Universidad de la República
Facultad de Ingeniería

Calidad de experiencia en videojuegos: ajuste automático de bitrate para plataforma de streaming interactivo y en tiempo real de videojuegos

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Alex Daniel Amaral Bordón
Alejandra Armendariz Voelker
Santiago Erramuspe Reyes

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

José Joskowicz ..... Universidad de la República

Tribunal

Eduardo Cota ..... Universidad de la República
Eduardo Grampin ..... Universidad de la República
Federico La Rocca ..... Universidad de la República

Montevideo, Uruguay
20 de Junio de 2023

Calidad de experiencia en videojuegos: ajuste automático de bitrate para plataforma de streaming interactivo y en tiempo real de videojuegos, Alex Daniel Amaral Bordón, Alejandra Armendariz Voelker, Santiago Erramuspe Reyes.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 93 páginas.
Compilada el sábado 29 julio, 2023.
http://iie.fing.edu.uy/

En algún lugar,
algo increíble está esperando ser descubierto.

Carl Sagan

Esta página ha sido intencionalmente dejada en blanco.

Agradecimientos

Queremos expresar nuestro más profundo agradecimiento a todas las personas que fueron fundamentales en la realización de este proyecto, cuyo valioso aporte fue crucial a lo largo de este camino.

En particular, deseamos reconocer y agradecer a nuestro tutor, José Joskowicz, quien se mantuvo atento y dispuesto a colaborar en todo momento. La generosidad con la que ha compartido sus conocimientos de manera profesional y humana ha sido invaluable para nosotros.

Asimismo, extendemos nuestro agradecimiento a Franco Miceli, Ignacio Barreto, Emiliano Pastorino y a todo el equipo de ABYA, cuya disposición y apoyo técnico resultaron indispensables para el desarrollo de este trabajo.

Finalmente, queremos expresar nuestro sincero agradecimiento a nuestras familias, quienes nos han brindado un apoyo incondicional a lo largo de nuestra carrera universitaria, así como en todas las facetas de nuestras vidas. Agradecemos a todos aquellos que nos acompañaron en cada etapa, contribuyendo de manera significativa tanto a nuestro crecimiento profesional como personal.

Esta página ha sido intencionalmente dejada en blanco.

A nuestros familiares y amigos
que nos han acompañado a lo largo de este trayecto.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El *Cloud Gaming*, también conocido como juegos en la nube, ha surgido como una alternativa prometedora para disfrutar de videojuegos de alta calidad en dispositivos de bajo costo. Estas plataformas ejecutan los juegos en servidores remotos y transmiten el contenido multimedia al dispositivo del jugador a través de la red de telecomunicaciones. Aunque ofrecen numerosas ventajas, también presentan desafíos y problemáticas, ya que dependen en gran medida de la calidad de la red que conecta los servidores con los usuarios.

En este trabajo, se investiga el impacto de los parámetros de la red en la calidad del video, con el objetivo de predecir en tiempo real posibles degradaciones visuales. Se ha determinado que la latencia es un indicador confiable de la pérdida de cuadros de video o frames, lo cual conduce a una disminución en la calidad visual. Basándose en esta premisa, se ha desarrollado un algoritmo en tiempo real capaz de predecir las degradaciones visuales antes de que ocurran, lo que permite tomar medidas preventivas.

Además, este algoritmo implementado tiene la capacidad de ajustar automáticamente la tasa de codificación del códec de video después de la predicción. De esta manera, se logra mantener una experiencia de juego fluida y mejorar la calidad de la experiencia para los usuarios respecto a condiciones no favorables de red sin la aplicación del algoritmo.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

El mundo de los videojuegos en dispositivos personales se ha hecho muy popular en los últimos 30 años. La agencia analista especializada y líder en el tema, Newzoo, muestra que las ganancias de la industria del ocio virtual están en constante crecimiento cada año. El reporte de esta agencia del año 2022 estima que las ganancias de la industria en su totalidad fueron de 182.000 millones de dólares a nivel mundial en ese año [1]. Por otra parte, las inversiones que hacen las empresas dedicadas al diseño y producción de videojuegos, se comparan con las grandes inversiones que hace la industria cinematográfica en las películas más taquilleras. El videojuego *Cyberpunk 2077*, que salió al mercado a fines del 2020, tuvo un presupuesto de 316 millones de dólares, cifra que se compara con la producción de una película de *Marvel* o *Star Wars* [2].

En los sistemas tradicionales, los jugadores deben instalar los videojuegos en sus propias máquinas físicas para poder jugar. Por otro lado, como se muestra en la figura 1.1, los juegos en la nube o *cloud gaming* (CG) utilizan una arquitectura cliente-servidor en la cual los juegos se cargan y procesan en un servidor, y el usuario visualiza el video del juego que está jugando en su dispositivo, también conocido como *thin client* o cliente ligero [3]. El CG proporciona un acceso más sencillo a los videojuegos, eliminando la necesidad de adquirir hardware costoso [4].

Formalmente, la recomendación ITU-T G.1032 [5] define *Cloud Gaming* como juegos en línea caracterizados por enviar el contenido del juego desde el servidor al cliente como una transmisión de video, y cuyos controles se transmiten desde el cliente al servidor. En este modelo, la ejecución, la lógica del juego, la representación de la escena virtual y la codificación del video se realizan en el servidor, mientras que el cliente es responsable de la decodificación del video y la captura de la entrada del cliente. En otras palabras, CG permite a un usuario jugar videojuegos localmente en su equipo, mientras que la ejecución real se lleva a cabo en la nube y el contenido multimedia se transmite desde el servidor al usuario a través de la red de telecomunicaciones.

En contraste, mientras que la transmisión de audio y video en tiempo real, como Spotify, Live YouTube y Zoom, parece ser una tecnología con un gran grado

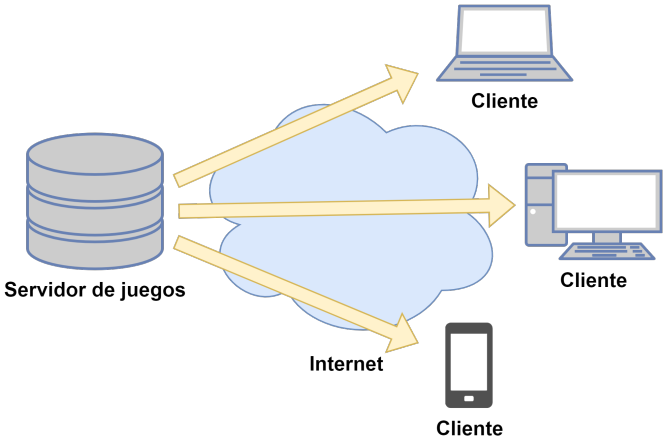


Figura 1.1: Plataforma de cloud gaming

de madurez, los sistemas de CG tienen muchas más tareas que realizar, como se menciona en el párrafo anterior. Además de ejecutar los juegos, deben manejar los controles de los jugadores, realizar la renderización de la pantalla, la codificación, la paquetización, la transmisión, la decodificación y la visualización en tiempo real, a diferencia de las aplicaciones ya mencionadas. Todo esto hace que un sistema de CG sea más difícil de optimizar en comparación con un sistema clásico de transmisión de multimedia ya que, a diferencia de este último, el retardo en CG es crítico [6].

Mientras que en CG la complejidad y los requisitos de ejecución del juego son independientes de las capacidades del dispositivo del usuario, estos requisitos dependen principalmente de la conexión a Internet y de las características de la red local del jugador, como por ejemplo conexión por cable UTP o Wi-Fi. Además de otros factores que influyen en la experiencia de juego, los principales cuellos de botella en la red son el ancho de banda limitado, la latencia y la pérdida de paquetes. Estos factores están correlacionados entre sí y afectan la transmisión de video y las interacciones de los jugadores.

El objetivo principal de este proyecto de fin de carrera es investigar, desarrollar y validar un algoritmo que permita predecir en tiempo real la capacidad del canal de comunicación entre el servidor y el dispositivo del usuario, además de detectar de forma preventiva potenciales degradaciones visuales durante las partidas. Con esta capacidad de predicción, se notifica al servidor para que ajuste automáticamente la tasa de transmisión, adaptándose a las variaciones del canal y evitando pantallas congeladas y/o artefactos indeseados de imagen en el juego.

Este proyecto surge como respuesta a un problema planteado por nuestro cliente ABYA, una empresa de origen uruguayo especializada en el negocio del *cloud gaming*. ABYA fue fundada en 2015, y en 2021 logró una alianza con Nvidia, fabricante de tarjetas gráficas e impulsor de GeForce Now, exitosa plataforma de CG en Norte América y Europa. Los responsables de GeForceNow se vincularon con ABYA para ofrecer este servicio en Latinoamérica.

Capítulo 2

Objetivos

El objetivo principal de este proyecto de fin de carrera es investigar, desarrollar y validar un algoritmo que pueda predecir en tiempo real la capacidad de comunicación entre el servidor y el dispositivo del usuario, así como identificar de manera anticipada posibles degradaciones visuales durante la partida. En base a esto, se podrá notificar al servidor sobre la situación para que pueda ajustar automáticamente la tasa de transmisión, con el fin de adaptarse a las variaciones del canal. Aunque esto podría resultar en cambios en la calidad de audio y video, el objetivo es mantener la fluidez en tiempo real y, en última instancia, mejorar la experiencia del usuario.

Como objetivos secundarios y deseables se plantean:

- Estudiar y comprender, a alto nivel, las arquitecturas de sistemas de videojuegos interactivos en la nube.
- Comprender las técnicas de codificación de audio y video utilizadas en los sistemas de videojuegos interactivos en la nube.
- Implementar el algoritmo desarrollado en un lenguaje de programación para poder probarlo en condiciones que se asemejen lo más posible a la realidad.
- Proponer mecanismos para medir la calidad de la experiencia (QoE) de los jugadores, de manera que se pueda cuantificar el desempeño del modelo propuesto.

Los criterios de éxito planteados al inicio del proyecto fueron los siguientes:

- El algoritmo debe ser capaz de predecir las pérdidas de paquetes antes de que se produzcan en al menos el 90 % de los casos de prueba basados en datos de sesiones del cliente. Además, el algoritmo debe ser capaz de estimar el ancho de banda al cual limitar el bitrate (tasa de bits) de la transmisión.
- El algoritmo debe mejorar la calidad de experiencia del usuario en comparación con las condiciones previas a su implementación. Esto se verificará mediante la realización de pruebas subjetivas utilizando criterios existentes para la medición de la calidad de experiencia.

Asimismo, al inicio del proyecto, el cliente determinó que el algoritmo deberá cumplir los siguientes criterios de éxito en orden de importancia:

- El algoritmo debe predecir las pérdidas de paquetes al menos 5 ms antes de que se produzcan en al menos el 90 % de los casos de prueba.
- En los casos de predicción de pérdidas, el algoritmo debe ser capaz de estimar el ancho de banda al cual limitar el bitrate de la transmisión en al menos el 90 % de los casos de prueba.
- Se debe obtener una estimación del ancho de banda que siga la curva de ancho de banda del canal, de forma de tener una estimación de capacidad de recepción del cliente.

Al principio del proyecto, en colaboración con el tutor, se distinguieron los criterios de éxitos generales y los específicos del cliente. Esto se debió a la dificultad de prever con precisión si se alcanzarían los criterios del cliente en su totalidad.

Capítulo 3

Antecedentes

3.1. Proveedores de cloud gaming

Los servicios de CG están madurando. Los proveedores están atrayendo más usuarios y atención que nunca a los beneficios de la nube. Al mismo tiempo, algunos de los principales actores de los videojuegos tradicionales (Xbox y PlayStation) están complementando sus experiencias de consolas tradicionales con el CG. Las ganancias de los proveedores de CG en 2022 fueron de 2.4 mil millones de dólares, y se estima que hay 31,7 millones de usuarios que pagan por el servicio [7]. Si bien esta cifra aún es menor si se compara con la totalidad de las ganancias de la industria (182 mil millones en 2022, donde se incluyen juegos de consola e instalados en *PCs gamers*, y juegos para celulares), para 2025 se estima que habrá 86,9 millones de usuarios que pagarán por el servicio de CG [7], lo que hará crecer las ganancias y la popularidad de jugar en la nube.

3.1.1. Oferta a nivel global

A fines de los 2000s, comenzaron a emerger servicios de CG ofrecidos por startups, como OnLive y Gaikai [3]. Luego Gaikai fue adquirida por Sony, uno de los fabricantes más importantes de consolas de videojuegos, marcando un hito en la industria. Hoy en día, una de las principales plataformas de CG en el mundo es GeForce Now de Nvidia. La misma compañía que fabrica GPUs para jugar a juegos de PCs tradicionales, es también una de las más exitosas en el mundo del CG. Al tener un largo catálogo de juegos compatibles, es una gran opción para los jugadores que han acumulado grandes bibliotecas de juegos y quieren jugarlos en la nube. GeForce Now se distingue de la competencia porque, además de tener una gran cantidad de juegos gratuitos (free-to-play, como Fortnite), permite jugar en la nube juegos por los que los usuarios ya han pagado en tiendas como Steam y Epic Games Store. El servicio está disponible para América del Norte y Europa, y para otras partes del mundo a través de alianzas con otros partners [8].

Otra plataforma que está teniendo éxito es Xbox Cloud Gaming, de Microsoft. Aunque es una plataforma reciente (2019), ofrece un sólido servicio de CG. Al

igual que GeForce Now, tiene una gran cantidad de juegos disponibles, incluyendo el famoso juego "Halo", propiedad de Microsoft. Los servicios de Xbox Cloud Gaming están disponibles para Estados Unidos (USA), gran parte de Europa, Japón y Australia [8].

El resto de la oferta se compone de plataformas que están emergiendo, como Luna de Amazon (2020, solo disponible en USA), y otras que si bien ya están hace un tiempo, han pasado por varios cambios, como PlayStation Now de Sony, ahora refundada como Playstation Plus (disponible en USA, Europa y Japón). La plataforma de Google, Stadia, habría cumplido 3 años en noviembre de 2022, pero Google la cerró antes de que pudiera establecerse [8].

3.1.2. ABYA: cloud gaming en Latinoamérica

La compañía ABYA, fundada en Uruguay por Nicolás Jodal, Waldemar Fernández y Franco Miceli en 2015, logró en 2021 una alianza con Nvidia, fabricante de tarjetas gráficas e impulsor de GeForce Now, mencionado en la sección anterior. Los responsables de GeForceNow se vincularon con ABYA para ofrecer el servicio de CG en Latinoamérica [9]. A su vez, la compañía de origen uruguayo, en una alianza con Antel (principal operadora de servicios de telecomunicaciones en Uruguay) utiliza su infraestructura de CG en un data center de gran capacidad, lo que le permite satisfacer un mercado más grande. La ubicación geográfica, su excelente conectividad internacional y su infraestructura de data center, permiten que esta plataforma tenga el potencial de cumplir con los altos requerimientos del servicio y captar un posible mercado de 100 millones de usuarios, especialmente en el cono Sur [10].

El portal de ABYA, que se muestra en la Figura 3.1, cuenta con dos tipos de suscripciones: ABYA Go y GeForce Now Powered by ABYA. ABYA Go está hecho para un usuario casual. Funciona como un "Netflix para videojuegos", es decir que el suscriptor tiene acceso a un catálogo de juegos ya precargado y todos free-to-play. Por otro lado, GeForce Now Powered by ABYA, al igual que versión del hemisferio norte, está pensado para usuarios más "hardcore", es decir jugadores más activos y con más experiencia, que ya tienen sus juegos favoritos en tiendas digitales, y los quieren jugar en la nube. "ABYA es la única empresa en el mundo que tiene una oferta desde el casual hasta el "hardcore"", dice Franco Miceli, CEO de ABYA [9].

Este proyecto de fin de carrera se desarrolló utilizando juegos de la plataforma ABYA Go. La empresa proporcionó al equipo de proyecto un espacio en el servidor de *staging* que se utiliza para realizar pruebas internas de la empresa. Además, se desarrolló un sistema de registro de datos de cada partida, el cual se describirá con más detalle en el Capítulo 5.

Figura 3.1: Portal de ABYA. Imagen extraída de [11]

3.2. Conceptos generales de cloud gaming

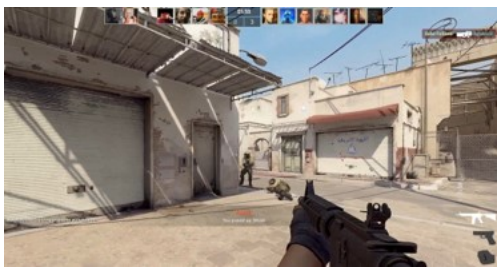
3.2.1. Tolerancia al retardo y categorías de juego

Como se menciona en la introducción, un sistema de CG debe pasar por múltiples etapas de procesamiento luego de capturar las acciones del jugador. Para lograr una buena interactividad, todas estas operaciones deben realizarse en cuestión de milisegundos. Este intervalo de tiempo se define como retardo de interacción [12].

Diferentes estudios muestran que la tolerancia al retardo de interacción varía según el tipo de juego. Los juegos en primera persona, como el famoso juego de disparos Counter Strike, se ven afectados en términos de jugabilidad cuando hay un retraso de tan solo unos 100 ms. Estos juegos se denominan *First Person Shooter* (FPS), y son particularmente sensibles al retardo.

Por otro lado, los juegos en tercera persona, también llamados *Role Playing Games* (RPG), pueden soportar hasta 500 ms de retardo sin que la mayoría de los jugadores lo perciba. Esto se debe principalmente a que los comandos del jugador, es decir, las acciones de su personaje en el juego, son realizadas por el avatar del jugador, por lo que no se espera una respuesta instantánea. Igualmente, una demora significativa también puede afectar la experiencia del jugador [12].

En una última categoría se encuentran los juegos de estrategia en tiempo real (RTS por sus siglas en inglés), en los cuales el jugador tiene una vista "omnipresente" desde arriba, supervisando las entidades que debe controlar. Estos juegos pueden tolerar retrasos de hasta 1000 ms, ya que las acciones en este tipo de juegos, como la construcción de un edificio en el mundo del juego, suelen durar varios segundos o incluso minutos, y el jugador difícilmente note esa demora [5]. Por este motivo, los juegos RTS son los que presentan menos dificultades para los sistemas de CG. En la Figura 3.2 se muestran juegos clásicos de las tres categorías mencionadas.



(a) FPS: Counter Strike (1999)

(b) RPG: GTA San Andreas (2004)

(c) RTS: Warcraft 3 (2002)

Figura 3.2: Categorías generales de juegos

3.2.2. Diferencias entre cloud gaming y otros sistemas

En los sistemas de videojuegos tradicionales, todo el procesamiento lógico y gráfico se realiza localmente, por lo que el retardo de interacción es prácticamente nulo, y solo se presenta en los casos de multijugador. En cambio, el retardo siempre está presente en mayor o menor medida en los sistemas de CG, y debe ser considerado incluso en los juegos de un solo jugador. Esto se debe a que el juego realiza todo el procesamiento lógico y gráfico en un servidor en la nube, y luego se transmite al jugador, por lo que el cliente no tiene la capacidad de ocultar el retardo de la interacción [12].

Como se muestra en la Figura 3.3, en los sistemas tradicionales de juegos multijugador u *online gaming*, la lógica del juego y la renderización de las escenas se ejecutan localmente en los clientes del juego, y los servidores solo son responsables de mantener los estados del juego entre los múltiples clientes. Por otro lado, en la arquitectura de CG, toda la lógica del juego, incluido el renderizado de video que consume muchos recursos, se traslada a los servidores del juego [13].

Por otra parte, los requisitos para la transmisión de audio y video en un sistema de CG son bastante similares a los de otros sistemas clásicos de transmisión de multimedia, como lo son plataformas de streaming de películas o música. Ambos tipos de sistemas deben codificar y comprimir rápidamente el contenido multimedia y distribuirlo a los usuarios finales. Sin embargo, existe una diferencia importante entre ambos tipos de sistemas: en comparación con un sistema clásico de streaming de multimedia, un sistema de CG no tiene la capacidad de almacenar en búfer los frames de video o *frames* en el lado del cliente. Esto se debe a que cuando un

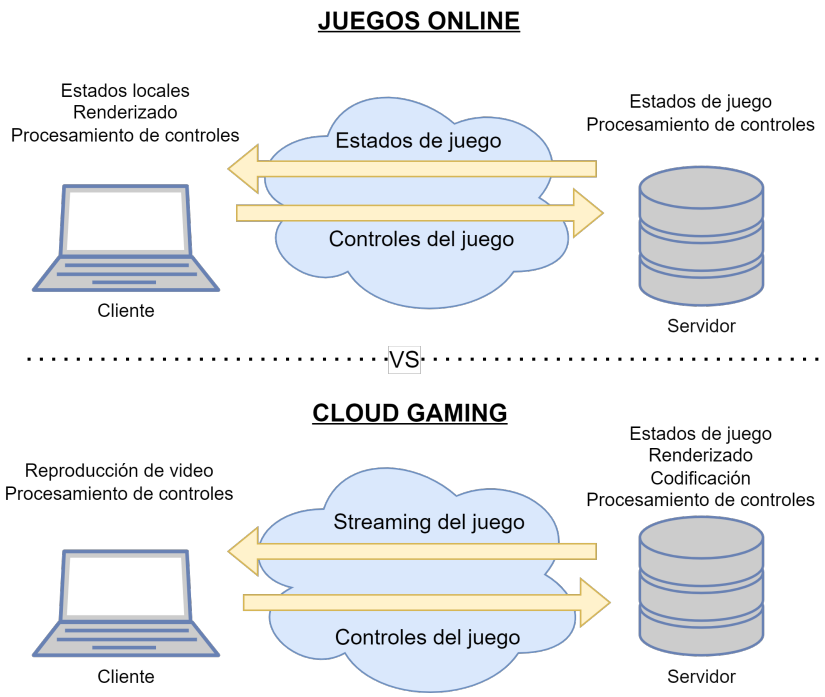


Figura 3.3: Juegos Online vs Cloud Gaming

jugador ejecuta un comando en el juego, dicho comando debe ser transmitido a través de Internet y pasar por todo el proceso mencionado previamente antes de que el usuario pueda ver la acción en el juego. Como se mencionó en párrafos anteriores, todo esto debe realizarse en un intervalo de tiempo de 100 a 500 ms, por lo que no hay demasiado margen para un búfer [12]. Por otro lado, los sistemas clásicos de multimedia pueden tolerar mayores retrasos y aplicar técnicas de *jitter buffer* para garantizar una buena experiencia del usuario.

3.3. Sistema de cloud gaming

3.3.1. Arquitectura

El acceso para jugar en una plataforma de CG comienza de la siguiente manera: un usuario se conecta primero al sistema a través de un portal que proporciona una amplia variedad de juegos disponibles. Luego, el usuario selecciona un juego y solicita jugar. Al recibir la solicitud, el servidor del portal busca un servidor de juegos disponible, inicia el juego seleccionado en el servidor y establece la conexión con el cliente.

Teniendo en cuenta los requisitos de un sistema de CG mencionados en las secciones anteriores, se presenta un esquema general de la arquitectura. En la Figura 3.4 se pueden observar los diferentes módulos necesarios para el sistema. Los comandos del jugador son enviados a través de Internet desde el cliente hasta la

plataforma de CG. Una vez que los comandos llegan a la plataforma, se convierten en acciones dentro del juego y son procesados por el módulo de lógica del juego, el cual traduce los cambios al mundo del juego [12]. Los cambios en el entorno del juego son procesados por la unidad de procesamiento gráfico (GPU) del sistema y se traducen en una escena renderizada. Posteriormente, la escena renderizada debe ser comprimida por el codificador de audio y video, y luego es enviada a un módulo de transmisión, que entrega el flujo de multimedia al cliente. Por último, el cliente decodifica el audio y video, y muestra los frames resultantes y el audio en el dispositivo del jugador [12].

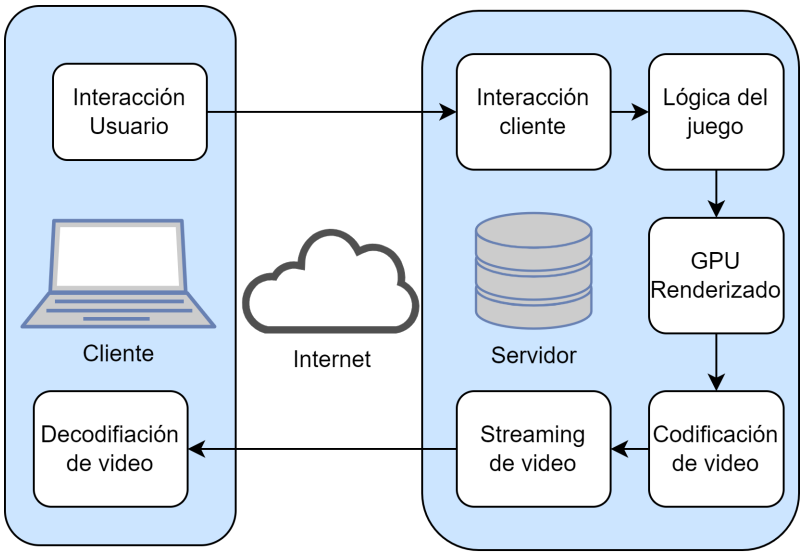


Figura 3.4: Arquitectura de un sistema de cloud gaming

En la arquitectura se definen dos tipos de flujos de red: el flujo de datos y el flujo de control. El flujo de datos se utiliza para transmitir frames de audio y video (A/V) desde el servidor hacia el cliente. Por otro lado, el flujo de control se ejecuta en sentido inverso, transmitiendo los controles del juego que envía el usuario desde el cliente hacia el servidor [14]. En la Figura 3.5 se muestra un esquema que representa ambos flujos y sus respectivos sentidos.

En la ejecución del juego, el servidor debe capturar los frames de A/V del juego, codificar dichos frames utilizando los codecs seleccionados, y luego transmitir los frames codificados al cliente a través del flujo de datos. Este flujo se transporta sobre *User Datagram Protocol* (UDP), utilizando los protocolos de tiempo real *Real Time Streaming Protocol* (RTSP) y/o *Real-time Transport Protocol* (RTP), u otros protocolos similares. Al mismo tiempo, el servidor debe interactuar con el juego, es decir, recibir los controles del usuario desde el cliente. Por otra parte, el cliente solo necesita contar con un reproductor multimedia compatible con RTP/RTSP y un decodificador para el códec utilizado, además de registrar y enviar las acciones del teclado, el mouse, entre otros.

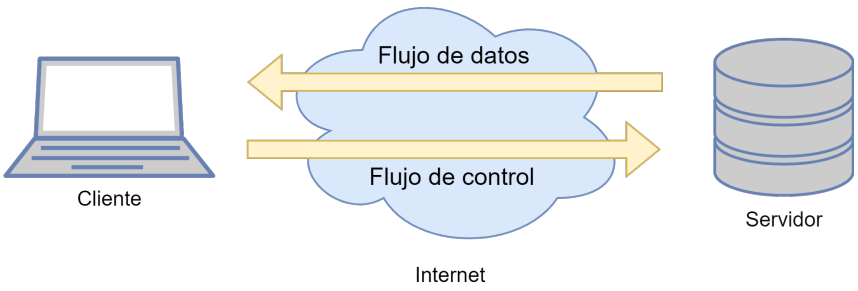


Figura 3.5: Flujos de datos y control

3.3.2. Códec de video H.264

Uno de los codecs más utilizados en los sistemas de CG es el códec H.264. Este códec es muy utilizado porque puede adaptarse bien a exigencias de tiempo real [12]. La codificación en H.264 utiliza técnicas de estimación y compensación de movimiento que permiten eliminar información temporal redundante entre frames. [15] Las secuencias de video se dividen en grupos de figuras (GOP), los cuales pueden incluir tres tipos distintos de frames:

- Intra (I): codificados únicamente con técnicas de compresión espacial y utilizados como referencia para las predicciones de frames P o B.
- Predictivos (P): codificados utilizando información de frames I o P previos.
- Predictivos Bidireccionales (B): codificados utilizando información de frames previos o siguientes. Este tipo de frames no son utilizados por el cliente debido a que ocasionan un aumento de la latencia, lo cual no es recomendable en sistemas de video de tiempo real.

Asimismo, se introdujeron nuevos tipos de frames en este estándar, como el *Switching P* (SP) y el *Switching I* (SI), los cuales permiten una reconstrucción precisa de valores específicos. También se emplean dos técnicas de codificación entrópica conocidas como *Context-Adaptive Variable-Length Coding* (CAVLC) y *Context-Adaptive Binary Arithmetic Coding* (CABAC). El segundo método es más complejo, pero posee una mayor eficiencia, reduciendo la tasa de bits entre un 10 % y un 15 % para la misma calidad [16].

La recomendación H.264 [17] establece diversos perfiles y niveles que definen los requisitos mínimos para codificadores y decodificadores. Los perfiles definen una sintaxis utilizada para generar un flujo de bits coherente, mientras que los niveles imponen restricciones en los valores de parámetros, como la tasa máxima de bits y la resolución de las imágenes, entre otros aspectos. Es obligatorio que todos los decodificadores que cumplen con un perfil específico admitan todas las características asociadas a dicho perfil, a fin de permitir una correcta decodificación de los flujos de bits recibidos. En total, la norma establece 17 perfiles, cada uno con características y usos específicos, como aplicaciones móviles, videoconferencias, TV

en definición estándar y alta definición, entre otros. Algunos de los perfiles más destacados son el *Baseline Profile* (BP), el *Main Profile* (MP) y el *High Profile* (HiP).

El sistema del cliente, que se presentó en detalle en la Sección 3.1, utiliza un GOP abierto. El tamaño de un GOP se define por la cantidad de frames existentes entre dos frames tipo I, en ese caso decimos que es cerrado. Sin embargo, en el sistema del cliente, al comienzo de la transmisión, el servidor envía un único frame de video de tipo I, seguido por frames de tipo P. Cada uno de estos frames está codificado en función del frame anterior, y se mantiene esta secuencia hasta el final de la partida. Idealmente, no se enviaría un segundo frame de tipo I, por lo que decimos que el GOP se encuentra abierto.

En la realidad, el retardo y la pérdida de paquetes en la red de telecomunicaciones pueden afectar la decodificación de los frames de tipo P. Si un paquete de un frame de tipo P no está disponible al momento de la decodificación, ya sea porque llegó tarde al decodificador o porque efectivamente se perdió, se producen degradaciones visibles en el cuadro decodificado actual y los siguientes. Esto puede resultar en pantallas congeladas y/o artefactos indeseados en el video, como se muestra en la Figura 3.6. En este contexto, se vuelve necesario retransmitir el frame de referencia (*Intra Refresh*), lo cual implica un mayor consumo de ancho de banda. Además, esta retransmisión debería ocurrir instantáneamente después de detectar la pérdida, lo que significa que ese consumo de ancho de banda puede producirse antes de que mejore la condición del canal, empeorando aún más la situación. Esto destaca la importancia de detectar de forma preventiva y evitar que se produzcan pérdidas en el sistema del cliente.

3.4. Calidad de experiencia

Aunque el CG ofrece varias ventajas, también plantea desafíos adicionales en términos de conexión a la red, como será descrito en la Sección 3.5. Por lo tanto, es fundamental que los proveedores de servicios de CG se aseguren de brindar una experiencia satisfactoria a sus clientes. La Calidad de Experiencia (QoE) se define como el grado de deleite o molestia del usuario respecto de una aplicación o servicio. [18] Esta calidad se puede evaluar mediante la realización de pruebas subjetivas, donde se recopila la opinión directa de los usuarios y se calcula un valor de opinión promedio. También es posible predecir la QoE utilizando modelos.

En este contexto, el grupo de estudio 12 de ITU-T decidió analizar los principales factores que afectan la QoE en CG. Sobre la base de esto se han desarrollado métodos subjetivos de evaluación y modelos predictivos para la calidad de los videojuegos en línea. Este estudio concluyó en la publicación de tres nuevas recomendaciones:

- ITU-T Rec. G.1032: *Influence factors on gaming quality of experience* [5]
  Esta recomendación tiene como objetivo identificar los principales factores

Figura 3.6: Capturas de pantalla del juego Alien Rage con y sin degradaciones visuales (arriba y abajo respectivamente).

que afectan la QoE del usuario en el contexto de CG. Estos factores se dividen en tres categorías: factores humanos, factores del sistema y factores del contexto. En particular, se destaca la pérdida de paquetes y, como consecuencia, la pérdida de frames, como un factor relacionado con el sistema que provoca una disminución en la QoE del usuario. Esto respalda la idea de que evitar la pérdida de frames tiene un impacto positivo en la QoE del usuario.

- ITU-T Rec. P.809: *Subjective evaluation methods for gaming quality* [19]
  El objetivo de esta recomendación es la descripción de métodos subjetivos para la evaluación de la calidad de experiencia en videojuegos en línea. Se definen dos enfoques de pruebas subjetivas: pruebas pasivas e interactivas. En particular, si se deseara evaluar cómo el algoritmo desarrollado en este proyecto mejora la experiencia del usuario, se deberían implementar pruebas de tipo interactivas.

- ITU-T Rec. G.1072: *Opinion model predicting gaming quality of experience for cloud gaming services* [20]
  Se presenta un modelo de predicción cuantitativa que proporciona una forma de predecir la calidad percibida por el usuario, tanto en términos generales como en aspectos individuales de la QoE, a partir de las características del sistema. Esta estimación se basa en el análisis del impacto de los parámetros de la red IP y los parámetros de codificación de video. Utilizando estos parámetros, el modelo puede calcular una estimación de la calidad que experimentará el usuario final, brindando información que permitirá mejorar y optimizar los sistemas de CG en función de la QoE deseada.

3.5. Desafíos de cloud gaming

Las plataformas de CG dependen en gran medida de la calidad de la red que conecta el servidor con el cliente. Un bajo rendimiento de la red puede tener un impacto significativo en la calidad de la experiencia del usuario. La alta latencia degrada la jugabilidad, especialmente en juegos que tienen una baja tolerancia al retardo interacción. Además, para proporcionar gráficos de alta calidad, se necesita un gran ancho de banda [21]. La mayoría de los proveedores de servicios de CG recomiendan una conexión de al menos 15 Mbps [22].

Servidores más cercanos a los usuarios, edge computing, el despliegue de la red celular 5G y el uso de redes Wi-Fi en la banda de 5 GHz, ayudarán a minimizar los problemas de red [23]. Sin embargo, en la actualidad estas redes no están desplegadas de forma universal, lo que resulta en degradaciones visuales para los jugadores de CG debido a problemas de red. Por lo tanto, es necesario implementar soluciones independientes de la red que mejoren la experiencia de juego. Estas soluciones pueden desarrollarse utilizando herramientas de optimización instaladas en el cliente y/o el servidor, empleando sistemas que predigan posibles degradaciones durante el juego.

3.6. Trabajos relacionados

Como se menciona en la introducción de este documento, el objetivo de este proyecto es desarrollar un algoritmo que permita predecir en tiempo real potenciales degradaciones visuales, con el fin de notificar al servidor y ajustar la tasa bits enviada al cliente. El modelo utilizado en la mayoría de los artículos consultados inicialmente se basa en la idea de que se puede pensar la red IP como una conexión entre el transmisor y el receptor dada por un solo enlace cuyo ancho de banda es igual al menor ancho de banda de todos los enlaces transitados por los paquetes [24]. Este enlace, llamado enlace de cuello de botella o *Bottleneck*, caracteriza a la red, y limita la velocidad de transmisión.

La latencia que experimenta un paquete en una red se compone del retraso de transmisión, de procesamiento y el retraso en la cola, es decir, el tiempo que demora

un paquete en ser procesado y transmitido. En estas condiciones, es intuitivo pensar que los retrasos de los paquetes están vinculados con las dimensiones del *buffer* asociado a este enlace, que se define como "Buffer de la red", así como a la cantidad de paquetes en espera que este contiene en cada instante de tiempo. Por lo tanto, la congestión se podría inferir a partir de las variaciones de la cola del Buffer de la red.

En este contexto, Alós et al. [25] proponen en su estudio sobre el control de congestión para una plataforma de cloud gaming un algoritmo basado en una adaptación del algoritmo BBR (*Bottleneck Bandwidth and Round-trip propagation time*) utilizado en TCP, pero aplicado a UDP. En su investigación, definen el punto óptimo de operación (OOP) de un enlace como la tasa de bits de transmisión que permite utilizar el enlace a su máxima capacidad con el mínimo retardo.

Cuando se transmite una tasa de bits inferior a OOP, el enlace se encuentra subutilizado. Sin embargo, si se intenta transmitir a una tasa mayor a OOP, los paquetes comienzan a acumularse en el buffer de la red, lo que aumenta el retraso experimentado por los paquetes sin incrementar efectivamente la tasa de bits entregada. En este caso, es necesario disminuir la tasa de transmisión para evitar la acumulación de paquetes en la cola, evitando así que el buffer se llene completamente y se produzca una pérdida de paquetes. A esta situación no deseada en el buffer se le conoce como *Bufferbloat*.

Debido a esto se asume que estamos en un estado de congestión cuando ocurre simultáneamente la presencia de paquetes en la cola de espera del buffer de la red y un aumento en la latencia. En su adaptación del algoritmo BBR, utilizan los valores de bitrate de transmisión y recepción para medir la existencia de paquetes en cola de espera, así como el valor de la latencia de vídeo de ida y vuelta RTVL (*Round Trip-time Video Latency*). Todos estos valores se consideran en un intervalo de tiempo determinado para evitar reaccionar ante fluctuaciones repentinas.

Otro trabajo tomado en cuenta en el desarrollo de este proyecto es el Algoritmo de Control de Congestión de Google (ACCG) [26]. El ACCG está basado en la idea de que el retardo de la red de extremo a extremo está correlacionado con la congestión en la misma. Utiliza el gradiente del retardo de la cola para inferir la congestión y estimar las tasas de envío según las variaciones observadas. En este trabajo se presenta un ejemplo de esto, donde se implementa un filtro Kalman como filtro de recepción de forma de predecir el valor del gradiente de retardo $m(t_i)$ a partir de la medida actual $dm(t_i)$. La realimentación corrige el error y, en este caso, $m(t_i)$ converge hacia el valor real de la señal.

Finalmente, otra alternativa analizada fue el algoritmo de transmisión DT3 propuesto por Zhang et al. [27] que tiene como objetivo mejorar la Calidad de Experiencia (QoE) del usuario para la transmisión de video en tiempo real. Se basa en la asignación de prioridades de los distintos frames transmitidos según el deadline de cada uno, mejorando así la tasa de finalización de cuadros. Sin embargo, este algoritmo no es aplicable en nuestro caso ya que el buffer de video del sistema del cliente es del tamaño de un único frame.

Según los artículos mencionados, existe una relación directa entre el retraso experimentado por los paquetes desde que son enviados por el servidor hasta que son recibidos por el cliente, y el incremento de paquetes en la cola del Buffer de la red. Por lo tanto, un aumento significativo del retraso puede ser usado como indicador de que el Buffer está próximo a llenarse y producir pérdidas de paquetes. La pérdida de paquetes a su vez está correlacionada con la pérdida de cuadros de videos, y esto último directamente relacionado con las degradaciones visuales, como se demostrará en las secciones siguientes.

En la bibliografía mencionada, se estudió la forma en se mide el retraso de los paquetes en la red, y con especial énfasis en una red para jugar videojuegos. Por ejemplo, Alós et al. [25] en su investigación propone detectar el instante en que el Buffer de la red comienza a llenarse, comparando los bits enviados por el servidor $\lambda(t)$ con los bits recibidos por el cliente $\mu(t)$. Esto se hace durante intervalos de tiempos iguales a lo largo de la partida. Si en alguno de estos intervalos de medición $\lambda(t) > \mu(t)$, se interpreta que existen paquetes acumulándose en el Buffer. A partir de este instante t_0, se comienza a medir la latencia RTLV. Si la misma supera cierto umbral al que llama $RTLV_{max}$, se le indica al servidor que debe bajar la tasa de envío para evitar que el Buffer se siga llenando. Esto hace que el Buffer comience a vaciarse y consecuentemente se reduzca la latencia. Con esta acción, se mejora la jugabilidad, no se pierden paquetes y por lo tanto se evita que se produzcan degradaciones visuales durante la partida .

En las siguientes secciones, se desarrollará cómo se llevaron a cabo y se adaptaron algunas de estas ideas propuestas en la bibliografía leída. En nuestra investigación, se constata que la pérdida de frames de video está correlacionada con la ocurrencia de degradaciones visuales durante la ejecución de una partida. Además, identificamos que instantes previos a la ocurrencia de estas pérdidas, la latencia de la red comienza a incrementarse y sostener valores muy por encima de los valores promedios en partidas sin degradaciones. Por lo tanto, concluimos que estos aumentos detectados en la latencia son indicadores de congestión de la red, análogamente a decir que el Buffer se está llenado.

Finalmente, elaboramos un método para medir el incremento de la latencia en tiempo real durante la ejecución de una partida, y buscamos un umbral que nos permitiera predecir la ocurrencia de degradaciones visuales una vez que la latencia supera ese umbral. Una vez logramos predecir la ocurrencia de degradaciones visuales, implementamos un sistema para reducir la tasa de envío, de forma en que el impacto en la disminución de la resolución de video sea lo menor posible.

Capítulo 4

Entorno de pruebas

Las pruebas iniciales utilizaron una red Wi-Fi doméstica típica para determinar las degradaciones más comunes percibidas por los usuarios en dichas redes. Basándonos en los resultados obtenidos, se desarrolló y desplegó un banco de pruebas controlado.

Nuestro sistema de pruebas consiste en una computadora cliente donde se ejecuta la aplicación de CG en simultáneo con el algoritmo a ser evaluado. Esta se encuentra conectada a Internet a través de otra computadora configurada como un puente a nivel de capa 2, en la que se ejecuta un emulador de red. El diagrama de configuración de la conexión se muestra en la Figura 4.1. El cliente está directamente conectado al puente mediante cables de red (UTP), así como también la conexión desde el puente al router a través del cual se tiene salida a internet. El router de Internet se utilizó exclusivamente para esta prueba, sin ninguna otra fuente de tráfico de Internet. Con esta configuración, el banco de pruebas tiene control total sobre la red cableada que simula el entorno del usuario. El servidor ABYA utilizado para esta prueba, ubicado en la nube, no se cargó con otros usuarios, por lo que no había otras fuentes de degradación o afectaciones presentes durante las pruebas.

Para implementar el puente de red, utilizamos una computadora con Ubuntu 20.04 y dos interfaces de red LAN. Empleamos el comando *brctl*, que forma parte del paquete *bridge-utils*, para conectar los segmentos de red adyacentes a ambas interfaces. Esto permitió el intercambio transparente de paquetes IP entre el cliente y el servidor CG ubicado en la nube.

La emulación de las condiciones de red se realiza mediante la aplicación *NetEm*, que también se ejecuta en la computadora que actúa como puente de red. Después de realizar varias pruebas preliminares y basándonos en la bibliografía consultada, se determinó que la limitación del ancho de banda del canal es el parámetro más crítico a considerar, ya que emula las condiciones de una red cableada o Wi-Fi doméstica sobrecargada. Al limitar el ancho de banda del canal, se produce una acumulación de paquetes en el buffer de la red, lo que provoca un aumento en la demora de los paquetes salientes en el sentido de *downlink*. Esta acumulación de

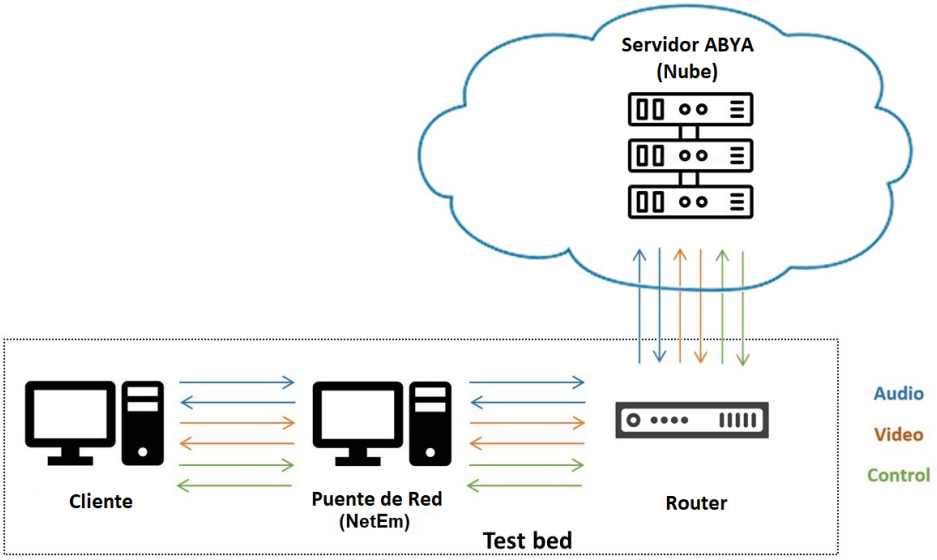


Figura 4.1: Diagrama de conexión del banco de pruebas

paquetes en el buffer ocasiona que los paquetes lleguen tarde al decodificador o que se pierdan en el caso de *Bufferbloat*. Esto deriva en pérdida de frames, lo que a su vez se percibe como degradaciones visuales.

Se utilizaron seis situaciones representativas típicas encontradas en contextos de Wi-Fi doméstico para evaluar el impacto en diferentes juegos. En la Figura 4.2 se muestran las diferentes configuraciones de prueba, cada una con una duración de 240 segundos. Además, con el objetivo de sistematizar la implementación de las condiciones de emulación, se desarrolló un script que automatiza los cambios en NetEm según cada configuración de prueba. El script implementado se encuentra en el Apéndice D.

Finalmente, también se considerará el caso en que el cliente se encuentra a una distancia superior a mil kilómetros del servidor. Por ejemplo, consideramos el caso en el que el cliente se encuentra en São Paulo, Brasil. En esta situación, se agregará a las condiciones implementadas previamente un retardo adicional de 50 ms, correspondiente a la latencia promedio de la red en estas condiciones.

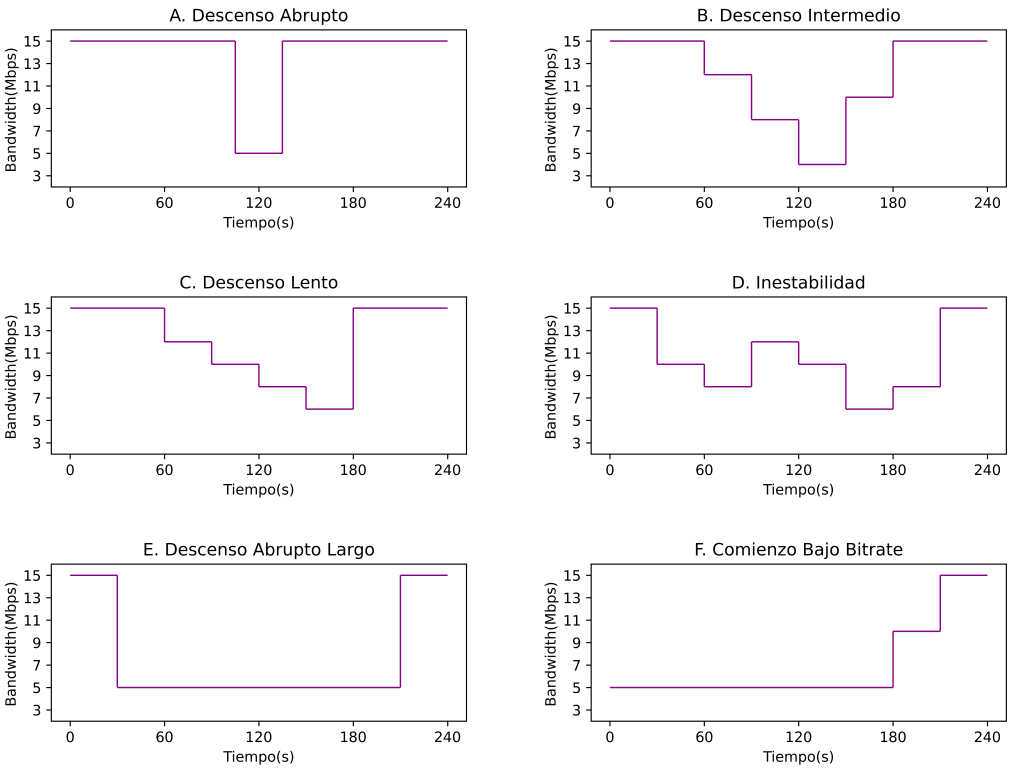


Figura 4.2: Condiciones de pruebas de limitación de ancho de banda implementadas con NetEm.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Arquitectura del sistema

Uno de los principales desafíos de este trabajo consistió en acceder en tiempo real a un conjunto de variables de interés generadas por el servidor de ABYA y modificar la tasa de codificación de video durante el transcurso de una partida. A su vez, el cliente de ABYA puede calcular estas y otras variables y dejarlas disponibles localmente. Sin embargo, este proceso sobre el servidor conlleva el riesgo de afectar el servicio que la empresa brinda a sus clientes. Por lo tanto, fue necesario diseñar una solución no invasiva que permita avanzar en el trabajo de manera efectiva, segura y sin poner en peligro el servicio de *streaming*.

5.1. Acceso a la plataforma

Como se menciona en la Sección 3.1, se ha tenido acceso a un catálogo de juegos de diversas categorías, como lo son FPS, RPG, Carreras, Lucha, para realizar las pruebas necesarias en el desarrollo de este proyecto. El servicio ofrece las mismas características que se brindan a un usuario convencional y se pone a disposición de cada miembro del equipo de proyecto mediante una cuenta personal. Todos los juegos se ejecutan en diferentes computadoras personales de propósito general que cumplen con los requisitos mínimos para garantizar un buen funcionamiento del servicio.

5.2. Descripción general del sistema

En esta sección se proporcionará una breve descripción del funcionamiento del sistema desarrollado. La Figura 5.1 ilustra el esquema general del sistema. Del lado del cliente se encuentra la aplicación de ABYA, una carpeta llamada "pruebaABRA", y el programa que contiene el algoritmo desarrollado en el transcurso de este proyecto. Este programa fue desarrollado para cumplir los objetivos descriptos en el capítulo 2, es decir la predicción de degradaciones visuales y el ajuste de bitrate automático para evitarlas. El funcionamiento general del programa se presentará en la Sección 5.5, y el fundamento teórico de su funcionamiento en los capítulos 6 y 7.

El programa y la aplicación cliente de ABYA tienen acceso a la carpeta "pruebaABRA", la cual contiene dos archivos. El primero es un archivo CSV o *log*, que recolecta los valores medidos de las variables de interés de la red del lado del cliente. Estos valores medidos son los datos que utiliza el programa para luego tomar acciones respecto a la predicción de degradaciones visuales y el ajuste automático de bitrate al ejecutarse una partida en la aplicación. El segundo es un archivo ".ini" donde se puede ingresar la frecuencia de obtención de los valores medidos de la red. También en este archivo se puede ajustar el valor inicial de la tasa de codificación de video en una partida, así como el valor de incremento y de decremento de la misma, como será detallado en la sección 5.4. Estos valores deben modificarse antes de iniciar un juego y no pueden cambiarse durante su desarrollo. Ambos archivos y sus variables serán explicados en la sección 5.3.

Para jugar una partida, el usuario accede al catálogo de juegos a través de la aplicación cliente y, una vez seleccionado el mismo, se realiza el lanzamiento en el servidor. Antes de seleccionar el juego, se debe ejecutar el programa implementado. Este entra en un bucle de espera hasta que comienza la partida. Cuando comienza la partida, se genera el archivo CSV mencionado en la carpeta "pruebaABRA". Durante la ejecución del juego, el cliente escribe en el archivo CSV creado el valor de las variables de red medidas. El programa implementado tiene acceso a este registro en todo momento durante la ejecución del juego, lee su contenido, y basándose en los datos registrados en tiempo real, decide si debe enviar una instrucción para aumentar o disminuir la tasa de codificación de video al servidor, como será explicado en capitulo 7.

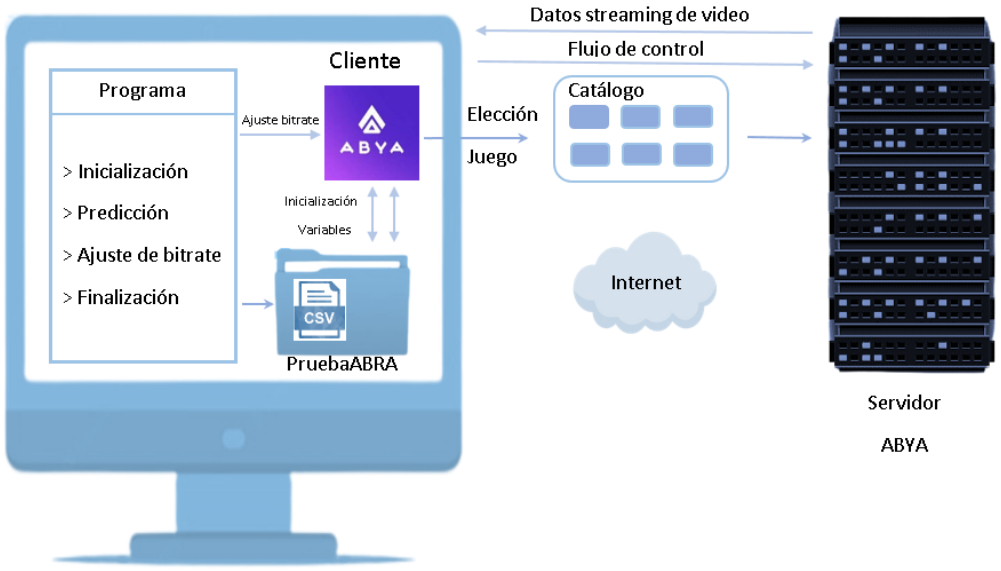


Figura 5.1: Esquema general del sistema.

5.3. Visualización de variables del sistema

La visualización de las principales variables del sistema se realiza mediante una versión de la aplicación proporcionada por el equipo de ABYA, la cual se aloja en la computadora del jugador. Al inicio de la partida, se genera un archivo en formato CSV que es accesible en todo momento. La aplicación registra en tiempo real las variables relevantes, generando una entrada de datos por cada unidad de tiempo configurada. Una vez iniciada la partida, el registro se etiqueta y se guarda en una carpeta accesible desde la computadora del jugador.

Debido a que no tenemos acceso al servidor ni al código de la aplicación, utilizamos para la implementación los datos encontrados en el CSV generado por la versión de la aplicación que nos proporcionó la empresa. De esta forma se puede medir y recibir los parámetros de red del cliente y del servidor de forma no invasiva del sistema de ABYA.

La frecuencia de registro de los datos puede ajustarse antes de comenzar la partida y se mantiene constante durante la misma. Dado que el servidor de ABYA envía frames a sus clientes a razón de uno cada 16 ms, se opta por obtener datos cada 20 ms de forma de recibir al menos un frame en cada intervalo de tiempo.

La selección de las variables de interés a registrar e imprimir en los *logs* se realiza en principio en colaboración con el equipo de ABYA, basándose en su experiencia de trabajo y en la documentación analizada por el grupo de proyecto. Posteriormente, a medida que se avanza en la experimentación, este conjunto se actualiza de manera dinámica y según la demanda. Esto se realiza con el objetivo de converger en un conjunto de variables identificadas como relevantes para esta investigación. En la Tabla 5.1 se puede encontrar una lista con las variables analizadas.

Del conjunto total de variables registradas, tres de ellas adquieren una importancia significativa en la implementación del algoritmo desarrollado. A continuación, se proporcionará una explicación más detallada sobre estas variables:

- *Average Network Latency* (*AvgNetLat*)
  Esta variable mide el retraso experimentado por los paquetes desde que son enviados por el servidor hasta que son recibidos por el cliente. Se calcula dividiendo el retraso total de todos los paquetes recibidos por el número de paquetes recibidos durante el intervalo de medición, como se muestra en la Ecuación 5.1.

$$AvgNetLat = \frac{\sum(T_{llegada} - T_{salida})}{Paquetes\ recibidos} \tag{5.1}$$

- *Estimated Bandwidth* (*estBW*)
  Esta métrica se calcula como la suma de todos los bits recibidos dividida por la duración de la ventana de medición, como se muestra en la Ecuación

Tabla 5.1: Parámetros registrados durante cada partida.

Variables	**Descripción**
Received Bytes	Número de bytes recibidos
Received Packets	Número de paquetes recibidos
Frames Lost	Número de frames perdidos
Frames Received	Número de frames recibidos
Out of order packets	Paquetes fuera de orden
Out of order bytes	Bytes fuera de orden
Average Video Latency	Latencia promedio de video
Average Network Latency	Latencia promedio de red
Time Stamps	Hora de cada log en el servidor
Last Received Packet Number	Último número de paquete recibido
Estimated Bandwidth	Ancho de banda estimado
ANTP Clock	Hora NTP en el cliente
Local PC Time Now	Hora local del PC del cliente
Max Delay Variation	Informa si se envió un frame de tipo I
Average Bitrate	Tasa de codificación del códec

5.2. Se debe tener en cuenta que esta estimación no refleja la capacidad real del canal de comunicación, sino el ancho de banda de los flujos de audio y video recibidos por el cliente.

$$estBW = \frac{\sum Bits\ recibidos}{Duración\ de\ la\ ventana} \tag{5.2}$$

- *Average Bitrate* (*AvgBitrate*)
  Este parámetro indica la tasa de codificación a la que opera el códec de video en el servidor. Puede tomar valores superiores a 1 Mbps y ser modificado durante el transcurso de una partida. Su valor inicial es configurable desde un archivo externo.

5.4. Modificación de parámetros en el servidor de ABYA

Durante la etapa de implementación del algoritmo en tiempo real, fue necesario realizar un monitoreo periódico de ciertas variables, evaluar indicadores y, basándose en ellos, ajustar la tasa de codificación de video, o *AvgBitrate*, en el servidor. Es importante tener en cuenta que cualquier modificación incorrecta en esta configuración podía afectar el servicio proporcionado a los suscriptores de la

empresa. Por esta razón, se optó por trabajar con una interfaz de comunicación restringida y controlada para acceder al servidor.

La versión de la aplicación que la empresa ha puesto a disposición del equipo de proyecto incorpora un mecanismo para modificar la tasa de codificación. Esta acción se realiza de manera manual, mediante el uso de las teclas F8 y F9 del teclado del cliente. Como se mencionó anteriormente, la magnitud del incremento o decremento en la tasa de codificación, entre otras variables, pueden configurarse previamente desde un archivo externo. Estas variables son especificadas en la Tabla 5.2. Los valores definidos se mantendrán constantes durante el transcurso de la partida, lo que significa que solo se podrán aumentar o disminuir la tasa en múltiplos de estas cantidades predefinidas.

Tabla 5.2: Parámetros configurables desde un archivo externo en el computador del cliente.

Parámetro	**Descripción**	**Valor configurado**
initial_bitrate	Tasa inicial de codificación	10 Mbps
up_bitrate_step	Incremento de la tasa de codificación	1 Mbps
down _bitrate_step	Decremento de la tasa de codificación	0.5 Mbps
stats_log_frequency	Frecuencia de registro.	20 ms

5.5. Estructura general del programa implementado

El programa desarrollado tiene dos objetivos principales. En primer lugar, se busca predecir las degradaciones visuales y, en segundo lugar, ajustar la tasa de codificación de video en el servidor de forma de evitar estas degradaciones o reducir su impacto. Además, se pretende aumentar la tasa de codificación para aprovechar al máximo el ancho de banda disponible en la red. El funcionamiento general del algoritmo, sus bases teóricas y su implementación se describen en detalle en los Capítulos 6 y 7. A continuación, se presenta una síntesis de los principales módulos del programa.

Al comienzo del programa, se crea un objeto de tipo cola FIFO (*First In, First Out*) utilizando la biblioteca *collections* de Python. Esta cola se utiliza como una ventana móvil y se inicializa con los valores de la variable *AvgNetLat*, los cuales se extraen de las primeras 25 líneas del registro generado durante el lanzamiento del juego. A partir de ese momento, el tamaño de la cola se mantiene constante durante toda la ejecución del programa. Además, se crean e inicializan variables adicionales que se utilizarán para predecir degradaciones y ajustar la tasa de codificación del códec.

Una vez que se ha inicializado la ventana móvil, el programa actualiza su contenido a medida que avanza el tiempo durante la partida. Con cada nueva entrada,

se ejecuta el algoritmo desarrollado y, en caso de que se produzca una predicción de degradación de video, se envía una instrucción al servidor para reducir la tasa de codificación. Por otra parte, si no se producen predicciones durante un período de tiempo predefinido, el algoritmo solicitará al servidor que aumente la tasa de codificación del códec de video.

Una vez que se completa una partida, se detiene la escritura en el *log* generado. El programa está diseñado para detectar esta situación y, si se mantiene durante al menos 5 segundos, se cierra automáticamente y muestra las trazas de los principales parámetros del sistema en formato gráfico. Esto permite visualizar y analizar de manera más conveniente la evolución de dichos parámetros durante la partida.

En la figura 5.2 se muestra el diagrama de flujo del algoritmo desarrollado, el código del programa completo implementado en python, se puede consultar en el Apéndice E.

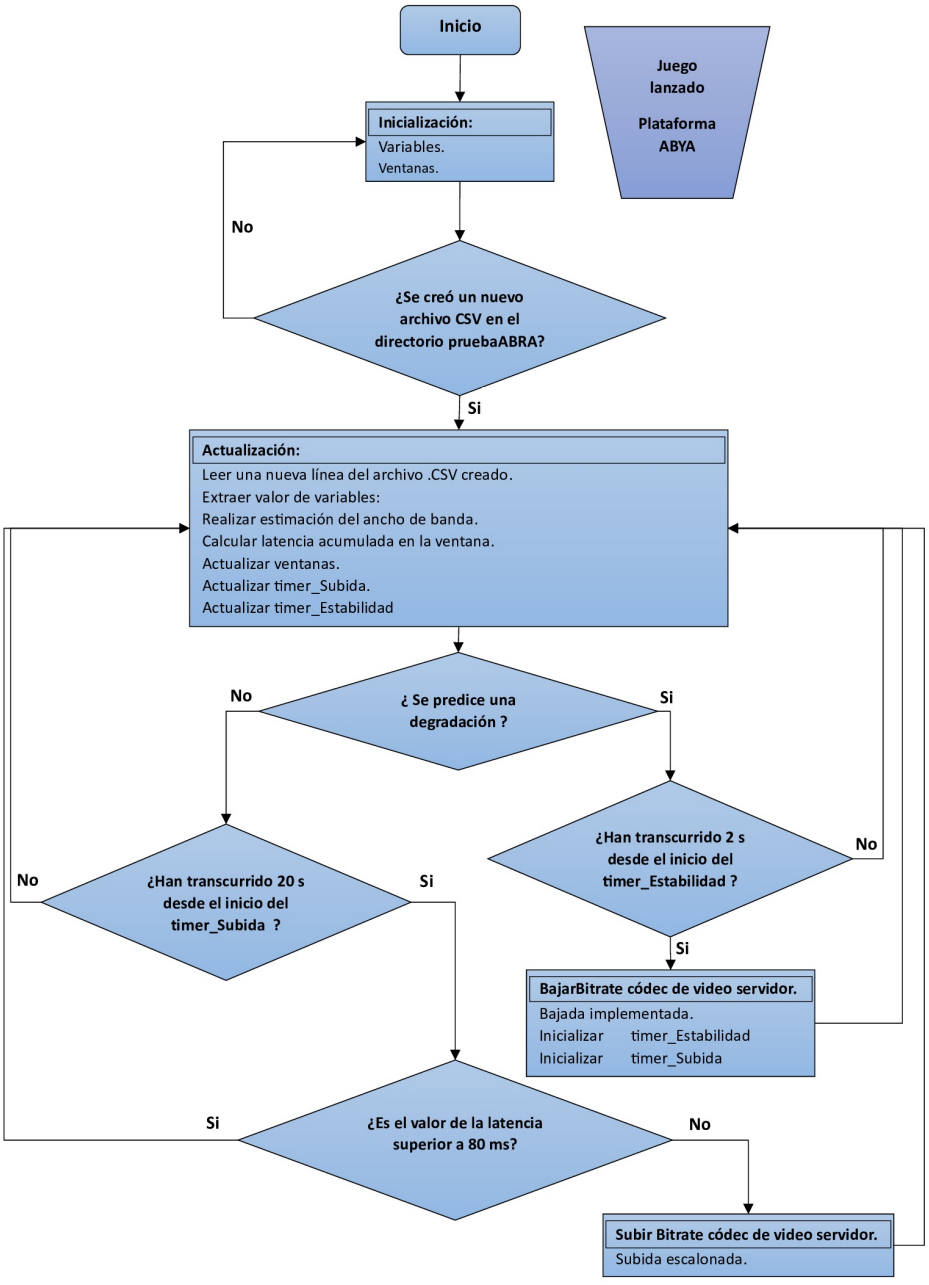


Figura 5.2: Diagrama de flujo algoritmo implementado.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Predicción de pérdidas

6.1. Implementación

Las degradaciones de video ocurren cuando se pierde un frame. Mediante el análisis de los logs obtenidos y diversas métricas de red, fue posible corroborar la idea presente en la bibliografía consultada de que existe una fuerte correlación entre la pérdida de frames y la latencia de red, observando que la latencia aumenta bruscamente justo antes de que ocurra la pérdida en todos los casos. La Figura 6.1 muestra un caso típico. En la misma se puede observar que la latencia de los paquetes recibidos aumenta repentinamente y, poco después, se pierden varios frames.

Basándonos en estos hechos, hemos desarrollado un programa predictor de pérdidas que permite detectar degradaciones en tiempo real. Durante el juego, este programa se ejecuta en paralelo en la computadora del cliente y monitoriza los registros en tiempo real para anticipar cualquier degradación inminente. Para lograrlo, implementamos ventanas móviles que calculan la latencia acumulada y estiman el ancho de banda disponible. Optamos por utilizar ventanas con sumatoria de valores de latencia para evitar falsas alarmas de degradación. Nuestras pruebas revelaron que los valores individuales de alta latencia u *outliers* no siempre indican una disminución en la calidad. En cambio, observamos que la acumulación de latencia alta en forma de rampas, como se muestra en la Figura 6.1, representa una clara indicación de degradación en la calidad del video.

El sistema utiliza ventanas de 500 ms de duración que siguen una estructura de cola tipo FIFO. Durante este periodo, se suman los valores de latencia cada 20 ms. A través de pruebas realizadas en diferentes juegos y condiciones de red, se ha optimizado un umbral para la latencia acumulada. Cuando el programa detecta que la acumulación de latencia supera este umbral, se activa una alarma de degradación, alertando al algoritmo de ajuste de bitrate sobre la presencia de una degradación inminente. Los resultados obtenidos se detallan en la siguiente sección.

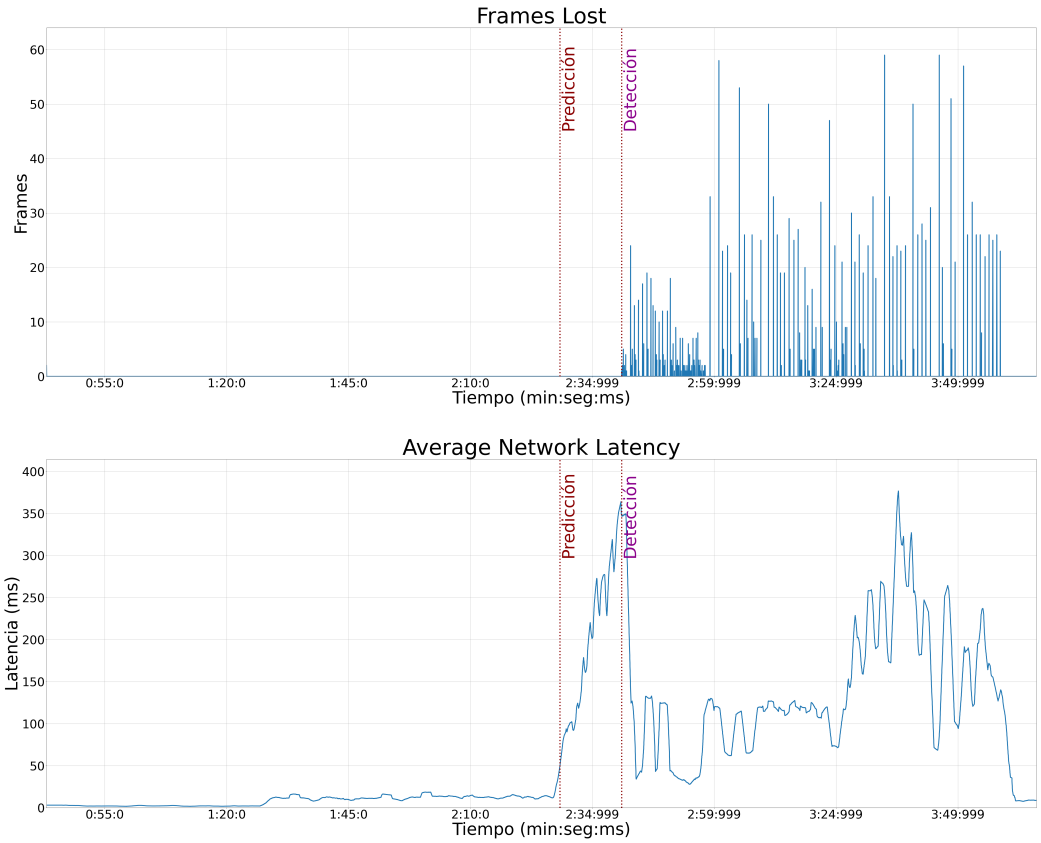


Figura 6.1: Arriba: frames perdidos. Abajo: latencia media de red. En ambos casos se destacan los instantes de predicción y detección de degradaciones.

6.2. Resultados

El algoritmo fue probado en un total de 183 partidas de 15 juegos diferentes. Estas partidas se realizaron con una latencia de base de hasta 7 ms. Tanto el servidor como los clientes se encontraban en la ciudad de Montevideo, Uruguay.

En la Tabla 6.1 se detallan los juegos utilizados, incluyendo sus géneros, descripciones breves y abreviaturas de los títulos. Se seleccionaron juegos dinámicos para poner a prueba el algoritmo, ya que requieren una conexión estable. Los géneros evaluados abarcaron RPG, FPS, carreras y luchas. Estas categorías son amplias y cuentan con numerosos subgéneros y combinaciones. No se incluyeron juegos de estrategia en tiempo real RTS en las pruebas, dado que suelen tener requisitos de ancho de banda más bajos debido a la menor capacidad de respuesta requerida por los usuarios. Por ende, se considera que las predicciones exitosas obtenidas para los juegos RPG y FPS, que demandan una mayor capacidad de respuesta, también serán aplicables a los juegos RTS.

La Tabla 6.2 presenta los resultados de las partidas realizadas en los 15 títulos, utilizando algunas de las configuraciones de red mencionadas en el Capítulo 4.

Tabla 6.1: Lista de juegos probados

Juego	Género	Breve descripción	Abreviatura
Alien Rage	FPS	Juego FPS ambientado en el espacio	AR
Cyber Hook	FPS	Juego de parkour 3D en primera persona	CH
Classic Racers	Carreras	Juego de carrera con autos clásicos	CR
Extinction	RPG	RPG de combate con espadas	EXT
Gravel	Carreras	Juego de carreras todoterreno	GRA
Grip	Carreras	Juego de carreras de combate futurista	GRI
King of Fighters XIII	Lucha	Juego de lucha de la serie King of Fighters	KOF
Lord of the Fallen	RPG	RPG ambientado en un mundo de fantasía medieval	LOTF
MXGP 2020	Carreras	Juego oficial del Campeonato Mundial de Motocross	MXGP
Outcast: Second Contact	RPG	RPG ambientado en un mundo alienígena	OSC
Stay Safe	Carreras	Juego de carreras con mecánica de “morir y volver a intentar”	SS
Styx: Master of Shadows	RPG	RPG ambientado en un universo de fantasía oscura	STYX
Tandem	RPG	Juego de plataformas con elementos de rompecabezas	TAN
V-Rally 4	Carreras	Juego realista de carreras de rally	VR4
WRC 8	Carreras	Juego oficial del Campeonato Mundial de Rally	WRC

A continuación, analizaremos detalladamente una partida en particular para una mejor comprensión de los datos recopilados.

Tabla 6.2: Resultados obtenidos con quince juegos diferentes probados en cuatro condiciones de red distintas. Los resultados completos por partida se encuentran en el Apéndice A.

Nombre	Cantidad de partidas	Cantidad total de degradaciones	Cantidad de Predicciones	Condiciones emuladas
AR	7	11	12	A-D
CH	20	32	37	A-B-C-D
CR	20	37	39	A-B-C-D
EXT	7	8	11	A-B
GRA	7	7	8	A-B
GRI	20	29	32	A-B-C-D
KOF	20	39	43	A-B-C-D
LOTF	20	25	25	A-B-C-D
MXGP	7	10	10	A-D
OSC	7	14	18	B-D
SS	7	8	9	A-B
STYX	7	10	13	A-C
TAN	7	17	20	A-B
VR4	7	7	7	A-B
WRC	20	15	16	A-B-C-D
TOTAL	183	269	300	

La Figura 6.2 muestra los datos de una partida del juego TAN bajo la condición de red de descenso abrupto, tal como se define en el Capítulo 4. Como se mencionó anteriormente, las degradaciones visuales están correlacionadas con la pérdida consecutiva de frames. Según la información presentada en el gráfico, se puede concluir que se produjeron dos instancias de degradación visual durante la partida.

Como se discutió en la sección anterior, el algoritmo realiza predicciones de pérdida consecutiva de frames cada vez que la latencia acumulada en una ventana de tiempo móvil supera un umbral determinado. Esta información se confirma en el gráfico central de la Figura 6.2, donde se muestran las dos predicciones realizadas durante la partida. Para contrastar estas predicciones, se utiliza la Tabla 6.2, que muestra las degradaciones visuales que efectivamente ocurrieron.

Finalmente, los resultados se resumen en la Tabla 6.3 y se expresan en forma de porcentajes. De un total de 183 partidas jugadas con 15 títulos diferentes y 4 escenarios de red, el algoritmo realizó 300 predicciones de un total de 269 degradaciones visuales observadas. Esto implica que no hubo degradaciones imprevistas (0 %) y una tasa de predicción de falsos positivos del 10.3 %.

Tabla 6.3: Resumen de los resultados obtenidos con el algoritmo de predicción de pérdidas.

	Número total de predicciones	Predicciones exitosas	Degradaciones no previstas	Falsos positivos
Cantidad	300	269	0	31
Porcentaje	100 %	89.7 %	0 %	10.3 %

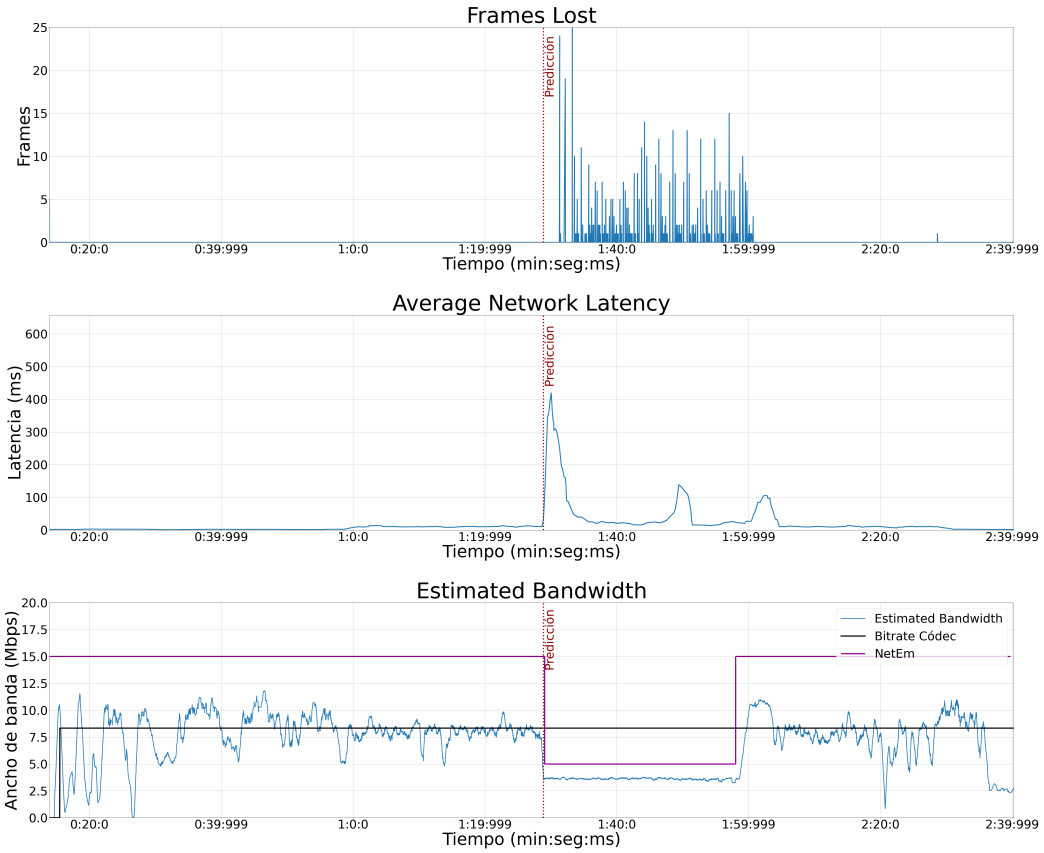


Figura 6.2: Arriba: frames perdidos, predicción y detección de degradaciones. Centro: latencia media de red. Abajo: ancho de banda estimado recibido por el cliente, valor medio del bitrate del códec y condición de red emulada con NetEm.

Es importante destacar que todas las degradaciones visuales fueron predichas exitosamente. En esta aplicación, es preferible tener una pequeña cantidad de falsos positivos que perder algún evento real de degradación, ya que esto ocasionaría una disminución de la QoE del usuario [5].

La investigación desarollada, el algoritmo implementado y los resultados obtenidos que se presentan en este capítulo fueron recopilados por el equipo en un artículo llamado *"Prediction of Video Quality Degradation on a Cloud Gaming Platform"*. Este trabajo fue presentado en el congreso internacional *"IEEE International Symposium on Broadband Multimedia Systems and Broadcasting 2023"* (BMSB 2023) que se realizó desde el 14 al 16 de junio en Beijing, China. [28]

Capítulo 7

Ajuste de bitrate automático

7.1. Bajada de bitrate

7.1.1. Implementación

El siguiente objetivo de este proyecto consiste en lograr un ajuste automático en la tasa de codificación del códec de video, con el fin de evitar degradaciones visuales. Durante las pruebas de predicción de degradaciones realizadas, se constató que antes de que se produzca pérdida de frames, la tasa de bits recibidos obtenida a partir de la variable *estBW* (cantidad de bits recibidos por el cliente una ventana de tiempo) disminuía considerablemente. Esto además explica el aumento de la latencia en este instante. Por esta razón, se implementó un estimador que se basa en el valor de la variable *estBW* en el momento en que se predice una pérdida. Esta nueva implementación requirió realizar pruebas de ajuste y validar estadísticamente el algoritmo.

El funcionamiento para predecir degradaciones es el mismo se explicó en la sección anterior. Instantáneamente, luego de que el algoritmo predice que se producirá una degradación, se ejecuta la función que estima el bitrate adecuado que debe enviar el servidor. En esta función, se intenta reducir la tasa a un bitrate que permita mantener la jugabilidad en la partida y que, al mismo tiempo, sea lo más alto posible dadas las condiciones del canal.

Para lograr esto, nuevamente se utiliza la variable *estBW* como referencia y se solicita al códec que disminuya el valor del bitrate de codificación por debajo de *estBW*. Se estableció un margen de 1.5 Mbps entre el *estBW* medido antes de la predicción, y el nuevo bitrate que debe ser enviado. Este valor se determinó empíricamente después de varias pruebas, donde se observó que un margen más pequeño no evitaba las degradaciones visuales y un margen más grande mantenía la jugabilidad pero con un alto costo de resolución del juego. La estimación del nuevo bitrate se realiza en pasos discretos de 0.5 Mbps mediante la siguiente Ecuación:

$$\text{cantidad\_Bajadas} = \left\lfloor \frac{\text{AvgBitrate} - \text{estBW} + \text{margen}}{\text{down\_bitrate\_step}} \right\rfloor \tag{7.1}$$

donde *margen* es el margen definido de 1.5 Mbps, y *down_bitrate_step* es el tamaño del paso discreto en que se disminuye la tasa de codificación. El resultado de la ecuación se trunca para determinar cuántos escalones discretos de 0.5 Mbps se debe disminuir el bitrate de codificación del servidor. Como se menciona en el Capítulo 5, esta reducción de la tasa solo puede realizarse mediante acumulación de eventos de teclado, por lo que para automatizar esta tarea dentro del programa implementado se utiliza una biblioteca especializada de manejo de teclado llamada *keyboard-Python*.

En la Figura 7.1 se puede observar que, después de una predicción, el algoritmo reduce automáticamente el bitrate del códec, evitando así las degradaciones. Es de notar que aquí ya no hay frames perdidos, a pesar de las condiciones de la red. Asimismo, se puede apreciar que la latencia vuelve a su estado inicial, lo que mantiene la jugabilidad en la partida. La prueba realizada corresponde a la condición de Descenso Intermedio.

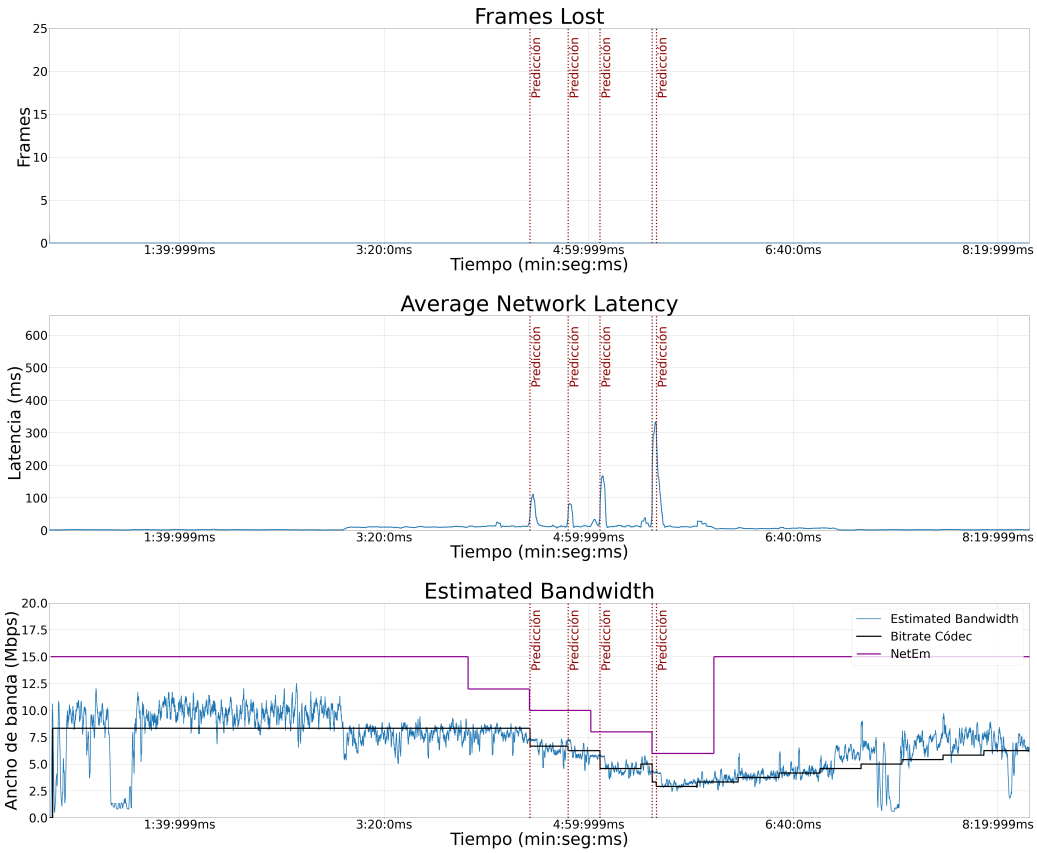


Figura 7.1: Arriba: frames perdidos y predicción de degradaciones. Centro: latencia media de red. Abajo: ancho de banda estimado recibido por el cliente.

7.1.2. Resultados

El algoritmo de ajuste automático de bitrate se probó en un total de 158 partidas, en 9 juegos de los 15 utilizados previamente para la predicción de degradaciones. Estas 158 partidas se jugaron con una latencia base de hasta 7 ms. Tanto el servidor como los clientes se encontraban en la ciudad de Montevideo, Uruguay. En la Tabla 7.1 se presentan los juegos, los cuales ya fueron descriptos en la Sección 6, junto con los resultados de cancelaciones de pérdidas de frames efectivas.

Tabla 7.1: Resultados obtenidos para la bajada del bitrate con nueve juegos diferentes probados en las seis condiciones de red mencionadas en el Capítulo 4. Los resultados completos por partida se encuentran en el Apéndice B.

Nombre	Cantidad de partidas	Cantidad total de predicciones	Cancelaciones de pérdida efectivas
CH	18	45	41
CR	15	42	28
EXT	18	35	33
GRA	18	27	23
GRI	18	32	31
KOF	18	43	34
LOTF	18	27	23
MXGP	18	33	29
TAN	17	47	39
TOTAL	158	331	281

Los juegos fueron probados en las seis condiciones de red presentadas en el Capítulo 4. Los resultados generales se muestran en la Tabla 7.2, donde se destaca que el algoritmo logró evitar las pérdidas de frames en un 84.9 % de los casos. Además, el algoritmo permitió seguir jugando las partidas el 100 % de las veces en todas las condiciones. Este último resultado es destacable, ya que durante las pruebas de predicción de degradaciones, donde no se aplicaba ningún algoritmo de ajuste, la calidad de la partida se vio deteriorada. En particular, en varias ocasiones, no era posible retomar la partida a pesar de retomar condiciones favorables de red.

El 15.1 % restante de cancelaciones no efectivas, en su gran mayoría (80 %), ocurrieron en situaciones en las que la condición de red requería que el algoritmo redujera el bitrate en más de 5 Mbps. En estas condiciones, a pesar de realizarse una predicción efectiva, no se logra disminuir el bitrate requerido a tiempo, lo que resultó en pérdida de frames. El promedio de la duración de las pérdidas en estas situaciones fue de 4.6 segundos, después de lo cual el juego pudo continuar normalmente con el bitrate adecuado.

Tabla 7.2: Resumen de los resultados obtenidos con el algoritmo de bajada de bitrate.

	Número total de predicciones	Cancelaciones de pérdida efectivas	Cancelaciones de pérdida no efectivas
Cantidad	331	281	50
Porcentaje	100 %	84.9 %	15.1 %

Por otra parte, en las partidas en las que condiciones de red requerían una reducción de hasta 4 Mbps, el algoritmo mostró una alta efectividad. Además, en las condiciones de red donde se mantuvo durante un tiempo prolongado un bitrate bajo en comparación con el inicial, el algoritmo logró mantener la jugabilidad en todas las ocasiones.

Finalmente, se probó comenzar la partida en condiciones de ancho de banda limitado. Se observó que el algoritmo presentó un buen desempeño en estas condiciones. Durante el inicio del juego, se detectaron incrementos en la latencia que superaban el umbral, lo que resultó en pérdidas de frames, como se muestra en la Figura 7.2. Sin embargo, estas pérdidas solo ocurrieron mientras se estaba cargando el juego y no afectaron la jugabilidad.

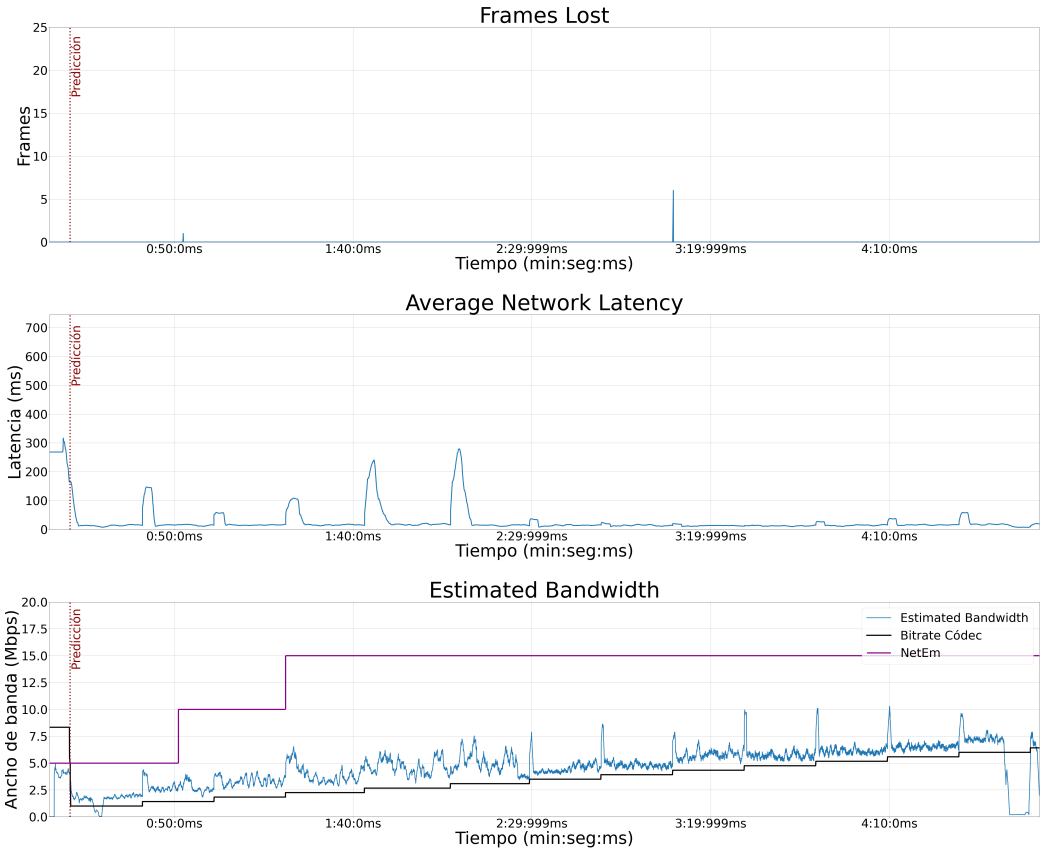


Figura 7.2: Comportamiento del sistema al iniciar la partida con un bitrate bajo.

7.2. Subida de bitrate

En una perspectiva general, el ajuste de bitrate consiste en maximizar la tasa de codificación de video en el servidor, sujeto a las restricciones de que el algoritmo prediga la ocurrencia de una degradación o que el sistema haya alcanzado la máxima tasa posible de transmisión dependiendo del servicio contratado.

El caso en el que durante la subida el algoritmo predice la ocurrencia de pérdidas de frames de video, lo cual conllevaría a degradaciones visuales y requiere reducir la tasa de codificación de video, fue tratado en secciones anteriores. Por lo tanto, esta sección se centrará en la acción de aumentar la tasa.

7.2.1. Implementación

El funcionamiento del servidor permite configurar, antes de iniciar una partida, el bitrate inicial del codificador de video a través de un archivo externo en el computador del cliente. Durante nuestras pruebas, se configuró este valor en 8,3 Mbps. En la Figura 7.3, se muestra la evolución de la tasa de codificación de video

durante el comienzo de una partida, y se puede observar cómo se mantiene en el valor configurado durante los primeros instantes.

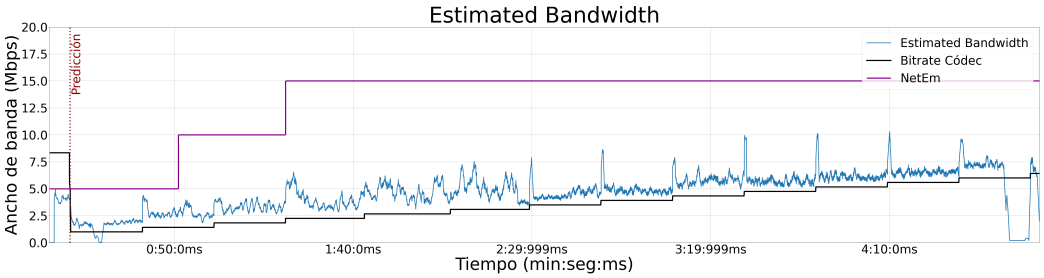


Figura 7.3: Evolución del bitrate de codificación del video.

La Figura 7.4 muestra un escenario típico del comportamiento del algoritmo durante la ejecución de una partida. En el momento en que se produce una predicción, se inicia un temporizador. Si no se produce ninguna predicción antes de transcurrir 20 segundos, se incrementa el valor de la tasa de codificación en 0.5 Mbps y se reinicia el temporizador. Por el contrario, si ocurre una nueva predicción durante ese tiempo, se implementa el algoritmo de reducción de bitrate explicado anteriormente. Se eligió el valor de 20 segundos para tener un margen de espera en caso de posibles predicciones de degradación, y no tener que bajar aún más abruptamente el bitrate luego de una subida.

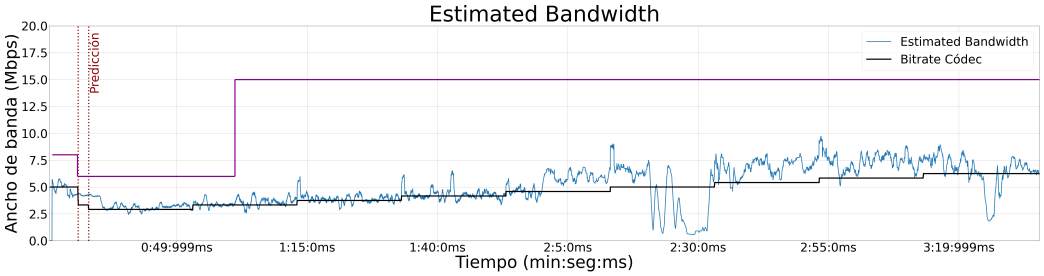


Figura 7.4: Comportamiento del algoritmo ante la ocurrencia de predicciones de degradación.

Se utilizó una estrategia de incremento lineal escalonado para la implementación de la subida de bitrate. Esta decisión se tomó debido a la falta de información sobre el ancho de banda del canal, lo cual dificulta la elaboración de un estimador para determinar la disponibilidad de ancho de banda del sistema.

El tiempo transcurrido t_{subida} desde que el algoritmo comienza su proceso de subida de la tasa de codificación, con una tasa r_{actual}, hasta alcanzar su valor óptimo $r_{óptima}$, es proporcional al número de incrementos realizados. Esto se puede calcular utilizando la Ecuación 7.2:

$$t_{subida} = 20 \cdot \frac{r_{óptima} - r_{actual}}{0{,}5Mbps} \tag{7.2}$$

7.2.2. Resultados

A continuación, se presentan los resultados obtenidos del algoritmo en cuanto a la subida efectiva de la tasa de codificación del códec. Para evaluar el desempeño del algoritmo en esta tarea, se utilizaron las mismas 158 partidas que se emplearon para evaluar el desempeño en la cancelación de pérdidas. Los resultados se detallan en las Tablas 7.3 y 7.4.

Tabla 7.3: Resultados obtenidos para la subida de bitrate con nueve juegos diferentes probados en las seis condiciones de red mencionadas en el Capítulo 4.

Nombre	Cantidad de partidas	Cantidad total de subidas efectivas
CH	18	18
CR	15	18
EXT	18	18
GRA	18	18
GRI	18	18
KOF	18	18
LOTF	18	18
MXGP	18	18
TAN	17	18
TOTAL	158	158

Tabla 7.4: Resumen de los resultados obtenidos

	Número total de partidas	Subidas efectivas	Subidas no efectivas
Cantidad	158	158	0
Porcentaje	100 %	100 %	0 %

Como se puede observar en las Tablas 7.3 y 7.4, los resultados en la subida del bitrate fueron altamente satisfactorios. En todas las pruebas realizadas, después de reducir el bitrate para evitar las pérdidas de frames, el algoritmo logró aumentar efectivamente el bitrate de codificación, siempre y cuando hubiera ancho de banda disponible en el canal.

7.3. Sistemas con alta latencia base

En esta sección, se presentan los resultados del algoritmo en condiciones de red donde la latencia base es mayor que las condiciones previamente emuladas. Se utilizó el mismo algoritmo que se empleó en las secciones 7.1 y 7.2. El objetivo fue evaluar el desempeño del algoritmo en situaciones donde el cliente y el servidor se encuentran separados por más de mil kilómetros. Se tomó como ejemplo el caso de un cliente que juega en São Paulo, Brasil, y se implementó una latencia base de 50 ms. Esta latencia adicional se simuló utilizando el software NetEm, como se menciona en la Sección 4.

En las Tablas 7.5 y 7.6 se presentan los resultados obtenidos en la cancelación de pérdidas. Por otro lado, los resultados de las subidas efectivas de bitrate se muestran en las Tablas 7.7 y 7.8. Los juegos se probaron en las condiciones A, B, C y D de la Sección 4, con una adición de 50 ms de latencia en cada condición. La Figura 7.5 muestra un ejemplo de una partida jugada en la condición D, donde se puede observar que la latencia base es superior a los casos anteriores.

Tabla 7.5: Resultados obtenidos con seis juegos diferentes probados en cuatro condiciones de red mencionadas en el Capítulo 4. Los resultados completos por partida se encuentran en el Apéndice C.

Nombre	Cantidad de partidas	Cantidad total de predicciones	Cancelaciones de pérdida efectivas	Condiciones emuladas
CH	4	10	9	A -B - C - D
EXT	4	9	9	A - B - C - D
GRA	4	6	5	A - B - C - D
GRI	4	7	7	A - B - C - D
LOTF	4	7	7	A - B - C - D
MXGP	4	7	6	A - B - C - D
TOTAL	24	46	43	

Tabla 7.6: Resumen de los resultados obtenidos con el algoritmo de bajada de bitrate.

	Número total de predicciones	Cancelaciones de pérdida efectivas	Cancelaciones de pérdida no efectivas
Cantidad	46	43	3
Porcentaje	100 %	93.5 %	6.5 %

Tabla 7.7: Resultados obtenidos en el caso de alta latencia base con seis juegos diferentes probados en cuatro condiciones de red mencionadas en el Capítulo 4.

Nombre	Cantidad de partidas	Cantidad total de subidas efectivas	Condiciones emuladas
CH	4	4	A-B-C-D
EXT	4	4	A-B-C-D
GRA	4	4	A-B-C-D
GRI	4	4	A-B-C-D
LOTF	4	4	A-B-C-D
MXGP	4	4	A-B-C-D
TOTAL	24	24	

Tabla 7.8: Resumen de los resultados obtenidos con el algoritmo de subida de bitrate.

	Número total de partidas	Subidas efectivas	Subidas no efectivas
Cantidad	24	24	0
Porcentaje	100 %	100 %	0 %

Las pruebas realizadas en este no fueron tan exhaustivas como las llevadas a cabo en las secciones 7.1 y 7.2. Sin embargo, los resultados obtenidos fueron alentadores y sugieren que el algoritmo podría implementarse de manera efectiva, mejorando la jugabilidad a pesar de tener condiciones de mayor latencia base.

En este caso, se demostró una mayor efectividad en comparación con el caso de menor latencia base, logrando un 93.5 % de cancelaciones efectivas. Como se mencionó anteriormente, estos resultados podrían variar ligeramente durante una validación más exhaustiva. La Figura 7.5 muestra cómo el algoritmo sigue siendo efectivo en la cancelación de pérdidas, a pesar de una latencia base más alta. En cuanto al aumento del bitrate, los resultados fueron consistentes con el caso de

menor latencia base, logrando aumentar efectivamente el bitrate el 100 % de las veces cuando el canal así lo permitía.

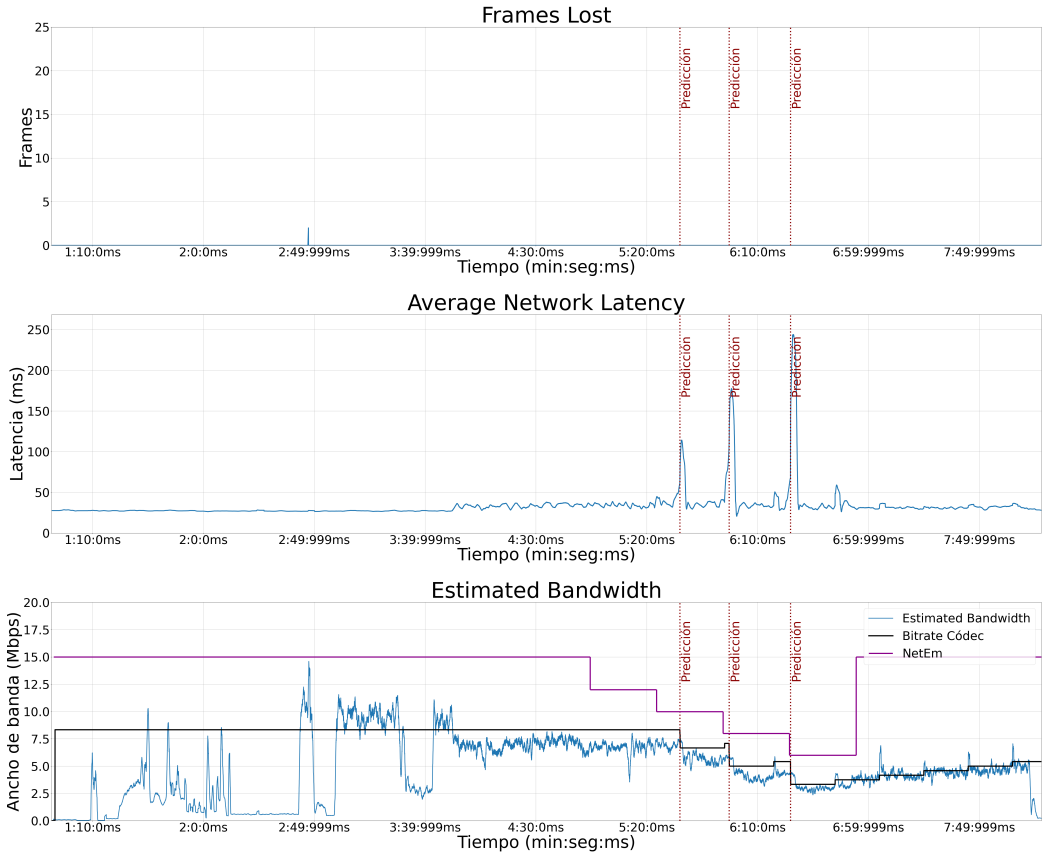


Figura 7.5: Arriba: frames perdidos y predicción de degradaciones. Centro: latencia media de red. Se puede ver que la latencia base es superior. Abajo: ancho de banda estimado recibido por el cliente.

Capítulo 8

Resultados generales

En esta sección, se realizará un análisis de los resultados generales del algoritmo, en contraste con los objetivos establecidos al inicio del proyecto. Este algoritmo implementado se puede apreciar en su totalidad en el Apéndice E. Además, se evaluará el cumplimiento de otros objetivos del proyecto, como la evaluación de la calidad de la experiencia y la mejora general en la jugabilidad de las partidas.

8.1. Predicción de pérdidas

El primer objetivo de este proyecto era lograr una correcta predicción de las pérdidas de paquetes en al menos el 90 % de los casos probados. A medida que se desarrolló el proyecto, se determinó que la predicción de pérdidas de paquetes estaba estrechamente relacionada con la predicción de degradaciones visuales. Como se vio, las degradaciones visuales son causadas por la pérdida de frames, como se mencionó en la Sección 6.

El algoritmo implementado para la predicción de degradaciones visuales fue sometido a pruebas en un total de 183 partidas, logrando predecir correctamente el 100 % de las degradaciones visuales, lo cual cumple el objetivo establecido de forma altamente satisfactoria. Sin embargo, se observó un 10.3 % de falsos positivos, es decir, predicciones en las que no se produjeron degradaciones visuales posteriormente. A pesar de esto, se logra mantener la jugabilidad en la partida en todo momento, ya que el algoritmo activa la alarma para reducir el bitrate y evitar posibles pérdidas, lo que resulta en una disminución temporal de la calidad del video que se recupera rápidamente si las condiciones del canal lo permiten.

El objetivo establecido por el cliente de detectar pérdidas con una anticipación de 5 ms también ha sido alcanzado. Aunque no se realizó un estudio detallado ni un análisis del promedio de tiempo de anticipación, se puede estimar a partir de la información de las gráficas obtenidas que las pérdidas fueron detectadas con al menos 500 ms de anticipación en todos los casos.

8.2. Ajuste automático de bitrate

El objetivo principal establecido al inicio del proyecto, relacionado con el ajuste del bitrate en la transmisión del servidor al cliente, fue lograr estimar el ancho de banda al cual limitar el bitrate. Este objetivo se ha cumplido al limitar el bitrate de codificación por debajo de la variable *estBW*, como se explica en el Capítulo 7. Al aplicar esta limitación, todas las partidas pudieron mantener la jugabilidad en la totalidad de la partida.

Las cancelaciones de pérdidas o degradaciones visuales alcanzaron una efectividad del 84.5 %. Aunque no se logró superar el objetivo del 90 % de efectividad, se observó que en el 80 % de los casos donde hubo pérdida de frames, la bajada necesaria era superior a 5 Mbps. Sin embargo, cuando la bajada requerida era de 4 Mbps o menos, la efectividad del algoritmo aumentó al 95.5 %. Esto indica que el algoritmo funciona bien en condiciones de red donde no hay disminuciones abruptas en el ancho de banda disponible. Incluso si el ancho de banda del canal fluctúa dentro del margen mencionado, el algoritmo es capaz de seguir la curva y mantener la jugabilidad sin interrupciones.

Asimismo, se observa que en los casos en que no fue posible evitar las pérdidas, la duración de la interrupción es en promedio de 4.6 segundos, siendo posible retomar la partida sin mayores complicaciones. Esta situación representa una mejora significativa en comparación con el sistema original, donde las pérdidas solían tener una duración superior a los 30 segundos e incluso en algunos casos el juego no podía ser retomado en absoluto, siendo necesario reiniciar la aplicación.

La subida del bitrate hasta alcanzar el valor inicial de la partida fue exitosa en el 100 % de las partidas jugadas. Como se describe en la Sección 7.2, no se dispone de una variable de referencia para estimar la tasa de bits a la que se puede aumentar. Se decidió incrementar el bitrate de forma lineal, añadiendo 1 Mbps cada 20 segundos. Se demostró que si hay suficiente ancho de banda disponible en el canal, el bitrate de transmisión puede aumentarse sin dificultades. En el caso de alcanzar un límite debido a restricciones en el ancho de banda del canal, el algoritmo demostró la capacidad de mantenerse en ese límite, fluctuando entre aumentos y disminuciones del bitrate, como se mencionó anteriormente.

Por último, el algoritmo demostró un buen desempeño en condiciones de red con una latencia base elevada, simulando un servidor a varios cientos de kilómetros del cliente. Al agregar una latencia adicional de 50 ms a las condiciones de red utilizadas en pruebas anteriores, el algoritmo logró una efectividad del 93.5 % en cancelaciones de pérdidas y, al igual que en las pruebas anteriores, un 100 % de subidas efectivas. Se resalta que, en las pruebas con una alta latencia base, no se realizaron alteraciones en ninguno de los parámetros del algoritmo.

8.3. Calidad de la experiencia

Otro objetivo del proyecto era que el algoritmo mejore la calidad de experiencia del jugador. Aunque no se realizaron mediciones de calidad basadas en estándares [19] [20], se obtuvo información subjetiva a través de las pruebas y experiencias de los participantes involucrados en el proyecto.

Se destaca como punto principal que el algoritmo implementado logró mantener la continuidad parcial y la jugabilidad de la partida en todas las pruebas realizadas. Aunque el algoritmo de ajuste automático de bitrate no pudo garantizar la continuidad total de la partida en algunos casos, sí mejoró significativamente la duración de los tiempos de corte. Esto permitió que todas las partidas pudieran retomarse después de esta breve interrupción, restablecer una jugabilidad aceptable y finalizar la partida de manera satisfactoria.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 9

Futuros trabajos

Este proyecto tiene varias líneas de trabajo que podrían continuar en el futuro. En primer lugar, se encuentra el estudio y mejora de la efectividad del algoritmo. La predicción de degradaciones visuales, como se explicó previamente, se basa en la medida de acumulación de latencia. Estas medidas pueden mejorarse al tener en cuenta, no solo el área bajo la curva de la latencia durante los últimos 500 ms, sino también al implementar otras funciones, como la medición de la pendiente. Esto ayudaría a reducir la cantidad de falsos positivos obtenidos.

Si bien la acumulación de latencia ha demostrado ser un factor relevante en la predicción de pérdidas de frames, es posible que existan otros parámetros de la red que puedan predecirlas con mayor precisión. Además, en la predicción se pueden aplicar técnicas de reconocimiento de patrones y procesamiento de señales que aún no se han utilizado, pero que podrían resultar muy útiles en esta problemática.

Otras mejoras que se pueden realizar en el algoritmo actual incluyen un mejor funcionamiento del ajuste de bitrate. En cuanto a la cancelación de pérdidas, se debe buscar mejorar la efectividad cuando es necesario reducir más de 5 Mbps instantáneamente para mantener la jugabilidad. Esta mejora puede ir de la mano con una mejora en la predicción, utilizando las técnicas mencionadas anteriormente. Una predicción más anticipada y prestando especial atención a los casos en los que se requiere una disminución abrupta del bitrate pueden proporcionar mejores resultados.

En relación a la subida del bitrate de codificación, se puede considerar reemplazar el modelo lineal actual por uno exponencial. De esta forma, a medida que el bitrate aumenta sin que se predigan pérdidas, el incremento del bitrate se volverá más abrupto. Esto permitiría alcanzar el valor de bitrate previo a la degradación en menos tiempo y mejorar la experiencia del jugador al aprovechar de manera más eficiente el canal disponible.

Finalmente, en cuanto a la calidad de experiencia, queda pendiente la realización de pruebas subjetivas interactivas para evaluar la calidad, siguiendo el estándar [19]. Estos valores obtenidos podrían ser comparados con los resultados obtenidos mediante el modelo presentado en [20].

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 10

Conclusiones

Las plataformas de CG han surgido como un medio prometedor para jugar a videojuegos de alta calidad en dispositivos de bajo costo. Sin embargo, esto también plantea varios problemas y desafíos, ya que estas plataformas dependen en gran medida del ancho de banda de la red que conecta los servidores con los clientes.

En este proyecto se comprobó que la latencia de la red puede ser utilizada como un predictor efectivo de las pérdidas de frames de video, que son la causa principal de las degradaciones visuales. En este contexto, se desarrolló un algoritmo en tiempo real para predecir futuras pérdidas de frames. Este algoritmo se ejecuta en el PC del cliente y puede integrarse fácilmente en la aplicación de la plataforma de CG. Se comprobó que el algoritmo logra predecir el 100 % de los eventos de degradación de video, aunque, también se detectó un 10.3 % de falsos positivos.

A su vez, se implementó un sistema de ajuste de bitrate que permite estimar la tasa a la cual se debe reducir el bitrate después de una predicción de pérdidas de frames. Esta estimación se basa en disminuir el bitrate un margen por debajo del valor actual recibido por el cliente, con el objetivo de evitar la saturación del canal y mantener una jugabilidad fluida en la partida. Una vez que se evitó la pérdida o se retomó la partida, el algoritmo comienza a aumentar linealmente el bitrate de codificación del servidor en intervalos de tiempo constantes. Este método de estimación de la tasa de codificación mostró una efectividad del 84.9 % en la cancelación de pérdidas y un 100 % de efectividad en el aumento del bitrate.

El algoritmo desarrollado se probó con 15 juegos de diferentes géneros, bajo 6 condiciones de red distintas, lo que resultó en más de 365 partidas jugadas. Aunque no se logró alcanzar la efectividad deseada al inicio del proyecto, creemos que es posible lograrlo mediante las acciones marcadas para el trabajo futuro. Además, se observó una notable mejora en la calidad de la experiencia, especialmente en la capacidad de reducir significativamente la duración de las pérdidas, lo que permitió una mejor continuidad en el juego y la posibilidad de finalizar las partidas en todos los casos.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice A

Predicción de pérdidas

Tabla A.1: Datos obtenidos durante la validación del algoritmo de predicción de pérdidas. Partidas 1, 2 y 3.

Juego	Cond.	Partida 1		Partida 2		Partida 3	
		Cant. degr.	Cant. pred.	Cant. degr.	Cant. pred.	Cant. degr.	Cant. pred.
AR	A	1	1	2	2	1	2
	D	3	3	1	1	2	2
CH	A	1	1	1	1	1	1
	B	2	3	1	2	1	1
	C	1	2	2	2	1	2
	D	2	2	2	2	2	2
CR	A	2	2	3	3	1	1
	B	3	3	3	4	1	1
	C	1	1	1	1	1	2
	D	2	2	3	3	2	2
EXT	A	1	1	1	1	1	1
	B	2	2	1	2	1	2
GRA	A	1	1	1	1	1	1
	B	1	1	1	1	1	2
GRI	A	1	1	1	1	1	1
	B	1	2	2	2	1	2
	C	1	1	1	1	1	1
	D	3	3	2	2	2	2
KOF	A	1	1	2	2	1	1
	B	2	2	2	2	2	2
	C	2	2	1	2	2	2
	D	4	4	3	3	3	4
LOTF	A	1	1	1	1	1	1
	B	1	1	1	1	1	1
	C	1	1	1	1	1	1

	D	2	2	2	2	2	2
MXGP	A	1	1	1	1	1	1
	D	2	2	2	2	2	2
OSC	B	2	3	1	2	1	2
	D	3	4	3	3	1	1
SS	A	1	1	1	1	1	1
	B	1	1	1	2	2	2
STYX	A	1	1	2	3	1	1
	C	1	3	1	1	1	1
TAN	A	2	2	2	3	2	2
	B	4	5	2	3	2	2
VR4	A	1	1	1	1	1	1
	B	1	1	1	1	1	1
WRC	A	1	1	1	1	1	1
	B	1	1	1	1	1	1
	C	1	1	1	1	1	1
	D	0	0	0	1	0	0

Tabla A.2: Datos obtenidos durante la validación del algoritmo de predicción de pérdidas. Partidas 4 y 5.

Juego	Cond.	Partida 4		Partida 5	
		Cant. degr.	Cant. pred.	Cant. degr.	Cant. pred.
AR	A	1	1	-	-
CH	A	2	3	1	1
	B	2	2	2	2
	C	2	2	1	1
	D	2	2	3	3
CR	A	1	1	2	2
	B	3	3	1	1
	C	1	1	1	1
	D	3	3	2	2
EXT	A	1	2	-	-
GRA	B	1	1	-	-
GRI	A	2	2	1	1
	B	1	1	1	2
	C	1	1	1	1
	D	3	3	2	2
KOF	A	1	1	1	1
	B	2	2	2	2
	C	1	1	1	1
	D	3	5	3	3

LOTF	A	1	1	1	1
	B	1	1	1	1
	C	1	1	1	1
	D	2	2	2	2
MXGP	A	1	1	-	-
OSC	D	3	3	-	-
SS	A	1	1	-	-
STYX	C	3	3	-	-
	B	3	3	-	-
VR4	B	1	1	-	-
WRC	A	1	1	1	1
	B	1	1	1	1
	C	1	1	1	1
	D	0	0	0	0

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Ajuste de bitrate automático

Tabla B.1: Datos obtenidos durante la validación del algoritmo de ajuste de bitrate. En rojo se encuentran los casos en que no se produjo una cancelación efectiva de pérdida de frames.

Juego	Cond.	Partida 1		Partida 2		Partida 3	
		Cant. pred.	Cant. baj. ef.	Cant. pred.	Cant. baj. ef.	Cant. pred.	Cant. baj. ef.
CH	A	1	1	2	2	1	0
	B	2	2	2	2	2	2
	C	3	3	4	4	3	3
	D	3	3	3	3	4	4
	E	3	3	2	1	3	3
	F	2	1	3	3	2	1
CR	A	2	1	3	2	3	1
	B	2	0	-	-	-	-
	C	3	3	3	3	-	-
	D	5	3	4	1	5	3
	E	3	3	3	3	1	1
	F	2	1	2	2	1	1
EXT	A	2	1	1	1	1	1
	B	2	2	2	2	2	2
	C	2	2	2	2	2	2
	D	3	3	2	2	2	2
	E	2	2	2	2	2	2
	F	2	1	2	2	2	2
GRA	A	1	1	1	1	1	0
	B	2	2	2	2	2	2
	C	2	2	1	1	1	1
	D	2	2	1	1	1	1
	E	1	0	2	1	2	1
	F	2	2	2	2	1	1
	A	1	1	1	1	1	1

GRI	B	2	2	2	2	2	2
	C	2	2	1	1	2	2
	D	2	2	2	2	2	2
	E	2	2	2	2	2	2
	F	2	1	2	2	2	2
KOF	A	1	1	3	2	1	0
	B	3	1	2	1	2	1
	C	3	3	2	2	3	3
	D	2	2	4	3	3	3
	E	2	2	3	2	3	2
	F	1	1	2	2	3	3
LOTF	A	1	0	1	1	1	1
	B	1	1	2	2	2	2
	C	1	1	2	2	1	1
	D	2	2	2	2	2	2
	E	2	1	2	1	2	1
	F	1	1	1	1	1	1
MXGP	A	1	0	1	0	1	0
	B	2	2	2	2	2	2
	C	1	1	2	2	2	2
	D	2	2	2	2	2	2
	E	2	1	2	2	2	2
	F	2	2	3	3	2	2
TAN	A	3	2	3	2	3	1
	B	3	3	3	1	-	-
	C	3	3	4	4	4	3
	D	3	3	4	4	2	2
	E	4	4	2	2	2	1
	F	1	1	2	2	1	1

Tabla B.2: Análisis de los casos en que no se produjo una cancelación efectiva de pérdida de frames.

Juego	Condición	Duración de la pérdida (s)	Cantidad de frames perdidos
CH	A	2.57	28
	E	0.89	12
	F	3.43	29
	F	2.24	62
CR	A	4.10	162
	A	1.89	136
	A	3.96	137
	F	6.40	205
EXT	A	1.99	48
	F	4.21	22
GRA	A	3.30	64
	E	4.45	73
	E	3.90	72
	E	3.21	28
GRI	F	0.12	4
KOF	A	2.70	34
	A	5.36	92
	E	3.35	54
	E	2.37	76
LOTF	A	0.45	10
	E	1.23	35
MXGP	A	3.60	88
	A	4.05	55
	E	3.82	68
TAN	A	0.35	15
	A	2.71	34
	A	5.35	79
	C	12.00	130
	E	8.25	72

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Ajuste de bitrate automático en sistemas con alta latencia base

Tabla C.1: Datos obtenidos durante la validación del algoritmo de ajuste de bitrate. En rojo se encuentran los casos en que no se produjo una cancelación efectiva de pérdida de frames. Se realizó una partida por cada condición de red emulada.

Juego	Cond.	Cant. pred.	Cant. baj. ef.
CH	A	1	0
	B	2	2
	C	4	4
	D	3	3
EXT	A	1	1
	B	2	2
	C	3	3
	D	3	3
GRA	A	1	0
	B	2	2
	C	1	1
	D	2	2
GRI	A	1	1
	B	2	2
	C	2	2
	D	2	2
LOTF	A	1	1
	B	2	2
	C	2	2
	D	2	2
	A	1	0

MXGP	B	2	2
	C	2	2
	D	2	2

Tabla C.2: Análisis de los casos en que no se produjo una cancelación efectiva de pérdida de frames.

Juego	Condición	Duración de la pérdida (s)	Cantidad de frames perdidos
CH	A	2.05	13
GRA	A	3.05	17
MXGP	A	3.74	59

Apéndice D

Emulación de condiciones

```
echo 'Ingresar interfaz downlink:'
read DL

echo 'Ingresar opción:'
echo '0: Configuración puente'
echo '1: Prueba A - Descenso Abrupto'
echo '2: Prueba B - Descenso Intermedio'
echo '3: Prueba C - Descenso Lento'
echo '4: Prueba D - Inestabilidad'
echo '5: Prueba E - Descenso Abrupto Largo'
echo '6: Prueba F - Comienzo Bajo Bitrate'
echo '7: Eliminar configuración puente'
read NUMBER

case $NUMBER in
  0)
    echo 'Configurando puente'
    echo 'Ingresar interfaz uplink:'
    read UL
    brctl addbr br1
    brctl setfd br1 0
    brctl addif br1 $DL
    brctl addif br1 $UL
    ifconfig br1 up
    for f in /proc/sys/net/bridge/bridge-*; do echo 0 > $f; done
    echo 'Listo'
    sh condiciones_ABRAGame.sh;;

  1)
    echo 'Realizando Prueba A: Descenso Abrupto'
    tc qdisc add dev $DL root netem rate 15Mbit;
```

```
    date; sleep 105
    tc qdisc change dev $DL root netem rate 5Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 15Mbit;
    date; sleep 105
    echo 'Listo'
    tc qdisc del dev $DL root netem;
    date
    sh condiciones_ABRAGame.sh;;

  2)
    echo 'Realizando Prueba B: Descenso Intermedio'
    tc qdisc add dev $DL root netem rate 15Mbit;
    date; sleep 60
    tc qdisc change dev $DL root netem rate 12Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 8Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 4Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 10Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 15Mbit;
    date; sleep 60
    echo 'Listo'
    tc qdisc del dev $DL root netem;
    date
    sh condiciones_ABRAGame.sh;;

   3)
    echo 'Realizando Prueba C: Descenso Lento'
    echo 'Esperar 60 segundos en 15Mbps'
    tc qdisc add dev $DL root netem rate 15Mbit;
    date; sleep 60
    tc qdisc change dev $DL root netem rate 12Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 10Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 8Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 6Mbit;
    date; sleep 30
    tc qdisc change dev $DL root netem rate 15Mbit;
    date; sleep 60
    echo 'Listo'
```

```
  tc qdisc del dev $DL root netem;
  date
  sh condiciones_ABRAGame.sh;;

  4)
  echo 'Realizando Prueba D: Inestabilidad'
  tc qdisc add dev $DL root netem rate 15Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 10Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 8Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 12Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 10Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 6Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 8Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 15Mbit;
  date; sleep 30
  echo 'Listo'
  tc qdisc del dev $DL root netem;
  date
  sh condiciones_ABRAGame.sh;;

5)
  echo 'Realizando Prueba E: Descenso Abrupto Largo'
  tc qdisc add dev $DL root netem rate 15Mbit;
  date; sleep 30
  tc qdisc change dev $DL root netem rate 5Mbit;
  date; sleep 180
  tc qdisc change dev $DL root netem rate 15Mbit;
  date; sleep 30
  echo 'Listo'
  tc qdisc del dev $DL root netem;
  date
  sh condiciones_ABRAGame.sh;;

6)
  echo 'Realizando Prueba F: Comienzo Bajo Bitrate'
  tc qdisc add dev $DL root netem rate 5Mbit;
  date; sleep 180
  tc qdisc change dev $DL root netem rate 10Mbit;
```

```
    date; sleep 30
    tc qdisc change dev $DL root netem rate 15Mbit;
    date; sleep 30
    echo 'Listo'
    tc qdisc del dev $DL root netem;
    date
    sh condiciones_ABRAGame.sh;;

  7)
    echo 'Eliminando configuración del puente'
    ifconfig br1 down
    brctl delbr br1
    echo 'Listo'
    sh condiciones_ABRAGame.sh;;

  *)
    echo  'Opción no válida';;


esac
```

Apéndice E

Algoritmo completo

```
import pandas as pd
from matplotlib import pyplot as plt
import keyboard
import os
import time
from collections import deque
import matplotlib.ticker as ticker

#-----------------------------------------------------------------
def ruta_archivo_csv(ruta_a_pruebaABRA):
    contenido = os.listdir(ruta_a_pruebaABRA)
    seGeneroArchivo = len(contenido)
    print(seGeneroArchivo,"Archivo en el directorio pruebaABRA")
    while seGeneroArchivo !=2:
        contenido = os.listdir(ruta_a_pruebaABRA)
        seGeneroArchivo = len(contenido)
    print("Ahora hay ",seGeneroArchivo,"archivos :",contenido)
    for archivo in contenido:
        if archivo != 'input.ini':
            a = 'C:\pruebaABRA'+ '\\'+archivo
    return a

#-----------------------------------------------------------------
def lee_UltimaLinea_csv(ruta_al_csv):
    data = open(ruta_al_csv)
    last_line = data.readlines()[-1]
    data.close()
    return last_line

#-----------------------------------------------------------------
def cambio_de_linea(ruta_csv):
    starTime = time.time()
```

```
    finArchivo = False
    last_line = lee_UltimaLinea_csv(ruta)
    last_last_line = last_line
    while last_last_line == last_line and (finArchivo == False):
        last_last_line = lee_UltimaLinea_csv(ruta)
        tiempo_trascurrido = time.time()-starTime
        finArchivo = (tiempo_trascurrido > 5)
    return [finArchivo,last_last_line]

#-----------------------------------------------------------------
def
↪  inicializacion_colaLatencia_Y_Area(last_line,cola_latencia,area):
    nombreLista = last_line.split(',')
    AverageNetworkLatency = int(nombreLista[9])
    cola_latencia.append(AverageNetworkLatency)
    area = area + AverageNetworkLatency
    return [cola_latencia,area]

#-----------------------------------------------------------------
def
↪  inicialización_colaEstimatedBandwidth(last_line,colaEstimatedBandwidth,EstBand):
   estBand = EstBand
   nombreLista = last_line.split(',')
   receivedBytes = int(nombreLista[0])
   estBand = receivedBytes*16 + estBand
   colaEstimatedBandwidth.append(receivedBytes*16)
   return [estBand,colaEstimatedBandwidth]

#-----------------------------------------------------------------
def actuaclización_cola_Latencia(last_line, colaLatencia , area):
    nombreLista = last_line.split(',')
    AverageNetworkLatency = int(nombreLista[9])
    sacoDLaCola = colaLatencia.popleft()
    nuevoEnCola = AverageNetworkLatency
    colaLatencia.append(nuevoEnCola)
    area = area + nuevoEnCola - sacoDLaCola
    return[colaLatencia,area]

#-----------------------------------------------------------------
def
↪  actualizacion_cola_estimatedBand(last_line,colaEstimatedBandwidth,EstBand):
    colaEStBand = colaEstimatedBandwidth
    nombreLista = last_line.split(',')
    estBand = EstBand
    nuevoByte = int(nombreLista[0])
```

```
    sacoEstBand = colaEStBand.popleft()
    colaEstBand.append(nuevoByte*16)
    estBand = estBand + nuevoByte*16-sacoEstBand
    return [estBand,colaEstBand]

#-----------------------------------------------------------------
def bajar_tasa(cantidad_bajadas):
    for q in range(cantidad_bajadas):
        keyboard.press("F8")
        time.sleep(0.02)

#-----------------------------------------------------------------
def graficas(tiempos_de_prediccion):
    df = pd.read_csv(ruta)
    df =df.assign(miEstimaciónAnchoDeBand =0)
    filtro = df['estimatedBandWidth'] > 0
    df=df[filtro]
    df['miEstimaciónAnchoDeBand']=
    ↪  df['estimatedBandWidth'].rolling(25).mean()

    def formatoHora(entrada):
        import math
        a= entrada
        b = (a/3600000)
        hora = math.floor(b)
        c= (b-hora)*60-math.floor((df.iloc[0,7]/3600000-
        ↪  math.floor(df.iloc[0,7]/3600000))*60)
        minuto = math.floor(c)
        d =(c-minuto)*60
        segundo = math.floor(d)
        milis=math.floor((d-segundo)*1000)
        horaf =str(str(minuto) + ":"+ str(segundo) +  ":"+
        ↪  str(str(milis) + "ms"))
        return horaf
    for i in range(100):
        df.iloc[i,18]= df.iloc[100,18]

    df['AverageNetworkLatency']=
    ↪  df['AverageNetworkLatency'].rolling(25).mean()
    for i in range(100):
        df.iloc[i,9]= df.iloc[100,9]

    plt.rcParams["figure.figsize"] = [40, 40]
    plt.rcParams["figure.autolayout"] = True
    fig, axs = plt.subplots(3)
```

```
    [fin,y]= df.shape
maximaLatencia = max(df['AverageNetworkLatency'].max(),0)
maximoAnchoDeBanda
↪  =max(df['AvgBitrate'].max(),df['miEstimaciónAnchoDeBand'].max())

axs[0].plot(df['timestamEstadisticas'],df['framesLost'])
axs[0].set_title('Frames Lost', fontsize = 70)
axs[0].set_ylim(0,df['framesLost'].max()+2)
axs[0].grid()
tiks_t =ticker.FuncFormatter(lambda x, pos: formatoHora(x))
axs[0].xaxis.set_major_formatter(tiks_t)
axs[0].tick_params(labelsize = 30)
axs[0].set_xlim(df.iloc[0,7],df.iloc[fin-1,7])

axs[1].plot(df['timestamEstadisticas'],
↪  df['AverageNetworkLatency'],color='k',linewidth= 3)
axs[1].set_title('Average Network Latency (ms)', fontsize = 70)
axs[1].set_ylim(0,maximaLatencia+0.1*maximaLatencia)
axs[1].set_xlim(df.iloc[0,7],df.iloc[fin-1,7])
axs[1].grid()
axs[1].xaxis.set_major_formatter(tiks_t)
axs[1].tick_params(labelsize = 30)

axs[2].plot(df['timestamEstadisticas'],
↪  df['AvgBitrate'],color='darkblue',linewidth= 3, label =
↪  "AvgBitrate")

↪  axs[2].plot(df['timestamEstadisticas'],df['miEstimaciónAnchoDeBand'],
↪  color='k',linewidth= 4,label = " EstimatedBandWidth")
axs[2].set_ylim(0,maximoAnchoDeBanda+0.1*maximoAnchoDeBanda)
axs[2].set_xlim(df.iloc[0,7],df.iloc[fin-1,7])
axs[2].grid()
axs[2].set_title('Estimated Bandwidth (bit/s)',fontsize = 70)
axs[2].xaxis.set_major_formatter(tiks_t)
axs[2].tick_params(labelsize = 30)

for i in range(len(tiempos_de_prediccion)):
    detectoAlgo=int(tiempos_de_prediccion.popleft())
    axs[0].axvline(detectoAlgo,color='Purple',ls=':',linewidth=
    ↪  4)
    axs[0].text(detectoAlgo, 0.8*df['framesLost'].max()+1,
    ↪  'Prediction', fontsize=40, color='purple', rotation=90)
    axs[1].axvline(detectoAlgo,color='Purple',ls=':',linewidth=
    ↪  4)
```

```
        axs[1].text(detectoAlgo, 0.8*maximaLatencia, 'Prediction',
        ↪  fontsize=40, color='purple', rotation=90)
        axs[2].axvline(detectoAlgo,color='Purple',ls=':',linewidth=
        ↪  4)
        axs[2].text(detectoAlgo, 0.8*maximoAnchoDeBanda,
        ↪  'Prediction', fontsize=40, color='purple', rotation=90)

    plt.legend(fontsize = 35)
    plt.show()

#-----------------------------------------------------------------
area = 0
umbral =2000
colaEstimatedBandwidth = deque()
colaDeDegradaciones= deque()
cola_latencia = deque()
almenosUna = False
EstBand = 0
j=0

#-----------------------------------------------------------------
ruta = ruta_archivo_csv('C:\pruebaABRA')
no_es_finArchivo = False
down_bitrate_step = 500000
margenBanda = 1500000
tiempo_Inicial = 0

for i in range(25):
    last_line = cambio_de_linea(ruta)[1]
    datos_utilesE =
    ↪  inicialización_colaEstimatedBandwidth(last_line,
    ↪  colaEstimatedBandwidth,EstBand)
    datos_utilesA =
    ↪  inicializacion_colaLatencia_Y_Area(last_line,cola_latencia,area)
    EstBand = datos_utilesE[0]
    colaEstimatedBandwidth = datos_utilesE[1]
    area = datos_utilesA[1]
    cola_latencia = datos_utilesA[0]
    j= j+1

while (no_es_finArchivo)== False:
    j=j+1
    datos_utiles = cambio_de_linea(ruta)
    no_es_finArchivo = datos_utiles[0]
    t_valido = datos_utiles[1].split(',')[7]
```

```
    latenciaMinima =int(datos_utiles[1].split(',')[9])
    if t_valido!='0' and j > 500 :
        varAuxLatencia =
        ↪  actuaclización_cola_Latencia(datos_utiles[1],
        ↪  cola_latencia , area)
        area = varAuxLatencia[1]
        cola_latencia = varAuxLatencia[0]
        varAuxEstBand =
        ↪  actualizacion_cola_estimatedBand(datos_utiles[1],
        ↪  colaEstimatedBandwidth,EstBand)
        EstBand = varAuxEstBand[0]
        AvgBitrate = int(datos_utiles[1].split(',')[14])
        colaEstimatedBandwidth = varAuxEstBand[1]
        tiempo_transcurrido = time.time()-tiempo_Inicial

        if area > umbral and t_valido!='0' and tiempo_transcurrido
        ↪  > 2:
            colaDeDegradaciones.append(int(t_valido))
            almenosUna = True
            cantidad_Bajadas =
            ↪  int((AvgBitrate-EstBand+margenBanda)/(down_bitrate_step*0.85))
            print("Bajando la tasa. Aumenta el AvgBitrate en ",
            ↪  0.5*cantidad_Bajadas, "Mbps ")
            bajar_tasa(cantidad_Bajadas)

            tiempo_Inicial = time.time()
            print(j," | area = ",area," | AvgBitrate =",AvgBitrate,
            ↪  " | EstBW =",EstBand," | Cantidad de bajadas =
            ↪  ",cantidad_Bajadas)

        else:
            tiempo_transcurrido = time.time()-tiempo_Inicial
            if latenciaMinima<80 and almenosUna and
            ↪  tiempo_transcurrido > 20:
                keyboard.press("F9")
                tiempo_Inicial = time.time()
                print("Subiendo la tasa. Aumenta el AvgBitrate en
                ↪  0.5 Mbps ")
                print(j,"|AvgBitrate = ", AvgBitrate, " | EstBW
                ↪  =",EstBand)

graficas(colaDeDegradaciones)

print(".....................ABRAGame.................")
print("..................END THE GAME..............")
```

Referencias

[1] https://newzoo.com/resources/blog/the-latest-games-market-size-estimates-and-forecasts. Mayo 2023.

[2] https://history-computer.com/video-games-with-the-biggest-budgets/. Mayo 2023.

[3] Wei Cai, Ryan Shea, Chun-Ying Huang, Kuan-Ta Chen, Jiangchuan Liu, Victor C. M. Leung, and Cheng-Hsin Hsu. A survey on cloud gaming: Future of computer games. *IEEE Access*, 4:7605–7620, 2016.

[4] Pranav Shrivastava and Madhavi Damle. Investment decision in cloud gaming-based businesses opportunities: An analysis of the cloud gaming industry. In *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, pages 1224–1228, 2022.

[5] ITU-T Recommendation G.1032: Influence factors on gaming quality of experience, 2017.

[6] Florian Metzger, Stefan Geißler, Alexej Grigorjew, Frank Loh, Christian Moldovan, Michael Seufert, and Tobias Hoßfeld. An introduction to online video game qos and qoe influencing factors. *IEEE Communications Surveys Tutorials*, 24(3):1894–1925, 2022.

[7] Newzoo. Global Cloud Gaming Report. 2022.

[8] https://www.cnet.com/tech/gaming/best-cloud-gaming-services/. Accedido Octubre 2022.

[9] https://enperspectiva.uy/en-perspectiva-programa/entrevistas/abya-el-netflix-para-videojuegos-que-apuesta-por-el-cloud-gaming-que-es-y-como-funciona/. Accedido Octubre 2022.

[10] https://www.antel.com.uy/institucional/sala-de-prensa/eventos/alianza-entre-antel-y-abya-un-proyecto-innovador-y-tecnologico-para-uruguay-y-la-region. Accedido Octubre 2022.

[11] ABYA. https://abya.com/es. Accedido en Mayo, 2022.

[12] Ryan Shea, Jiangchuan Liu, Edith C.-H. Ngai, and Yong Cui. Cloud gaming: architecture and performance. *IEEE Network*, 27(4):16–21, 2013.

[13] Kuan-Ta Chen, Yu-Chun Chang, Hwai-Jung Hsu, De-Yu Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, 16(2):480–495, 2014.

[14] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. Gaminganywhere: An open cloud gaming system. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, New York, NY, USA, 2013. Association for Computing Machinery.

[15] José Joskowicz. Codificación de voz y video. *Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República*, 2021.

[16] G.J. Sullivan and T. Wiegand. Video compression - from concepts to the h.264/avc standard. *Proceedings of the IEEE*, 93(1):18–31, 2005.

[17] ITU-T Recommendation H.264: Advanced video coding for generic audiovisual services, 2021.

[18] José Joskowicz. Calidad de la experiencia en servicios de telecomunicaciones. *Tecnologías de Redes y Servicios de Telecomunicaciones. Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República*, 2022.

[19] ITU-T Recommendation P.809: Subjective evaluation methods for gaming quality, 2018.

[20] ITU-T Recommendation G.1072: Opinion model predicting gaming quality of experience for cloud gaming services, 2020.

[21] Saeed Shafiee Sabet, Steven Schmidt, Saman Zadtootaghaj, Carsten Griwodz, and Sebastian Moller. Towards the impact of gamers strategy and user inputs on the delay sensitivity of cloud games. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–3, 2020.

[22] https://www.nvidia.com/en-us/geforce-now/system-reqs/. Accedido Mayo 2022.

[23] Hanlin Sun. Research on latency problems and solutions in cloud game. *Journal of Physics: Conference Series*, 2019.

[24] Jim Gettys. Bufferbloat: Dark buffers in the internet. *IEEE Internet Computing*, 15(3):96–96, 2011.

[25] Alberto Alós, Francisco Morán, Pablo Carballeira, Daniel Berjón, and Narciso García. Congestion control for cloud gaming over udp based on round-trip video latency. *IEEE Access*, 7:78882–78897, 2019.

[26] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems*, MMSys '16, New York, NY, USA, 2016. Association for Computing Machinery.

[27] Lei Zhang, Yong Cui, Junchen Pan, and Yong Jiang. Deadline-aware transmission control for real-time video streaming. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–6, 2021.

[28] BMSB 2023. https://www.bmsb2023.com/. Accedido en Mayo, 2022.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el sábado 29 julio, 2023.
http://iie.fing.edu.uy/