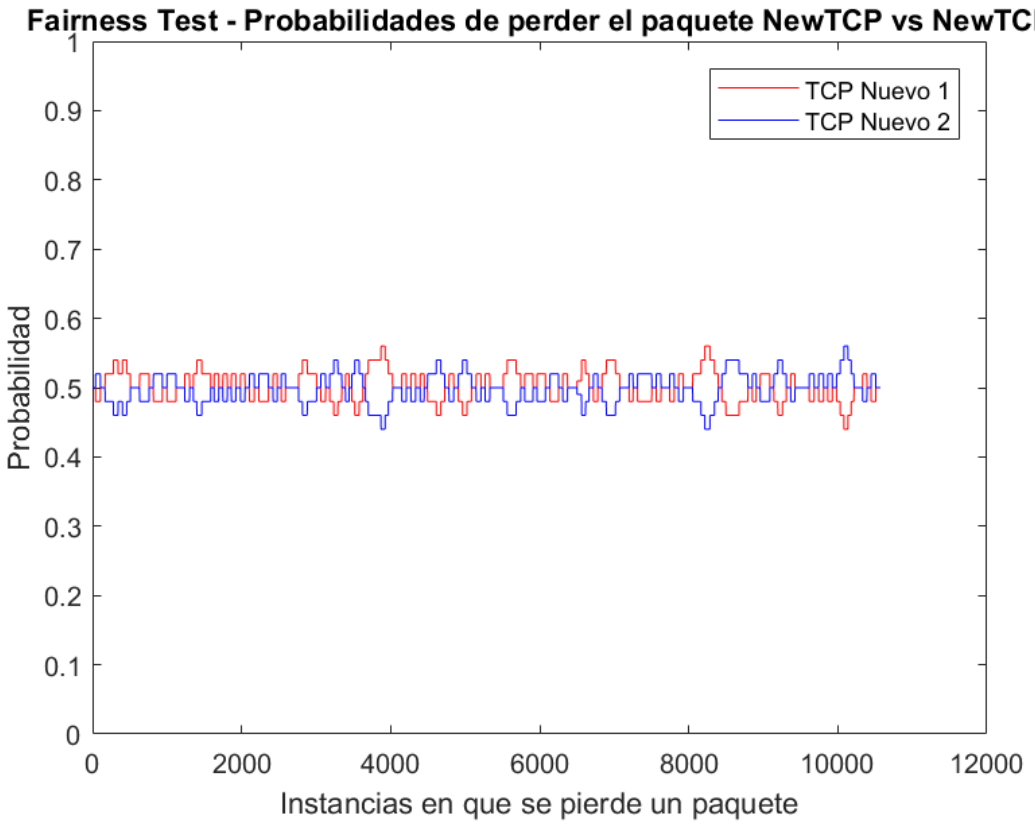


Figura 3.7: Fairness Test - NewTCP vs NewTCP.

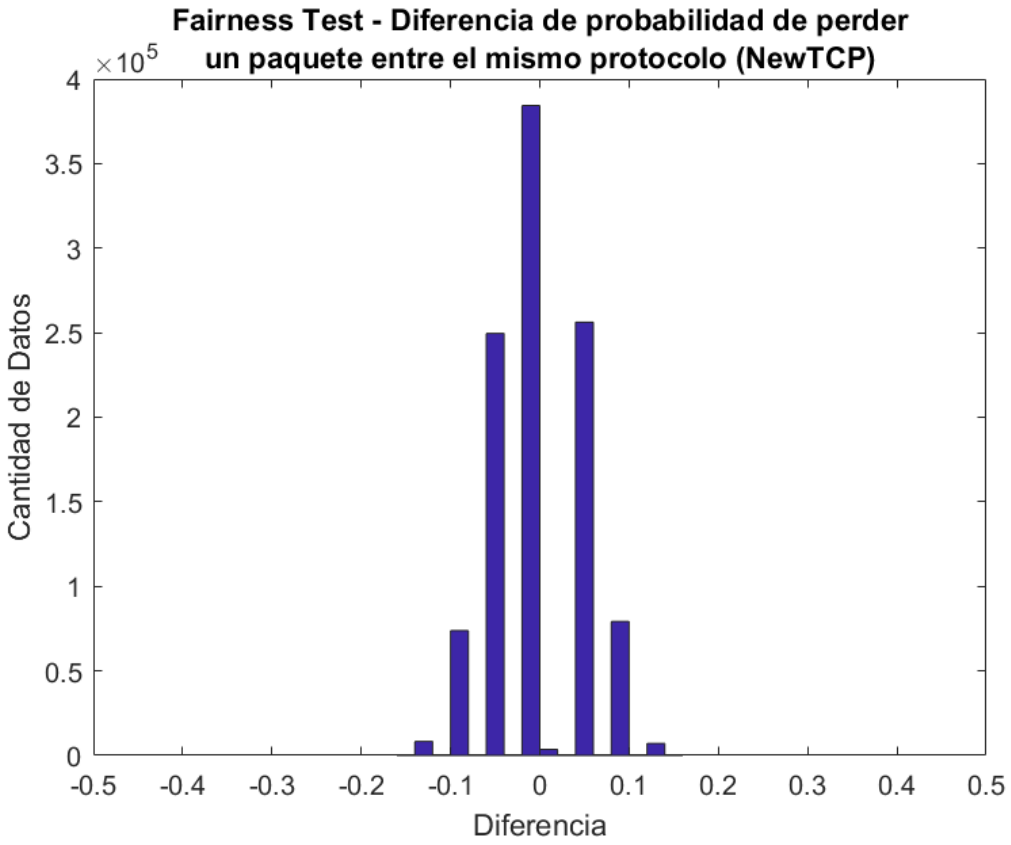


Figura 3.8: Histograma - NewTCP vs NewTCP.

De la Figura 3.7 y 3.8 se observa que el nuevo protocolo tiene un comportamiento bastante estable en el corto y largo plazo. Además del histograma, con desviación estándar de 0.0437, se infiere que en la mayor cantidad del tiempo, ambos usuarios estarían perdiendo paquetes con la misma frecuencia, es decir, ninguno se ve favorecido de manera importante en ningún momento.

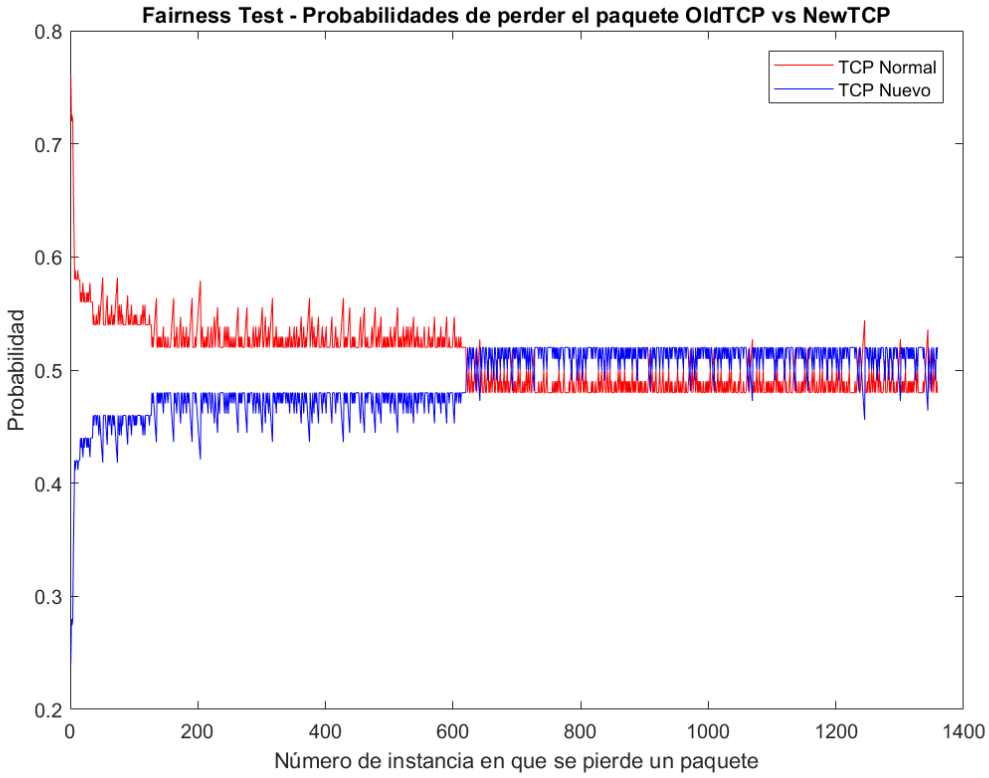


Figura 3.9: Fairness Test - NewTCP vs OldTCP.

Muy parecido al caso anterior, se muestra un comportamiento justo entre ambos protocolos. De hecho, en este casos se obtiene una desviación estándar levemente menor de 0.0362, lo que muestra de forma aún más concreta el efecto del ajuste personalizado que se le hizo al nuevo protocolo para co-existir de mejor manera con el antiguo.

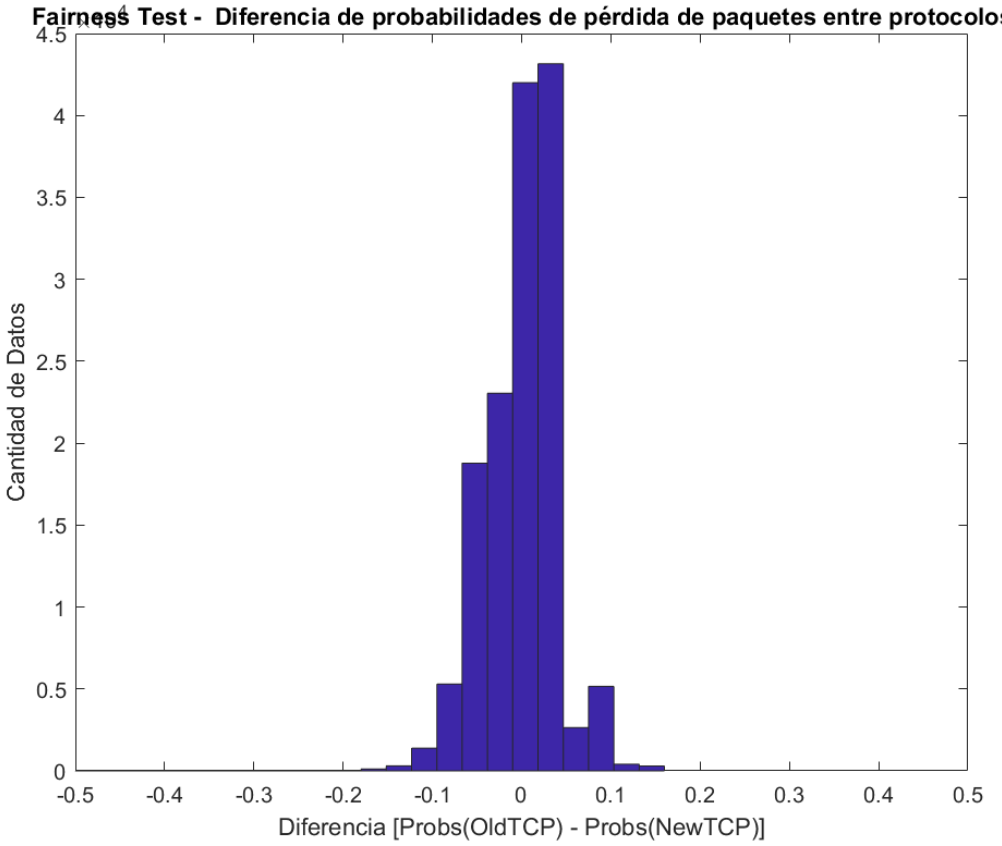


Figura 3.10: Histograma - NewTCP vs OldTCP.

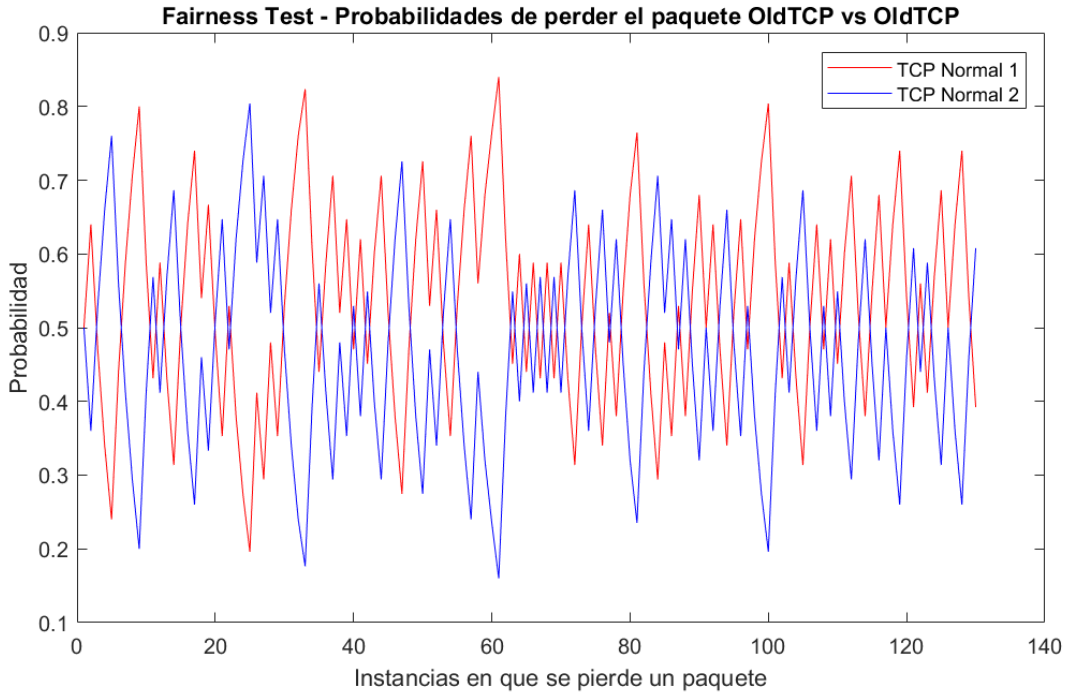


Figura 3.11: Fairness Test - OldTCP vs OldTCP.

Por último, la Figura 3.11 muestra lo que sería el comportamiento normal de una red actual. El resultado de la desviación estándar de los datos es de 0.2978, lo cual es aproximadamente 10 veces mayor a los casos anteriores. Aunque se observe del histograma de la Figura 3.12 que su comportamiento natural es similar a una *Campana de Gauss* centrada en el origen, se tendrá, que con muy alta frecuencia, el protocolo es excesivamente injusto en el corto plazo y medianamente justo en el largo plazo en comparación al nuevo algoritmo.

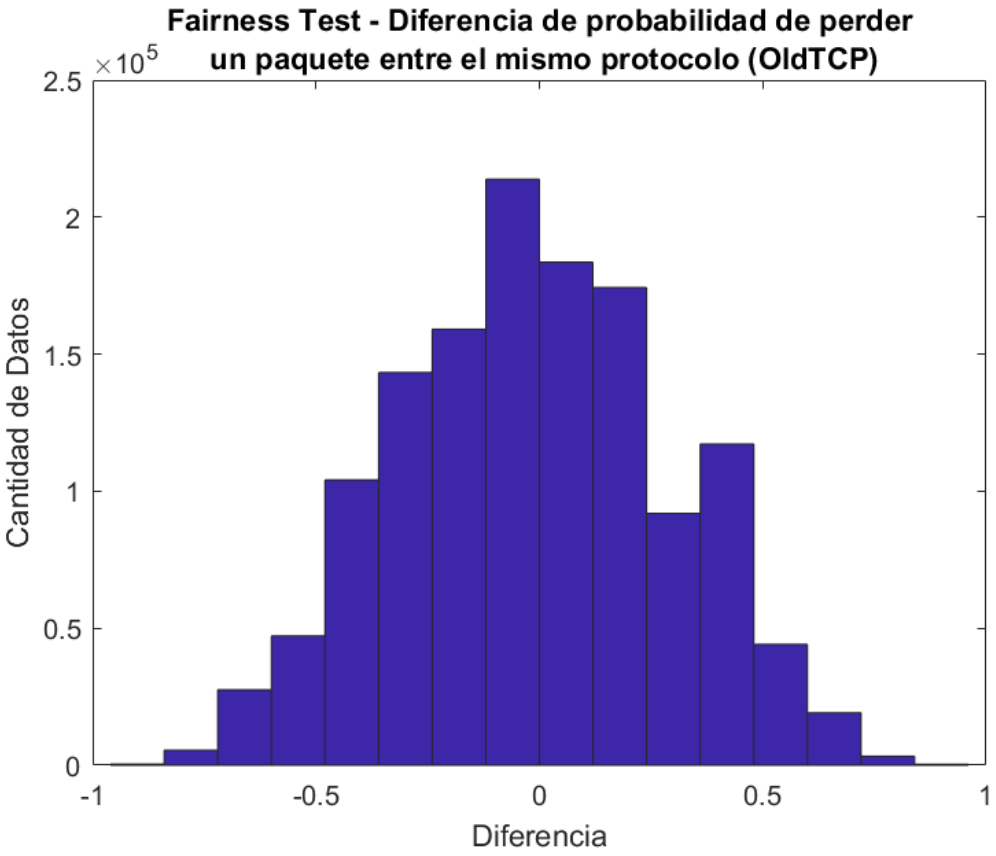


Figura 3.12: Histograma - OldTCP vs OldTCP.

3.4. TCP y Algoritmo de Deadline

El *algoritmo de deadline* permite mostrar claramente el tiempo que demoran los *frames* de cada tipo en llegar con y sin el uso de este (para un RTT = 100 ms). En la Figura 3.13 se observa la importante diferencia en los tiempos de llegada de los *frames* (uno es 4 veces mayor que el otro) cuando el algoritmo está en funcionamiento. Además cabe destacar que la curva rosada (protocolo en funcionamiento) logra evitar pérdidas de calidad importantes, en condiciones normales ($Loss\ Rate = 0{,}01$) , dado que en ningún momento se acerca demasiado al *deadline*, sin embargo se ve un comportamiento algo oscilatorio lo que en la práctica indica una variación de la calidad de imagen constante a lo largo del tiempo.

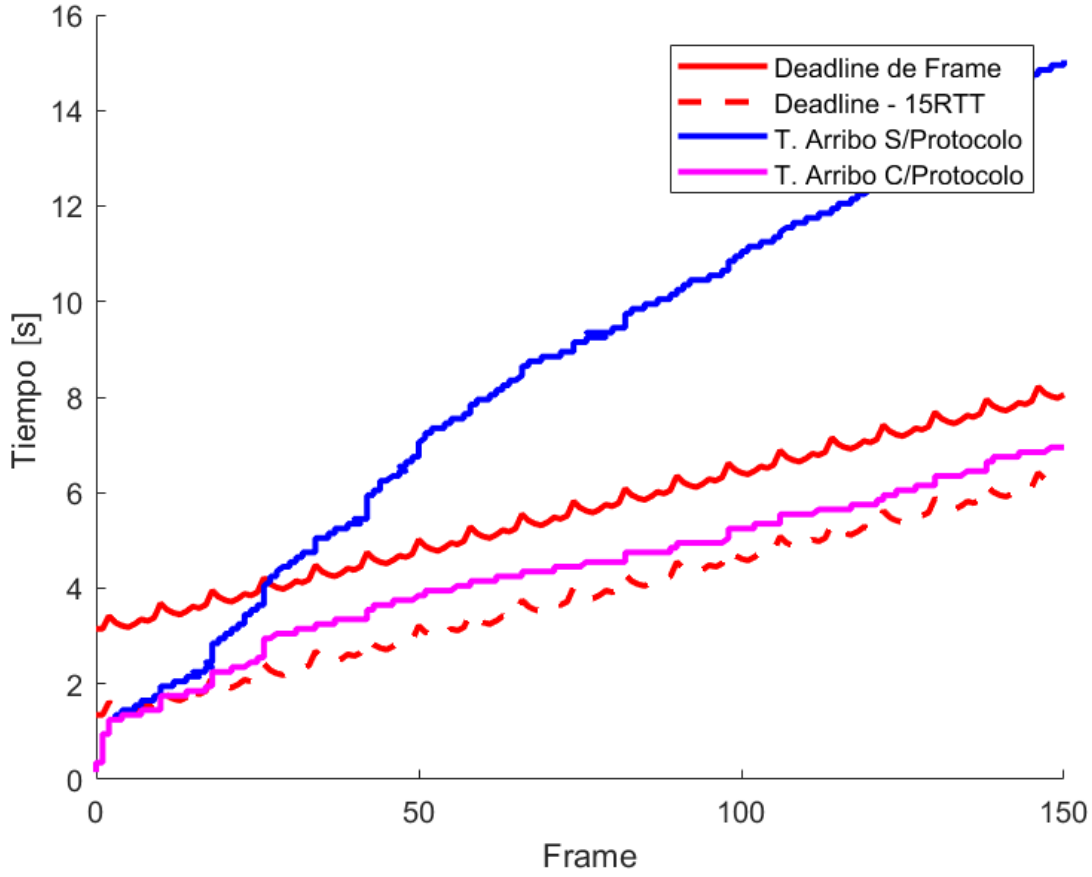


Figura 3.13: Tiempos de Llegada - OldTCP - Condiciones Normales de Red.

En este punto, se cambia el protocolo de congestión usado dentro del *algoritmo de deadline* por el creado en este trabajo. Dado lo estable que es el comportamiento del nuevo TCP, se espera a priori obtener así mismo tiempos de llegada más lineales.

En la siguiente figura se muestran los nuevos tiempos de llegada de cada *frame* para el nuevo TCP:

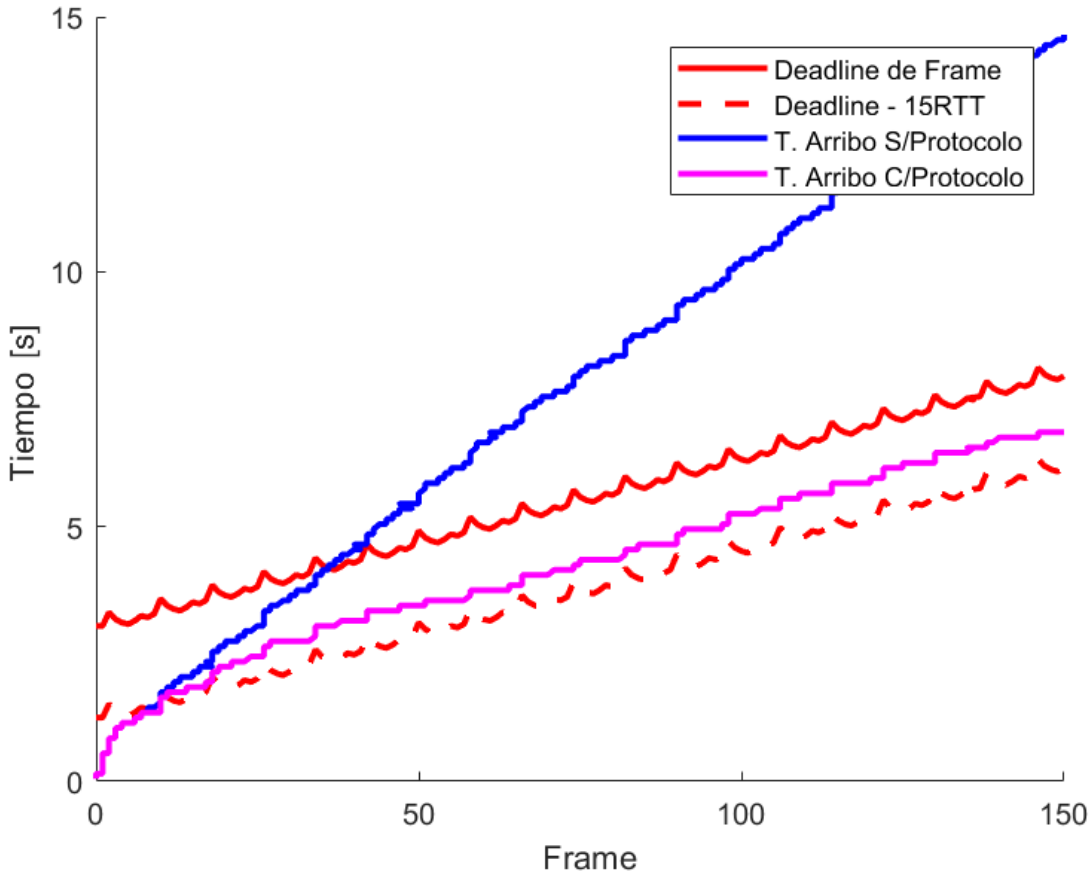


Figura 3.14: Tiempos de Llegada - NewTCP - Condiciones Normales de Red.

Se observa una clara diferencia entre las figuras 3.13 y 3.14. El *algoritmo de deadline* se comporta de forma menos oscilatoria utilizando el nuevo protocolo TCP, de hecho mantiene aproximadamente la misma distancia del *deadline* de manera constante, sin embargo es posible que para el usuario esta diferencia sea aún imperceptible.

Finalmente se simulará el algoritmo con ambos protocolos TCP en condiciones más criticas, es decir, con $LossRate = 0{,}05$.

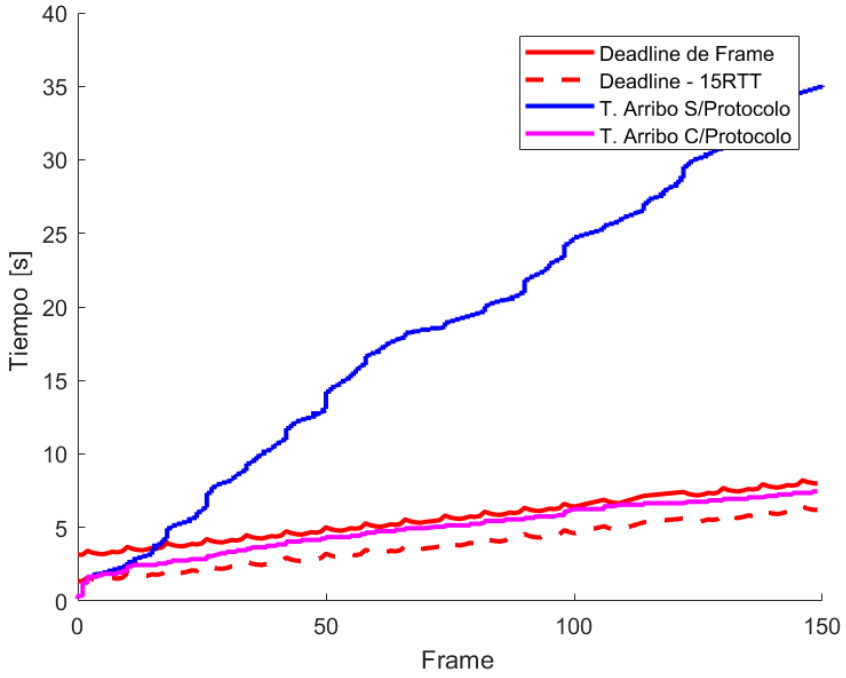


Figura 3.15: Tiempos de Llegada - OldTCP - Malas Condiciones de Red.

Nuevamente se muestra un comportamiento más estable al usar el nuevo protocolo TCP. Adicionalmente, se muestra una notoria mejora en la calidad promedio que tendría cada frame dado que en todo momento se está bastante alejado del *deadline*, esto puede deberse a que en los instantes donde se solicita retransmisión de paquetes, el protocolo nuevo siempre sufrirá el efecto del RTT en la misma magnitud, sin embargo TCP tradicional es muy dependiente de su valor instantáneo de ventana por lo que se verá muy afectado cuando este valor sea muy pequeño.

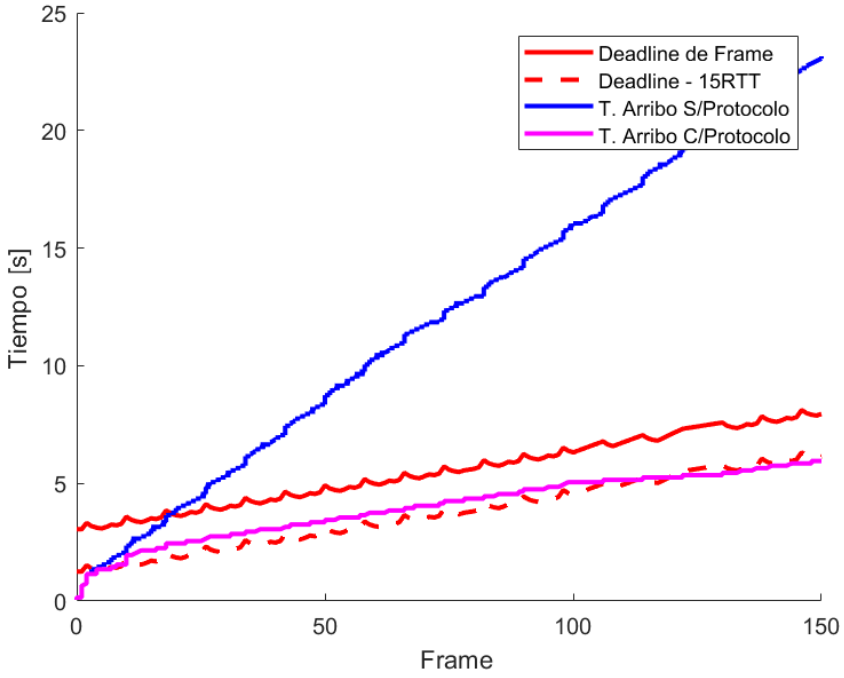


Figura 3.16: Tiempos de Llegada - NewTCP - Malas Condiciones de Red.

3.4.1. Comportamiento en el Largo Plazo

Es de suma importancia destacar que anteriormente se mostró el comportamiento del *algoritmo de deadline* para situaciones particulares donde se pierden paquetes en momentos arbitrarios acorde a un cierto *Loss Rate*. Es por esto, que dada la gran volatilidad del protocolo de TCP tradicional, se puede inferir que es posible que en ciertas situaciones se muestren tiempos mucho menores en comparación al nuevo algoritmo y otra veces mucho mayores (como en el caso anterior). Por lo que los resultados de las Figuras 3.13 - 3.16 solo estarían enseñando un comportamiento esperable en el corto plazo. Es por esto que, como último *test*, se realizará una nueva simulación que permita mostrar las rutas que toman las curvas de tiempo de arribo con el protocolo de *deadline*, de tal forma de ubicar una tendencia (densidades) en el largo plazo.

La Figura 3.17 muestra los caminos que toman los tiempo de llegada de los frames para un *Loss Rate* de 0.01. De este gráfico se puede inferir que en el largo plazo existe una notable oscilación del tiempo de arribo de los *frames* dado los pequeños huecos que se forman entre los caminos. Además vale notar que entre *frames* número 20 y 60 no existe una tendencia clara, por lo que es bastante probable que el usuario experimente artefactos o una calidad de imagen poco definida en esos *frames*.

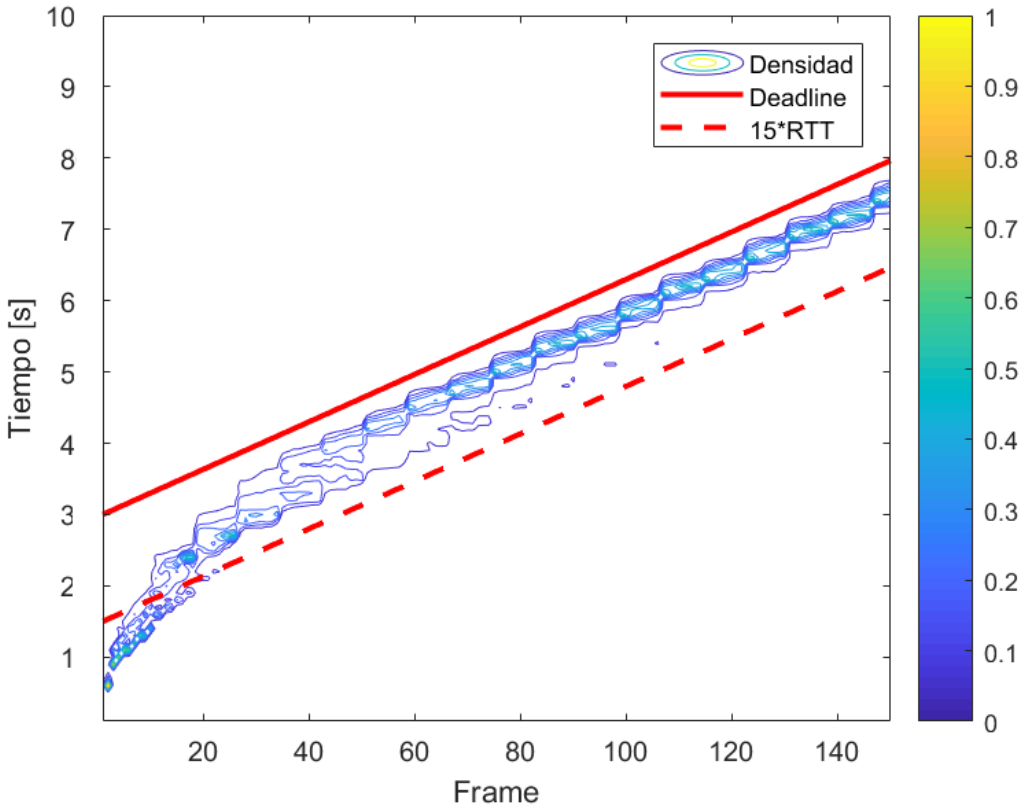


Figura 3.17: Densidad de Tiempos de Llegada - TCP Normal.

En la Figura 3.18, se muestra de forma clara el contraste que existe respecto de la figura anterior. Se observa una tendencia significativa al recorrer las zonas marcadas con color amarillo. Además estas zonas son más angostas y se tiene consistencia a lo largo de toda la curva, es decir, a pesar de haber una leve oscilación entre los *frames* 40 y 60, la estabilidad y por tanto calidad de imagen que será visualizada por el usuario será consistente en todo momento de la transmisión.

Finalmente, se procederá a codificar y decodificar un archivo de video simulando su transmisión con ambos algoritmos para así definir de manera empírica y visual que tan relevante o importante es la diferencia que experimentaría el usuario.

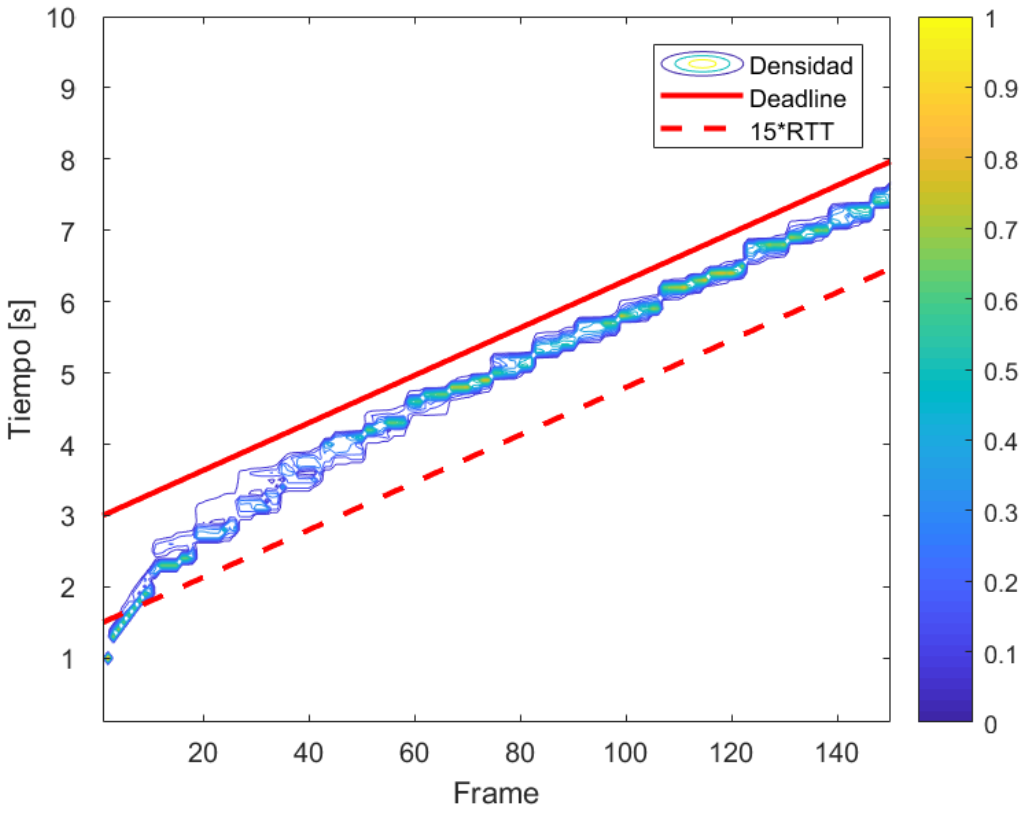


Figura 3.18: Densidad de Tiempos de Llegada - TCP Nuevo.

Capítulo 4

Resultados

Para mostrar el efecto del nuevo protocolo TCP sobre la transmisión de un video que utiliza el *algoritmo de deadline*, se realizará, en primera instancia, un contraste junto a TCP tradicional para distintos valores de *Round Trip Time* (RTT), aunque en este caso se utilizará un *Loss Rate* fijo de 0.01. Esto dado que el *Loss Rate* varía para cada usuario según distintas condiciones (servidor, horario, distancia, etc.) además de afectar la suma de los RTTs (tiempo de llegada neto) de todos los paquetes. Cada escenario distinto dispondrá de imágenes del video original y reconstruido, el PSNR de cada *frame* y por último un gráfico del *algoritmo de deadline* que muestre ambos protocolos TCP en funcionamiento.

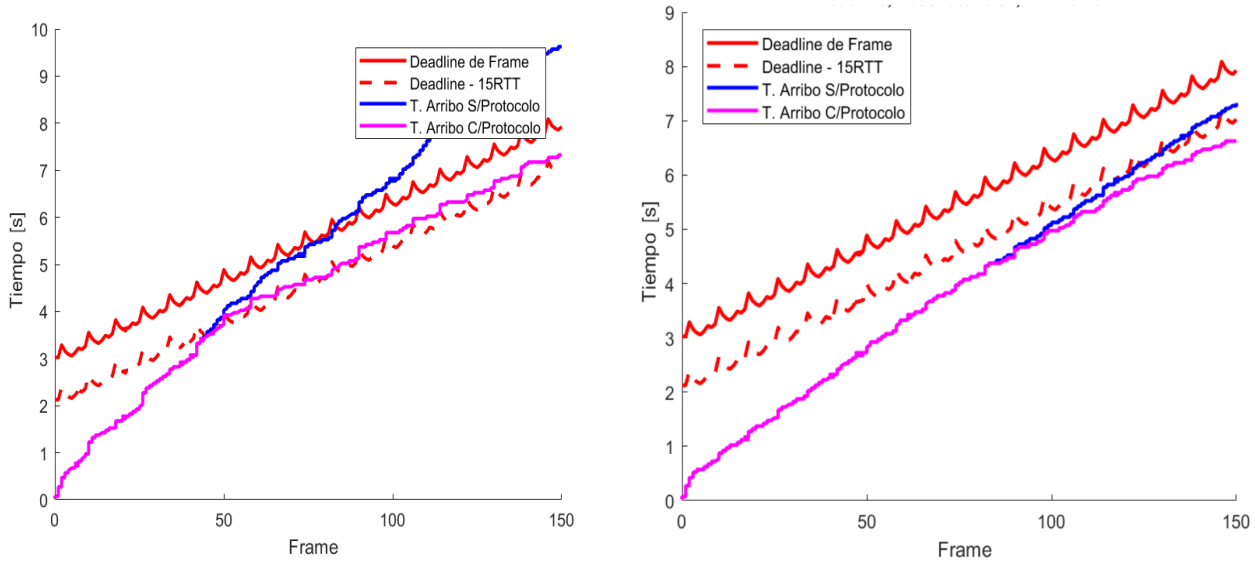
Para el cálculo del PSNR por *frame* e imágenes del video se utilizará la versión 11.1 del software **MSU Video Quality Measurement Tool** [6]. La versión gratuita de este producto permite comparar, afortunadamente, 3 versiones de un mismo video (original, codificación con TCP tradicional y TCP nuevo en este caso), entregándonos todos los detalles de información importantes requeridos para este trabajo.

El primer escenario consiste en una transmisión a nivel **nacional**, es decir, un RTT máximo de 50ms. El segundo es en una transmisión más lejana, desde Santiago de Chile a la costa este de **Estados Unidos** (RTT = 180 ms). Por último, dado que se considera que una perdida igual o mayor a un 5 % de paquetes, disminuye significativamente la calidad de transmisión de audio y video [15], el tercer escenario corresponderá a una red más congestionada (Loss Rate = 0.05) dentro del continente (RTT = 100 ms).

4.1. Transmisión Nacional

Dentro de Chile, dada su longitud del país, el RTT puede variar entre 5-50ms. Es por esto que en este escenario tomaremos 50 *ms* como un peor caso y de esta manera obtener información más relevante al comparar los protocolos.

La figura 4.1 muestra una leve diferencia en los tiempos de llegada y forma de la curva que representa el *algoritmo de deadline* en acción para ambos TCP. Esto se debe a que el RTT es tan pequeño, que la velocidad de transmisión del usuario crece hasta el punto de solo ver un pequeña baja en la calidad del video respecto al original debido a que apenas se llega a tocar la curva punteada que indica el momento donde se comienzan a omitir *frames* de baja información (B y P). Sin embargo, se puede apreciar que a partir del *frame* 50 la curva sobrepasa levemente el *threshold* en el cual se comienzan a omitir *frames* del tipo B y P, por lo que a priori debería existir una diferencia en la calidad de imagen final.



(a) Deadline, RTT = 50 ms - TCP Normal.

(b) Deadline, RTT = 50 ms - TCP Nuevo.

Figura 4.1: Tiempos de Llegada (NewTCP & OldTCP) - Transmisión Nacional.

En la siguiente figura se observa el comportamiento del PSNR a lo largo del video para ambos protocolos, mientras más grande es este valor mejor es la calidad que se observa.

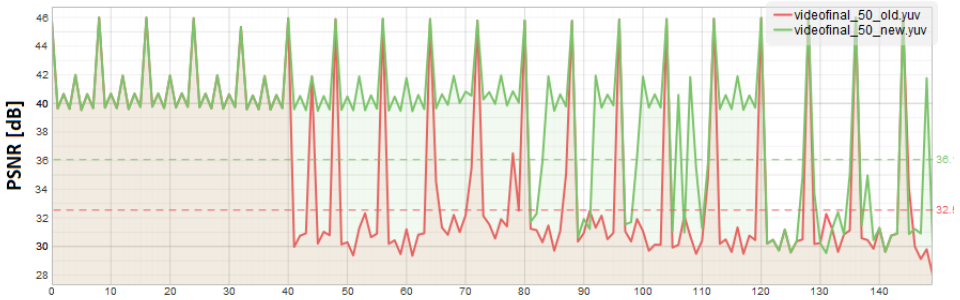


Figura 4.2: PSNR - NewTCP vs OldTCP - RTT = 50 ms.

La Figura 4.2 muestra que el video transmitido utilizando el nuevo protocolo TCP se ve beneficiado a lo largo de toda la transmisión con un PSNR promedio mas grande, con un valor de 36 para el nuevo protocolo y 32 para el tradicional. Por otro lado, desde el *frame* número 40 en adelante, se observan bajas en el PSNR, por tanto el usuario recibirá un video con una calidad algo más baja que el original. Sin embargo, al tomar en cuenta los datos de la figura 4.1, se conoce a priori que los *frames* que se omitieron son los de menor importancia en ambos TCP, por lo cual es posible que la calidad final del video tenga una diferencia imperceptible para el usuario. Para comprobar esta teoría se mostrará una comparación visual de algunos frames relevantes comparados con el video original.

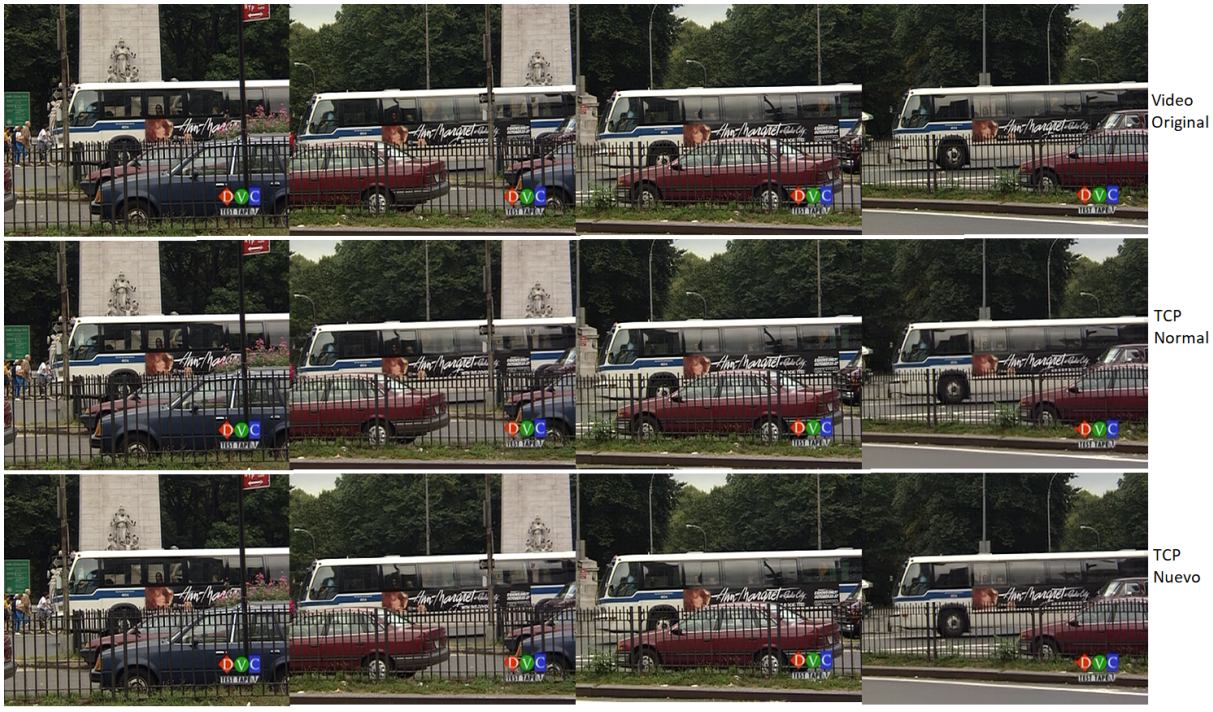


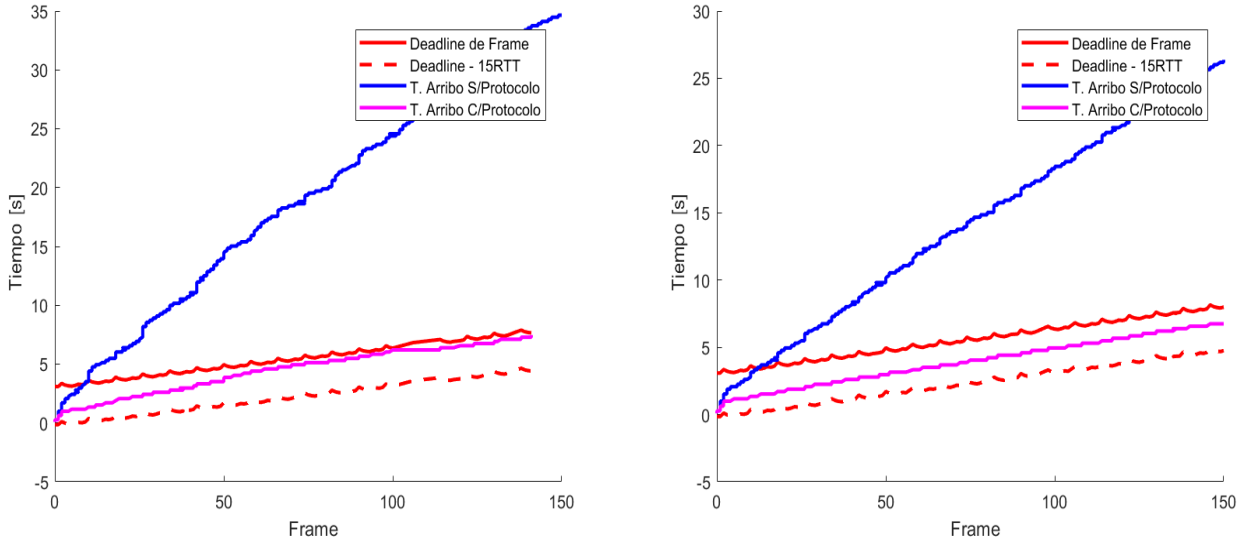
Figura 4.3: Video Original vs Decodificaciones - RTT = 50 ms.

Los *frames* mostrados en la figura 4.3 son los número: 40, 81, 125 y 147 de izquierda a derecha. El primer y último *frame* tienen un PSNR bastante alto (en referencia a la figura 4.2), por lo que su calidad es prácticamente igual a la del video original. Por otra parte los *frames* número 81 y 125 muestran una baja en el PSNR, sin embargo visualmente, la diferencia en calidades es irrelevante para el ojo humano para ambos protocolos TCP.

Se concluye entonces, que los resultados del uso de un nuevo protocolo TCP para la transmisión de video a nivel nacional, no favorece significativamente la experiencia final del usuario.

4.2. Transmisión desde Estados Unidos

Asumimos un RTT aproximado de 180 ms desde la costa este de los Estados Unidos a Santiago de Chile (valor algo mayor para la costa oeste). Al igual que en el escenario anterior, se mostrará a continuación los tiempos de llegada de los paquetes para ambos protocolos tanto con el uso del *algoritmo de deadline* como sin este.



(a) Deadline, RTT = 180 ms - TCP Normal.

(b) Deadline, RTT = 180 ms - TCP Nuevo.

Figura 4.4: Tiempos de Llegada (NewTCP & OldTCP) - Transmisión desde Estados Unidos.

La Figura 4.4 muestra un importante aumento en los tiempos de llegada respecto del escenario nacional. Además se tiene una notable concavidad en la curva de tiempo de llegada con *algoritmo de deadline* utilizando TCP tradicional, pero la diferencia es tan pequeña en cuanto a los tiempos que puede no ser relevante en la calidad del video final.

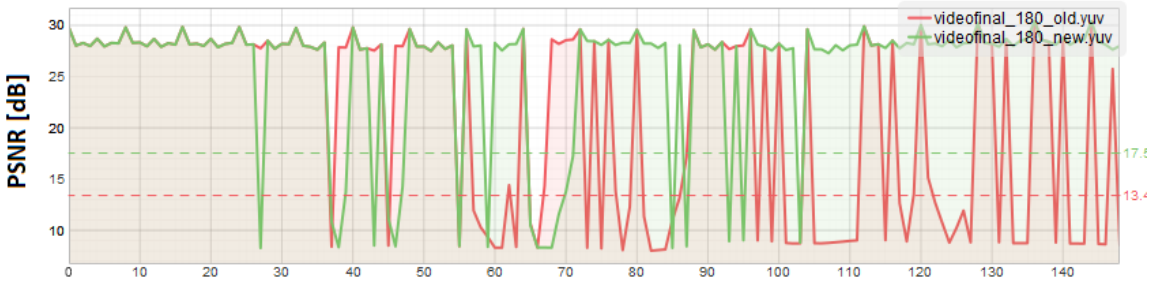


Figura 4.5: PSNR - NewTCP vs OldTCP - RTT = 180 ms.

La Figura 4.5 indica una diferencia más relevante en la calidad de imagen de ambos videos decodificados, sin embargo el PSNR de ambos se vio igualmente afectado por el aumento en el RTT. Cabe notar que las lineas punteadas indican el PSNR promedio de ambos videos, siendo algo mayor el que usó el nuevo protocolo.

A continuación, en la figura 4.6 se muestran algunos *frames* (30, 45, 58, 88 y 134) con pérdidas significativas de calidad para ambos protocolos.

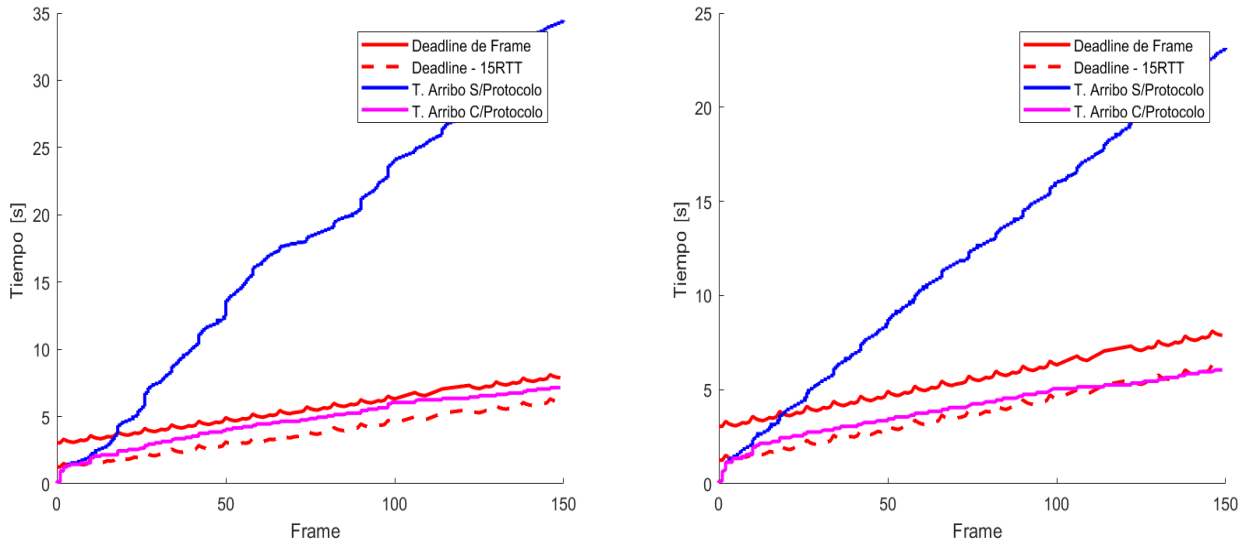
Figura 4.6: Video Original vs Decodificaciones - RTT = 180 ms.

Lo más destacable que se puede observar de este escenario, es la mayor cantidad de pérdidas de *frames* tipo *I* usando el protocolo tradicional. Dado la forma en que estas pérdidas se traducen en los **frames**, el usuario si obtendrá una mejor experiencia en cuanto a la fluidez y calidad de imagen utilizando el nuevo protocolo TCP.

4.3. Transmisión por Red Congestionada

En esta sección, se simula la transmisión del video para un RTT con valor estándar (100 ms), pero con pérdidas más significativas (Loss Rate = 0.05). De esta forma se espera obtener resultados mas concluyentes dado que en este escenario se afecta directamente a la congestión de la red.

La siguiente figura (4.7), muestra los tiempos de llegada para ambos protocolos TCP, en donde se observa una importante diferencia en el comportamiento de ambos, específicamente en la distancia que mantienen de la curva que representa el *deadline* de cada *frame*.



(a) Deadline, Loss Rate = 0.05 - TCP Normal.

(b) Deadline, Loss Rate = 0.05 - TCP Nuevo.

Figura 4.7: Tiempos de Llegada (NewTCP & OldTCP) - Red Congestionada.

Dado lo consistente en la lejanía de la transmisión que utiliza el nuevo protocolo TCP al *deadline*, se deduce que existe una menor omisión de frames en promedio de todos los tipos. Es posible que esto ocurra ya que que la ventana de congestión se mantiene de forma más constante en el tiempo, por lo que cada vez que se necesiten retransmitir ciertos paquetes, siempre se contará con los mismos tiempos, al contrario de TCP tradicional, donde puede ocurrir que la ventana de congestión haya sido tan pequeña en ese instante que se necesiten múltiples retransmisiones por lo que se utiliza más tiempo para un mismo objetivo.

La Figura 4.8 muestra una situación algo similar al escenario anterior (RTT=180ms), es decir, la cantidad de *frames* tipo I perdidos es algo mayor utilizando el protocolo tradicional de TCP, sin embargo en esta ocasión se tiene un promedio de PSNR bastante similar para ambos protocolos.

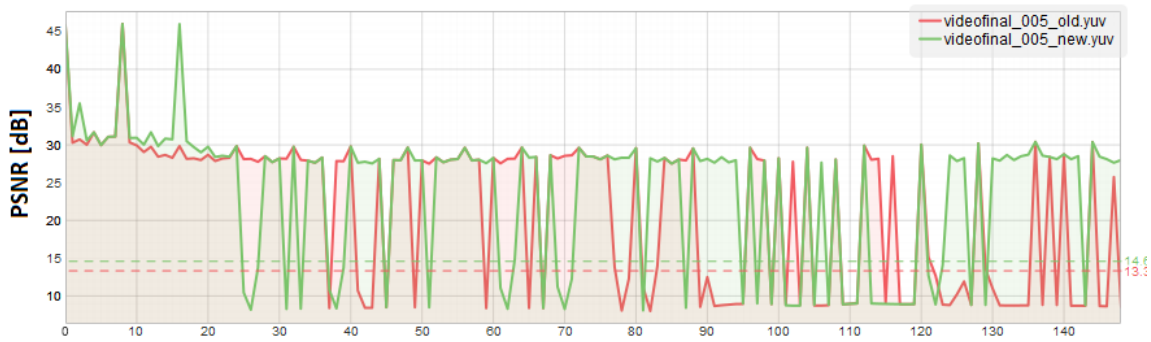


Figura 4.8: PSNR - NewTCP vs OldTCP - LossRate=0.05.

La Figura 4.9 muestra los *frames* 16, 25, 41, 90 y 129 de forma de representar la calidad general del video para cada caso.

Figura 4.9: Video Original vs Decodificaciones - Loss Rate = 0.05

Es importante notar que en los primeros 25 *frames* (tiempo del *buffer*) la calidad mostrada por el nuevo protocolo es bastante similar a la del video original. Dado lo estable de los caminos que se toman utilizando el nuevo TCP, como se mostró anteriormente en la Figura 3.18, se tendrá que este comportamiento se repetirá de forma frecuente. Por tanto el usuario observará una buena calidad de imagen al comienzo de cada transmisión.

Finalmente, se tiene que el nuevo protocolo TCP es más beneficioso en los casos donde el RTT es mayor, y solo de forma leve cuando existen pérdidas por congestión.

Capítulo 5

Conclusiones

A partir de lo mostrado en la sección anterior de resultados, se presentará una conclusión junto a las bases para un posible trabajo futuro.

5.1. Conclusiones

Dado cada escenario mostrado en el cual se utilizó el nuevo algoritmo de TCP, es posible afirmar que se cumplen con los objetivos propuestos, de mejorar en el corto y largo plazo no solo la calidad si no también la estabilidad en la imagen que observa el usuario. Esto se reafirma gracias a los resultados mostrados específicamente en el escenario de transmisión de larga distancia y con red congestionada.

En cualquier situación donde el ancho de banda disponible sea muy alto, no se tiene inclinación por el uso de ningún protocolo puesto que el *algoritmo de deadline* no tendrá efecto absoluto en la transmisión.

El nuevo protocolo TCP cumple con todos los estándares propuestos de manera de coexistir sin problemas con otros protocolos de congestión. Además este tiene un funcionamiento mas bien simple, y no requiere de cálculos complejos. Al mismo tiempo, logra utilizar todo el ancho de banda que tiene disponible sin perjudicar a otros usuarios.

En cuanto al efecto del trabajo sobre el *algoritmo de deadline* en sí mismo, se tiene un clara tendencia y por tanto estabilidad en los tiempos de llegada de los paquetes, lo que permitiría que cada usuario experimente una visualización del video más consistente.

En los escenarios menos favores para el usuario, se tendrá un baja inevitable en la calidad general del video, sin embargo el nuevo protocolo permitirá amenizar este efecto.

Se concluye finalmente que el nuevo protocolo resulta de gran utilidad en escenarios más bien normales (ni muy favorable o desventajoso), donde las distintas calidades que puede tener el video son visualmente perceptibles para el usuario. Por tanto se recomienda el uso

de este algoritmo para plataformas populares que ofrezcan servicios de transmisión de video, dado que tendrán más control sobre la experiencia del usuario.

5.1.1. Trabajo Futuro

En cuanto al enfoque sobre TCP, es de gran interés comparar el impacto de este protocolo con otro más moderno como TCP Cubic y revisar nuevamente cual co-existe mejor en la red. Por otro lado, a pesar de que el nuevo protocolo está enfocado en la transmisión de video, en teoría, podría tener también un efecto positivo en otro tipo de escenarios como transmisión de audio, transferencia de archivos y redes celulares.

Para la sección de video, inmediatamente surge la idea de experimentar con calidades más altas como *FullHD* o *UltraHD*. Adicionalmente, se pueden agregar técnicas de codificación de video que se aprovechen de los beneficios que entrega el nuevo protocolo TCP. Por otro lado se puede intentar reemplazar los artefactos de color verdes observables en la sección de resultados, por píxeles de *frames* futuros o más antiguos. Finalmente, el hecho de realizar una comparación directa de este trabajo usando otra codificación como DASH, podría generar resultados que entreguen más razones para su uso comercial.

Anexos

-Archivos, pruebas y resultados disponibles en:

https://github.com/spate19/TCP-Protocol-for-Video-Transmision

-Algoritmo principal de ventana de congestión:

%% Inicialización

```
loss = 0.01;
lp = 1>(rand(1,Npkts/2)/loss);
cwnd(1) = 4;
decCwnd = cwnd(1)*3;
alpha = 0.97743;
averageCwnd = cwnd(1);
var = 0.5; incrementIni = 1;
incrementPerm = 0.08;
increment = incrementIni;
fastStart = 0;
```

%% Loop

```
for j=1:Npkts

    comp = sum(lp(ptr:min(ptr+cwnd(j)-1,Npkts)));

    if comp>0
        maxCwnd = decCwnd;
        averageCwnd = abs(maxCwnd+minCwnd)/2;
        decCwnd = averageCwnd*alpha;
        minCwnd = decCwnd;
        increment = increment/2;
        if fastStart <= 1
            cwnd(j+1) = cwnd(j) + 1;
            fastStart = fastStart + 1;
        end
```

```
        if increment <= incrementPerm
            increment = incrementPerm;
        end
    else
        decCwnd = decCwnd + averageCwnd/decCwnd*increment;
    end

    if fastStart <= 1 && comp == 0
        cwnd(j+1) = cwnd(j) + 1;
    end

    if (decCwnd-cwnd(j))/cwnd(j) >= var
        cwnd(j+1) = cwnd(j) + 1 ;
    elseif (decCwnd-cwnd(j))/cwnd(j) <= -var
        cwnd(j+1) = cwnd(j) - 1;
    else
        cwnd(j+1) = cwnd(j);
    end

    ptr = ptr+cwnd(j);

    if ptr>length(lp)
        break
    end

end


    %% Graficar

cwnd = cwnd(1:j);
plot(cwnd, '-r')
```

Bibliografía

[1] V. Paxson M. Allman and E. Blanton. *TCP Congestion Control*. RFC 5681, International Computer Science Instite (ICSI), September 2009.

[2] Gulshan Amin and Syed Shahdad. Tcp cubic-congestion control transport protocol. 3, 11 2014.

[3] Mohamed el Amine Brahmia, Abdelhafid Abouaissa, and Pascal Lorenz. Optimal bandwidth consumption for iptv services over wimax multihop relay networks. pages 40–45, 01 2012.

[4] V. Zapex Technol. Chatterjee S. Paulsamy. *Network convergence and the NAT/Firewall problems*. Grundlehren der mathematischen Wissenschaften. Technical report, School of Information Science Claremont Graduate University, 2003.

[5] Jian-Wen Chen, Chao-Yang Kao, and Youn-Long Lin. Introduction to h.264 advanced video coding. volume 2006, pages 6 pp.–, 02 2006.

[6] Dr. M. Smirnov. Dr. D. Vatolin. *MSU Quality Measurement Tool*. Available in: http://compression.ru/video/quality_measure/vqmt_download.html . Accessed on 2019-06-15.

[7] Andrés Sanhueza Gutiérrez. *Scalable Video Coding Sobre TCP*, Universidad de Chile, 2015.

[8] Z. Haitham and M. K. Mahmood Al-Azawi. Video compression based on motion compensation and contourlet transform. In *2018 Third Scientific Conference of Electrical Engineering (SCEE)*, pages 90–94, Dec 2018.

[9] Wu Hua and Jian Gong. Analysis of tcp bic congestion control implementation. pages 781–784, 08 2012.

[10] National Instruments. *Peak Signal-to-Noise Ratio as an Image Quality Metric*. https://www.ni.com/es-cl/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html . Accessed on 2019-07-01.

[11] Yueqiu JIANG, Junkun ZHANG, and Qixue GUAN. Improvement of tcp reno congestion control protocol. *Sensors and Transducers*, 163:308–315, 01 2014.

[12] Cheng Jin, D.X. Wei, and Steven Low. Fast tcp: Motivation, architecture, algorithms, performance. volume 14, pages 2490 – 2501 vol.4, 04 2004.

[13] D. Kozen, Y. Minsky, and B. Smith. Efficient algorithms for optimal video transmission. In *Proceedings DCC '98 Data Compression Conference (Cat. No.98TB100225)*, pages 229–238, March 1998.

[14] Luis Lucas, Nuno Rodrigues, Sergio De Faria, Eduardo da Silva, Murilo Carvalho, and Vitor Silva. Intra-prediction for color image coding using yuv correlation. pages 1329–1332, 09 2010.

[15] J.L. Mansfield, K.C. Antonakos. *Computer Networking from LANs to WANs: Hardware, Software, and Security.* Boston: Course Technology, Cengage Learning, 2010.

[16] Mohd Mohamad, Mudassar Ahmad, and Md Ngadi. Experimental evaluation of tcp congestion control mechanisms in short and long distance networks. *Journal of Theoretical and Applied Information Technology*, 71:153–166, 01 2015.

[17] Sandvine: Inteligent Broadband Networks. *Global internet phenomena report.* Available in: https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf. Accessed on 2019-07-01.

[18] Lisong Xu. Sangtae Injong. Cubic: A new tcp-friendly high-speed tcp variant. *Proceedings of the third PFLDNet Workshop*, 2, 2005.

[19] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17:1103 – 1120, 10 2007.

[20] Joel Sing and Ben Soh. Tcp new vegas: Improving the performance of tcp vegas over high latency links. pages 73–82, 08 2005.

[21] Networking Study. *The TCP/IP and OSI Networking Models.* Available in: https://4anm.wordpress.com/2013/07/05/the-tcpip-and-osi-networking-models . Accessed on 2019-06-03.

[22] Christian Timmerer. *MPEG DASH Tutorials.* Available in: http://multimediacommunication.blogspot.co.at/2013/09/mpeg-dash-tutorials.html. Accessed on 2019-06-12.