



## **GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA**

Escuela de Ingeniería de Fuenlabrada

Curso académico 2024-2025

### **Trabajo Fin de Grado**

Desarrollo de un Sistema ABR para Transmisión de Vídeo en  
Tiempo Real en Entornos Robóticos

**Tutor:** José Miguel Guerrero Hernández

**Autor:** Álvaro Gutiérrez García



Este trabajo se distribuye bajo los términos de la licencia internacional CC BY-NC-SA International License (Creative Commons AttributionNonCommercial-ShareAlike 4.0). Usted es libre de *(a) compartir*: copiar y redistribuir el material en cualquier medio o formato; y *(b) adaptar*: remezclar, transformar y crear a partir del material. El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia:

- *Atribución.* Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciatante.
- *No comercial.* Usted no puede hacer uso del material con propósitos comerciales.
- *Compartir igual.* Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

# Agradecimientos

---

Quiero expresar mi más profundo agradecimiento y dedicar este trabajo a la persona que más hubiera querido verme convertido en ingeniero, pero que, lamentablemente, partió de este mundo al inicio del grado: mi padre.

Aunque no logró vencer en su batalla contra el cáncer, su lucha incansable y su valentía, me enseñaron a buscar el camino del esfuerzo y la dedicación que hoy guían mi vida.

Hoy cumplo la promesa que te hice, para que al fin puedas descansar orgulloso de mí. Yo, en cambio, llevaré siempre tu recuerdo, tu fuerza y tus enseñanzas contigo.

Esta es mi forma de despedirme de ti, como te mereces. Hasta siempre, Papá.

*A Jesús María Gutiérrez Cebrián;  
En paz descanse.*

Madrid, 13 de octubre de 2024

*Álvaro Gutiérrez García*

# Resumen

---

En el ámbito de la robótica avanzada y los sistemas autónomos, *Robot Operating System 2* (ROS 2) se destaca como un marco de trabajo pionero, caracterizado especialmente por su arquitectura modular fundamentada en nodos que operan de manera interconectada a través de una capa de *middleware*. En este entorno, la eficiencia y efectividad del sistema dependen directamente del rendimiento de la red, ya que un rendimiento por debajo de lo óptimo puede generar cuellos de botella significativos que amenazan con comprometer la funcionalidad y eficiencia del sistema, sobre todo en operaciones críticas.

Dado el considerable impacto que el contenido de vídeo tiene en el volumen de datos transmitidos por la red, este proyecto explora cómo la implementación de técnicas avanzadas de codificación y *streaming* adaptativo puede representar una estrategia valiosa para mejorar las capacidades de los robots. Este enfoque se centra en su impacto sobre el rendimiento y la fiabilidad del sistema. Además, se hará disponible al público todo el código utilizado para su implementación o que sirva de inspiración, promoviendo así la colaboración y el avance en este campo.

# Acrónimos

---

**ROS 2** *Robot Operating System 2*

**ABR** *Adaptative Bit Rate*

**CCA** *Congestion Controll Algorithm*

**FEC** *Forward Error Correction*

**AL-FEC** *Aplication Layer Forward Error Correction*

**UAV** *Unmanned Aerial Vehicle*

**GOP** *Group of Pictures*

**HAS** *HTTP Adaptive Streaming*

**HVS** *Human Visual System*

**SNR** *Signal-to-Noise Ratio*

**RF** *Radiofrecuencia*

**EM** *Electromagnetic*

**HTTP** *Hypertext Transfer Protocol*

**TCP** *Transmission Control Protocol*

**UDP** *User Datagram Protocol*

**IP** *Internet Protocol*

**QoE** *Quality of Experience*

**QoS** *Quality of Service*

**IPTV** *Internet Protocol Television*

**OTT** *Over-The-Top*

**DCT** *Discrete Cosine Transform*

**PID** *Proportional-Integral-Derivative*

**CNN** *Convolutional Neural Network*

# Índice general

---

<b>1. Introducción</b>	<b>1</b>
1.1. Contexto histórico de la transmisión de contenidos multimedia . . . . .	1
1.2. Introducción a ROS 2 . . . . .	7
<b>2. Objetivos</b>	<b>8</b>
2.1. Descripción del problema . . . . .	8
2.2. Requisitos . . . . .	11
2.3. Metodología . . . . .	12
2.4. Plan de trabajo . . . . .	13
<b>3. Marco teórico</b>	<b>15</b>
3.1. Códigos de vídeo y capacidad de procesamiento del robot . . . . .	15
3.1.1. Límites en el cliente . . . . .	17
3.2. Sistema <i>Adaptive Bit Rate</i> (ABR) . . . . .	18
3.2.1. Modelo de latencia . . . . .	18
3.2.2. Medición del tiempo y del throughput . . . . .	20
3.2.3. Tipos de algoritmos ABR . . . . .	21
3.2.4. Estrategia de ajuste . . . . .	29
3.2.5. Estrategia de arranque: tiempo transitorio . . . . .	31
3.2.6. Tamaño de los segmentos de vídeo . . . . .	32
3.2.7. Consumo humano y consumo máquina . . . . .	33
3.3. Comunicación entre nodos . . . . .	34
3.3.1. Configuración de la red . . . . .	35
3.3.2. Mensajería . . . . .	37
3.4. Nivel físico . . . . .	37
3.4.1. Comunicaciones internas . . . . .	38
3.4.2. Comunicaciones externas (radio frecuencia) . . . . .	38

<b>4. Diseño</b>	<b>42</b>
4.1. Visión general . . . . .	42
4.1.1. Rol servidor . . . . .	45
4.1.2. Rol Cliente . . . . .	49
<b>5. Experimentos realizados</b>	<b>56</b>
5.1. Comparación entre predictores . . . . .	57
5.2. Respuesta a múltiples artefactos . . . . .	60
5.2.1. Tramo A . . . . .	61
5.2.2. Tramos B y C . . . . .	64
5.3. Comparación con Theora . . . . .	64
5.4. Experimento real . . . . .	66
<b>6. Conclusiones y trabajos futuros</b>	<b>69</b>
6.1. Conclusiones . . . . .	69
6.2. Trabajos futuros . . . . .	71
<b>Bibliografía</b>	<b>73</b>
<b>A. Anexo 1</b>	<b>78</b>
<b>B. Anexo 2</b>	<b>84</b>

# Índice de figuras

---

1.1. Comparativa entre las capas modificables en la transmisión de vídeo en un modelo tradicional de proveedor <i>Internet Protocol Television</i> (IPTV) y un modelo de proveedor <i>Over-The-Top</i> (OTT). . . . .	5
2.1. Diagrama conceptual simplificado de los sistemas de comunicación más comunes en robots. . . . .	9
3.1. Diagrama que ilustra el funcionamiento de un sistema ABR basado en el cliente. Es importante resaltar que, en este enfoque, . . . . .	18
3.2. Representación que muestra las variaciones en la estabilidad de la red ante un cambio en la calidad del vídeo. . . . .	32
3.3. Comparación de un <i>frame</i> utilizando una representación de escaneo entrelazado frente al escaneo progresivo. Ilustración creada a partir de la reconocida imagen 'Mandrill' (alternativamente 'Baboon') proveniente de la USC SIPI Image Database ([USC SIPI, 2024]). . . . .	34
3.4. Diagrama de comunicación del canal de datos entre el cliente y el servidor, exemplificando cada uno de los procesos que pueden ocurrir. .	38
4.1. Diagrama simplificado de la mecánica principal de ajuste llevada por el sistema ABR. . . . .	44
4.2. Diagrama mostrando la lógica de cambio de calidad implementada. .	55
5.1. Comparación de rendimiento entre diferentes predictores en relación a la capacidad del canal y en relación a la tasa máxima. . . . .	58
5.2. Comparación de rendimiento en mayor detalle de los mejores predictores en relación a la tasa de transmisión ideal. . . . .	60
5.3. Comparación sistema ABR completo. . . . .	61
5.4. Comparación del sistema ABR completo para el tramo A, tras establecer un límite superior de <i>bitrate</i> . . . . .	63

5.5. Comparación del sistema propuesto (izquierda) frente a Theora (derecha) en el tramo A. . . . .	65
5.6. Comparación del sistema propuesto (izquierda) frente a Theora (derecha) en el tramo B. . . . .	66
5.7. Comparación del sistema propuesto (izquierda) frente a Theora (derecha) en el tramo C. . . . .	66
5.8. Robot TIAGo utilizado en el experimento. . . . .	67
5.9. Comparación del sistema propuesto en un robot real, en concreto, un robot TIAGo conectado a un receptor a través de una conexión Wi-Fi irregular. . . . .	68

# Listado de códigos

---

4.1. Definición de la interfaz para el <i>topic</i> de datos. . . . .	45
B.1. Script bash utilizado para simular las condiciones de red. . . . .	85

# Listado de ecuaciones

---

3.1.	Modelo simple de latencia en una transmisión de vídeo. . . . .	19
3.2.	Predicción de <i>throughput</i> en base al último segmento descargado. . . .	23
3.3.	Predicción de <i>throughput</i> en base a la media aritmética en una ventana temporal. . . . .	24
3.4.	Predicción de <i>throughput</i> en base a la media armónica en una ventana temporal. . . . .	24
3.5.	Ecuación para la selección escalonada del nivel de <i>bitrate</i> . . . . .	30
3.6.	Ecuación de Friis para calcular la pérdida de propagación en espacio libre. . . . .	39
4.1.	Fórmula para el cálculo del número de macrobloques por segundo. . . .	48
4.2.	Fórmula para el cálculo de la tasa máxima . . . . .	49
4.3.	Condiciones del sistema de inestabilidad. . . . .	51
4.4.	Condiciones del sistema de votación. . . . .	51
4.5.	Condiciones para la activación del filtro anti-rizado. . . . .	53

## Índice de cuadros

A.1. Información de la información de interés de las tablas A-1,A-2 y A-3. . .	79
A.2. Velocidad de decodificación en tiempo real por nivel . . . . .	80
A.4. Información de la información de interés de las tablas A-1,A-2 y A-3, corregida para transmisiones en tiempo real. . . . .	82
A.3. Tabla enumerada de formatos de resolución . . . . .	83

---

# Capítulo 1

## Introducción

---

*Although we routinely take for granted the extraordinary performance of multimedia systems, such as high definition video, high fidelity audio, and interactive games, these systems have always relied heavily on state-of-the-art signal processing.*

Alan V.Oppenheim — Ronald W.Schafer, *Discrete-Time Signal Processing*.

### 1.1. Contexto histórico de la transmisión de contenidos multimedia

Con la adopción generalizada de internet en los hogares de todo el mundo a principios de la década de los dos mil, los navegadores web se han establecido como el principal acceso a este medio. Este desarrollo fue fundamental para establecer los modelos de comunicación actuales en internet, destacando particularmente el nacimiento del protocolo *Hypertext Transfer Protocol* (HTTP). La transmisión de contenido multimedia en las últimas dos décadas ha enfrentado el reto de integrar este tipo de contenido en la tecnología HTTP, abarcando tanto la visualización de vídeos en diferido y en directo como las comunicaciones en tiempo real, incluyendo llamadas y videollamadas. Además, el interés en este protocolo creció en popularidad debido a que permitía atravesar los cortafuegos de las máquinas de los usuarios, característica que facilitaba que los servicios pudieran llegar a una gran cantidad de clientes sin tener que preocuparse de que se rechazara el servicio.

Actualmente, esta forma de consumo de contenido multimedia se ha convertido en la norma en nuestra comunicación, consumo de información y entretenimiento, marcando así el ritmo de la cultura popular. Sin embargo, lograrlo ha requerido un esfuerzo global y multidisciplinario por parte de ingenieros especializados, debido a los desafíos asociados con el protocolo HTTP y sus protocolos subyacentes. La razón detrás de

esto es que estos protocolos fueron diseñados originalmente para transferir textos de tamaño limitado, lo cual resulta insuficiente para gestionar el gran volumen de datos involucrados en la transmisión de audio y video.

La gran cantidad de datos necesarios para estos contenidos puede saturar el ancho de banda de la red y generar problemas adicionales, como la fragmentación de paquetes al transmitir grandes volúmenes de información, como los fotogramas de vídeo. Esta fragmentación aumenta la probabilidad de pérdida de paquetes, especialmente en transmisiones que requieren un flujo constante de datos. El desafío se intensifica en las transmisiones en tiempo real, ya que el protocolo de transporte *Transmission Control Protocol* (TCP), que permite la retransmisión de paquetes perdidos, resulta ineficaz en estos escenarios. Esto se debe a que la recuperación de paquetes perdidos en tiempo real provoca una degradación visual instantánea, lo que hace que cualquier paquete recuperado sea desecharido y que la retransmisión consuma un ancho de banda que podría utilizarse para transmitir paquetes más recientes.

Para adaptarse a los nuevos requisitos de transmisión multimedia, el modelo de *streaming* se ha consolidado como el estándar en internet. Este enfoque permite a los usuarios acceder a contenido multimedia en tiempo real sin necesidad de descargar el archivo completo, lo que no solo facilita la visualización de contenidos generados instantáneamente, sino que también minimiza la demanda de ancho de banda que implicaría la descarga total de archivos para su visualización en diferido.

La creciente popularidad de los servicios que utilizan esta tecnología, junto con el incremento de usuarios dispuestos a adoptarla impulsados por la expansión de los *smartphones*, ha desencadenado una competencia tecnológica orientada a optimizar al máximo la calidad de experiencia del usuario (*Quality of Experience* (QoE)). Este concepto, el cual es una evolución del *Quality of Service* (QoS), abarca en el ámbito multimedia todo el proceso de transmisión, desde la codificación del contenido hasta la percepción subjetiva final por parte del usuario. En términos de vídeo, el incremento en la QoE se refleja tanto en la mejora de la calidad visual como en la reducción del tiempo de espera para acceder al contenido y en la minimización de interrupciones debidas a limitaciones de la red, lo que marcó una nueva era de innovación que revolucionó el sector.

Cuando hablamos de codificación, nos referimos a un proceso que modifica la representación de los datos con diversos propósitos, tales como reducir la cantidad de bits necesarios para representarlos, haciéndolos más ligeros o proteger la información para optimizar su transmisión o almacenamiento. Este proceso se realiza a través de un codificador (*coder*) y un decodificador (*decoder*), que se encarga de devolver la

información a su estado original para que el contenido pueda ser consumido.

Desde su concepción, estos sistemas se volvieron imprescindibles para el manejo, almacenamiento y transmisión de contenido de vídeo, respaldados por una considerable cantidad de literatura y destacados por su notable rendimiento. Habitualmente, implementan la codificación con pérdidas, un método que reduce significativamente el tamaño de los datos al alterar parte del contenido original. Esta técnica se basa en el entendimiento del sistema visual humano (*Human Visual System (HVS)*), enfocándose en la eliminación de detalles de menor importancia perceptiva. De esta manera, se logra un equilibrio entre la calidad de imagen y el volumen de datos, medido en bits por segundo.

Algunas de las estrategias habitualmente utilizadas son la reducción de la tasa de cuadros (*framerate*), la cuantificación de las imágenes con menos bits y la modificación espectral de los coeficientes menos perceptibles en transformadas como la *Discrete Cosine Transform (DCT)* o *Wavelets*. Los códecs también trabajan para minimizar la información a través de sistemas predictivos, tanto a nivel de *frame* individual (*intraframe*), analizando similitudes entre píxeles vecinos, como entre diferentes *frames* (*interframe*), donde se analiza el movimiento de objetos en la escena. Finalmente, tras esta reducción en la información visual, se aplican técnicas de codificación entrópica o sin perdidas que logran comprimir el contenido un paso más sin introducir ningún tipo de degradación en la imagen.

Los códecs no se limitan a ofrecer un único formato de codificación; más bien, presentan un amplio abanico de configuraciones que permiten ajustar las preferencias visuales del vídeo, como la resolución y fluidez, en función del contenido que se esté codificando. A su vez, posibilitan la aplicación de estrategias de compresión más o menos agresivas para alcanzar *bitrates* específicos, o adoptar configuraciones que, siendo menos costosas en términos computacionales, resultan en una calidad de imagen inferior. Esta flexibilidad es clave en escenarios de tiempo real, donde es necesario codificar el vídeo lo más rápidamente posible.

En sus orígenes, el *streaming* se empleaba principalmente en comunicaciones de radio y televisión terrestres, regidas por el estándar DVB-T. Este estándar especificaba no solo la configuración a seguir a nivel físico —como la potencia de radiación, modulación y multiplexación— sino también el *bitrate* máximo permitido para la transmisión. Al tener control sobre el canal físico terrestre, se pudieron establecer qué códecs y configuraciones eran adecuados para asegurar que la comunicación fuese satisfactoria, partiendo de una base sólida que garantizaba el éxito de la comunicación en la mayoría de situaciones y abordando las limitaciones del canal físico cuando era

necesario.

Sin embargo, la transmisión de este contenido a través de internet presentaba desafíos adicionales, ya que no solo se desconocía el canal o canales físicos por los que la información transitaría hasta llegar a su destinatario, sino que, además, el internet, a diferencia de la transmisión por radiofrecuencia terrestre, es multiconectada. Los usuarios se interrelacionan a través de capas de protocolos que actúan de manera abstracta en distintos niveles, según el modelo TCP/*Internet Protocol* (IP).

Por ello, la capacidad de descarga de cada usuario variaría significativamente, lo que plantea la incógnita sobre cuál configuración del códec es la óptima para maximizar la Calidad de Experiencia (QoE) de los clientes. Una calidad de vídeo elevada con un *bitrate* alto podría provocar la saturación de la red para los usuarios con conexiones más débiles, mientras que una calidad de vídeo baja reduciría la QoE para aquellos con mejor acceso a internet. Ante este desafío, surgieron dos filosofías predominantes: IPTV y OTT. IPTV estaba dominado por empresas de telecomunicaciones que mantenían control total sobre la comunicación; desde el servidor que almacena y distribuye el contenido a través de una red de entrega de contenidos (CDN), pasando por el canal físico del cliente hasta la capa de aplicación, permitiendo adaptar la red de la que eran propietarios para optimizar la transmisión de contenido multimedia. Esto implicaba la utilización de protocolos de red específicos y estrategias de priorización de paquetes multimedia, algoritmos de control de congestión (*Congestion Control Algorithm* (CCA)), entre otros. Tal control sobre la red permitía realizar los ajustes necesarios para maximizar la QoE de sus usuarios, aunque los costos de despliegue eran considerablemente altos.

Por otro lado, los servicios *Over the Top* o OTT buscan desplegarse a través de las redes de internet existentes, lo que les permite reducir los elevados costos asociados al establecimiento de una infraestructura propia, como ocurre con la IPTV. Sin embargo, estos servicios deben primero enfrentar las dificultades previamente mencionadas relacionadas con la transmisión de contenido multimedia en el entorno hostil que representa internet y la tecnología web. Además, deben ser capaces de competir en términos de QoE con las soluciones de IPTV ya implementadas. Al tener control únicamente sobre la capa superior del modelo TCP/IP, la de aplicación, los servicios OTT se encuentran en una desventaja significativa, como se puede ver en la figura 1.1.

Para que los servicios OTT pudieran competir con los servicios IPTV, fue necesario optimizar las áreas donde estos servicios tenían control: el codificador y la aplicación en el lado del cliente. Como solución, surgieron los sistemas ABR (*Adaptive Bitrate*), que

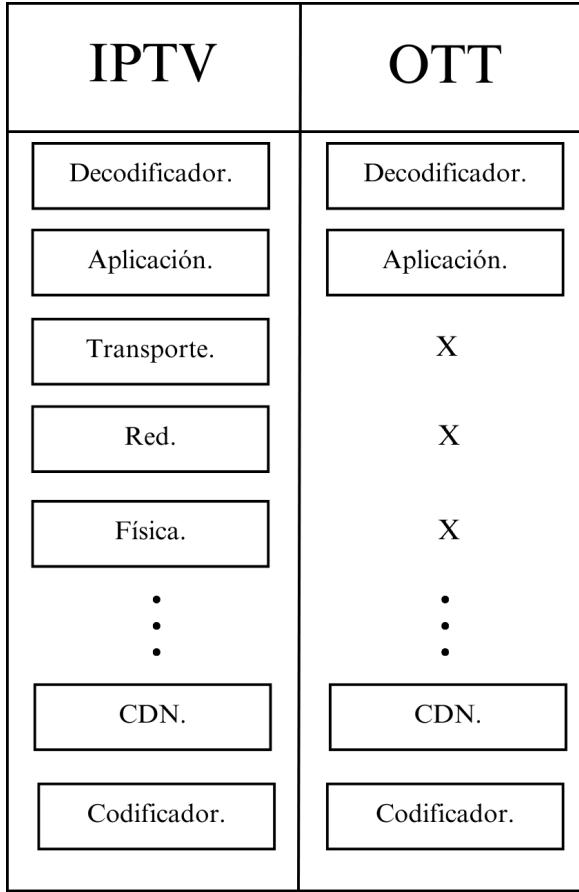


Figura 1.1: Comparativa entre las capas modificables en la transmisión de vídeo en un modelo tradicional de proveedor IPTV y un modelo de proveedor OTT.

permiten modificar la calidad del vídeo en tiempo real durante la descarga del *stream*, adaptándose así a la calidad de la red del cliente para maximizar la QoE. El análisis puede realizarse desde ambos lados de la comunicación, pero predominantemente, la implementación más efectiva ha sido que el lado del cliente se encargue de este análisis y envíe una retroalimentación solicitando que los siguientes segmentos del *stream* se envíen con una calidad determinada. Esta metodología se adoptó porque reduce la complejidad computacional del servidor y porque el lado del cliente demostró una mejor capacidad de análisis de la situación, a pesar de que este análisis se realiza exclusivamente a nivel de aplicación, con las limitaciones que esto implica.

Al analizar la comunicación desde la capa superior de aplicación, se podía, hasta cierto punto, sintetizar las limitaciones de las capas subyacentes, ya fueran físicas o de transporte, y se lograba un rendimiento que permitió a los servicios OTT competir y perdurar hasta nuestros días. Aunque este sistema también podía emplearse en tecnologías IPTV, se trataría de una optimización de la red sobre una infraestructura que ya habría sido mejorada a través de otros mecanismos en las capas inferiores. Por lo tanto, aunque la implementación tenía un impacto positivo, no generaba tanto impacto

como lo hizo en los sistemas OTT.

La implementación de los sistemas ABR en aplicaciones web basadas en HTTP recibió el nombre de *HTTP Adaptive Streaming* (HAS) y ganó mucha popularidad, marcando una era en el desarrollo de las tecnologías multimedia. Aunque el desafío de la implementación de HAS hoy en día puede considerarse resuelto y está implementado en los servicios de audio/vídeo y telecomunicaciones que consumimos actualmente, aún existe un amplio campo de estudio enfocado en cómo mejorar las tecnologías existentes. Se espera que el peso del tamaño del contenido multimedia en internet continúe aumentando debido al consumo de contenidos de mayor calidad y más exigentes con la latencia, como el uso de multimedia en realidad aumentada o videojuegos en la nube. Además, las nuevas investigaciones en el campo están convergiendo con la creciente tendencia actual en el área del análisis de datos y el aprendizaje automático, y posteriormente en este trabajo se desarrollará más detalle sobre las sinergias entre ambos campos.

Por último, es relevante destacar que la implementación de sistemas ABR ha dado lugar a la creación de soluciones completas y propietarias como Microsoft Smooth Streaming (MSS), HTTP Live Streaming (HLS) y Adobe Open Source Media Framework (OSMF). Estos sistemas, que incorporan tecnologías ABR cerradas, han tenido un impacto significativo en la industria. Sin embargo, es importante señalar que, en respuesta a estas soluciones propietarias, se desarrolló el estándar MPEG-DASH de código abierto. Este estándar ha demostrado ser capaz de competir con soluciones anteriores, ofreciendo resultados similares mediante el uso de algoritmos públicos o versiones modificadas de los mismos, lo que ha llevado a MPEG-DASH a establecerse como la implementación más utilizada en los servicios de *streaming* HAS.

Además, DASH, y en particular su versión en JavaScript para el lado del cliente, ha sido ampliamente utilizado por investigadores. Esta accesibilidad ha permitido que sea fácilmente modificado para incorporar nuevos algoritmos propuestos en investigaciones académicas, facilitando así la experimentación y el desarrollo de mejoras en el campo del *streaming* adaptativo.

Este trabajo busca reutilizar el concepto de ABR y adaptarlo de manera creativa al campo de la robótica, demostrando cómo estos conceptos desarrollados pueden ser relevantes en otras ramas de la ingeniería. La intención es explorar aplicaciones innovadoras de ABR más allá de su uso tradicional, ilustrando su potencial para mejorar la eficiencia y efectividad en sistemas robóticos. Este enfoque no solo extiende la aplicación de tecnologías existentes a nuevos dominios, sino que también abre nuevas vías de investigación y desarrollo en la intersección de la robótica y las

telecomunicaciones.

## 1.2. Introducción a ROS 2

ROS 2, o *Robot Operating System*, es un marco de trabajo de código abierto diseñado para el despliegue y desarrollo de sistemas robóticos que opera sobre otros sistemas operativos, especialmente sobre Linux (Ubuntu). Se caracteriza por una arquitectura de nodos modulares e independientes que se comunican mediante una red local, sobre la cual opera un *middleware* que facilita la comunicación. Este *middleware* proporciona una capa de abstracción de mensajería a nivel de aplicación, ofreciendo múltiples formas de comunicación entre nodos, como servicios y acciones. En los servicios, un nodo cliente envía una solicitud a un nodo servidor que procesa esta solicitud y devuelve una respuesta. Las acciones, por otro lado, son útiles para tareas de duración considerable, proporcionando retroalimentación sobre el progreso de la tarea. Sin embargo, la modalidad más utilizada es la de publicador/subscriptor, donde los nodos publicadores pueden transmitir información en un tema (*topic*) específico y los nodos subscriptores pueden suscribirse a él para procesar la información recibida. Las interfaces de mensajes entre nodos son completamente personalizables, permitiendo el envío de estructuras de mensajes a medida.

ROS 2 admite código escrito tanto en C++ como en Python, permitiendo que ambos lenguajes se ejecuten de manera independiente y simultánea dentro de la misma aplicación. Esto facilita el manejo de un amplio rango de herramientas, aprovechando la velocidad de procesamiento de C++ y la sencillez, junto con las capacidades matemáticas y de aprendizaje automático, que ofrece Python. Para una distribución eficiente del código, ROS 2 emplea un sistema de 'paquetes' de código, que constituyen la estructura básica para organizar y distribuir código, datos y recursos. Cada paquete puede incluir nodos, bibliotecas, conjuntos de datos, configuraciones y otros archivos necesarios para implementar funcionalidades específicas. Esta combinación de elementos convierte a ROS 2 en un entorno multidisciplinar, facilitando la integración del mundo de la robótica a profesionales de diversos campos.

Además, ROS 2 es compatible con múltiples herramientas de simulación, siendo GAZEBO una de las más destacadas. Esta herramienta permite a ingenieros y desarrolladores probar algoritmos, diseñar robots y simular escenarios complejos en un entorno 3D dinámico y detallado con físicas avanzadas. Esto facilita la experimentación y validación de conceptos en un espacio virtual antes de su implementación en el mundo real.

---

# **Capítulo 2**

# **Objetivos**

---

En este capítulo se expondrán en detalle los objetivos fundamentales del proyecto, así como la definición precisa del problema que se busca resolver. Se abordarán los resultados finales esperados, destacando su relevancia en el contexto del proyecto. Asimismo, se realizará un análisis exhaustivo sobre el enfoque adoptado para llevar a cabo el proyecto de manera exitosa, describiendo paso a paso las estrategias implementadas. Se discutirá de forma detallada las metodologías seleccionadas, justificando su elección y su aplicación en las diferentes fases del desarrollo. Finalmente, se presentará el plan de trabajo estructurado que ha permitido avanzar de forma coherente y eficiente hacia la consecución de los objetivos planteados.

## **2.1. Descripción del problema**

Un sistema robótico en ROS 2, gracias a su flexibilidad, puede adoptar múltiples arquitecturas. Por un lado, podemos encontrar sistemas que se asemejan a la idea general de lo que es un robot, como puede ser un *Unmanned Aerial Vehicle* (UAV) manejado a distancia o un robot humanoide. Por otro lado, también es posible configurar sistemas compuestos por una gran cantidad de componentes que ensamblan elementos en una planta industrial, mientras gran parte de los datos se procesan en un servidor en la nube ubicado a cientos de kilómetros de distancia. En general, ROS 2 facilita la comunicación de estructuras de procesamiento de datos como si fueran un sistema distribuido, independientemente de si los nodos se están procesando en la misma máquina, constituyen un conjunto de sistemas embebidos, u otras configuraciones posibles. Esto abre un amplio abanico de posibilidades para generar aplicaciones robóticas; sin embargo, también hace que el sistema sea sensible a los desafíos inherentes a la transmisión de información a través de medios de comunicación físicos. Dada la diversa disponibilidad física de los nodos, que pueden estar interconectados de varias maneras, la integración y la eficiencia de la comunicación

pueden variar significativamente.

En este trabajo, principalmente nos centraremos en un modelo de robot sencillo para explicar y probar los conceptos desarrollados teniendo en cuenta que estos conceptos pueden extrapolarse a otras arquitecturas y diseños. En concreto, nos basaremos en una arquitectura donde existe una máquina central encargada de escuchar a una gran cantidad de *topics* que están transmitiendo flujos de datos de los sensores conectados. Esta arquitectura cuenta con una interfaz de red central y propia para conectar todos estos elementos y habitualmente se incorpora un sistema de antenas que generan un punto de acceso al que conectarse externamente, todo en un mismo objeto de *hardware*. Un ejemplo de este tipo de robots puede ser entendido con el siguiente esquema simplificado de la estructura interna de estos robots y sus comunicaciones en la figura 2.1. Aunque este esquema es flexible y admite diversas implementaciones, generalmente se considera que una división en comunicaciones internas y externas, junto con la clasificación de los componentes en sensores y actuadores según sus roles, proporciona una comprensión completa de la mayoría de los escenarios posibles.

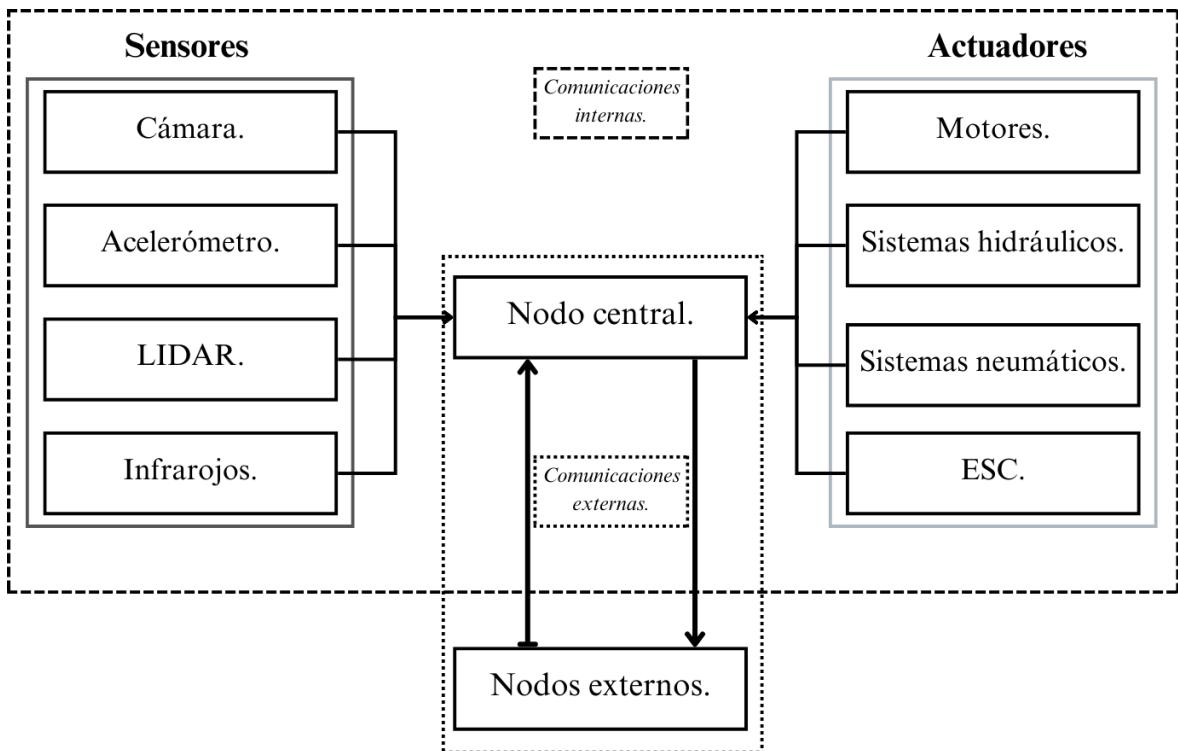


Figura 2.1: Diagrama conceptual simplificado de los sistemas de comunicación más comunes en robots.

Se conoce que la capacidad de la red desplegada por el robot estará estrechamente ligada al canal por el que la comunicación debe realizarse atendiendo a criterios físicos como la distancia, la conductividad o permeabilidad de los medios, la calidad del

*hardware* y el estándar con el que se ha codificado la información.

En este punto, se evidencia la relevancia de adoptar sistemas *Adaptive Bitrate* (ABR) en el ámbito de la robótica. No obstante, es importante reconocer los retos presentes al aplicar esta tecnología en contextos significativamente distintos a aquellos para los que originalmente fue desarrollada, como es el protocolo HTTP. Además, se destaca una relevante inversión en la problemática: típicamente, en un sistema ABR, el servidor mantiene una posición estática con capacidades superiores de procesamiento, almacenamiento y transmisión, mientras que el cliente presenta limitaciones en estas áreas. Sin embargo, en el escenario actual, donde el robot actúa como el nodo central y punto de acceso (AP), se invierte este papel. El robot, actuando como servidor, se ve limitado por las capacidades de procesamiento del nodo en tiempo real y el procesamiento de otros nodos en la red, limitado a su vez por el tamaño físico del robot y el hecho de que pueda estar en movimiento. Estos aspectos se explorarán con mayor detalle en el capítulo 3.

Al definir ciertos parámetros de la calidad de la comunicación, es crucial entender el uso previsto del vídeo, distinguiendo principalmente entre consumo humano o consumo máquina. Si el vídeo está destinado a ser visualizado por humanos, es esencial seleccionar un sistema códec y ABR que se alineen con la investigación clásica en la transmisión de contenido multimedia, considerando la seguridad de las misiones del robot y, posteriormente, la calidad de servicio (QoS) y la calidad de experiencia del usuario (QoE) como factores prioritarios.

Por otro lado, si el flujo de vídeo se utiliza como entrada para algún tipo de procesamiento de datos, como *computer vision* o de inteligencia artificial, que son campos prominentes en la robótica, el sistema ABR debe adaptarse a las necesidades de estos algoritmos. Estos sistemas son particularmente sensibles a modificaciones en el formato de entrada respecto a su entrenamiento. Por ello, es fundamental determinar qué parámetros del vídeo, desde el *framerate* y la resolución hasta la cuantificación de bits y la calidad de imagen en términos generales, deben mantenerse constantes para que puedan operar eficazmente sin priorizar esquemas de codificación basados en el sistema visual humano, que podrían degradar su rendimiento. Por ejemplo, los cambios rápidos en la calidad de la imagen pueden hacer que un sistema sea inestable pero permiten una mejor adaptabilidad al ancho de banda. Generalmente, los sistemas muy inestables se ha observado que deterioran la ABR, por lo que se prefiere sacrificar algo de eficiencia de ancho de banda a favor de la estabilidad, maximizando así la QoE. Sin embargo, esta limitación se basa completamente en la experiencia humana y un algoritmo de procesado de vídeo podría preferir sistemas más inestables.

## 2.2. Requisitos

El trabajo será considerado exitoso si la propuesta introducida mejora la actual implementación de los sistemas de vídeo en ROS 2. Actualmente, estos sistemas utilizan *plugins* que codifican y decodifican vídeo a través de la solución nativa del paquete 'Image Transport', el cual emplea el códec Theora. Aunque Theora es de uso libre, presenta signos de obsolescencia. Existen también soluciones de terceros que implementan códecs como h264, pero no permiten realizar un ajuste adaptativo de *bitrate* (ABR) de manera dinámica y en tiempo real. Por lo tanto, el objetivo es lograr un mejor rendimiento que el ofrecido por estas implementaciones, evaluando cómo la propuesta de ABR introduce mejoras en el sistema.

El objetivo final será desarrollar una solución de vídeo adaptativo en tiempo real, regulada mediante un algoritmo ABR diseñado con base en el conocimiento existente en la bibliografía. Este algoritmo estará específicamente enfocado en las situaciones que probablemente enfrentaría un sistema robótico bajo las condiciones de este proyecto.

Para lograr el objetivo global del proyecto, se deberán cumplir una serie de requisitos específicos que permitirán completar cada una de las fases del desarrollo:

- Explorar ROS 2, su comunidad y las tendencias emergentes de desarrollo dentro de la plataforma.
- Analizar las implementaciones nativas de códecs en ROS 2, entendiendo su funcionamiento y características.
- Realizar un estudio exhaustivo de la bibliografía relacionada con los temas clave: codificación de vídeo, algoritmos ABR, y comunicaciones en robots (sistemas móviles), entre otros.
- Diseñar la propuesta pertinente, lo que incluirá la selección del códec a implementar, la definición de roles de las máquinas, la estructura de los mensajes a enviar y recibir, y el algoritmo ABR final.
- Implementar en código la propuesta diseñada.
- Realizar experimentos en un entorno simulado, comparando el rendimiento bajo determinadas condiciones para evaluar si la propuesta supera a las implementaciones actuales.

- Llevar a cabo experimentos en un entorno real, realizando comparaciones similares para determinar si la propuesta introducida ofrece mejoras en comparación con las soluciones existentes.

## 2.3. Metodología

La metodología aplicada en el proyecto es de carácter mixto. Por un lado, es cuantitativa, ya que implica la implementación física del algoritmo propuesto, el cual operará en un entorno con datos simulados y reales, permitiendo medir su rendimiento para obtener conclusiones. Por otro lado, también requiere un enfoque cualitativo, debido a la necesidad de una exhaustiva revisión de los sistemas ABR actuales, las problemáticas que enfrentan los sistemas robóticos en términos de comunicación y las contribuciones de la comunidad ROS. Todo esto es fundamental para asegurar una implementación exitosa que permita adaptar un sistema ABR al contexto robótico, resolviendo los desafíos existentes.

Esta doble naturaleza del proyecto se refleja en dos pilares fundamentales. El primero es una extensa revisión de la literatura actual en áreas clave como vídeo, ciencia de datos, comunicaciones, modelos de propagación y artefactos específicos en el contexto robótico. Para ello, se han utilizado trabajos de alta calidad provenientes de fuentes confiables, incluyendo el IEEE (Instituto de Ingenieros Eléctricos y Electrónicos), la ITU (Unión Internacional de Telecomunicaciones) y publicaciones de la comunidad ROS, entre otros. El resultado de este análisis se presentará de manera destacada en el capítulo dedicado al marco teórico. El segundo pilar, de naturaleza más práctica, consistirá en el desarrollo de software dentro del entorno robótico ROS 2, específicamente en su distribución 'Humble', alineándose con los desarrollos actuales de la comunidad. El código se programará en C++ para garantizar un rendimiento óptimo y evitar cuellos de botella, todo ello ejecutado sobre el sistema operativo Linux Ubuntu 22.04 LTS (Jammy Jellyfish). Además, se hará uso de Python y la plataforma web 'Canva' para visualizar los datos obtenidos y generar diagramas explicativos que se incluirán en la memoria del proyecto.

El análisis de los resultados se llevará a cabo de manera cuantitativa mediante la medición de la tasa de transmisión (*throughput*) respecto a la capacidad del canal (*bitrate*), es decir, la cantidad de bits enviados por segundo cuando los sistemas de prueba son sometidos a determinadas condiciones. Además, dado que no existe un modelo único consensuado de QoE (Calidad de Experiencia) en el sector de vídeo, y que las necesidades de un robot de este tipo, como se explicará más adelante, son distintas,

se evaluará cualitativamente si el robot cumple con los criterios de calidad deseados o si presenta artefactos indeseados que lo hagan inviable. Finalmente, los resultados obtenidos se presentarán y analizarán de forma conjunta para llegar a conclusiones específicas.

## 2.4. Plan de trabajo

El plan de trabajo ha seguido una estructura flexible pero claramente diferenciada en cuanto a las etapas del proyecto. Es importante destacar que las estimaciones de tiempo han tenido en cuenta que el desarrollo del proyecto se realizó de manera parcial, ya que se ha compaginado con responsabilidades laborales y académicas. Esto ha requerido una planificación cuidadosa para asegurar el progreso constante, ajustándose a las limitaciones de tiempo disponibles en cada fase.

- **Primera etapa.** El objetivo de esta fase fue familiarizarse con el entorno de trabajo de ROS 2, así como con los lenguajes de programación C++ y Python aplicados a esta plataforma. Para lograrlo, se realizó un curso en línea específico sobre el tema, se llevaron a cabo pequeños proyectos y se participó activamente en la comunidad ROS. Durante esta etapa, el tema del proyecto aún no estaba definido, y el enfoque principal fue explorar una propuesta que pudiera mejorar las implementaciones actuales de vídeo en sistemas robóticos. Esta fase tuvo una duración aproximada de seis meses, motivada por el interés personal en profundizar en la plataforma.
- **Segunda etapa.** Tras definir una propuesta interesante para la resolución del problema, esta fase se centró en la búsqueda de recursos bibliográficos y una profunda síntesis del contenido, con el objetivo de comprender en mayor detalle cómo llevar a cabo la implementación y cómo abordar los principales focos de error. Paralelamente, se realizaron pequeños experimentos con datos sintéticos para validar los conceptos teóricos. Esta etapa tuvo una duración aproximada de dos meses.
- **Tercera etapa.** En esta fase se desarrolló la propuesta a nivel de software, siguiendo las directrices que se detallarán en el capítulo de diseño. Este proceso incluyó la implementación completa del sistema, asegurando que cumpliera con los requisitos establecidos. La duración de esta etapa fue de aproximadamente dos meses.

- **Cuarta etapa.** Esta fase estuvo dedicada a la realización de pruebas del sistema propuesto mediante experimentos con entornos de red limitados y robots reales en la Universidad Rey Juan Carlos, en el campus de Fuenlabrada. Finalmente, se redactaron los capítulos correspondientes a los experimentos y las conclusiones, dando por concluido el proyecto. Para esta etapa, se estima que la carga de trabajo sería de un mes.

---

## Capítulo 3

# Marco teórico

---

En esta sección, se abordarán los retos asociados con la transferencia e implementación de un sistema de códec de vídeo adaptativo (ABR) en el entorno ROS 2. Se analizarán detalladamente las diversas etapas implicadas en la implementación de un sistema ABR, proporcionando una visión exhaustiva de los diferentes niveles de complejidad requeridos. Además, se discutirán las ventajas y desventajas en comparación con los sistemas multimedia tradicionales.

### 3.1. Códigos de vídeo y capacidad de procesamiento del robot

Como se vio en el capítulo de introducción (capítulo 1), los códigos de vídeo son un elemento clave en la transmisión de vídeo por lo que conocer más en detalle como estos pueden ser configurados es una parte esencial para su correcto dominio. La configuración de los códigos se estructura en torno a dos conceptos claves: 'Nivel' y 'Perfil'.

- **Perfil.** El 'Perfil' en un códec de vídeo se refiere al conjunto de técnicas utilizadas para la compresión de vídeo a nivel técnico. Estas técnicas incluyen aspectos como el número de bits utilizados para codificar los píxeles, la configuración de la estructura del *Group of Pictures* (GOP), y el tamaño de los macrobloques, entre otros. A medida que se aumenta el perfil de un códec, también se incrementa la complejidad del procesamiento. Sin embargo, este aumento en la complejidad se traduce en una compresión más eficiente, lo que puede resultar en una mejora significativa en la calidad del vídeo bajo restricciones de ancho de banda.
- **Nivel.** El 'Nivel' en un códec de vídeo permite seleccionar un conjunto de especificaciones deseadas para el vídeo, tales como la tasa de *frames* por segundo y la resolución espacial. Al aumentar estas especificaciones, inevitablemente se

incrementa el tamaño del archivo de vídeo. Este ajuste es esencial para adaptar la calidad del vídeo a los requisitos específicos de visualización o almacenamiento, equilibrando la calidad y la fluidez del vídeo con el uso eficiente del espacio de almacenamiento y el ancho de banda.

Estos parámetros se establecen principalmente para asegurar la compatibilidad entre dispositivos emisores y receptores, garantizando así la comunicación. Además, reflejan el equilibrio necesario entre los tres elementos clave en la codificación de vídeo: calidad de imagen, *bitrate* y coste computacional.

También es fundamental conocer mejor el sistema ABR más utilizado actualmente, HAS, con el fin de distinguir claramente los puntos a favor y en contra al momento de realizar nuestra implementación. Los servidores HAS tradicionales suelen ofrecer servicios de distribución en diferido, por lo que preparan el contenido ya precodificado y fragmentado en segmentos, almacenando cada segmento en URIs específicas que son solicitadas por el cliente al servidor. Cada segmento, correspondiente a un fragmento de tiempo y configuración, se conoce como presentación. Sin embargo, si el contenido está siendo grabado y emitido en tiempo real, generalmente se dispone de suficiente potencia para codificar vídeos en múltiples niveles simultáneamente, incluso llegando a utilizar uno o varios códecs para lograr una mayor diversidad de configuraciones.

Por ello, la capacidad de procesamiento de un robot se presenta como uno de los principales desafíos al implementar tecnología ABR en robótica, ya que deben gestionar su capacidad de procesamiento de manera más crítica. Esta necesidad marca un cambio de paradigma en comparación con el uso convencional de la tecnología ABR, que es orientado principalmente a optimizar la transmisión de contenido multimedia en diferentes condiciones de red, mientras que en robótica, el enfoque estaría en maximizar la eficiencia y efectividad operativa dentro de los límites de recursos computacionales disponibles.

Para afrontar esta limitación, se considera recomendable que el códec implementado en el robot codifique en tiempo real una única configuración, ajustando su agresividad basándose en la retroalimentación recibida del lado del cliente.

Además, la selección del códec es crucial. Los códecs de última generación, como AV1 y VVC, ofrecen un mejor equilibrio entre calidad visual y cantidad de bits, conocido como eficiencia de compresión. Esto se consigue a través de aplicar técnicas de codificación más avanzadas que mejoran la QoE y permiten manejar mayores volúmenes de datos, resoluciones y niveles de cuantificación de color. Sin embargo, el mayor peso computacional que llevan estas técnicas avanzadas puede ser limitante en

dispositivos con menor capacidad. Por lo tanto, la elección del códec estará restringida por las limitaciones técnicas del diseño del *hardware* y viceversa.

Para este trabajo, se ha realizado una implementación del códec h264/AVC , también conocido como MPEG-4 Part 10 o AVC (Advanced vídeo Coding), el cual fue desarrollado conjuntamente por el Moving Picture Experts Group o MPEG y vídeo Coding Experts Group o VCEG. Fue lanzado en 2003 y se ha convertido en uno de los códecs de vídeo más ampliamente utilizados y populares en el mundo y presenta cierto renombre por conseguir a día de hoy una buena eficiencia de codificación frente a la complejidad que supone. Si buscásemos un códec con mejor eficiencia de compresión podríamos optar por su sucesor H265 aunque este suponga un salto computacional considerable, o por opciones todavía mas costosas como VP8, VP9, AV1 o VVC. En cambio, si buscásemos opciones de menor complejidad tendríamos estándares anteriores como Theora, MPEG-2 o MPEG-1 sin embargo estas opciones se consideran obsoletas por su baja eficiencia de compresión.

Las especificaciones del códec h264/AVC se pueden encontrar en el documento citado [Union, 2021], siendo de especial interés las tablas A-1, A-2 y A-6. Para condensar toda la información contenida en estas tablas en un único recurso, se ha elaborado la tabla A.1. Además, se ha realizado un mapeo de las configuraciones disponibles en A.3, que se encuentra en el anexo 1. Este enfoque unificado permite una visión más clara y accesible de las opciones de configuración del códec, facilitando su consulta y aplicación en proyectos relacionados.

### 3.1.1. Limites en el cliente

Las limitaciones técnicas en el lado del cliente desempeñan un papel crucial en los algoritmos de tasa de bits adaptable (ABR). En primer lugar, la capacidad de cómputo y almacenamiento del dispositivo, que depende directamente de su *hardware*, influye en el rendimiento general. Además, cuando el contenido está orientado a un usuario final, las características de la pantalla, como la resolución y la tasa de refresco, imponen restricciones significativas. Por ejemplo, codificar vídeos en 4K para una pantalla con soporte máximo de 2K implicaría un uso ineficiente del ancho de banda, lo que permite descartar opciones de mayor *bitrate*. De igual forma, codificar en resoluciones superiores a las que la cámara puede capturar no solo es ineficaz, sino que además exige interpolación de píxeles, haciendo que el proceso sea igualmente ineficiente.

Por último, las preferencias del usuario también son relevantes, como por ejemplo, elegir entre una mayor fluidez a cambio de menor resolución o una mejor calidad de imagen con un *bitrate* reducido puede llegar a ser crucial en contextos específicos.

## 3.2. Sistema ABR

La primera distinción que podemos hacer es sobre qué lado de la comunicación se encargará de estimar las capacidades de la red: *Receiver-driven*, si es el lado del cliente, o *Sender-driven*, si es el lado del servidor. Las tendencias actuales en *streaming* multimedia tienden a ser *Receiver-driven*, lo que permite que el servidor maximice su capacidad y ofrezca servicio al mayor número de clientes posible. En nuestro caso específico, optar por un enfoque *Receiver-driven* también permitiría liberar un porcentaje de la CPU que podría ser destinado a otras funciones. Por lo tanto, este proyecto se centrará en explorar alternativas *Receiver-driven*. Estas alternativas habitualmente siguen una estructura como la descrita en la figura 3.1. En este esquema, la estimación del *bitrate* requiere una combinación tanto de las capacidades del dispositivo como de las condiciones de congestión de la red.

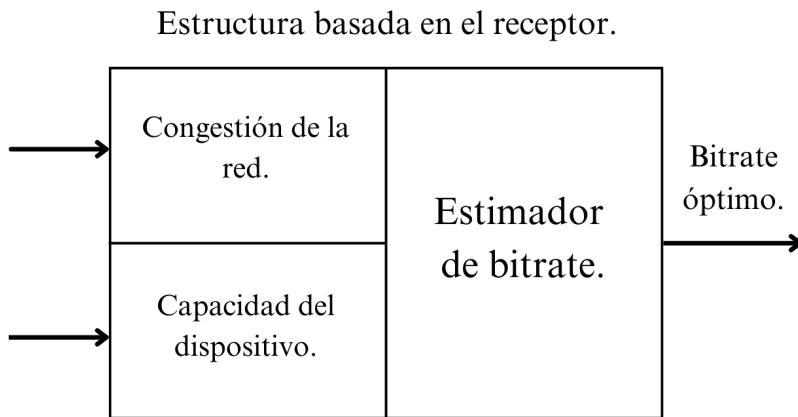


Figura 3.1: Diagrama que ilustra el funcionamiento de un sistema ABR basado en el cliente. Es importante resaltar que, en este enfoque,

### 3.2.1. Modelo de latencia

Además, a diferencia de los sistemas HAS, donde la mayoría de los algoritmos están diseñados para una estructura de esquema '*request-response*', adaptar esta estructura a un esquema de publicador-subscriptor de ROS 2 puede introducir diferencias significativas que afecten negativamente su rendimiento. Este cambio de estructura requiere una consideración detallada de cómo los algoritmos ABR pueden ser optimizados o modificados para funcionar eficientemente dentro de ROS 2.

Antes de profundizar en más detalles, es importante considerar ciertos aspectos sobre implementar el sistema ABR al *middleware* de ROS 2. Uno de los mayores beneficios de usar este sistema es que el esquema de publicador-subscriptor permite mantener un flujo continuo de mensajes en vez del sistema de fases ON-OFF común en

HAS, facilitando regular el número y el ritmo de los mensajes transmitidos. Este fue uno de los mayores desafíos enfrentados por la transmisión de contenido multimedia sobre HTTP en su día, que requirió de estándares y tecnologías como MPEG-DASH para ser resuelto.

Este hecho nos permite anticipar un modelo de recepción de mensajes ideal y estimar artefactos específicos en la capa de transporte. Podemos realizar un modelo total de la latencia como:

$$t_{\text{emisor receptor}} = t_{\text{GOP}} + t_{\text{codificación}} + t_{\text{canal}} + t_{\text{decodificación}} \quad (3.1)$$

Ecuación 3.1: Modelo simple de latencia en una transmisión de vídeo.

donde:

- $t_{\text{emisor receptor}}$  es el tiempo hasta que el nodo receptor es capaz de reproducir vídeo.
- $t_{\text{GOP}}$  es la latencia introducida por la necesidad de capturar y almacenar varios *frames* para poder codificarlos conjuntamente, el cual viene definido por el tamaño del GOP.
- $t_{\text{codificación}}$  es la latencia introducida por la codificación de la señal.
- $t_{\text{decodificación}}$  es la latencia introducida por la decodificación de la señal.
- $t_{\text{canal}}$  es la latencia en el canal de transmisión.

De la ecuación anterior, parámetros como el tiempo de GOP y de codificación pueden obtenerse desde el lado del servidor y transmitirse al cliente para mejorar sus estimaciones. El tiempo de decodificación también es medible en el lado del cliente y proporciona información compacta sobre las capacidades de procesamiento del sistema en ese momento. Las variables restantes están asociadas a elementos que influyen en el canal, como la distancia, la cobertura espacial de la posición del robot o errores puntuales, los cuales pueden caracterizarse mediante un análisis más profundo.

Otro aspecto favorable es que este modelo de comunicación resulta ser más eficiente en términos energéticos en comparación con el HAS. Esto se debe a que, al funcionar mediante un flujo continuo de vídeo, el cliente no requiere solicitar al servidor la descarga de segmentos, sino que únicamente envía la retroalimentación al servidor cuando es necesario.

### 3.2.2. Medición del tiempo y del throughput

La estimación precisa del tiempo es fundamental en estos sistemas para garantizar una sincronización y estimación de la latencia correcta, ya que incluso pequeñas variaciones de milisegundos pueden introducir errores significativos. La sincronización de los relojes entre los subsistemas de la red es especialmente crítica cuando un canal físico conecta dos nodos, ya que cualquier variación en la propagación del tiempo debe ser comunicada a los demás nodos. Sin embargo, la latencia de la red puede introducir desincronización en las comunicaciones. Para abordar estos desafíos, se proponen las siguientes estrategias.

#### Sincronización entre dos máquinas

Para lograr una sincronización precisa, se pueden utilizar diversas estrategias que generalmente son más precisas que la latencia de las comunicaciones de la red. No obstante, estas estrategias suelen basarse en suposiciones sobre la continuidad del tiempo, lo cual puede interferir en la simulación de entornos que utilizan ROS Time. Por tanto, es fundamental considerar estas particularidades al realizar experimentos simulados.

Una alternativa efectiva es el uso de fuentes de tiempo externas, como sistemas *GPS* (*Global Positioning System*) o sistemas sincronizados mediante NTP. Estas fuentes proporcionan una referencia temporal que sincroniza a todos los nodos que se suscriben al tópico '/clock', lo que ayuda a mitigar la desincronización provocada por la latencia en la transmisión de cambios temporales dentro de la red.

Además, es posible medir con precisión la latencia desde el momento en que se envía un mensaje hasta que es recibido, lo que permite caracterizar el retardo del canal. A partir de esta medida de latencia, se puede estimar la capacidad del canal de una manera sencilla: dividiendo el tamaño del mensaje entre el tiempo de transmisión, midiéndose en bits por segundo.

#### Medición relativa

En este enfoque, se utiliza una única máquina para realizar todas las mediciones de tiempo, eliminando la necesidad de sincronizar varios dispositivos. Sin embargo, esto conlleva ciertos inconvenientes, especialmente al medir procesos cíclicos sobre los cuales no se tiene control directo.

En términos de latencia, si el receptor está transmitiendo, no es posible calcular con exactitud el tiempo desde que el emisor envía un paquete hasta que es recibido, incluso

si el emisor registra la marca de tiempo del momento del envío, debido a la falta de sincronización entre los relojes del emisor y del receptor. La única forma de medir la latencia en este caso es utilizando el tiempo de llegada del paquete anterior y asumiendo una periodicidad de publicación, lo cual permite estimar la latencia relativa respecto al paquete anterior. Este enfoque establece una latencia mínima correspondiente al periodo de transmisión.

Como resultado, al aplicar este método para estimar la velocidad de transmisión, los valores obtenidos tienden a ser inferiores, limitados por el máximo teórico del tamaño del mensaje dividido entre la tasa de publicación. En lugar de calcular la capacidad total del canal, se obtiene una medida de eficiencia relativa comparada con un canal ideal, lo cual complica la caracterización precisa del canal.

### 3.2.3. Tipos de algoritmos ABR

El algoritmo ABR, encargado de evaluar tanto la calidad de la red como las capacidades del cliente, constituye el núcleo de esta tecnología, lo que hace esencial un análisis exhaustivo en esta área. Cabe destacar que la evolución de estos algoritmos ha estado fuertemente influenciada por las tendencias tecnológicas de cada época. En sus primeras etapas, las investigaciones se centraban en el desarrollo de heurísticas que representaran de manera efectiva la calidad de la conexión. Sin embargo, las tendencias actuales se inclinan hacia enfoques basados en modelado estadístico, que han evolucionado hasta integrar modelos de Machine Learning y Deep Learning, los cuales predominan en la literatura contemporánea.

Este enfoque más avanzado no solo permite maximizar el aprovechamiento de las mediciones, sino también clasificar y comprender con mayor precisión los eventos que ocurren en las capas subyacentes de la comunicación. Además, ofrece a los algoritmos mayores capacidades predictivas, lo que les permite anticiparse a cambios en la red y ajustar su comportamiento de manera más eficaz. Como resultado, estos avances han ampliado significativamente las capacidades de los algoritmos ABR, mejorando tanto la eficiencia como la adaptabilidad en entornos dinámicos.

A pesar de esto, estos algoritmos de machine learning suelen procesar el mismo tipo de *input* y generar el mismo tipo de *output* que los métodos tradicionales , por lo que no serán incluidos como una categoría independiente, sino que serán incorporados dentro de las categorías ya existentes ya que se considera que esta clasificación genera un mejor entendimiento del tipo de transmisión que se quiere llevar a cabo

Los algoritmos ABR se pueden dividir en cuatro categorías: basados en el estado del búfer, en el *throughput*, en otras heurísticas y mixtos, que combinan varios enfoques.

Además, pueden ser activos (intrusivos), añadiendo contenido a la red para realizar mediciones, o pasivos, que se basan únicamente en el contenido transmitido. El enfoque pasivo es más seguro, ya que evita saturar el canal cuando se opera al límite de capacidad. Actualmente, los algoritmos ABR han evolucionado hacia enfoques pasivos en la capa de aplicación, mientras que en los CCAs es más común ver enfoques activos, utilizando el *RTT* (*Round Trip Time*) para estimar la capacidad del canal y la latencia.

### Algoritmos basados en búfer

Estos algoritmos están diseñados específicamente para transmisiones de vídeo donde la captura y la transmisión del vídeo no se realizan en tiempo real, ya que se basan en la capacidad del cliente de almacenar en un búfer segmentos de vídeo futuros, como ocurre en plataformas de vídeo bajo demanda (VoD). Este tipo de algoritmos no son adecuados para nuestras necesidades ya que pueden llegar a introducir latencias en el sistema además de que el vídeo en directo implicaría trabajar con búfers pequeños que no permitirían una mayor capacidad de análisis, pero es relevante mencionarlos y explicar por qué se descartan para nuestro uso. Estos algoritmos toman decisiones de cambio de calidad principalmente basándose en el estado del búfer de reproducción del reproductor. La idea es mantener la salud del búfer para evitar el agotamiento (*búfer underflow*) que causaría rebuffering o pausas en la reproducción. Si el búfer está lleno o en un nivel saludable, el algoritmo puede decidir solicitar segmentos de mayor calidad. Si el búfer disminuye, podría optar por una calidad más baja para garantizar la continuidad de la reproducción. Ejemplos de estos algoritmos incluyen BBA ([Huang et al., 2014]) considerado el primer algoritmo en establecer la estructura base para este tipo de algoritmos, BIEB ([Sieber et al., 2013]), BOLA ([Spiteri et al., 2020])

Para el correcto uso de estos algoritmos es convenientemente entender las dinámicas entre la capacidad del sistema y el estado del búfer por lo que se recomienda leer la explicación dada en [Huang et al., 2014]. Ademas, otro trabajo muy interesante es el algoritmo ELASTIC ([De Cicco et al., 2013]), el cual aunque es un algoritmo híbrido, expande el entendimiento en las relaciones establecidas del búfer con el tiempo de descarga de los segmentos de vídeo

Según [Bentaleb et al., 2019], en general, los esquemas de adaptación basados en búferes sufren de numerosas limitaciones, incluyendo una calidad de experiencia (QoE) general baja y problemas de inestabilidad, especialmente en casos de fluctuaciones de ancho de banda a largo plazo. Estos algoritmos están diseñados para un tipo de flujo de vídeo diferente y, por tanto, no son compatibles con aplicaciones que requieren

transmisión y captura de vídeo en tiempo real, lo que justifica su exclusión en nuestro contexto.

### Algoritmos basados en Available Bandwidth o Throughput

Estos algoritmos están diseñados para predecir el *throughput* o ancho de banda disponible (*Throughput Prediction, TPP*) midiendo el tiempo de descarga de los segmentos de vídeo y ajustando la tasa de bits a las condiciones actuales de la red. La precisión en esta estimación es fundamental para su óptimo desempeño.

Estudios como los de Biernacki ([Biernacki, 2017]) y Lazaris ([Lazaris and Koutsakis, 2009]) han demostrado que el *throughput* sigue patrones de *Long Range Dependence (LRD)* y autosemejanza. De forma complementaria, trabajos como el de Xie ([Xie et al., 2016]) también han observado un comportamiento similar en canales LTE, asociado a factores físicos como la distancia y la potencia de las antenas. Esto sugiere la posibilidad de analizar los fenómenos físicos de las capas inferiores como un problema de evolución del *throughput* en el tiempo, mediante el análisis de series temporales.

Este enfoque está cobrando importancia en el contexto del aumento en el análisis de datos y se considera crucial para los algoritmos que dependen de esta técnica. Además, introduce el concepto de predictibilidad, lo que no solo permite estimar la relación del *throughput* con la capacidad total del canal, sino también anticipar futuras tasas y ajustarse de manera más eficiente. De esta forma, se pueden prever los efectos de los cambios en la calidad del vídeo solicitados por el sistema ABR, evaluando si los ajustes son adecuados. Este enfoque predictivo y adaptable promete mejorar notablemente la eficiencia y efectividad de los sistemas ABR en entornos dinámicos.

Retomando el enfoque autoregresivo, se parte del modelo más elemental e intuitivo, que consiste en predecir el *throughput* basado en la medición del último segmento descargado. Este enfoque asume una serie temporal constituida por una única muestra, representada por la siguiente relación:

$$\text{Throughput}_{N+1} = \text{Throughput}_N \quad (3.2)$$

Ecuación 3.2: Predicción de *throughput* en base al último segmento descargado.

Si avanzamos en términos de complejidad, la propuesta consistiría en analizar el *throughput* estimado a lo largo de una ventana temporal reciente al instante actual 'n' y calcular su media aritmética. Este proceso facilita el suavizado de la serie temporal, lo que ayuda a mitigar el impacto de valores atípicos. Además, proporciona múltiples

valores históricos para mejorar la precisión de las predicciones futuras, permitiendo una evaluación más detallada de las muestras dentro de dicha ventana para medir el *throughput* en el estado posterior ( $n+1$ ) y su variabilidad a través de la varianza.

$$\text{Throughput}_{N+1} = \frac{1}{N} \sum_{i=0}^N \text{Throughput}_i \quad (3.3)$$

Ecuación 3.3: Predicción de *throughput* en base a la media aritmética en una ventana temporal.

Aunque estos enfoques son intuitivos y han sido implementados con éxito en diversas ocasiones, han demostrado ser insuficientes frente a las cambiantes condiciones de la red. En respuesta a esta limitación, surge el reconocido algoritmo ABR FESTIVE ([Jiang et al., 2014]), que se distingue por introducir mejoras clave, entre ellas, el uso de la media armónica en lugar de la media tradicional para la estimación del *throughput*. Esta modificación ha ganado popularidad debido a su elevado rendimiento y ha sido adoptada en otros algoritmos, extendiendo su aplicación más allá del contexto original de FESTIVE.

$$\text{Throughput}_{N+1} = \frac{n}{\sum_{i=1}^n \frac{1}{\text{Throughput}_i}} \quad (3.4)$$

Ecuación 3.4: Predicción de *throughput* en base a la media armónica en una ventana temporal.

Estos tres predictores han sido ampliamente utilizados en la literatura académica, y existe un consenso general que sugiere que los períodos breves permiten capturar mejor la variabilidad en redes, especialmente en canales físicos inestables. Sin embargo, dada la naturaleza del problema, que esencialmente implica la predicción de una serie temporal, podrían implementarse otros enfoques, como el Promedio Móvil Exponencial, la Media Móvil Adaptativa de Kaufman o la regresión mediante Support Vector Machines (SVR). Por ejemplo, Biernacki ([Biernacki, 2017]) propone el uso de modelos ARIMA/FARIMA como alternativa para superar las limitaciones de FESTIVE.

Otra propuesta interesante es el algoritmo LOLYPOP presentado por Miller ([Miller et al., 2016]), que se centra en transmisiones en vivo y resulta particularmente relevante para canales físicos de *Radiofrecuencia* (RF), como los abordados en este estudio. Este algoritmo implementa una serie de ventanas de predicción de *throughput* de multiresolución (de 1 a 10 segundos), adaptando dinámicamente el tamaño de la ventana para alinearse lo máximo posible con el tramo temporal desde el inicio hasta

la finalización de la descarga, lo que permite una predicción más precisa. Además, gestiona factores críticos como la latencia máxima permitida para evitar el rebúfering, el número de segmentos omitidos y la variabilidad entre cambios de calidad, mejorando significativamente la experiencia de transmisión en entornos dinámicos.

Para garantizar una estimación precisa del *throughput*, es fundamental que los segmentos de vídeo descargados mantengan una longitud uniforme. Según trabajos como [Juluri et al., 2015], la velocidad media de descarga puede estar influenciada por la distribución del tamaño de los archivos. Por lo tanto, resulta ventajoso emplear una configuración de GOP fijo en el codificador o aplicar una ponderación basada en el tamaño de los segmentos, como se ha implementado en dicho estudio. Otros autores, como [Wang et al., 2016], han demostrado que el tamaño de los segmentos no solo afecta la precisión en la estimación del *throughput*, sino que distintos tamaños de segmentos pueden generar comportamientos significativamente diferentes en un mismo algoritmo.

Un aspecto adicional importante a considerar en ROS 2 para el cálculo del *bitrate* es que, debido a la capa de abstracción del *middleware*, no se puede medir el tamaño real de los bits transmitidos en la comunicación, sino únicamente los bits correspondientes al contenido de los mensajes ROS 2. Esto implica que la información sobre las cabeceras de los protocolos subyacentes o si el mensaje está fragmentado en varios paquetes de transporte no será accesible, lo que impide conocer el peso real de los paquetes o el *bitrate* exacto. Además, no es posible estimar el número de bits reales ni la fragmentación de los paquetes, ya que estos aspectos son gestionados íntegramente por la implementación del *middleware*. Actualmente, ROS 2 cuenta con diversas implementaciones de *middleware*, como 'Fast DDS', 'Connext DDS' y 'GurumDDS', y las características de gestión de paquetes pueden variar entre ellas. No obstante, la estimación de la tasa efectiva sigue siendo útil para el cálculo de la congestión de la red, y puede calcularse como el peso efectivo del paquete recibido en bits entre el tiempo medido frente a la marca de tiempo presente en la cabecera del mensaje.

Dependiendo del modelo de medición de la latencia, estos algoritmos son capaces de predecir el *throughput* o la capacidad del canal. No obstante, la implementación de la lógica de decisión para subir o bajar en la escalera de calidades puede ser ambigua, ya que las acciones a seguir no son tan evidentes como en los algoritmos basados en búfer.

Algunos métodos optan por solicitar una calidad de transmisión ligeramente inferior al umbral estimado, mientras que otros se centran en analizar la variabilidad de las mediciones y tomar decisiones basadas en la tendencia del comportamiento de la

conexión. Esta tendencia puede clasificarse como ascendente, descendente o estable, priorizando el análisis de la dinámica del sistema en lugar de valores numéricos puntuales. De este modo, se busca optimizar la calidad de la transmisión al considerar patrones de comportamiento más amplios en lugar de decisiones basadas únicamente en métricas instantáneas.

### **Basados en otras heurísticas**

En esta categoría se incluyen estrategias que, aunque basadas en diferentes heurísticas y potencialmente interesantes, no se han consolidado como estándar en el sector. No obstante, han sido parcialmente útiles para complementar los algoritmos actuales, destacando principalmente por su capacidad para caracterizar eventos en la red y apoyar a los algoritmos de categorías previas.

Normalmente, se emplean sistemas que almacenan datos en ventanas temporales mediante búferes de dimensiones preestablecidas, permitiendo que las características evolucionen de un estado escalar a uno secuencial. Este enfoque mejora la capacidad de análisis al incorporar una dimensión temporal. El manejo de estos búferes está estrechamente ligado al rendimiento del sistema, por lo que entenderlos desde una perspectiva de ciencia de datos resulta especialmente interesante.

El tamaño óptimo de los búferes implica un compromiso: aquellos de mayor tamaño favorecen una estimación más precisa de la capacidad de la red gracias a su amplia representatividad temporal. Sin embargo, esto requiere esperar a que el búfer se actualice con nuevos datos para proporcionar una respuesta al servidor, lo que puede ralentizar el sistema y resultar en una eficiencia reducida. Por otro lado, los búferes más pequeños, aunque ofrecen menor representatividad y tienden a dar resultados menos exactos y más erráticos, permiten una respuesta y adaptabilidad más rápidas.

Típicamente se extraen características que permiten parametrizar su rendimiento, tales como la latencia entre mensajes, el *jitter* y el número de paquetes descartados en comunicaciones en tiempo real y se buscan relaciones entre ellos como en la propuesta de [Pozueco et al., 2013] donde la linealidad de la latencia a través del coeficiente de Pearson. Además, propuestas como [Liu et al., 2011] la extensión de su trabajo en [Liu et al., 2012] han introducido características que relacionan la duración de segmentos temporales multimedia reales (como un segundo de vídeo o un GOP) con el tiempo de transmisión del contenido, los cuales pueden ser indicadores útiles del estado de la red. Este enfoque puede ser especialmente relevante en contextos de transmisiones en tiempo real. Existen tres tipos de características a considerar: características de encabezados de paquetes, características de carga y características de flujo de datos,

como se discute en [Wang et al., 2018].

Además, el conocimiento del *hardware* del robot puede enriquecer las características obtenidas al combinarlas principalmente con elementos de la capa física, tales como la Potencia de Señal de Referencia Recibida (RSRP), el Indicador de Calidad de Canal (CQI), y el Indicador de Fuerza de Señal Recibida (RSSI). A esto se suman características que permiten modelar el canal, como la Matriz Dominante del Canal y el Rango de Índices Únicos por Segundo. En un tercer nivel, aunque con menos éxito, se incluyen características de información del vehículo, como la velocidad del robot, la geolocalización y la altura de las antenas. Estas características físicas han demostrado ser capaces de estimar y predecir el rendimiento en la capa física de una red móvil de RF, como se evidencia en [Palaios et al., 2021].

De esta forma, los algoritmos que integran información tanto de la capa de red como de la capa de aplicación podrían superar los efectos del desconocimiento de las capas de red y transporte, que suelen ser abstractas por el *middleware* y sus características. Los algoritmos ABR que utilizan estimaciones de *throughput* y QoS a través de características de todo el *stack* representan una oportunidad considerable para la mejora de estos sistemas.

## Algoritmos híbridos

Las propuestas de esta categoría combinan múltiples enfoques de las categorías anteriores, para tomar decisiones de cambio de calidad. Un ejemplo de esto es el algoritmo implementado en Microsoft Smooth Streaming v1, que utiliza un análisis del búfer de vídeo para establecer límites máximos y mínimos de la calidad de imagen; para afinar un *bitrate* intermedio, añaden una estimación de *throughput* y una estimación de riesgo de saturación del búfer de reproducción ([Famaey et al., 2013]). También siguiendo este enfoque tendríamos PANDA ([Li et al., 2014]), que aunque principalmente está basado en *throughput*, utiliza el estado del búfer para ajustar el tiempo hasta la próxima solicitud de descarga de segmentos de vídeo. Si el búfer está por debajo de un cierto umbral, el algoritmo ajusta la tasa de bits para evitar el agotamiento del búfer y mantener la reproducción del vídeo sin interrupciones.

Además, destaca una creciente tendencia en algoritmos que combinan tanto la información del *throughput*, que introduce datos sobre la calidad del vídeo, como la información del búfer en un único modelo para la percepción subjetiva de la QoE que intentan maximizar. Un ejemplo de esto es el MPC, descrito en uno de los trabajos de referencia sobre el tema [Yin et al., 2015], que decide la próxima escala de calidad de la tasa de bits basándose en el estado actual del búfer y una predicción del *throughput*,

resolviendo un problema de maximizaron de QoE. Sin embargo, la sobrecarga de cálculo es considerable, lo que destaca la implementación de la variante FastMPC, que resuelve el problema de forma previa y almacena los resultados en tablas. Existe otra variante, ROBUSTMPC, que busca minimizar el impacto de uno de los factores clave: la predicción del *throughput*. Para ello, busca el mejor desempeño en el peor de los casos posibles dentro de un conjunto definido de variaciones posibles del *throughput*. Este artículo es relevante por proporcionar un entendimiento completo no solo de los sistemas ABR actuales sino también de fenómenos físicos que pueden influir en el rendimiento, destacando que, en sus experimentos, la predicción del *throughput* es peor en redes móviles, incluso utilizando la media armónica como predictor, destacando que este aspecto representa una limitación significativa del MPC.

Otro ejemplo podría ser Pensieve ([Mao et al., 2017]), siendo la propuesta que mas destaca dentro de los algoritmos de reinforcement learnning. Este sistema emplea una red neuronal convolucional (*Convolutional Neural Network (CNN)*) cuya función de recompensa se basa en la modelización de la calidad de experiencia (QoE), utilizando información sobre el *throughput* y el estado del búfer. Una característica notable de Pensieve es que, a diferencia de otros sistemas donde la calidad del vídeo es seleccionada por el cliente, en este caso es el servidor quien determina y selecciona la calidad óptima de transmisión tras recibir la retroalimentación del cliente. Dentro de la misma subcategoría de aprendizaje por refuerzo, se encuentra ABRL ([Mao et al., 2020]), que integra la predicción de *throughput*, el estado del búfer y la escala de *bitrate* como entradas para una red neuronal. Adicionalmente, ABRL incorpora técnicas avanzadas como la reducción de varianza y la optimización bayesiana restringida para gestionar la estocasticidad de las condiciones de la red.

Continuando con las propuestas de aprendizaje por refuerzo, se encuentra el trabajo presentado en [Chen et al., 2023], donde se emplea el algoritmo D3QN para procesar características como el *throughput* actual, el estado del búfer, el tiempo de descarga y el *bitrate* de los chunks anteriores, además del tamaño del próximo *chunk* y el tiempo de *rebuffering*. Este método se complementa con la predicción del próximo valor del *throughput* utilizando redes LSTM, dando lugar al algoritmo LD-ABR. Este enfoque ha demostrado ser eficaz en la transmisión de vídeo sobre canales de RF entre terminales base y móviles, mostrando mejoras significativas especialmente en situaciones de baja relación señal/ruido (*Signal-to-Noise Ratio (SNR)*).

Una limitación de los algoritmos basados en modelos de calidad de experiencia (QoE) radica en que, a pesar de la existencia de recomendaciones y modelos previos, cada autor tiende a asignar un peso diferente a los factores que influyen en la percepción,

resultando en una falta de consenso generalizado sobre cómo medir la QoE. Además, este enfoque no resulta adecuado para nuestro estudio debido a la dualidad en el tipo de consumidores de vídeo que se considera.

Otra subcategoría relevante dentro de los algoritmos híbridos son aquellos basados en teoría de control. Un ejemplo destacado es el algoritmo PIA ([Yanyuan et al., 2017]), que emplea un enfoque híbrido utilizando un control *Proportional-Integral-Derivative* (PID) para la gestión del búfer. La implementación de técnicas PID ha sido ampliamente cuestionada en este contexto debido a que están diseñadas principalmente para mantener el control del sistema y no necesariamente para optimizar la (QoE) ([Yin et al., 2015]). Sin embargo, se argumenta que el PID, si se aplica adecuadamente para mantener el búfer dentro de un rango óptimo, puede alcanzar como consecuencia una QoE óptima al estabilizar el búfer. En esta misma categoría, se encuentra el algoritmo ELASTIC ([De Cicco et al., 2013]), que modela la relación entre el *throughput*, el búfer y el nivel de calidad del vídeo mediante una ecuación diferencial no lineal con retardo. Este controlador se enfoca en mantener el búfer cerca de un punto de referencia predefinido, con el objetivo de evitar los patrones ON-OFF de HAS y mejorar la transmisión a través de optimizar utilización del ancho de banda.

Otra forma de establecer esta mezcla de algoritmos híbridos puede ser como la vista en DYNAMIC, la cual ofrece una doble estrategia similar a la vista en BBA ([Huang et al., 2014]) donde se usa un algoritmo basado en *throughput* en el momento transitorio pero cuando logra estabilizarse se mantiene utilizando el algoritmo BOLA basado en búfer ([Spiteri et al., 2020]) el cual fue la implementación por defecto en el estándar DASH durante años.

### 3.2.4. Estrategia de ajuste

Una vez estimada la calidad óptima del segmento, la elección de una estrategia de ajuste de *bitrate* es clave para evitar respuestas no deseadas en la transmisión. Los cambios abruptos en el *bitrate* pueden generar comportamientos exponenciales indeseados, como se ha observado en modelos estimadores previos.

Al estar limitados a *bitrates* específicos definidos por la escalera de *bitrate* del códec, no tenemos la flexibilidad de seleccionar un valor intermedio entre dos niveles disponibles. Por lo tanto, debemos optar por el *bitrate* más alto que no supere el valor óptimo estimado. Esta estrategia, de tipo '*stateless*', puede complementarse con restricciones adicionales para mejorar la estabilidad, equidad u otros objetivos de comunicación, lo que corresponde a una estrategia *stateful* ([Jiang et al., 2014]).

Como resultado, la precisión y adaptabilidad que podemos alcanzar están

condicionadas por las configuraciones del códec en el servidor, lo que restringe nuestra capacidad para hacer ajustes finos dentro de este marco estructurado.

En un extremo del espectro de posibilidades, encontramos la estrategia de rampa lineal. En esta, una vez estimado el *bitrate* ideal, el cliente solicita la configuración del códec más cercana, pero siempre inferior al valor estimado. Esta estrategia permite una adaptación rápida, ya que solo requiere una estimación para determinar el *bitrate* óptimo. No obstante, es la menos suave y puede generar respuestas caóticas en la red, afectando incluso las predicciones futuras. Por ello, su uso está orientado a sistemas de alta velocidad, donde la rapidez de respuesta es crítica, acompañada de un sistema predictivo optimizado para el caso.

En el otro extremo, están las estrategias escalonadas (*step-wise*). Aquí, al estimar un *bitrate* óptimo que abarca varios niveles superiores o inferiores, el sistema solo puede solicitar un cambio a un nivel adyacente. Esto implica que se necesitan varias rondas de estimación para alcanzar el *bitrate* óptimo. Aunque la respuesta es más lenta, dado que requiere múltiples rondas para ajustar el *bitrate*, esta estrategia es más suave y segura, ya que los cambios son menos agresivos, lo que facilita una estabilización más rápida de la red. Además, al basarse en estimaciones sucesivas, se puede refinar la precisión y mitigar el impacto de errores o valores atípicos que puedan surgir en rondas anteriores.

$$\begin{aligned} & \min_{x_n \in \{-k, -k+1, \dots, k\}} \quad \text{Escalera de Bitrate } [L_{n-1} + x_n] - \text{Bitrate Estimado}[n] \\ & \text{sujeto a } 0 \leq L_{n-1} + x_n \leq N \end{aligned} \tag{3.5}$$

Ecuación 3.5: Ecuación para la selección escalonada del nivel de *bitrate*

- $x_n$ : Número de escalones que puede ascender o descender en la escalera de *bitrate* para el instante n.
- $L_n$ : Nivel de *bitrate* seleccionado en el instante actual.
- $N$ : Índice máximo en la escalera de *bitrate*.

Entre estas dos filosofías, se pueden explorar enfoques intermedios, como limitar el número máximo de niveles que el sistema de *Adaptive Bitrate* (ABR) puede ajustar, implementar estrategias de búsqueda más sofisticadas o realizar ajustes dinámicos sobre la restrictividad del sistema en tiempo real.

La agresividad del cambio de ajuste se puede definir mediante un parámetro 'k', que representa la capacidad de salto entre niveles de *bitrate*. Un valor de 'k' igual a uno solo permitirá avanzar o retroceder una capa a la vez, mientras que en una estrategia de rampa lineal, k podría considerarse infinito, ya que no existirían restricciones.

### 3.2.5. Estrategia de arranque: tiempo transitorio

La estrategia de inicialización es un aspecto clave en los algoritmos de *bitrate* adaptativo (ABR). Al arrancar el sistema, la falta de información sobre la capacidad del canal exige un período inicial para evaluar la red y lograr estabilidad. Aunque esta fase transitoria puede ser subóptima, intentar minimizarla excesivamente podría comprometer la calidad del servicio, en comparación con estrategias menos eficientes pero que priorizan la calidad de experiencia (QoE). En nuestro caso, una pérdida de conexión que sature la comunicación entre nodos sería crítica, por lo que hemos optado por una estrategia conservadora que inicia con el *bitrate* más bajo y asciende gradualmente a través de los niveles disponibles.

Durante este periodo, un valor más elevado de 'k' permite ajustes más agresivos y podría reducir el tiempo de transición. Sin embargo, seleccionar un 'k' alto debe hacerse con cautela, ya que la limitada disponibilidad de datos iniciales puede llevar a una sobreestimación del *bitrate* óptimo en el primer ajuste, provocando congestión. Esto podría generar una cadena de subestimaciones y sobreestimaciones en descargas sucesivas, creando un ciclo de inestabilidad. Por tanto, aunque un 'k' alto puede acortar el periodo transitorio, también puede tener el efecto contrario si no se gestiona adecuadamente.

Por otro lado, una estrategia conservadora puede penalizar a los usuarios con mejor conexión, quienes experimentarían una calidad innecesariamente baja durante el periodo transitorio. La estrategia opuesta, de ajustes descendentes, podría resolver este problema, pero en la mayoría de los casos no compensa el riesgo, y se prefiere un enfoque tradicional. Entre ambos extremos, existen enfoques intermedios que aplican sistemas de búsqueda más avanzados.

Además, es posible ajustar dinámicamente el valor de 'k' mediante una estrategia progresiva que incremente dicho valor a medida que se recopilan más datos, permitiendo que el algoritmo adopte un enfoque más agresivo conforme aumenta la precisión de las estimaciones. Así, el sistema comenzaría con ajustes más conservadores, incrementando gradualmente la agresividad a medida que se completa el período transitorio.

Otro aspecto crítico en la gestión de sistemas ABR es el manejo de los períodos transitorios que ocurren al ajustar el códec en tiempo real. Estos ajustes tienden a desestabilizar la red, provocando períodos transitorios breves que afectan la estabilidad general. La magnitud de la inestabilidad es proporcional al número de cambios significativos (saltos en la escala de calidad del códec) realizados respecto a la configuración actual, siendo directamente proporcional al valor de k seleccionado. Esto

incrementa el tiempo necesario para la recuperación de la red. La representación de la figura 3.2 busca ilustrar este comportamiento con el fin de alcanzar una mejor comprensión de este aspecto. En esta figura se destaca como el impacto en la estabilidad depende tanto del tiempo que el sistema tarda en recuperar su equilibrio como de la forma de la curva de respuesta. Ambas variables son inciertas, ya que dependen en gran medida del sistema implementado y de la existencia de mecanismos para mitigar dicho impacto. No obstante, se asume que un mayor salto en el *bitrate* resultará en un impacto más significativo en ambas dimensiones.

Es fundamental considerar estos períodos transitorios y desarrollar mecanismos que eviten tomar como referencia el estado de la red en estos momentos críticos, ya que no reflejan fielmente las condiciones normales de comunicación. Ignorar esta precaución puede provocar un efecto cascada en el sistema. Por ejemplo, si no se vacían los búferes de información tras realizar un cambio en el códec, pueden acumularse datos correspondientes al *bitrate* anterior, al periodo transitorio y a la nueva configuración, complicando aún más la estabilidad de la red.

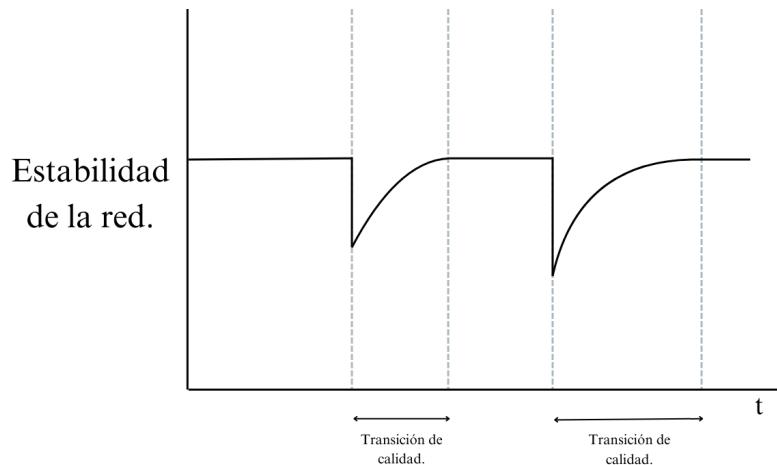


Figura 3.2: Representación que muestra las variaciones en la estabilidad de la red ante un cambio en la calidad del vídeo.

### 3.2.6. Tamaño de los segmentos de vídeo

El tamaño de los segmentos de vídeo es un factor clave a considerar en la implementación. La duración mínima de un segmento está determinada por el tamaño mínimo del GOP disponible, ya que los segmentos descargados deben ser decodificables de manera independiente. A partir de esta duración mínima, se pueden definir segmentos que contengan uno o varios GOPs, con la estructura general basada en la longitud del GOP y el número de GOPs incluidos en cada descarga. En sistemas tradicionales de transmisión adaptativa (HAS), los segmentos largos son beneficiosos

porque reducen el número de solicitudes al servidor y disminuyen su complejidad. Sin embargo, en nuestra situación, donde se transmiten segmentos de vídeo de forma constante, esta ventaja pierde relevancia, y los segmentos cortos son preferibles por su mejor adaptabilidad al canal.

El tamaño y la estructura del GOP es fundamental para la calidad del vídeo. Estructuras GOP más largas maximizan la predicción entre *frames*, reduciendo el *bitrate*, pero requieren la decodificación de todos los *frames* del GOP, lo que incrementa la latencia. Además, el tamaño ideal del GOP depende del tipo de contenido. vídeos dinámicos, con cambios frecuentes entre *frames*, se benefician de estructuras GOP cortas, mientras que contenidos más estáticos, con poco movimiento, aprovechan mejor las estructuras GOP largas.

La configuración del GOP se define por la presencia y distribución de tres tipos de *frames*: I, P y B. Los *frames* I (intra) contienen toda la información necesaria para ser decodificados por sí mismos y funcionan como puntos de referencia dentro del GOP. Los *frames* P (predictivos) almacenan solo los cambios en relación con *frames* anteriores, y los *frames* B (bi-predictivos) aprovechan tanto *frames* anteriores como futuros para lograr una compresión aún mayor.

Es posible optar por una estructura fija del GOP, que establece tanto su tamaño (la distancia entre *frames* I) como una disposición predeterminada de *frames* P y B. Alternativamente, una estructura variable permite ajustar la disposición de *frames* P y B según el contenido del vídeo, logrando un mejor equilibrio entre ventajas y desventajas al adaptarse a las variaciones de movimiento y complejidad en cada escena.

### 3.2.7. Consumo humano y consumo máquina

Uno de los principales desafíos en la implementación en un sistema robótico es la diferenciación entre el tipo de nodo consumidor del vídeo. Los códecs de vídeo actuales están diseñados en gran medida considerando las capacidades del Sistema Visual Humano (HVS). De esta manera, se busca eliminar información que resulta imperceptible para los humanos o minimizar su impacto visual.

Sin embargo, a diferencia de los consumidores humanos, las máquinas no están limitadas por el HVS. De hecho, pueden optimizarse para superar estas limitaciones y mejorar el rendimiento en áreas donde el HVS no es tan eficiente. Es fundamental identificar y preservar aquellas características del vídeo que son esenciales para las máquinas, garantizando que no se elimine información relevante para sus procesos. Un ejemplo destacado son las redes de deep learning para imágenes, que actúan como consumidores no humanos, capaces de extraer y procesar características visuales clave,

como bordes, texturas o patrones específicos, optimizando el análisis y la interpretación visual más allá de lo que el HVS podría percibir.

Aspectos cruciales como la resolución, la frecuencia de cuadros por segundo y la profundidad de bits en la codificación influyen significativamente en el desempeño de las redes neuronales. Un ejemplo notable de la importancia de la configuración del vídeo es el tipo de escaneo de cuadros empleado. El escaneo progresivo muestra cada línea de píxeles en cada cuadro, resultando en una imagen completa en cada *frame*. En cambio, el escaneo entrelazado divide la imagen en líneas pares e impares, alternando su transmisión en cada cuadro, lo que permite reducir a la mitad la cantidad de información por *frame*. Aunque los seres humanos perciben ambos cuadros como una única imagen gracias a la persistencia visual, este método puede generar artefactos en vídeos con contenido dinámico. Esta diferencia entre ambos tipos de escaneo se ilustra en la figura 3.3. Sin embargo, las redes neuronales, al procesar características visuales detalladas, tendrían dificultades para operar eficientemente en escenarios de escaneo entrelazado.

A pesar de que el uso del escaneo entrelazado ha disminuido debido a su antigüedad, siendo en gran medida reemplazado por el escaneo progresivo, aún se emplea en ciertos códigos por su eficiencia en la reducción del *bitrate* con un bajo costo de procesamiento. Esto sigue siendo relevante en dispositivos con capacidades limitadas y en situaciones donde el ancho de banda es un recurso restringido.

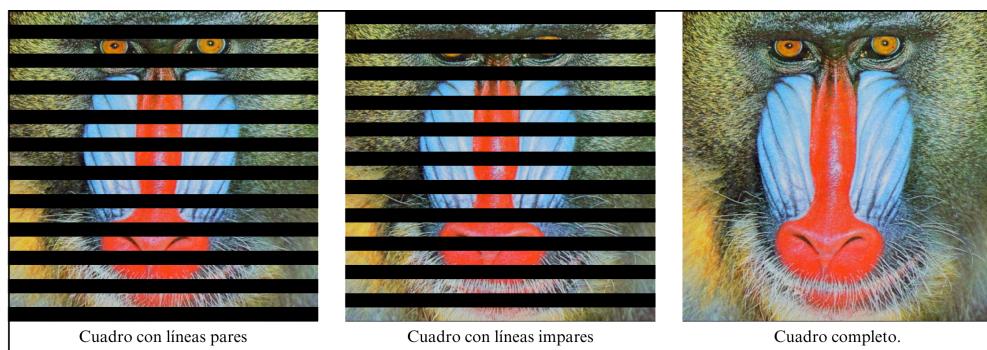


Figura 3.3: Comparación de un *frame* utilizando una representación de escaneo entrelazado frente al escaneo progresivo. Ilustración creada a partir de la reconocida imagen 'Mandrill' (alternativamente 'Baboon') proveniente de la USC SIPI Image Database ([USC SIPI, 2024]).

### 3.3. Comunicación entre nodos

En esta sección, se discuten los desafíos clave relacionados con la implementación del sistema de *Adaptive Bitrate Streaming* (ABR) desde una perspectiva estructural en

el entorno de ROS 2. Se explicará detalladamente el proceso de creación, configuración y coordinación de los mensajes dentro de la arquitectura ROS 2, enfatizando la importancia de una integración sistemática para asegurar una transmisión efectiva y eficiente en entornos ABR.

### 3.3.1. Configuración de la red

Para la implementación, utilizaremos dos *topics* distintos dentro del sistema. El principal estará dedicado a la transmisión de vídeo desde el nodo servidor al nodo cliente, mientras que el segundo gestionará una comunicación bidireccional entre ambos nodos. Este segundo canal será responsable de transmitir la retroalimentación necesaria para que el servidor ajuste el *bitrate*, además de manejar mensajes secundarios como los de tipo *handshake*. Esta configuración busca replicar un esquema orientado a sesiones a nivel de aplicación, similar a los protocolos RTP/RTCP/RTSP o QUIC.

Cada uno de estos canales de comunicación presenta requisitos distintos, especialmente al tratarse de una transmisión en vivo. Para el canal de vídeo, es fundamental utilizar protocolos que no requieran retransmisión de paquetes, como los basados en *User Datagram Protocol* (UDP), ideales para minimizar la latencia en la entrega de datos en tiempo real. Por otro lado, el canal de retroalimentación debe garantizar la fiabilidad de la transmisión, por lo que, en caso de pérdida de paquetes, estos deben retransmitirse rápidamente. Para ello, es necesario utilizar tecnologías basadas en TCP, que aseguren la correcta recepción de los ajustes críticos para el sistema.

La capa *middleware* DDS en ROS 2 utiliza el protocolo RTPS (Real-Time Publish-Subscribe), el cual es la base del estándar DDS (Data Distribution Service) adoptado por diversas implementaciones de RMW. RTPS, diseñado para aplicaciones en tiempo real, proporciona mecanismos eficientes para gestionar el tiempo y la calidad del servicio (QoS). Este protocolo opera principalmente en la capa de sesión del modelo OSI y se basa en subprotocolos y tecnologías de red, como UDP y TCP. No obstante, el manejo de los protocolos subyacentes en el *middleware* es invisible para la capa de aplicación y puede variar según la implementación del sistema DDS utilizado.

Por esta razón, ROS 2 ofrece a través de su sistema de *Quality of Service* (QoS) distintos parámetros que podemos configurar al iniciar el nodo para establecer ciertas preferencias en la comunicación de manera abstracta. Es el *middleware* el encargado de implementar estas preferencias, asegurando que se adapten a las necesidades específicas de cada canal dentro del entorno de comunicación. Estas son:

- **Reliability**

- *Reliable*: Garantiza la entrega de mensajes, retransmitiendo datos si no se reciben confirmaciones.
- *Best Effort*: No garantiza la entrega de mensajes; ideal para datos en tiempo real donde recibir información antigua puede ser inútil.

- **Durability**

- *Transient Local*: El nodo guardará un registro de al menos algunos mensajes y los proporcionará a los suscriptores que se unan más tarde.
- *Volatile*: No se conservan los mensajes después de que se entregan, adecuado para datos que se actualizan frecuentemente y no necesitan historial.

- **History**

- *Keep Last*: Solo se almacena un número definido de los mensajes más recientes.
- *Keep All*: Se intenta almacenar todos los mensajes, lo que puede requerir consideraciones de memoria importantes.

- **Depth** Configura el tamaño de la cola utilizada con la política Keep Last. Afecta cuántos mensajes se almacenarán antes de empezar a descartar los más antiguos.
- **Deadline** Permite especificar un período máximo entre mensajes; útil para detectar nodos que han dejado de publicar.
- **Liveliness** Define cómo se determina si un nodo sigue activo o no y qué tan frecuentemente debe confirmar su presencia.
- **Lease Duration** Tiempo después del cual un nodo se considera como no activo si no ha mostrado signos de vida.

Es importante señalar que estos parámetros no garantizan la implementación de protocolos específicos como TCP o UDP en las capas inferiores, por lo que es posible que la presencia de mecanismos de gestión de la capacidad del canal, como los CCA's, afecte al modelaje de la capa de aplicación de forma inesperada. Estos detalles son cruciales y requieren consideración, especialmente en entornos donde las sinergias entre estos mecanismos a nivel multicapa pueden tener un impacto significativo, como se aborda en estudios como [Matsumoto et al., 2020] o [Lebreton and Yamagishi, 2021]

### 3.3.2. Mensajería

Como se mencionó anteriormente, gracias a la estructura de ROS 2, tenemos la capacidad de establecer modelos de comunicación más similares a IPTV que a OTT. Esto nos permite ofrecer un mejor servicio de vídeo sin las limitaciones impuestas por la estructura HTTP/TCP clásica de los sistemas OTT. Por esta razón, se ha diseñado una comunicación a nivel de aplicación orientada a conexión a través del canal de datos, lo que requiere que se establezca una estructura de mensajes bien definida y conocida tanto por el lado del servidor como por el del cliente. Esta estructura facilita una interacción más fluida y eficiente, permitiendo un intercambio de datos más directo y controlado, optimizando así la entrega de contenido multimedia en tiempo real. En concreto tendremos:

- Handshake: El cliente enviará al servidor un mensaje de tipo handshake al servidor que contendrá todos los parámetros máximos como tasa de refresco, resolución máxima, *bitrate* máximo, preferencia de fluided, etc. A este mensaje, el servidor responderá con un True junto con un diccionario con la escalera de *bitrate* disponible ofertada por el códec, si ha encontrado al menos una configuración que se adapte a estos requisitos o False si no ha sido así junto con un mensaje de error.
- Status: Para el lado del cliente este tipo de mensajes contendrá la referencia a la nueva configuración del códec calculada a través del algoritmo ABR. En cambio, los mensajes Status del servidor se mandarán como respuesta confirmando la configuración aplicada en el códec de manera que el cliente pueda saber si el cambio se ha realizado y facilitar la estimación del tiempo de espera del algoritmo ABR para tomar mediciones después de que la red se haya estabilizado tras realizar el cambio de tasa de bits.

En ROS 2, la construcción de los diferentes tipos de mensajes se lleva a cabo mediante interfaces, plantillas predefinidas diseñadas para especificar el contenido y el formato de los mensajes. Estas interfaces pueden ser las estándar que ofrece ROS 2 o bien interfaces personalizadas, como las que se desarrollarán para este proyecto. La implementación de estas interfaces personalizadas se realizará utilizando un enfoque simple e intuitivo para adaptarse específicamente a las necesidades del proyecto.

## 3.4. Nivel físico

En esta sección se abordarán los principales canales físicos de comunicación a los que puede enfrentarse un sistema robótico. Seguiremos el modelo de robot mencionado

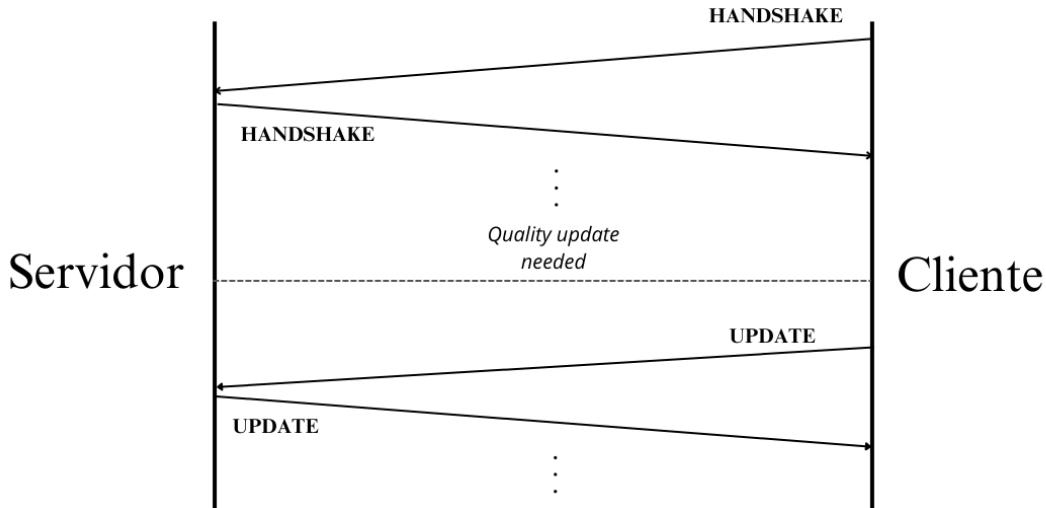


Figura 3.4: Diagrama de comunicación del canal de datos entre el cliente y el servidor, exemplificando cada uno de los procesos que pueden ocurrir.

anteriormente, que presenta una alta densidad de nodos de comunicación (sensores, componentes móviles, etc.), todos ellos funcionando como un conjunto de sistemas embebidos bajo un mismo *hardware*, lo que denominaremos comunicaciones internas. También se describirán los flujos de comunicación que permiten la interacción de este sistema embebido con otros nodos a mayor distancia, denominadas comunicaciones externas.

### 3.4.1. Comunicaciones internas

Las comunicaciones internas de un robot ocurren dentro de su propio *hardware*, que suele estar compuesto por varios microprocesadores y sistemas embebidos, coordinados por un nodo central encargado de gestionar todas las operaciones. Estas comunicaciones se realizan a través de cables eléctricos, como pares trenzados, coaxiales o pistas de circuitos impresos (PCB), utilizando protocolos como I2C, SPI, UART, HDMI, Ethernet o USB. Como las distancias son cortas y el *hardware* se puede diseñar a medida, el reto principal es asegurar que no haya interferencias electromagnéticas (EMC) que comprometan su correcto funcionamiento.

### 3.4.2. Comunicaciones externas (radio frecuencia)

Las comunicaciones entre un robot y un nodo externo suelen realizarse a través de radiofrecuencia, dado que el uso de conexiones no inalámbricas es poco práctico debido a la movilidad del robot. Sin embargo, en ciertos casos específicos, podrían existir excepciones que ROS 2 está diseñado para manejar de forma eficiente.

ROS 2 proporciona un marco común de programación para una amplia variedad de

sistemas robóticos, lo que es crucial para adaptarse a la flexibilidad que requieren distintas tareas en contextos específicos. Esta diversidad de escenarios hace difícil establecer un único modelo de propagación para las comunicaciones.

Aunque, como se ilustrará más adelante, cada robot requiere un análisis específico, el sistema adaptativo propuesto debe ser lo suficientemente flexible para funcionar en una amplia variedad de escenarios. Al situarse el sistema ABR en la capa de aplicación, nos enfocaremos en los aspectos generales de las comunicaciones por radiofrecuencia (RF) y cómo estas pueden ser gestionadas desde dicha capa.

En transmisión RF, el objetivo es maximizar la potencia recibida en las antenas y minimizar el ruido. Si bien una señal más débil no implica necesariamente la pérdida de la conexión, sí afecta la tasa de bits transmitidos, ya que los mecanismos de codificación de canal protegen la señal. No obstante, si la potencia cae por debajo de un umbral crítico, según las capacidades del *hardware*, la señal se vuelve irrecuperable, lo que provoca una pérdida total de la comunicación entre los nodos.

La propagación de las ondas electromagnéticas (*Electromagnetic (EM)*) depende del entorno, el canal y los obstáculos presentes, lo que puede influir significativamente en la calidad de la transmisión. Sin embargo, existe una pérdida mínima de propagación aplicable a todos los escenarios, descrita por la ley de Friis para el espacio libre:

$$L = \left( \frac{4\pi d f}{c} \right)^2 \quad (3.6)$$

Ecuación 3.6: Ecuación de Friis para calcular la pérdida de propagación en espacio libre.

donde:

- $L$  es la pérdida de potencia por propagación en el espacio libre.
- $d$  es la distancia entre el transmisor y el receptor (en metros).
- $f$  es la frecuencia de la señal transmitida (en hertz, Hz).
- $c$  es la velocidad de la luz en el vacío ( $3 \times 10^8$  m/s).

Esta ley describe cómo la potencia de la señal emitida disminuye con el cuadrado de la distancia, incluso en condiciones ideales. Este fenómeno representa un riesgo significativo para robots en movimiento que potencialmente pueden alejarse del elemento con el que han establecido conexión, especialmente cuando la calidad de la comunicación es crítica.

En situaciones como esta, los sistemas de *bitrate* adaptativo son clave, ya que ajustan la calidad del vídeo según las consecuencias derivadas de las variaciones en la potencia de la señal, lo que permite mantener la comunicación a mayores distancias o en condiciones adversas. Un ejemplo claro es el uso de robots en zonas peligrosas para los humanos, donde son operados a distancia desde un nodo de control. La capacidad de ajustar el *bitrate* no solo garantiza la continuidad de la comunicación, sino también el correcto funcionamiento del robot. Esto es crucial, ya que tiene el potencial de determinar el éxito de la misión, permitiendo que el robot regrese de forma segura a su base o, en el peor de los casos, evitando una pérdida total de comunicación.

Es relevante diferenciar entre las comunicaciones por RF en entornos interiores y exteriores, ya que la geometría tridimensional de los espacios cerrados influye significativamente en la calidad de la señal. A diferencia de los entornos exteriores, donde la propagación de las ondas es más directa, en interiores las reflexiones y los obstáculos, como muebles, personas o incluso el propio robot, alteran la potencia de la señal, generando un entorno más difuso y con fluctuaciones más notables en los niveles de señal.

Para abordar estos retos, se han desarrollado varios modelos de propagación específicos para interiores, como el Log-Normal Shadowing Path Loss Model, COST 231, ITU-R, Keenan-Motley Model y el Multi-Wall Model. Estos modelos calculan las pérdidas de señal en función de la distancia, incorporando ajustes por el número y tipo de paredes o suelos atravesados, que afectan la permeabilidad de los materiales. Estos factores son esenciales para caracterizar correctamente la propagación en entornos interiores ([International Telecommunication Union, 2002]).

En entornos domésticos, como hogares o pequeñas empresas, es habitual el uso de bandas de frecuencia de espectro libre, que suelen estar entre 433 MHz y 5 GHz. Tecnologías como Wi-Fi y Bluetooth son predominantes, lo que simplifica la implementación al evitar la necesidad de utilizar estándares más complejos en bandas superiores. En cambio, en grandes empresas con operaciones más avanzadas, como inventarios masivos o cadenas de montaje complejas, se opta por redes industriales 5G o 6G, diseñadas específicamente para estos entornos. Estas redes garantizan un flujo constante y robusto de grandes volúmenes de datos, adaptándose a las exigencias del espacio y la maquinaria utilizada.

En entornos exteriores, los robots utilizan arquitecturas móviles como LTE, 5G o 6G para conectarse a la estación base más adecuada mientras se desplazan, operando en canales multirayecto. Es esencial analizar el terreno para garantizar una cobertura que satisfaga las necesidades del robot, basándose en modelos de comunicaciones móviles.

Estas comunicaciones exteriores se caracterizan por mayores distancias y la presencia de múltiples trayectorias, donde las reflexiones pueden afectar la señal de manera constructiva o destructiva según la latencia relativa al rayo directo. Esta dinámica puede inducir fenómenos como el desvanecimiento selectivo en frecuencia, especialmente importante en modulaciones OFDM, el efecto Doppler, especialmente notable en robots que se mueven a altas velocidades y otros artefactos característicos de este tipo de comunicaciones.

Muchas de las problemáticas relacionadas con la transmisión por RF se abordan mediante técnicas avanzadas de modulación y codificación de señales a nivel de transporte, utilizando estándares que regulan estas comunicaciones. Además, muchos sistemas modernos ya emplean esquemas adaptativos que ajustan la codificación en esta capa, logrando un balance entre la redundancia transmitida y la velocidad de bits para optimizar el rendimiento. Un sistema ABR no reemplaza estas tecnologías, ya que no introduce redundancia para proteger la información ni opera directamente en la capa de transporte. Sin embargo, en la capa de aplicación, se pueden observar efectos de las capas inferiores, y el sistema ABR podría optimizar el rendimiento general.

Por otro lado, los modelos tradicionales de redes móviles y arquitectura de red fueron desarrollados para usuarios terrestres, lo que los hace aplicables a robots que operan en la superficie. Sin embargo, con el creciente uso de UAVs, estos modelos son subóptimos en situaciones de comunicación aire-tierra y aire-aire.

En la comunicación aire-tierra entre un UAV y una estación base, la propagación tiene menos obstáculos, pero depende más del rayo directo y de la correcta orientación de las antenas. Esto es un reto cuando se usan estaciones base públicas, en lugar de una infraestructura dedicada, ya que las estaciones base suelen estar diseñadas para propagar horizontalmente y ligeramente hacia abajo, lo que no es ideal para robots aéreos que puedan alcanzar grandes altitudes.

La comunicación aire-aire, por su parte, se asemeja más a una propagación en espacio libre, con algunas reflexiones dependiendo de la altitud. Estas problemáticas, junto con los procesos y consideraciones relevantes, están detalladas en el trabajo de [Daniel Bonilla Licea and Saska, 2024].

---

# Capítulo 4

# Diseño

---

En este capítulo se abordarán principalmente las cuestiones técnicas necesarias para el desarrollo del proyecto, profundizando en los desafíos técnicos y matemáticos que se han resuelto para asegurar una implementación efectiva. Este capítulo está estructurado en tres partes, comenzando con una visión general del diseño, seguido por detalles específicos del diseño en el lado del cliente y, finalmente, en el lado del servidor.

Es importante destacar que todo el código desarrollado en el marco de este proyecto será publicado de manera abierta en la plataforma GitHub, permitiendo así acceso y colaboración continua. El código esta disponible en el siguiente enlace: [Álvaro Gutiérrez García, 2024b].

## 4.1. Visión general

En esta sección se analizarán los conceptos fundamentales que comparten ambos extremos de la comunicación en ROS 2. Cabe destacar que ROS facilita estas comunicaciones principalmente mediante nodos, concepto que hemos mencionado con frecuencia para simplificar las discusiones previas. Sin embargo, ROS 2 también ofrece la posibilidad de establecer comunicaciones a través de *plugins*. Aunque estos requieren una sintaxis de código distinta, se integran en el sistema como componentes reutilizables por otros nodos y permiten una intercambiabilidad en tiempo real con otros *plugins*. Aunque la implementación podría haberse realizado utilizando cualquiera de las dos estrategias, se ha decidido optar por una estructura basada en *plugins* en lugar de nodos, ya que es el enfoque empleado en la solución nativa de ROS 2 mediante el paquete *image\_transport* ([Mihelich and Carroll, 2009]), replicando su modo de funcionamiento.

Adicionalmente, con el objetivo de alinearse con los desarrollos actuales de la comunidad ROS, este trabajo se ha basado en una implementación de código abierto ya existente de los codificadores H264 y H265 en formato *plugin*, aunque

sin retroalimentación adaptativa implementada. El repositorio de esta implementación está disponible en GitHub [Utilities, 2024], concretamente en la rama 'Humble'. Esta solución utiliza los codificadores H264 y H265 a través de la biblioteca FFmpeg, una plataforma multimedia también de código abierto, ampliamente reconocida por su versatilidad y capacidad para manejar una amplia gama de formatos y códecs [Developers, 2024].

La decisión de unirse a los esfuerzos de la comunidad garantiza que se parta de estructuras de código familiares, facilitando la comprensión del proyecto por parte de desarrolladores ya familiarizados con el código base, y asegurando que el resultado final sea una contribución útil y comprensible.

Otra de las cuestiones a tratar es el numero de componentes en el que vamos a abstraer la implementación. Aunque previamente se mencionaron dos roles principales, servidor y cliente, el desarrollo se descompone en realidad en tres bloques esenciales: el codificador, ubicado en el servidor; el decodificador, en el cliente; y el sistema ABR , que se puede implementar tanto en cliente, como en el servidor, como un nodo tercero. La ubicación del componente ABR dependerá principalmente del tipo de datos que va a consumir y su proximidad al nodo generador, dado que si se limita a utilizar información adquirida en el lado del cliente, como en la mayoría de los algoritmos descritos en la sección 3.2.3, enviar esta información de vuelta al servidor implicaría una latencia adicional proporcional al canal que deba atravesar. Aunque esta latencia no es crítica, tampoco es necesaria. Del mismo modo, si se requiere el uso de información proveniente de los sensores del robot para adoptar un enfoque similar al presentado en [Palaios et al., 2021], entonces sería más conveniente situar este componente físicamente cerca de los nodos que emiten dicha información, es decir, cerca del servidor.

A su vez, el codificador y el decodificador deben optimizar el rendimiento, puesto que la velocidad de codificación y el consumo de recursos afectan directamente la calidad del vídeo recibido. Por esta razón, su implementación se realizará en C++, un lenguaje conocido por su eficiencia en rendimiento y gestión de memoria. Por otro lado, el algoritmo ABR no requiere respuestas ni cálculos tan rápidos, dado que necesita tiempo para recopilar datos y ofrecer una respuesta. Este componente ofrece mayor flexibilidad en su implementación. De hecho, la tendencia actual hacia el uso de técnicas estadísticas y de aprendizaje automático sugiere programar este componente en lenguajes de alto nivel como Python, que dispone de múltiples librerías que facilitan la implementación de estos algoritmos de manera sencilla y acorde con la filosofía de ROS 2.

Por lo tanto, se ha decidido que, para la versión actual abordada en este trabajo, el lado del cliente incluya un algoritmo ABR integrado, desarrollado en C++, lo que elimina la necesidad de implementar soluciones adicionales más allá de los dos *plugins* cliente-servidor, como se muestra en la figura 4.1. No obstante, en futuros trabajos sería interesante modificar el código para permitir desactivar este sistema ABR a través de un parámetro de entrada, de manera que las métricas de red se publiquen en un tópico externo. Esto permitiría que un nodo externo analice la red, implemente su propio algoritmo ABR en el lenguaje de su preferencia y responda al *plugin* del codificador.

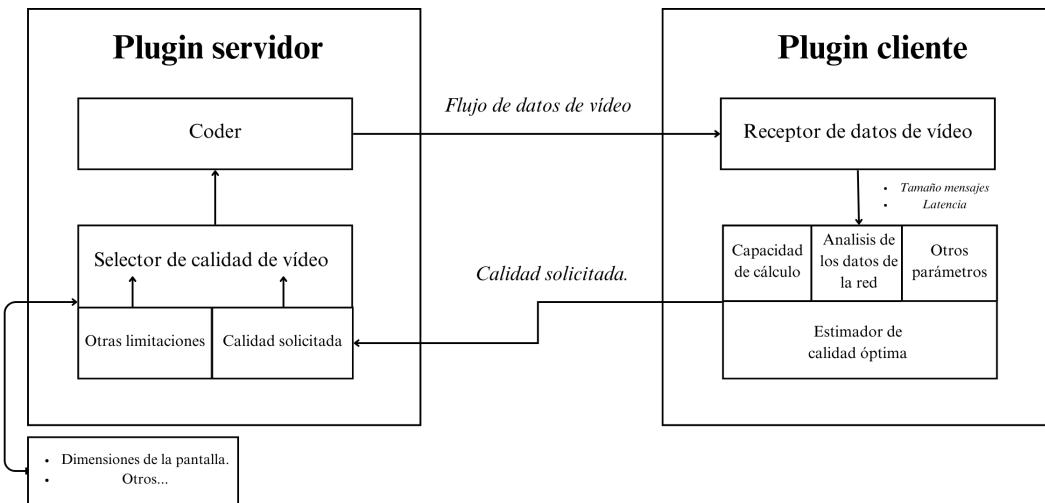


Figura 4.1: Diagrama simplificado de la mecánica principal de ajuste llevada por el sistema ABR.

Respecto a la configuración de la transmisión del vídeo, utilizaremos una configuración 'Best Effort', 'Volatile', y 'Keep Last', con un tamaño de búfer de paquetes de 0. Esto significa que buscamos una entrega de datos rápida sin garantías de entrega o almacenamiento de mensajes en caso de que el suscriptor no esté disponible para recibirlos inmediatamente. En contraste, para el canal de datos utilizaremos una configuración 'Reliable', 'Transient Local', 'Keep Last' y un 'depth' de búfer mayor a 1, específicamente se ha elegido un 'depth' de 10. Esta configuración asegura la fiabilidad en la entrega de los mensajes y mantiene una copia de los últimos diez mensajes en el búfer para nuevos suscriptores o en caso de pérdida de paquetes. Los parámetros de 'Deadline', 'Liveliness' y 'Lease Duration' afectan principalmente a cómo el *middleware* identificará como activos o no los nodos, por lo que se han dejado los valores por defecto en ambos casos.

Profundizando en los detalles de la comunicación, en ROS 2, la construcción de los diferentes tipos de mensajes se lleva a cabo mediante interfaces, las cuales son plantillas predefinidas diseñadas para especificar el contenido y el formato de los mensajes. Estas

interfaces pueden ser las estándar que ofrece ROS 2 o bien interfaces personalizadas, como las que se desarrollarán para este proyecto. La implementación de estas interfaces personalizadas se realizará utilizando un enfoque simple e intuitivo para adaptarse específicamente a las necesidades del proyecto.

Para el *topic* de vídeo, se reutilizará la interfaz ya implementada en [Utilities, 2024] ya que permite una integración eficiente y efectiva del flujo de vídeo dentro del ecosistema de ROS 2, asegurando que los datos del vídeo se manejen de manera óptima.

En cuanto al *topic* de datos, se adoptará una interfaz más flexible diseñada para manejar dos tipos de mensajes distintos. Esta interfaz incluirá un componente de tipo *string* que identifica el rol del remitente, ya sea servidor o cliente, y un identificador de tipo de mensaje en formato uint8. Los valores se asignarán de la siguiente manera: el valor 0 corresponderá a un mensaje de tipo 'handshake', el 1 a un mensaje de tipo 'estatus'. Además, incluirá un segundo *string* con el contenido del mensaje en formato JSON, lo que facilitará su decodificación y manejo en ambos extremos de la comunicación.

---

Listing 4.1: Data topic Interface Definition

---

```
# Data topic Interface
string role # Server or Client
uint8 msg_type # 0=Handshake/Requirements, 1=Status
string msg_json # JSON formatted message
```

---

Código 4.1: Definición de la interfaz para el *topic* de datos.

#### 4.1.1. Rol servidor

El rol del servidor se caracterizará específicamente por la configuración del codificador y la definición de la escalera de *bitrate*. Se establecerá una escalera de *bitrate* tradicional basada en la regulación del parámetro '*bitrate*' ofrecido por el códec libx264, en una configuración recomendada para permitir un ajuste dinámico y óptimo al valor objetivo especificado. Sin embargo, uno de los principales inconvenientes de esta configuración es que el *bitrate* puede verse afectado significativamente por el contenido de vídeo que se esté transmitiendo, incluso cuando se mantienen constantes la resolución y la tasa de *frames*. Esto puede resultar en variaciones de calidad visual para una misma configuración. Para mitigar este efecto, se puede definir un rango de valores dentro del cual el códec pueda ajustarse tanto por encima como por debajo del valor objetivo, aunque esto puede introducir incertidumbre e irregularidad en la red. Por esta razón, se ha optado por restringir la oscilación del códec a  $\pm 1\%$  de la configuración activa.

Otras configuraciones relevantes son de la implementación del códec por [Developers, 2024] que son los parámetros '*tune*' y '*preset*'. El parámetro *tune* ajusta el comportamiento del codificador para optimizar la calidad del vídeo en función del tipo de contenido. Entre las configuraciones disponibles, como '*film*', que optimiza para contenido cinematográfico, o '*stillimage*', diseñado para imágenes estáticas o secuencias con poco movimiento, destaca la opción '*zerolatency*'. Esta configuración es ideal para aplicaciones en tiempo real o aquellas que requieren la mínima latencia posible, como en el *streaming*. En nuestro caso, '*zerolatency*' ha demostrado ofrecer el mejor rendimiento en términos de sensación de inmediatez, superando considerablemente al resto de opciones. Por esta razón, se ha seleccionado como la configuración predeterminada, dado que es esencial para la aplicación descrita. Como consecuencia, se desactivan los *frames* B, resultando en una estructura de GOP con un primer *frame* I seguido exclusivamente de *frames* P, entre otros ajustes internos del codificador.

Por otro lado, el parámetro *preset* también ha demostrado ser clave, aunque en menor medida que *tune*, ya que controla la velocidad de codificación en relación con la calidad del vídeo. Partiendo del valor por defecto '*medium*', se presentan opciones como '*ultrafast*', '*superfast*', '*veryfast*', '*faster*', y '*fast*', que ofrecen mayor velocidad de codificación a costa de una menor eficiencia, lo que genera un mayor flujo de datos. En el extremo opuesto, opciones como '*slow*', '*slower*', '*veryslow*', y '*placebo*' priorizan la calidad, pero a cambio de una mayor carga de recursos y tiempos de procesamiento. Es importante destacar que '*placebo*' no se recomienda, ya que consume una cantidad significativa de recursos sin ofrecer mejoras notables en la calidad respecto a '*veryslow*'.

La selección de este parámetro puede variar considerablemente según los recursos de *hardware* disponibles, ya que implica un compromiso entre velocidad de codificación, latencia, y eficiencia. *Hardware* más potente puede manejar mejor este equilibrio, permitiendo mayores rendimientos en términos de ancho de banda. Además, estas configuraciones de velocidad no están disponibles para todos los perfiles; por ejemplo, '*ultrafast*' no es compatible con perfiles avanzados como '*high*', estando limitado a perfiles más básicos, lo que es relevante en escenarios donde los robots cuentan con capacidades computacionales reducidas. Finalmente, se ha optado por la configuración '*medium*', que es el valor predeterminado del códec y ofrece un equilibrio razonable entre velocidad y calidad.

Adicionalmente, entre otras configuraciones implementadas, se ha dado un amplio margen de factor de compresión '*q*' del rango de 5 a 50 y también se ha optado por activar el filtro '*deblock*' con el fin de sacar el máximo partido a las calidades visuales más bajas. También se ha tomado la decisión de no utilizar la extensión SVC debido

al considerable aumento del costo computacional que implica la codificación de vídeo en tiempo real

Al analizar el GOP (Group of Pictures), su tamaño debe ajustarse en función del *framerate*, ya que *framrates* más altos permiten utilizar GOPs más largos sin afectar significativamente la latencia. Por ejemplo, a un *framerate* de 120 fps, un GOP de 10 genera una latencia mínima de 83 milisegundos, mientras que a 10 fps la misma configuración alcanzaría una latencia de 1 segundo. Para una configuración óptima, se recomienda que el tamaño del GOP esté entre 1 y 10, dependiendo de los requisitos de latencia y el *framerate* de la aplicación.

Un GOP de tamaño 1, en el cual todos los cuadros son *frames* I, elimina la codificación de redundancia temporal, minimizando así la latencia. Esta configuración es ideal para aplicaciones que requieren tiempo real absoluto, aunque demanda un ancho de banda elevado. En contraste, un GOP de hasta 10 *frames* permite mantener una latencia aceptable y mejorar la eficiencia de compresión.

La elección del GOP debe equilibrar la latencia, la tasa de cuadros y la calidad de vídeo deseada según los objetivos específicos de cada aplicación. En este caso particular, se ha optado por un GOP de tamaño uno para minimizar la latencia.

Por último, el servidor también es responsable de analizar, al inicio de la comunicación, la resolución recibida junto con el *framerate* de los datos en bruto que está recibiendo. Esta información se utiliza para filtrar la escala del *bitrate* y descartar resoluciones incompatibles con los niveles y perfiles seleccionados, mejorando así la eficiencia y evitando problemas de compatibilidad.

Sin embargo, el *framerate* se recomienda encarecidamente que se preconfigure de forma manual este valor en el archivo 'config.json' provisto en función de la cámara a la que va a ser conectada ya que esta información es vital para el correcto funcionamiento del sistema ABR y su estimación puede sufrir ligeras variaciones afectando al rendimiento del sistema.

## Niveles y perfiles

Aunque el sistema implementado se abstrae de la elección de nivel y perfil de forma dinámica dejando al códec ajustándose de forma óptima, es posible que en determinados contextos multimedia, la fijación y elección de estos parámetros esté sujetas a tanto limitaciones que el *bitrate* puede extraer, resoluciones y capacidades en tiempo real. De necesitar modificar el código para solventar esto, se mencionan las tablas A-1 y A-2 del documento [Union, 2021]. La tabla A-2 clasifica las configuraciones de perfiles disponibles, donde los perfiles de mayor complejidad ofrecen una mayor eficiencia de

codificación a cambio de un costo computacional más elevado, mientras que los perfiles menos complejos resultan en un mayor *bitrate*. Estos perfiles se dividen en cuatro categorías, cada una de las cuales ajusta al alza el *bitrate* máximo del códec. Por otro lado, la tabla A-1 proporciona una visión global de las diferentes capacidades técnicas de cada nivel, destacando aspectos como la 'Velocidad máxima de procesamiento de macrobloques' y la 'Tasa máxima de bits de vídeo'.

También será necesario familiarizarse con el factor 'cpbBrVclFactor', el cual es un coeficiente que cuantifica el incremento en el *bitrate* debido al uso de codificadores con perfiles específicos que incorporan características de imagen menos restrictivas. Por lo tanto, es fundamental multiplicar el *bitrate* base de cada nivel, estipulado en la tabla A-1, por el 'cpbBrVclFactor' para determinar la cantidad de datos por segundo reales que presentará cada representación en la escalera.

Para obtener una visión completa de la información relevante de ambas tablas, se ha elaborado una nueva tabla que condensa todo este contenido en una sola. Dado su tamaño, esta tabla se encuentra en el Anexo 1 en el cuadro A.1.

No obstante, esta tabla es orientativa, ya que solo es válida para contenido de vídeo precodificado. En el caso de trabajar en tiempo real, tendremos una limitación impuesta por el número máximo de macrobloques por segundo que el códec es capaz de procesar para un nivel determinado, información que se encuentra en la columna 'Velocidad máxima de procesamiento de macrobloques' de la tabla A-1. Esta columna se presenta de forma resumida en el Anexo 1, en el cuadro A.2. Dado que en H.264 el tamaño máximo de macrobloque es de 16x16, podemos calcular el número de macrobloques necesarios que nuestro codificador requiere utilizando la siguiente fórmula:

$$\text{MB/s} = \frac{\text{framerate} \times (\text{Ancho} \times \text{Alto})}{16 \times 16} \quad (4.1)$$

Ecuación 4.1: Fórmula para el cálculo del número de macrobloques por segundo.

Dado que la representación en macrobloques está definida por los valores máximos indicados en la tabla A.2, se ha realizado una versión de la tabla A.1 que opera en tres modos: uno para una tasa de fotogramas inferior a 24 fotogramas por segundo, otro que funciona hasta 60 fotogramas por segundo, y un último modo que alcanza hasta 120 fotogramas por segundo. De esta forma, se mantiene la consistencia y se limita la posibilidad de que el codificador colapse en tiempo real al exceder la velocidad máxima de procesamiento de macrobloques. Esta nueva restricción hace que la tabla A.1 quede obsoleta y se requiera una versión aún más restrictiva. Esta segunda tabla actualizada también está documentada en el anexo, como A.4.

### 4.1.2. Rol Cliente

#### Decoder

En la implementación del lado del cliente, se integrará un decodificador capaz de interpretar y convertir mensajes codificados a una interfaz de vídeo ROS 2, asegurando su correcta utilización. Esta implementación seguirá una metodología similar a la empleada en el codificador y se desarrollará utilizando la biblioteca FFmpeg. El objetivo principal es que el consumidor final del vídeo no perciba visualmente ningún impacto derivado de las operaciones ejecutadas ni de los *plugins* intermedios.

#### Algoritmo ABR

Como se mencionó anteriormente, se ha optado por implementar el algoritmo ABR basado en el *throughput*. Esta decisión se tomó debido a que la incorporación de un búfer para los segmentos de vídeo resultaría en una latencia adicional, y su tamaño reducido podría comprometer el análisis adecuado de la transmisión. Sin embargo, estos algoritmos dependen estrechamente de dos factores: el método de medición de la latencia y la lógica de gestión de la calidad.

- **Medición de la latencia.** El método escogido para la medición de la latencia es el de latencia reactiva, que implica medir la latencia exclusivamente con las mediciones de tiempo obtenidas en el lado del cliente, independientemente de las marcas de tiempo enviadas por el servidor. Esto simplifica la implementación, ya que evita la necesidad de coordinar ambas máquinas involucradas. Sin embargo, esta aproximación conlleva la limitación de que la medición de la latencia se basa en el tiempo de recepción del paquete anterior, lo que provoca que el *bitrate* estimado siempre sea inferior al máximo posible debido a la relación entre el tamaño del mensaje y la cadencia de transmisión de paquetes, que en este caso corresponde al *framerate*. La tasa máxima se puede calcular siguiendo la ecuación 4.2.

$$M_{bs,max} \text{ (Mbps)} = B_{rec} \text{ (bits)} \times F_{ideal} \text{ (FPS)} \quad (4.2)$$

Ecuación 4.2: Fórmula para el cálculo de la tasa máxima

En este contexto, la estimación obtenida no representa una evaluación global, como sería el caso con un método basado en sincronización, sino que proporciona una relación relativa respecto a la latencia mínima y al *bitrate* máximo. Estos

factores deben considerarse en la implementación de la lógica de gestión de la calidad, la cual se detalla en los siguientes apartados.

- **Estrategia de arranque.** Se ha adoptado una estrategia de *bitrate* ascendente, comenzando con la calidad de imagen más baja y escalando gradualmente hasta alcanzar la calidad óptima. Esta metodología asegura una introducción suave y adaptativa al entorno de transmisión, permitiendo ajustes basados en las condiciones reales del medio mientras se optimiza la experiencia del usuario final.
- **Estrategia de cambio.** Para los ajustes en la escalera de *bitrate*, se ha adoptado una estrategia conservadora, permitiendo un incremento de 'k=1' en cada paso. Esta aproximación busca minimizar la posibilidad de que el algoritmo entre en un estado caótico, manteniendo la estabilidad del sistema mediante ajustes graduales del *bitrate*.
- **Estimación del Throughput.** La estimación del *throughput* se realiza mediante un búfer diseñado para almacenar las mediciones del *throughput* instantáneo durante una ventana temporal y determinar un valor escalar final utilizando un predictor, como se discute en el capítulo del marco teórico (capítulo 3). Finalmente, se ha optado por utilizar la media armónica como predictor, y su justificación se presenta en el capítulo de experimentos (capítulo 5), donde se realiza un análisis comparativo de múltiples predictores para evaluar sus ventajas y desventajas.

La elección del tamaño de la ventana se puede ajustar a través de un archivo JSON (config.json) proporcionado en el paquete, que permite regular este y otros parámetros. En nuestro caso específico, se ha establecido un valor de ventana temporal de 0,5 segundos. Aumentar el tamaño de la ventana proporciona un mayor suavizado y un mayor conjunto de muestras, pero también puede ralentizar la adaptabilidad del sistema y capturar datos no representativos debido a la naturaleza no estacionaria del entorno.

El número de muestras almacenadas en el búfer será diferente para cada cámara ya que varía en función de la tasa de *frames* que se estén recibiendo.

- **Sistema de inestabilidad y búfer de inestabilidad.** Una vez implementado el estimador de *throughput* recibido, este valor se almacena e inicia un sistema de votación en el cual se comparan la predicción de la ventana recién obtenida y la ventana anterior. Si ambas exceden una fracción del *framerate* máximo, se considerará que la comunicación es altamente inestable.

Cuando se detecta inestabilidad, se añade una entrada al búfer de inestabilidad. Si este búfer se llena, se solicita inmediatamente una configuración de calidad 'k' posiciones inferior, es decir, se cumple 4.3

$$I = \begin{cases} \text{Añadir al búfer} & \text{si } (V_{n+1} > U_{\text{porc}} \text{ y } V_n > U_{\text{porc}}) \\ \text{Sin acción} & \text{si } (V_{n+1} \leq U_{\text{porc}} \text{ o } V_n \leq U_{\text{porc}}) \\ \text{Reducir calidad en } k \text{ posiciones} & \text{si } B_{\text{inst}} \text{ lleno} \end{cases} \quad (4.3)$$

Ecuación 4.3: Condiciones del sistema de inestabilidad.

Esto resulta útil cuando algún artefacto en la red genera un cuello de botella temporal, que provoca la ausencia de paquetes seguida por su llegada en un corto período. Este método permite identificar la naturaleza de la conexión, evitar un análisis erróneo y reaccionar de manera rápida reduciendo la calidad para estabilizar la red.

- **Sistema de votación.** Si la conexión es estable, los valores de las ventanas temporales anteriores se comparan con un umbral determinado como una fracción del *bitrate* máximo teórico.

Después de aplicar un umbral a las mediciones obtenidas en ambas ventanas, se obtienen dos valores con valores de verdadero o falso, y el algoritmo toma las siguientes decisiones:

- Ambos valores superan el umbral: Se realiza una petición para aumentar 'k' posiciones en la escalera del *bitrate*, incrementando la calidad de imagen.
- Ambos valores están por debajo del umbral: Se solicita una reducción de 'k' posiciones en la escalera del *bitrate*, disminuyendo la calidad de imagen.
- Los valores no coinciden: Se asume que el sistema ha alcanzado cierta estabilidad y no se toma ninguna acción.

Es decir, se aplican las condiciones de 4.4

$$\text{Acción} = \begin{cases} \text{Aumentar bitrate} & \text{si } (V_{n+1} > \text{Umbral} \text{ y } V_n > \text{Umbral}) \\ \text{Reducir bitrate} & \text{si } (V_{n+1} < \text{Umbral} \text{ y } V_n < \text{Umbral}) \\ \text{Sin acción} & \text{si } (V_{n+1} > \text{Umbral} \text{ y } V_n < \text{Umbral}) \\ & \text{o } (V_{n+1} < \text{Umbral} \text{ y } V_n > \text{Umbral}) \end{cases} \quad (4.4)$$

Ecuación 4.4: Condiciones del sistema de votación.

- **Tasa máxima de transmisión (opcional).** De forma opcional, se permite al usuario establecer un límite máximo de calidad de imagen mediante la restricción del *bitrate* máximo que el códec puede transmitir. Esto elimina la posibilidad de alcanzar escalones de *bitrate* superiores al valor fijado, asegurando una transmisión dentro de los parámetros establecidos.
- **Detección de calidades ineficientes.** Si tras ascender a un escalón superior solicitando un mayor *bitrate*, no se observa un incremento significativo en el *throughput* medido, se considera un claro indicativo de que se ha alcanzado la máxima calidad disponible para el contenido de vídeo. En consecuencia, se detendrá el ascenso en ese escalón hasta que se requiera un ajuste. Esto evita ascensos innecesarios a escalones superiores, los cuales podrían generar descensos repentinos, mejorando así el rendimiento general del sistema.
- **Sistema de emergencia.** Se ha incorporado un modo de emergencia para situaciones en las que la latencia de los paquetes recibidos alcance un umbral alto, también configurable en el archivo JSON, cuyo valor por defecto es 5. Si la latencia supera este umbral, se considera que la comunicación es demasiado lenta, permitiendo al sistema solicitar la calidad de imagen más baja en la escalera, omitiendo la limitación de descenso 'k'. Aumentar el valor del umbral proporcionaría una mayor flexibilidad al sistema, pero implica un mayor riesgo en comunicaciones críticas, mientras que un umbral demasiado bajo provocaría activaciones constantes, evitando así alcanzar la mejor calidad de imagen posible.
- **Filtro anti-rizado.** Se ha introducido un filtro anti-rizado con el objetivo de suavizar los patrones de oscilaciones repetitivas que pueden producirse si la calidad óptima se encuentra en un punto intermedio entre dos escalones de calidad. Esto permite reducir el efecto de rizado sin afectar el rendimiento en el escalón inferior, generando un umbral de decisión suficiente para que el sistema logre estabilizarse adecuadamente.

El filtro utiliza la información del *bitrate* actualizado en el paso anterior ( $n - 1$ ), el *bitrate* actual ( $n$ ) y el *bitrate* estimado para el siguiente paso ( $n + 1$ ). Su lógica de activación se basa en la siguiente condición: si  $\text{bitrate}[n - 1]$  y  $\text{bitrate}[n + 1]$  son iguales y mayores que  $\text{bitrate}[n]$ , el filtro se activa, denegando el ascenso de la calidad. En este caso, se establece que:

$$\text{bitrate}[n + 1] = \text{bitrate}[n]$$

de modo que se evitan oscilaciones no deseadas y se promueve la estabilización en un nivel adecuado de calidad.

Para representar esta lógica, podemos definir la función de activación del filtro anti-rizado como:

$$\mathcal{F}(n) = \begin{cases} \text{bitrate}[n + 1] = \text{bitrate}[n], & \text{si } \text{bitrate}[n + 1] = \text{bitrate}[n - 1] > \text{bitrate}[n] \\ \text{bitrate}[n + 1], & \text{en caso contrario} \end{cases} \quad (4.5)$$

Ecuación 4.5: Condiciones para la activación del filtro anti-rizado.

Es importante notar que este filtro no actúa como un filtro en el sentido clásico, pero tiene la capacidad de suavizar oscilaciones de alta frecuencia en tiempo real. La implementación está diseñada para detectar únicamente el patrón  $\text{bitrate}[n + 1] = \text{bitrate}[n - 1] > \text{bitrate}[n]$ , ya que es más crítico prevenir un aumento en la calidad que supere la capacidad del canal. En contraste, el patrón invertido  $\text{bitrate}[n + 1] = \text{bitrate}[n - 1] < \text{bitrate}[n]$  simplemente se traduce en una pérdida puntual de calidad en la imagen, lo cual permite al sistema responder más rápidamente a tendencias descendentes en la calidad.

Es importante señalar que este filtro no puede mantenerse activo de manera continua, ya que podría interferir con el flujo general del sistema. En situaciones en las que la calidad de la red disminuye, el filtro podría impedir un ascenso oportuno en la calidad de la transmisión. Para abordar esto, es necesario definir un número máximo de activaciones consecutivas del filtro, de modo que su acción sea temporal y se reduzca la frecuencia del rizado a medida que este número aumenta.

De manera análoga a los filtros reales, este parámetro se denomina 'orden del filtro'. Un orden elevado permite un filtrado más agresivo, similar a un filtro de orden alto que suprime con mayor fuerza las fluctuaciones. Sin embargo, esto también puede introducir artefactos. El filtro propuesto, al configurarse con un orden elevado, otorga mayor estabilidad al sistema, pero a costa de reducir su capacidad de adaptación, lo cual afecta la eficiencia del sistema adaptativo. Por ello, es fundamental encontrar un equilibrio adecuado: un orden demasiado alto puede suavizar el sistema en exceso, mientras que uno demasiado bajo podría no ser suficiente para controlar el rizado.

Esta configuración permite que el sistema suavice su respuesta ante artefactos de rizado específicos, manteniendo una tasa de cambio de calidad elevada en la

mayoría de las demás situaciones. De esta forma, se evita la necesidad de utilizar ventanas de análisis más amplias, las cuales ralentizarían la capacidad del sistema para adaptarse a las fluctuaciones de la red. Así, se optimiza la respuesta del sistema sin comprometer su flexibilidad ni su rendimiento global.

Con el objetivo de clarificar el flujo de trabajo del sistema implementado, se ha elaborado un diagrama explicativo que se muestra en la figura 4.2.

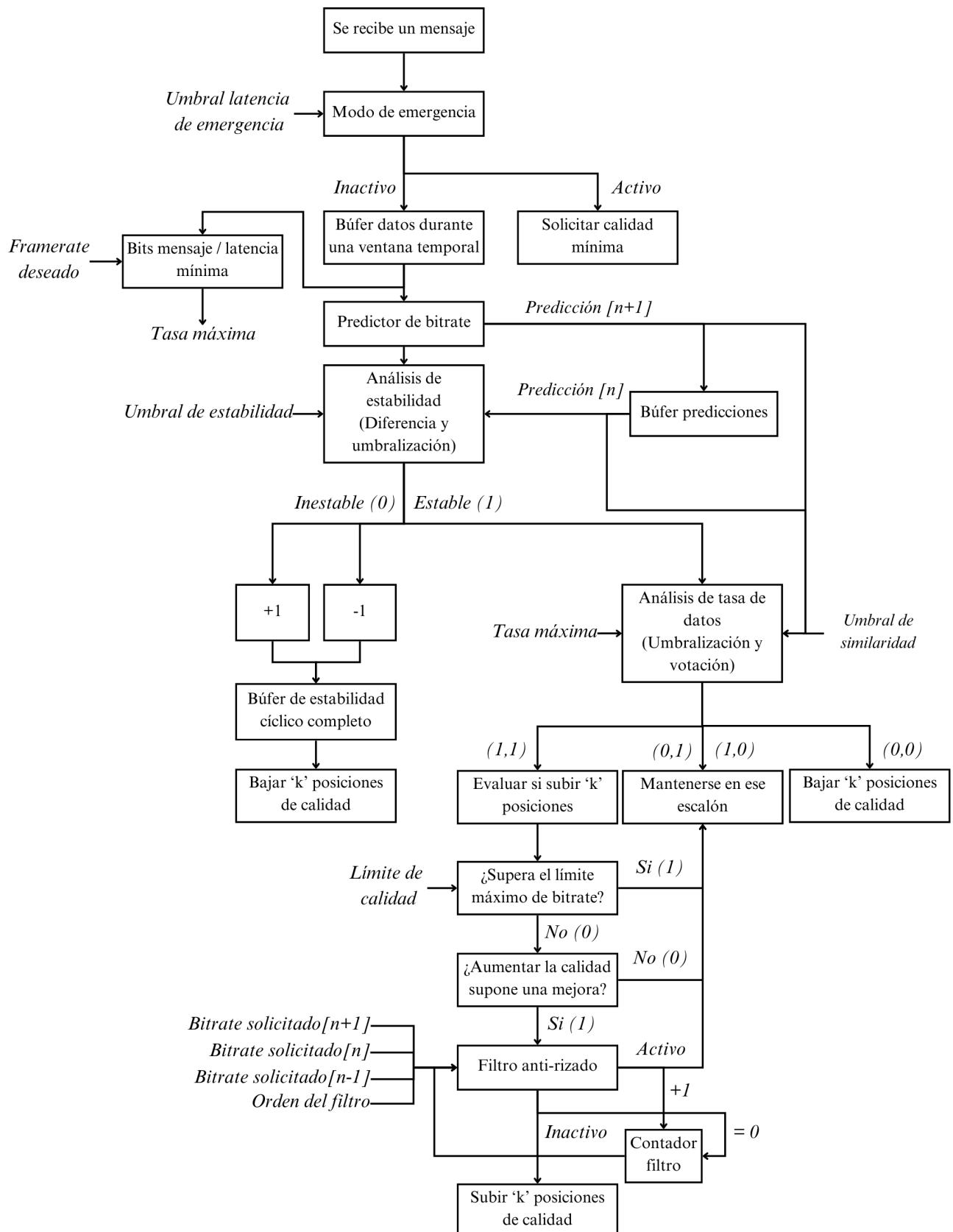


Figura 4.2: Diagrama mostrando la lógica de cambio de calidad implementada.

---

## Capítulo 5

# Experimentos realizados

---

En este capítulo se presentan los experimentos diseñados para evaluar el sistema propuesto y verificar las mejoras derivadas de su implementación. Se llevarán a cabo dos tipos principales de experimentos: simulados y reales.

El primer conjunto de experimentos corresponde a las simulaciones, las cuales se realizarán utilizando dos ordenadores, uno actuando como transmisor y otro como receptor, conectados directamente a través de una red local generada por el transmisor. Este equipo regulará artificialmente el ancho de banda mediante un *script* en bash (ver código B.1 en el anexo 2), con el objetivo de simular diferentes condiciones de red. El comportamiento del sistema se estudiará bajo tres escenarios distintos:

1. **Situaciones de estabilidad:** Se fija la capacidad del canal a 15 Mbit/s (Tramo A, primeros 60 segundos).
2. **Situaciones de irregularidad en el *bitrate*:** La tasa de transmisión oscila alrededor de 14 Mbit/s con una amplitud de 1 Mbit/s (Tramo B, 38 segundos).
3. **Situaciones de descenso progresivo de la capacidad del canal:** Se inicia con una caída abrupta a 8 Mbit/s (Tramo C, 50 segundos).

Estos experimentos permitirán observar cómo se ajusta el sistema en cada uno de los escenarios propuestos, evaluando su capacidad para mantener una transmisión eficiente y de calidad, así como su respuesta ante cambios en la capacidad del canal y fluctuaciones del *bitrate*.

Para este conjunto de experimentos se hará uso del vídeo 'Big Buck Bunny' el cual es una animación hecha por la asociación Blender de uso publico que es muy utilizada para la realización de pruebas de vídeo, en concreto se hará uso de su versión Full HD (1920 x 1080) a 30 *frames* por segundo ([Blender Foundation, 2008]. Además, dentro de esta categoría se harán tres evaluaciones que permitan entender el comportamiento del sistema en su conjunto.

1. Comparación entre diferentes predictores.
2. Simulaciones de artefactos de red.
3. Comparativa con la implementación de `image_transport` basada en Theora.

Por otro lado, el experimento real consistirá en una sesión de movimiento a través de una habitación, utilizando un robot TIAGo de la compañía PAL Robotics ([PAL Robotics, 2023]), que estará conectado directamente a un ordenador portátil mediante una conexión Wi-Fi directa entre ambos dispositivos. Este experimento tiene como objetivo verificar si el comportamiento del sistema es aplicable a un entorno real, aunque no se puedan reproducir todos los artefactos de red previstos ni los modelos de propagación deseados.

De forma común a ambos tipos de experimentos, las mediciones se calculan cada segundo, lo que permite sincronizarlas con el tiempo de solicitud de una nueva calidad, correspondiente a dos ciclos del búfer de la tasa de bits calculada. En condiciones estables, es necesario llenar estos ciclos para tomar una decisión sobre la calidad que se debe solicitar al receptor. Sin embargo, esta frecuencia de muestreo de 1 Hz no siempre se cumple, ya que la ventana temporal se calcula en función de la tasa de recepción de mensajes y, por lo tanto, del *framerate*. Esto provoca que, al inicio de la comunicación, en condiciones de red cambiantes o después de solicitar una nueva calidad, el *framerate* no sea constante.

Para corregir estas irregularidades, se aplica un procesamiento a las series de datos obtenidas utilizando la función `interp1d` de la librería SciPy de Python [SciPy Community, 2023], con el objetivo de regularizar la tasa de muestreo a un segundo exacto.

## 5.1. Comparación entre predictores

Los primeros experimentos descritos en este capítulo están enfocados en la evaluación del mejor predictor, con el objetivo de establecerlo como la opción predeterminada en el sistema. Esta sección no tiene como propósito justificar el rendimiento de la propuesta en su conjunto, sino más bien respaldar las decisiones de diseño tomadas en el capítulo anterior. En particular, se evaluarán cuatro predictores de *throughput*: el *throughput* instantáneo (una medición del valor muestreado del último paquete recibido entre el tiempo de latencia respecto al paquete anterior), la media aritmética, la media armónica (siguiendo el reconocido trabajo de FESTIVE ([Jiang et al., 2014])), y finalmente la mediana de los valores instantáneos almacenados

durante una ventana temporal de medio segundo, como se estableció en el capítulo de diseño (capítulo 4).

Para realizar estas pruebas, se utilizará un ordenador transmisor conectado a otro receptor mediante una conexión Wi-Fi tipo hotspot, cuya capacidad de canal será modulada por un textitscript bash con el objetivo de simular un comportamiento irregular y artefactos variados. El propósito es observar cómo cada predictor es capaz de estimar el *bitrate* en condiciones cambiantes de la red. Todas estas mediciones serán comparadas con la velocidad de transmisión ideal, calculada utilizando la ecuación 4.2.

El resultado del experimento se presenta en la figura 5.1. En esta figura, se observa claramente la relación entre la capacidad del canal, representada por una línea discontinua, y el *throughput* máximo ideal, mostrado en rojo. Este *throughput* ideal se calcula como el tamaño del mensaje recibido dividido entre la latencia mínima entre dos mensajes (idealmente, la inversa del *framerate*). A partir de los primeros cuarenta segundos de la etapa A, se observa una estabilización y una aparente correlación entre ambas métricas. Sin embargo, también se aprecia que, en ciertos picos, la tasa máxima supera la capacidad del canal, lo cual en principio no debería ser posible. Esto se debe a que la tasa máxima refleja el rendimiento del códec en el lado del servidor, sin tener en cuenta la latencia introducida por el canal.

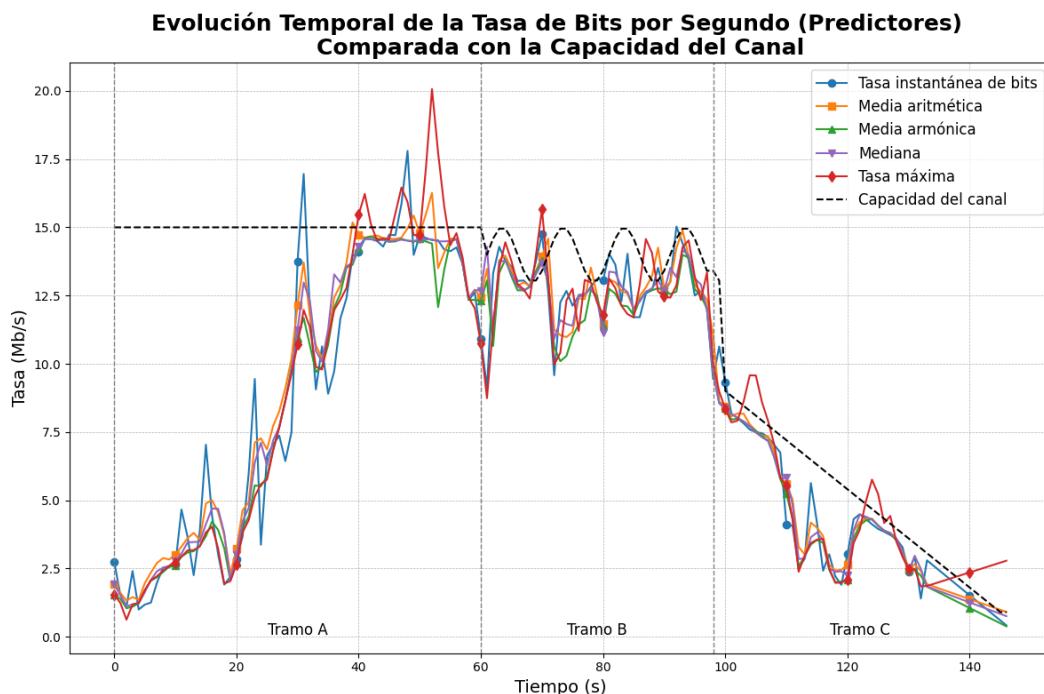


Figura 5.1: Comparación de rendimiento entre diferentes predictores en relación a la capacidad del canal y en relación a la tasa máxima.

Así, cada vez que la tasa máxima supera la capacidad del canal, el contenido de vídeo

experimenta una irregularidad en el *framerate*, un efecto que se busca mitigar dada su importancia en el contexto de aplicación robótica. La similitud entre la tasa máxima y la respuesta del canal se debe al mecanismo regulador ABR, que ajusta el emisor para que se adapte a la calidad del códec, como se analizará en la siguiente sección. Cabe destacar que, aunque el codificador tenga un *bitrate* objetivo, en función del contenido transmitido, puede realizar codificaciones puntuales a un *bitrate* ligeramente superior, lo cual también se refleja en la tasa máxima.

Es precisamente esta diferencia entre el *bitrate* transmitido por el códec y la tasa del canal la que se busca aprovechar y caracterizar para regular la tasa de bits, con el fin de mantener un *framerate* estable.

Por último, el resto de elementos en la gráfica representan predictores cuyo objetivo es estimar la capacidad del canal. Entre ellos, el *bitrate* instantáneo y la media presentan el peor rendimiento, ya que generan un comportamiento irregular, especialmente notable durante las irregularidades de la conexión, introduciendo picos de alta amplitud. En cambio, tanto la media armónica como la mediana muestran un ajuste excepcionalmente preciso a la respuesta del canal.

Para poder analizar con mayor detalle el rendimiento entre ambos predictores, se ha ajustado la visualización de la figura 5.1, resultando en una representación más clara que se muestra en la figura 5.2. En esta nueva representación se aprecia que ambos se adaptan especialmente bien a la morfología de la serie temporal, y que los picos en el caso del predictor de la mediana están suavizados. No obstante, cabe destacar que en el predictor de la media armónica estos picos están completamente ausentes, lo que en términos generales se traduce en un mejor rendimiento.

Aunque la media armónica se comporta muy bien en general, su naturaleza tiende a priorizar los valores más pequeños y penalizar aquellos de mayor amplitud. Esto la convierte en una opción extremadamente valiosa en el contexto actual, pero también implica que si se recibe un paquete con un volumen reducido de datos, la latencia al presentar un valor mínimo asociado a la tasa de cuadros del vídeo por ser medida exclusivamente en el receptor, puede derivar en una velocidad de transmisión muy baja, afectando gravemente al sistema y haciendo al predictor inutilizable en esas circunstancias siguiendo así las advertencias vistas en [Juluri et al., 2015] y [Wang et al., 2016].

Para mitigar este problema, se podrían aplicar técnicas de *padding*, que consisten en llenar los paquetes hasta alcanzar un número mínimo de bits en caso de transmitir un mensaje con menor contenido. Sin embargo, esto supondría un desperdicio de recursos de red, lo que podría ser inaceptable en situaciones críticas. Por otro lado, el predictor

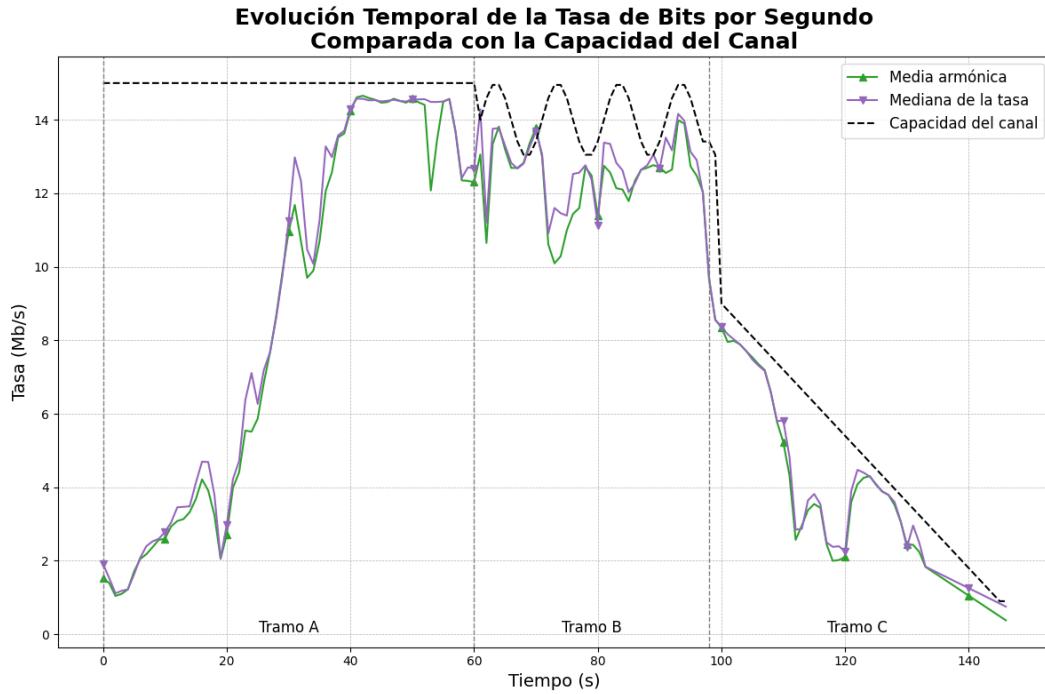


Figura 5.2: Comparación de rendimiento en mayor detalle de los mejores predictores en relación a la tasa de transmisión ideal.

de la mediana es más robusto ante tamaños de mensajes irregulares, ya que pondera negativamente los valores atípicos tanto por encima como por debajo del promedio, lo que lo convierte en una opción muy interesante.

En estos experimentos, se ha optado por utilizar la media armónica como la opción por defecto debido a su buen rendimiento general. No obstante, es importante resaltar que la mediana también puede ser una excelente alternativa en ciertas situaciones.

## 5.2. Respuesta a múltiples artefactos

En esta segunda parte del sistema, evaluaremos la lógica ABR en su totalidad, en lugar de enfocarnos exclusivamente en el predictor. Nuestro objetivo principal es maximizar el *bitrate* solicitado sin exceder la capacidad del canal, optimizando así la calidad de imagen mientras mantenemos un *framerate* estable. Para lograrlo, compararemos la capacidad máxima del canal con el *bitrate* que es solicitado y confirmado por el servidor.

Es importante destacar que exceder la capacidad del canal no implica una pérdida inmediata de comunicación, sino retrasos en la transmisión de la imagen y una reducción del *framerate*. Sin embargo, si la diferencia entre el *bitrate* solicitado y la capacidad del canal es demasiado grande, existe el riesgo de perder la comunicación por completo.

En la figura 5.3, se observa que, en términos generales, el sistema opera de manera

adecuada, procurando mantenerse por debajo del umbral de la capacidad del canal. Además, se adapta a los diversos artefactos presentados en los distintos tramos del experimento, demostrando así su capacidad de ajuste y optimización bajo diferentes condiciones de red.

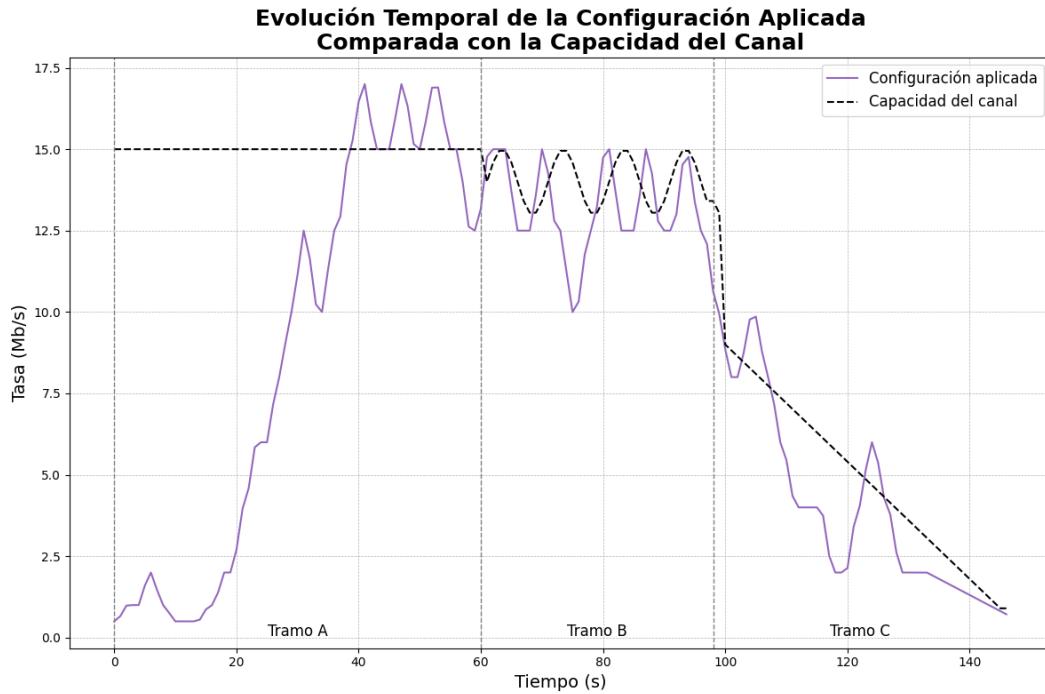


Figura 5.3: Comparación sistema ABR completo.

### 5.2.1. Tramo A

En el tramo de estabilidad (tramo A) de la simulación, observamos que, desde el inicio hasta alcanzar la estabilidad, transcurren 38 segundos. Aunque se comienza desde la calidad más baja, el ascenso es relativamente lento, presentando dos irregularidades en los momentos iniciales y durante el salto de 10 a 12,5 megabits por segundo. Este rendimiento se debe a la configuración del sistema ABR, que se ha ajustado para seguir una filosofía que priorice la seguridad. En este contexto, los incrementos de *bitrate* solo pueden ascender de uno en uno, empezando desde la configuración más baja, lo cual prolonga el periodo transitorio especialmente si se utiliza una escalera de *bitrate* con múltiples escalones.

Por otro lado, la limitación del *bitrate* máximo permite reducir la escalera de *bitrate* solo a configuraciones por debajo de este umbral, lo cual acorta significativamente el tiempo de ascenso sin sacrificar la calidad de resolución al utilizar escalones de mayor tamaño. Asimismo, se han presentado dos momentos de inestabilidad de la red que han ralentizado el proceso de subida. Estos momentos de inestabilidad tienen un

efecto negativo amplificado por el filtro anti-rizado, que impide que patrones irregulares asciendan consecutivamente, aumentando así el tiempo total necesario para alcanzar la estabilidad.

En el resto del periodo A, se observa uno de los tres comportamientos posibles, que son regulables según la configuración:

1. El *bitrate* máximo esta configurado.
2. La capacidad del canal esta muy por encima del *bitrate* máximo disponible para el escalón mas alto y el contenido del vídeo.
3. El canal esta limitado y ninguna configuración ha sido definida previamente.

De las tres opciones, la que estaría activa en este momento es la última y más compleja, en la cual el sistema intenta ajustarse a la tasa ideal mediante el análisis de los datos disponibles. Esta alternativa destaca una de las principales debilidades del sistema, ya que, como se muestra en la figura 5.3, existe la posibilidad de que el sistema solicite una configuración que supere el umbral máximo del canal. Esto puede provocar un impacto negativo en el *framerate* al implementar el cambio, para luego revertir a la configuración anterior, generando un efecto de rizado entre la configuración óptima y la subsiguiente. Este fenómeno se produce debido a que, en esta situación, la diferencia entre la tasa máxima y el valor predicho es tan pequeña, incluso estando cerca del límite del canal, que el umbral de decisión se vuelve extremadamente estrecho.

Cuando este efecto de rizado ocurre, se activa el filtro anti-rizado, el cual previene la repetición de este patrón ‘n’ veces consecutivas, según el orden del filtro configurado. Actualmente, con un filtro de orden cuatro activo, se permite que el filtro se reactive solo cada cuatro ventanas temporales, suavizando así la respuesta del sistema.

En comparación con los otros dos escenarios planteados, si la capacidad del canal excediera significativamente la calidad máxima del vídeo, la respuesta del sistema se estabilizaría completamente en los momentos de estabilidad. En este caso, aun sin alcanzar el nivel máximo de la escalera de *bitrate*, el sistema detectaría que la tasa recibida no cambiaría en configuraciones subsecuentes, manteniendo dicho nivel hasta que se produzca un cambio en las condiciones de red o en el contenido del vídeo. Por otra parte, en un escenario donde el *bitrate* máximo del códec esté limitado, la escalera de *bitrate* se acorta, impidiendo así que el sistema ascienda más allá de lo establecido y mitigando, de este modo, el problema del rizado.

Como conclusión para este análisis, se observa que, en términos generales, el sistema responde de manera robusta y adecuada. Sin embargo, presenta dos problemas

asociados: la dificultad de alcanzar un máximo óptimo en condiciones de regularidad, debido a que la estimación temporal se realiza exclusivamente en el lado del cliente, y la proximidad de la tasa máxima al límite del canal, que limita el umbral de decisión. Estos problemas pueden resolverse de manera efectiva introduciendo un umbral máximo de calidad deseada y suavizando el orden del filtro de rizado, lo cual optimizaría aún más el rendimiento del sistema.

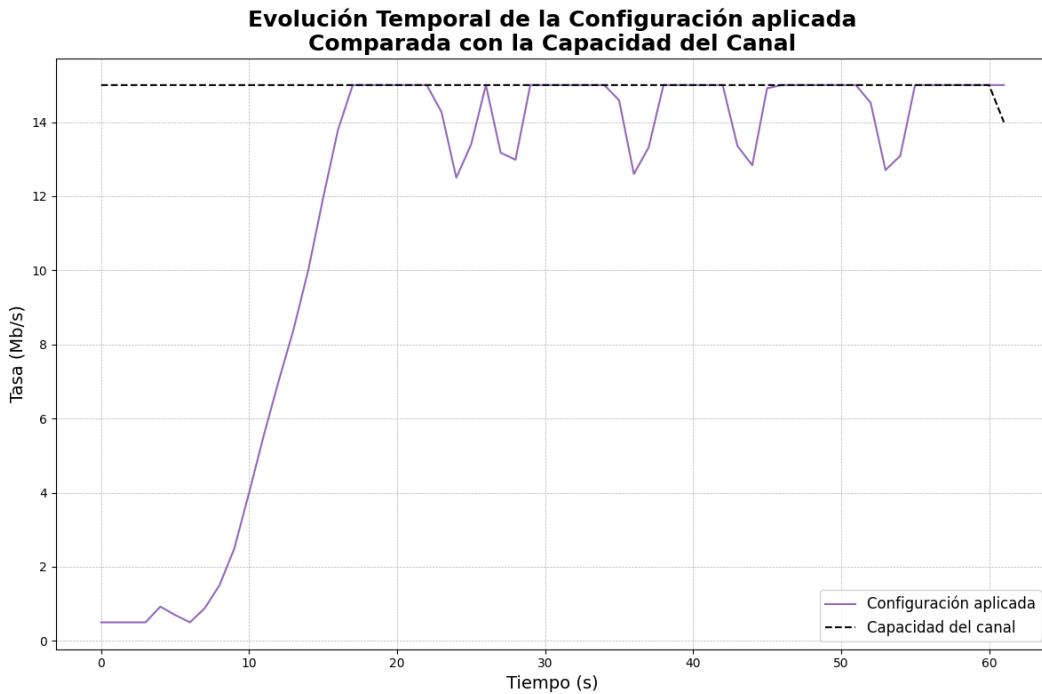


Figura 5.4: Comparación del sistema ABR completo para el tramo A, tras establecer un límite superior de *bitrate*.

Al repetir el experimento estableciendo un límite de 15 Mbit/s y reduciendo el orden del filtro a dos, se obtiene el resultado mostrado en la figura 5.4. En este nuevo escenario, el tiempo necesario para alcanzar la estabilidad disminuye considerablemente, alcanzando un estado estable en aproximadamente quince segundos. Además, se observa una reducción en la frecuencia de aparición del rizado, que permanece consistentemente por debajo del límite máximo de calidad.

Como se puede apreciar, la introducción de un límite de calidad máxima contribuye a minimizar dos de los principales puntos débiles del sistema: el tiempo de estabilización y la frecuencia del rizado. Estos ajustes no solo mejoran la eficiencia del sistema, sino que también optimizan su rendimiento general, permitiendo un control más preciso y una adaptación más rápida a las condiciones de red cambiantes.

### 5.2.2. Tramos B y C

En el tramo B, se observa que el ajuste realizado resulta adecuado, ya que no se detecta ningún rizado que supere el límite óptimo de calidad establecido en 15 Mbit/s. Este ajuste permite prever el comportamiento de la red y, en gran medida, mantenerse por debajo del umbral de la capacidad del canal. Además, el sistema aprovecha las fluctuaciones de la red para incrementar la calidad de manera oportuna, logrando optimizar la calidad de imagen de forma eficiente.

En cuanto al tramo C, la configuración del canal demuestra su capacidad para anticiparse a una caída abrupta de 8 Mbit/s. El algoritmo no solo responde adecuadamente ante esta disminución repentina, sino que además se adapta a la tendencia descendente que continúa, logrando mantenerse por debajo del umbral de manera constante. Sin embargo, es importante señalar que en dos momentos específicos, alrededor de los segundos 110 y 125, este umbral es superado, lo que genera una ralentización temporal en el *framerate*.

En conclusión, el algoritmo proporciona un ajuste óptimo en condiciones desafiantes, maximizando la calidad de imagen sin introducir irregularidades significativas en el *framerate*. No obstante, es fundamental recordar que la eficacia de este ajuste también depende de la configuración inicial y del criterio del usuario. Por ello, enfoques más conservadores o más arriesgados pueden ser preferidos según las necesidades y expectativas del entorno. Para el experimento, se optó por una configuración equilibrada, con el objetivo de lograr un balance entre calidad de imagen y estabilidad del *framerate*.

## 5.3. Comparación con Theora

En esta sección se realizará una comparación con la implementación del códec no adaptativo de 'image transport', Theora. El experimento se desarrollará bajo los escenarios de calidad descendente y comportamiento hostil, ya que, al no ser adaptativo, el códec Theora mantendría un flujo de datos constante siempre que la capacidad del canal se mantenga por encima de la tasa a la que esté transmitiendo.

Es importante señalar que la evaluación de Theora no puede llevarse a cabo en condiciones completamente iguales, dado que las mediciones obtenidas en el sistema ABR corresponden a los mensajes codificados que llegan al receptor. Esto implica que no podríamos hacer lo mismo con el códec Theora sin modificar el código. Por esta razón, se ha optado por un enfoque visual, en el que se comparan *frames* de vídeo correspondientes al mismo instante temporal, permitiendo así observar cómo responden

ambos vídeos bajo las mismas condiciones. Aunque en esta sección se presenta una descripción textual de la comparación, también es posible visualizarla directamente en un vídeo sincronizado disponible en [Álvaro Gutiérrez García, 2024a].

La primera comparación entre *frames* se muestra en la figura 5.5, donde, unos instantes después de comenzar el vídeo y aún dentro del tramo A, se puede observar cómo el vídeo ABR (imagen a la izquierda) presenta un significativo adelantamiento temporal en relación a su contraparte Theora (imagen a la derecha). Esto se debe a la mayor fluidez del *framerate* en la solución ABR en comparación con Theora. Sin embargo, la calidad de imagen es notablemente superior en el códec Theora, debido a su transmisión a un *bitrate* constante y elevado desde el inicio, mientras que el códec ABR en H.264 todavía se encuentra en sus niveles de calidad más bajos.

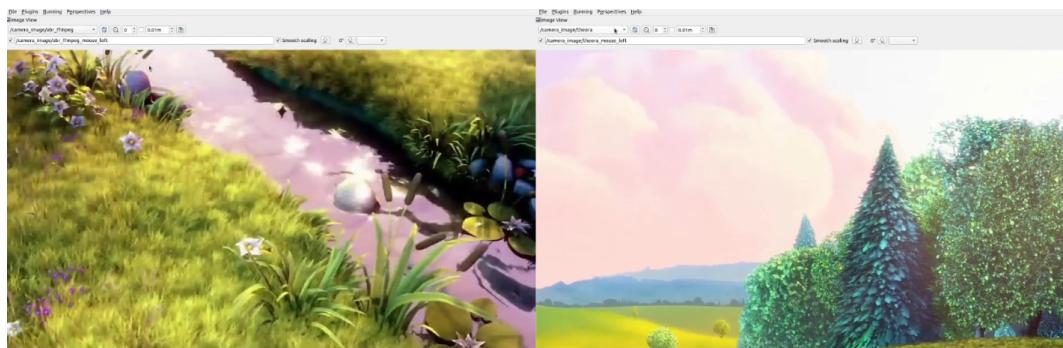


Figura 5.5: Comparación del sistema propuesto (izquierda) frente a Theora (derecha) en el tramo A.

En la figura 5.6 se observa el comportamiento durante el tramo B del experimento, en el que la conexión es irregular. Aquí, la calidad de imagen de ambos *frames* es similar, dado que el sistema ABR ha alcanzado su punto de estabilidad y solo reduce la calidad cuando las condiciones de la red lo requieren. Cabe destacar que, gracias a la eficiencia de codificación de H.264, es posible obtener una calidad de imagen comparable a la de Theora, pero con un menor flujo de datos.

Además, se aprecia que el sistema ABR muestra un avance notablemente mayor en el vídeo, debido a que mantiene un *framerate* estable, mientras que Theora enfrenta dificultades para preservar una sensación de fluidez en la imagen, una diferencia que se acentúa con el paso del tiempo.

Finalmente, como se muestra en la figura 5.7, al final de la etapa C del experimento, se observa que, aunque Theora mantiene una buena calidad de imagen, no consigue sostener la transmisión durante más que unos pocos segundos adicionales. Esto resulta en una pérdida completa de comunicación durante el proceso de descenso progresivo de calidad, lo que provoca que la imagen se congele. En cambio, el códec H.264 ha logrado

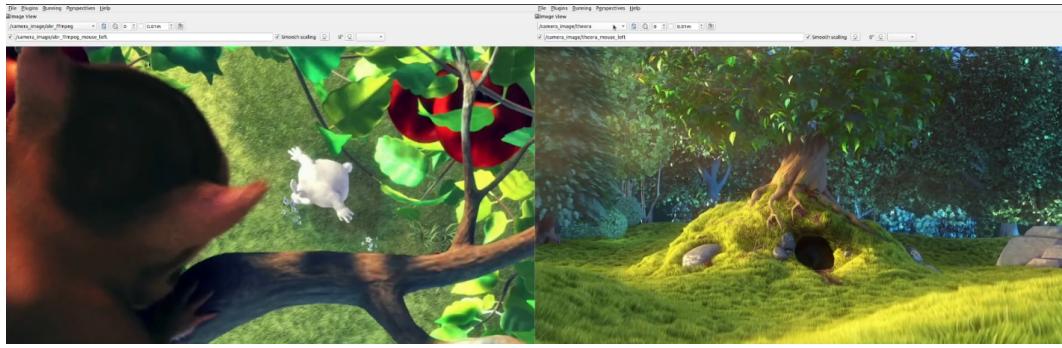


Figura 5.6: Comparación del sistema propuesto (izquierda) frente a Theora (derecha) en el tramo B.

mantener la fluidez del vídeo hasta el final del segmento C, ajustando progresivamente la calidad de imagen a medida que las condiciones de la red lo requieren.

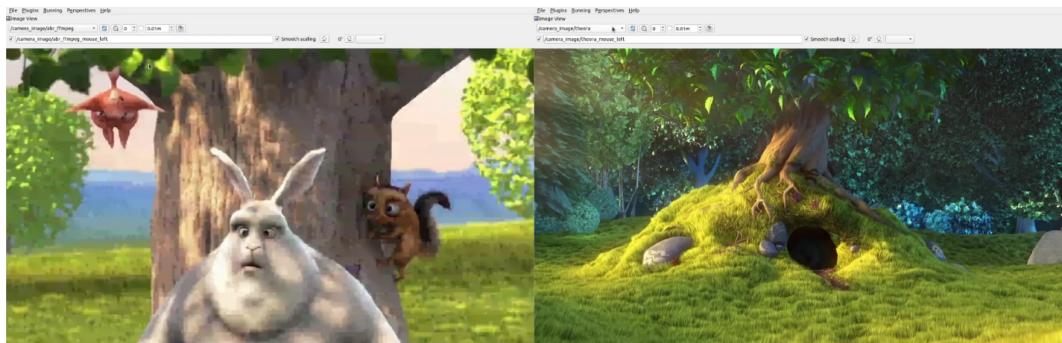


Figura 5.7: Comparación del sistema propuesto (izquierda) frente a Theora (derecha) en el tramo C.

Cabe destacar que el resultado de esta comparación no resalta exclusivamente la mejora introducida por el sistema ABR, sino también los beneficios aportados por la eficiencia de codificación de H.264. Esto permite entender la propuesta completa como una mejora respecto a la tecnología actual proporcionada por el paquete 'Image transport'. Sin embargo, el rendimiento del sistema ABR al extender la duración de la transmisión ajustándose a la capacidad del canal representa un valor añadido especialmente relevante para robots en misiones críticas, donde mantener una comunicación estable y prolongada es crucial.

## 5.4. Experimento real

Finalmente, se va evaluar el rendimiento del sistema propuesto cargando el sistema adaptativo en un robot TIAGo de la compañía PAL Robotics ([PAL Robotics, 2023]) perteneciente a la escuela de ingenieros de Fuenlabrada de la Universidad Rey Juan Carlos. El robot utilizado puede ser visualizado en la figura 5.8.



Figura 5.8: Robot TIAGo utilizado en el experimento.

Para el experimento, el robot cargará con el rol de transmisor transmitiendo el contenido visual de su cámara a un ordenador externo. Una vez establecida la conexión con el robot y tras un breve periodo inicial establecido para que el sistema pudiera superar el periodo transitorio inicial, el receptor se desplazó por el laboratorio, alejándose progresivamente del robot, en un entorno de comunicación inalámbrica altamente fluctuante.

El resultado se puede observar en 5.9 donde encontramos como la calidad del vídeo efectivamente cambia a lo largo del tiempo de forma adecuada tras una sencilla instalación.

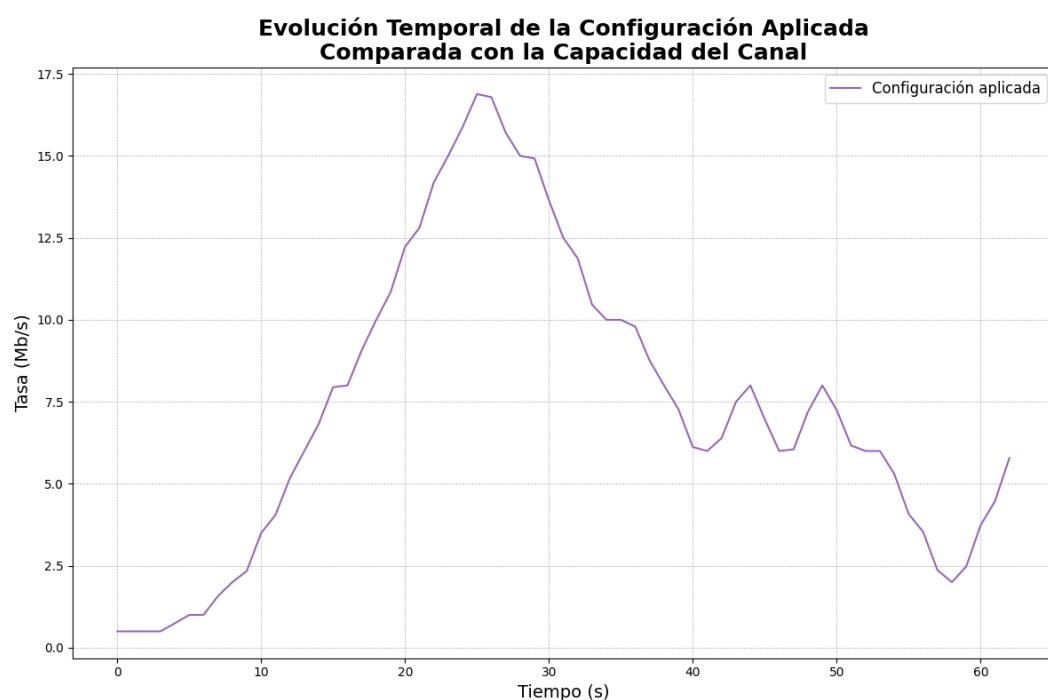


Figura 5.9: Comparación del sistema propuesto en un robot real, en concreto, un robot TIAGo conectado a un receptor a través de una conexión Wi-Fi irregular.

---

# Capítulo 6

# Conclusiones y trabajos futuros

---

En este capítulo final se abordarán las conclusiones obtenidas tanto en el diseño, implementación y experimentos realizados en este trabajo con el fin de clarificar las ideas más importantes y establecer el camino de ruta de los siguientes pasos de este tipo de tecnología en trabajos futuros.

## 6.1. Conclusiones

En este trabajo se ha presentado una visión completa sobre la implementación de un sistema de *Bitrate Adaptativo* (ABR) en el sector de la robótica, adaptando esta tecnología, tradicionalmente vinculada al sector multimedia, a la plataforma de código abierto ROS 2, documentando las similitudes y diferencias entre ambos sectores, resaltando las particularidades que requieren ajustes para lograr una implementación efectiva.

Para ello, se han ido cumpliendo progresivamente todos los objetivos parciales descritos en la sección de requisitos (2.2) del capítulo 2. Actividades como la exploración de la comunidad ROS 2, el análisis de las implementaciones actuales y el estudio exhaustivo de la bibliografía se llevaron a cabo exitosamente y se reflejan principalmente en el capítulo 3.

Estos objetivos dieron lugar a un análisis enfocado en dos aspectos principales: por un lado, ajustar y adaptar el concepto ABR para trasladarlo con éxito desde su contexto y aplicaciones multimedia tradicionales al entorno de ROS 2, y por otro lado, identificar y atender las necesidades específicas que un sistema robótico puede requerir. Estas necesidades se resumen en tres pilares fundamentales: la codificación en tiempo real con recursos limitados, la baja latencia y la fluidez en la transmisión de video en tiempo real, donde una pérdida completa de conexión resulta altamente indeseable. Además, se realizó una descripción de los posibles elementos físicos que podrían afectar la comunicación, considerando los diferentes tipos de robots y sus

contextos de aplicación, con el fin de obtener una visión general.

Adicionalmente, se logró diseñar una propuesta de sistema ABR basada en el conocimiento previo obtenido, la cual se describe detalladamente en el capítulo 4 y fue llevada a una implementación de código satisfactoriamente.

Los aspectos más destacados de esta fase del proceso fueron la selección y configuración del códec para asegurar un funcionamiento óptimo en tiempo real, el diseño de los canales de comunicación (o *topics*) y su configuración en relación con el *middleware*, y finalmente, el diseño de la lógica ABR. Esta última, además de seguir criterios basados en la literatura para los elementos principales, incorpora componentes específicos orientados a reducir artefactos y mejorar el rendimiento del sistema.

Dentro del diseño del algoritmo ABR, el núcleo central radica en la elección de un enfoque basado en *throughput*, utilizando un predictor de esta métrica que analiza ventanas temporales de medio segundo, el cual acabaría siendo la media armónica. La solución propuesta busca ser adaptable a distintos modelos de propagación de comunicación, y está concebida para abordar el problema desde una perspectiva general, cubriendo robots terrestres, aéreos, en entornos interiores y exteriores. Con este enfoque, se evita una caracterización detallada de artefactos específicos, difíciles de identificar desde la capa de aplicación, y se logra una mejora en el rendimiento global.

Finalmente, se completaron ambos tipos de experimentos propuestos, tanto en un entorno controlado como en un entorno físico, con el fin de evaluar el rendimiento del sistema, cuyos resultados quedan documentados en el capítulo 5.

El experimento en un entorno físico consistió en la implementación en un robot real operando con normalidad a pesar de estar expuesto a una red inestable donde el sistema propuesto fue instalado y se mantuvo funcionando en todo momento, garantizando así su fiabilidad y que la implementación fue correcta y libre de fallos.

Por otro lado, los experimentos en un entorno controlado, diseñados para transmitir vídeo entre máquinas y exponer la red a tres situaciones distintas para evaluar su desempeño, se dividen en tres etapas. En el primer subexperimento, se analiza el rendimiento de una variedad de predictores de forma independiente. En el segundo, se repite el proceso con un enfoque en el comportamiento global del sistema ABR. Finalmente, en el tercero, se realiza una comparación directa cualitativa con el códec Theora, sometiéndolo a las mismas condiciones hostiles.

De la comparación entre predictores destaca a la media armónica en primer lugar, seguida de cerca por la mediana, destacando muy por encima de otras alternativas como la media o el *throughput* instantáneo, ademas de encontrar convergencia en la

respuesta con los resultados reportados en la literatura.

Al analizar la respuesta global, el sistema propuesto ha demostrado una buena adaptación general a diversos mecanismos de red, aunque exhibe comportamientos subóptimos en la oscilación entre niveles cuando la conexión es estable. No obstante, en comparación con la solución de codificación estándar (Theora), el sistema ABR propuesto ofrece ventajas significativas: una mejor calidad de imagen en condiciones estables, un mantenimiento más constante del *framerate* cuando la capacidad de la red disminuye, y una mayor duración de comunicación útil cuando la calidad de la conexión cae por debajo del umbral fijo de Theora. Este comportamiento permite extender el rango operativo de los robots y mejorar su capacidad para manejar zonas de baja cobertura, facilitando su regreso a áreas con conexión estable.

De esta manera, se logra cumplir el objetivo final definido de desarrollar una solución de vídeo adaptativo en tiempo real, regulada mediante un algoritmo ABR que es capaz de mejorar la actual implementación de Theora.

Ademas, los experimentos realizados muestran que existe una transferencia de conocimiento directa, lo cual facilita la implementación y conecta ambos campos, destacando la relevancia de continuar investigando y mejorando los algoritmos ABR.

Como conclusión final, este trabajo, a través de un desarrollo teórico-práctico, demuestra cómo la incorporación de un sistema ABR en el marco de ROS 2 representa una estrategia altamente positiva, con una sinergia efectiva entre ambas tecnologías. Este estudio establece las bases necesarias para su integración y abre la puerta a futuras mejoras, fomentando un aprendizaje mutuo entre los sectores de la robótica y las telecomunicaciones.

## 6.2. Trabajos futuros

Al solo asentar las bases para la adaptación de la tecnología ABR al campo de la robótica, la propuesta ofrece un amplio margen de mejora para implementaciones futuras, en un abanico de posibilidades de distintas complejidades. Por ello, en esta sección se abordarán una serie de propuestas para evolucionar el sistema en trabajos futuros.

En orden de complejidad ascendente, una posible mejora es el ajuste fino (*Fine-tuning*) de los parámetros de configuración ya incorporados en este trabajo, explorando cuáles de ellos presentan una mejor adaptación para determinados tipos específicos de robots y modelos de comunicación, incrementando el rendimiento de manera sencilla.

Aumentando el nivel de complejidad, otra posibilidad es ampliar la oferta de

códecs, utilizando software especializado para las capacidades del *hardware* disponible. Este enfoque se beneficiaría de las mejoras introducidas por el estado del arte en el área multimedia y permitiría optimizar aspectos específicos como la eficiencia de compresión, la velocidad de codificación y el coste computacional, que son tres aspectos fundamentales en el contexto definido.

Otra línea de mejora es la adaptación de la lógica de ajuste del *bitrate* a diferentes condiciones de la red, o la incorporación de nuevas características de la red previamente no consideradas considerando factores como la distancia del robot al receptor o la potencia de radiación. Sería especialmente interesante incorporar información del contexto del robot, como la posición o datos de otros sensores, para mejorar el rendimiento del ABR en situaciones específicas. ROS 2, al incorporar Python, facilita la integración de estos datos con modelos de aprendizaje automático que puedan identificar nuevas relaciones y mejorar la adaptación.

Siguiendo con enfoques basados en análisis avanzado de datos, es posible mejorar el sistema ABR mediante un entendimiento más profundo del canal. En este sentido, sería interesante explorar predicciones de *throughput* a largo plazo, y no únicamente para el instante siguiente, empleando modelos capaces de captar patrones complejos en las mediciones de la red y de detectar tendencias a mayor plazo. Este enfoque también abre la posibilidad de intentar caracterizar fenómenos específicos que ocurren en el canal físico a través de la capa de aplicación, permitiendo una caracterización del canal en tiempo real que mejore la toma de decisiones.

Este enfoque se alinea con la posibilidad de caracterizar fenómenos específicos en el canal físico desde la capa de aplicación, permitiendo así una caracterización más precisa del canal en tiempo real y una toma de decisiones más informada.

Paralelamente, otra área de mejora podría ser el uso de la codificación de canal para incrementar la robustez en la comunicación, complementando la codificación de fuente utilizada para reducir el volumen de datos. Esta codificación consistiría en añadir redundancia al contenido original para que, en caso de pérdida, el contenido pueda reconstruirse a partir de la redundancia introducida, técnica conocida en inglés como *Forward Error Correction* (FEC). Al aplicar esta técnica desde la capa de aplicación, se haría uso de la codificación *Aplication Layer Forward Error Correction* (AL-FEC) ([Casu, 2017],[Rizzo, 1997]), permitiendo reforzar la entrega de paquetes de manera relativamente sencilla, sin introducir una alta complejidad computacional.

# Bibliografía

---

- [Bentaleb et al., 2019] Bentaleb, A., Taani, B., Begen, A. C., Timmerer, C., and Zimmermann, R. (2019). A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys Tutorials*, 21(1):562–585.
- [Biernacki, 2017] Biernacki, A. (2017). Improving quality of adaptive video by traffic prediction with (f)arima models. *Journal of Communications and Networks*, 19(5):521–530.
- [Blender Foundation, 2008] Blender Foundation (2008). Big buck bunny, 30 fps. Video. Disponible en <https://peach.blender.org/>.
- [Casu, 2017] Casu, F. (2017). Optimization of protection techniques based on fec codes for the transmission of multimedia packetized streams. Unpublished.
- [Chen et al., 2023] Chen, C., Liu, K., Dong, C., and Liu, G. (2023). Ld-abr: An adaptive bitrate algorithm for video transmission in wireless network. In *2023 IEEE 3rd International Conference on Computer Communication and Artificial Intelligence (CCAI)*, pages 370–375.
- [Daniel Bonilla Licea and Saska, 2024] Daniel Bonilla Licea, M. G. and Saska, M. (2024). When robotics meets wireless communications: An introductory tutorial. *Proceedings of the IEEE*, 112(2):140–151.
- [De Cicco et al., 2013] De Cicco, L., Calderalo, V., Palmisano, V., and Mascolo, S. (2013). Elastic: A client-side controller for dynamic adaptive streaming over http (dash). pages 1–8.
- [Developers, 2024] Developers, F. (2024). Ffmpeg. <https://ffmpeg.org/>. Free and open-source software for handling multimedia data.
- [Famaey et al., 2013] Famaey, J., Latré, S., Bouten, N., Van de Meerssche, W., Vleeschauwer, B., Van Leekwijck, W., and De Turck, F. (2013). On the merits of svc-based http adaptive streaming.

- [Huang et al., 2014] Huang, T.-Y., Johari, R., McKeown, N., Trunnell, M., and Watson, M. (2014). A buffer-based approach to rate adaptation: evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 187–198, New York, NY, USA. Association for Computing Machinery.
- [International Telecommunication Union, 2002]
- International Telecommunication Union (2002). *Terrestrial Land Mobile Radiowave Propagation in the VHF/UHF Bands*. Disponible en: <http://handle.itu.int/11.1002/pub/800c94ad-en>. Último acceso el 7 de abril de 2024.
- [Jiang et al., 2014] Jiang, J., Sekar, V., and Zhang, H. (2014). Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *IEEE/ACM Transactions on Networking*, 22(1):326–340.
- [Juluri et al., 2015] Juluri, P., Tamarapalli, V., and Medhi, D. (2015). Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1765–1770.
- [Lazaris and Koutsakis, 2009] Lazaris, A. and Koutsakis, P. (2009). Modeling video traffic from multiplexed h.264 videoconference streams. pages 1 – 6.
- [Lebreton and Yamagishi, 2021] Lebreton, P. and Yamagishi, K. (2021). Network and content-dependent bitrate ladder estimation for adaptive bitrate video streaming. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4205–4209.
- [Li et al., 2014] Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A. C., and Oran, D. (2014). Probe and adapt: Rate adaptation for http video streaming at scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733.
- [Liu et al., 2011] Liu, C., Bouazizi, I., and Gabbouj, M. (2011). Rate adaptation for adaptive http streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 169–174.
- [Liu et al., 2012] Liu, C., Bouazizi, I., Hannuksela, M., and Gabbouj, M. (2012). Rate adaptation for dynamic adaptive streaming over http in content distribution network. *Signal Processing: Image Communication*, 27:288–311.

- [Mao et al., 2020] Mao, H., Chen, S., Dimmery, D., Singh, S., Blaisdell, D., Tian, Y., Alizadeh, M., and Bakshy, E. (2020). Real-world video adaptation with reinforcement learning. *CoRR*, abs/2008.12858.
- [Mao et al., 2017] Mao, H., Netravali, R., and Alizadeh, M. (2017). Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17*, page 197–210, New York, NY, USA. Association for Computing Machinery.
- [Matsumoto et al., 2020] Matsumoto, T., Goto, K., and Yamamoto, M. (2020). On fairness issue of abr and tcp algorithms in video streaming. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–2.
- [Mihelich and Carroll, 2009] Mihelich, P. and Carroll, M. (2009). Ros2 image\_transport package. [https://index.ros.org/p/image\\_transport/](https://index.ros.org/p/image_transport/). The package provides efficient image transport in ROS2, supporting plugins for JPEG, PNG, and Theora formats. License: BSD.
- [Miller et al., 2016] Miller, K., Al-Tamimi, A.-K., and Wolisz, A. (2016). Qoe-based low-delay live streaming using throughput predictions. *ACM Trans. Multimedia Comput. Commun. Appl.*, 13(1).
- [PAL Robotics, 2023] PAL Robotics (2023). Tiago robot. Robot Móvil de PAL Robotics. Disponible en <https://pal-robotics.com/robots/tiago/>.
- [Palaios et al., 2021] Palaios, A., Vielhaus, C., Külzer, D. F., Geuer, P., Sattiraju, R., Fink, J., Kasparick, M., Watermann, C., Fettweis, G., Fitzek, F. H. P., Schotten, H. D., and Stańczak, S. (2021). Effect of spatial, temporal and network features on uplink and downlink throughput prediction. In *2021 IEEE 4th 5G World Forum (5GWF)*, pages 418–423.
- [Pozueco et al., 2013] Pozueco, L., Pañeda, X. G., García, R., Melendi, D., and Cabrero, S. (2013). Adaptable system based on scalable video coding for high-quality video service. *Computers Electrical Engineering*, 39(3):775–789. Special issue on Image and Video Processing Special issue on Recent Trends in Communications and Signal Processing.
- [Rizzo, 1997] Rizzo, L. (1997). Effective erasure codes for reliable computer communication protocols. *SIGCOMM Comput. Commun. Rev.*, 27(2):24–36.

- [SciPy Community, 2023] SciPy Community (2023). *SciPy Reference Guide: scipy.interpolate.interp1d*. Accessed: 2023-11-05.
- [Sieber et al., 2013] Sieber, C., Hossfeld, T., Zinner, T., Tran-Gia, P., and Timmerer, C. (2013). Implementation and user-centric comparison of a novel adaptation logic for dash with svc. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1318–1323.
- [Spiteri et al., 2020] Spiteri, K., Urgaonkar, R., and Sitaraman, R. K. (2020). Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711.
- [Union, 2021] Union, I. T. (2021). Advanced video coding for generic audiovisual services. Technical Report H.264, ITU-T. Tabla A-1 en p. 294. Disponible en: <https://www.itu.int/rec/T-REC-H.264-202108-I>.
- [USC SIPI, 2024] USC SIPI (2024). Image from USC SIPI Image Database. <https://sipi.usc.edu/database/database.php?volume=misc&image=10#top>. [Online; accessed 7-September-2024].
- [Utilities, 2024] Utilities, R. M. (2024). `ffmpeg_image_-transport` github repository. [https://github.com/ros-misc-utilities/ffmpeg\\_image\\_transport/tree/master](https://github.com/ros-misc-utilities/ffmpeg_image_transport/tree/master). Accessed: 2024-09-12.
- [Wang et al., 2016] Wang, C., Rizk, A., and Zink, M. (2016). Squad: a spectrum-based quality adaptation for dynamic adaptive streaming over http. In *Proceedings of the 7th International Conference on Multimedia Systems, MMSys ’16*, New York, NY, USA. Association for Computing Machinery.
- [Wang et al., 2018] Wang, K., Chen, L., Wang, S., and Wang, Z. (2018). Network traffic feature engineering based on deep learning. *Journal of Physics: Conference Series*, 1069(1):012115.
- [Xie et al., 2016] Xie, X., Zhang, X., Kumar, S., and Li, L. (2016). pistream: Physical layer informed adaptive video streaming over lte. *GetMobile: Mobile Computing and Communications*, 20:31–34.
- [Yanyuan et al., 2017] Yanyuan, Q., Jin, R., Hao, S., Pattipati, K., Qian, F., Sen, S., Wang, B., and Yue, C. (2017). A control theoretic approach to abr video streaming: A fresh look at pid-based rate adaptation. pages 1–9.

[Yin et al., 2015] Yin, X., Jindal, A., Sekar, V., and Sinopoli, B. (2015). A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev.*, 45(4):325–338.

[Álvaro Gutiérrez García, 2024a] Álvaro Gutiérrez García (2024a). Abr h264 vs theora [ros 2]: Fhd@30fps comparison video in changing network conditions. [Video] Disponible en: <https://www.youtube.com/watch?v=8eLEeWR9lw8>.

[Álvaro Gutiérrez García, 2024b] Álvaro Gutiérrez García (2024b). Abr image transport tfg. Repositorio en GitHub con código y documentación, enfocado en la creación de un plugin para ROS2 que implementa un transporte de imágenes adaptativo mediante FFmpeg para optimizar el ancho de banda en redes hostiles. Disponible en [https://github.com/agutig/ABR\\_image\\_transport\\_tfg](https://github.com/agutig/ABR_image_transport_tfg).

## Apéndice A

### Anexo 1

---

Level	Category	Base Rate (Kbits/s)	Factor	Max Rate (Kbits/s)	$FPS < 24$	$FPS \geq 24$	$FPS \geq 60$
1	0	64	1	64	2	1	
	1	64	1.25	80			
	2	64	3	192			
	3	64	4	256			
1.b	0	128	1	128	2	1	
	1	128	1.25	160			
	2	128	3	384			
	3	128	4	512			
1.1	0	192	1	192	[3-5]	2	1
	1	192	1.25	240			
	2	192	3	576			
	3	192	4	768			
1.2	0	384	1	384	[3-5]		[1-2]
	1	384	1.25	480			
	2	384	3	1152			
	3	384	4	1536			
1.3	0	768	1	768		[3-5]	[1-2]
	1	768	1.25	960			
	2	768	3	2304			
	3	768	4	3072			
2	0	2000	1	2000		[3-5]	[1-2]
	1	2000	1.25	2500			
	2	2000	3	6000			
	3	2000	4	8000			
2.1	0	4000	1	4000		[5-7]	[1-4]
	1	4000	1.25	5000			
	2	4000	3	12000			
	3	4000	4	16000			
2.2	0	4000	1	4000	8	[5-7]	[1-4]
	1	4000	1.25	5000			
	2	4000	3	12000			
	3	4000	4	16000			
3	0	10000	1	10000		[7-12]	[1-6]
	1	10000	1.25	12500			

	2 3	10000 10000	3 4	30000 40000			
3.1	0	14000	1	14000		[13-15]	[1-12]
	1	14000	1.25	17500			
	2	14000	3	42000			
	3	14000	4	56000			
3.2	0	20000	1	20000		[16-17]	[1-15]
	1	20000	1.25	25000			
	2	20000	3	60000			
	3	20000	4	80000			
4	0	20000	1	20000		[16-22]	[1-15]
	1	20000	1.25	25000			
	2	20000	3	60000			
	3	20000	4	80000			
4.1	0	50000	1	50000		[16-22]	[1-15]
	1	50000	1.25	62500			
	2	50000	3	150000			
	3	50000	4	200000			
4.2	0	50000	1	50000			[1-23]
	1	50000	1.25	62500			
	2	50000	3	150000			
	3	50000	4	200000			
5	0	135000	1	135000		[24-27]	[1-23]
	1	135000	1.25	168750			
	2	135000	3	405000			
	3	135000	4	540000			
5.1	0	240000	1	240000		[25-31]	[1-24]
	1	240000	1.25	300000			
	2	240000	3	720000			
	3	240000	4	960000			
5.2	0	240000	1	240000		31	[1-30]
	1	240000	1.25	300000			
	2	240000	3	720000			
	3	240000	4	960000			
6	0	240000	1	240000		[32-34]	[1-31]
	1	240000	1.25	300000			
	2	240000	3	720000			
	3	240000	4	960000			
6.1	0	480000	1	480000			ALL
	1	480000	1.25	600000			
	2	480000	3	1440000			
	3	480000	4	1920000			
6.2	0	800000	1	800000			ALL
	1	800000	1.25	1000000			
	2	800000	3	2400000			
	3	800000	4	3200000			

Cuadro A.1: Información de la información de interés de las tablas A-1,A-2 y A-3.

Nivel	Velocidad de decodificación (macroblocks/s)
1	1,485
1b	1,485
1.1	3,000
1.2	6,000
1.3	11,880
2	11,880
2.1	19,800
2.2	20,250
3	40,500
3.1	108,000
3.2	216,000
4	245,760
4.1	245,760
4.2	522,240
5	589,824
5.1	983,040
5.2	2,073,600
6	4,177,920
6.1	8,305,840
6.2	16,711,680

Cuadro A.2: Velocidad de decodificación en tiempo real por nivel

Level	Category	Base Rate (Kbits/s)	Factor	Max Rate (Kbits/s)	MB/s limit	FPS < 24	FPS < 60	FPS < 120
1	0	64	1	64	1,485	1		
	1	64	1.25	80				
	2	64	3	192				
	3	64	4	256				
1.b	0	128	1	128	1,485	1		
	1	128	1.25	160				
	2	128	3	384				
	3	128	4	512				
1.1	0	192	1	192	3000	2	1	1
	1	192	1.25	240				
	2	192	3	576				
	3	192	4	768				
1.2	0	384	1	384	6000	2	1	1
	1	384	1.25	480				
	2	384	3	1152				
	3	384	4	1536				
1.3	0	768	1	768	11,880	3-5	2	1

	1	768	1.25	960				
	2	768	3	2304				
	3	768	4	3072				
2	0	2000	1	2000	11,880	3-7		[1-2]
	1	2000	1.25	2500				
	2	2000	3	6000				
	3	2000	4	8000				
2.1	0	4000	1	4000	19,800	4-7	3	[1-2]
	1	4000	1.25	5000				
	2	4000	3	12000				
	3	4000	4	16000				
2.2	0	4000	1	4000	20,250	6,7	4,5	[1-2]
	1	4000	1.25	5000				
	2	4000	3	12000				
	3	4000	4	16000				
3	0	10000	1	10000	40,500	8-12	6,7	[1-5]
	1	10000	1.25	12500				
	2	10000	3	30000				
	3	10000	4	40000				
3.1	0	14000	1	14000	108,000	13-15	8-12	[1-7]
	1	14000	1.25	17500				
	2	14000	3	42000				
	3	14000	4	56000				
3.2	0	20000	1	20000	216,000	15-23	13,14	[1-12]
	1	20000	1.25	25000				
	2	20000	3	60000				
	3	20000	4	80000				
4	0	20000	1	20000	245,760	16-23	14,15	[1-13]
	1	20000	1.25	25000				
	2	20000	3	60000				
	3	20000	4	80000				
4.1	0	50000	1	50000	245,760	16-23	14,15	[1-13]
	1	50000	1.25	62500				
	2	50000	3	150000				
	3	50000	4	200000				
4.2	0	50000	1	50000	522,240	24-26	16-23	[1-15]
	1	50000	1.25	62500				
	2	50000	3	150000				
	3	50000	4	200000				
5	0	135000	1	135000	589,824	24-27	17-23	[1-16]
	1	135000	1.25	168750				
	2	135000	3	405000				
	3	135000	4	540000				
5.1	0	240000	1	240000	983,040	[25-31]	[21-24]	[1-20]
	1	240000	1.25	300000				
	2	240000	3	720000				
	3	240000	4	960000				
5.2	0	240000	1	240000	2,073,600	[30-31]	[25-29]	[1-24]

	1	240000	1.25	300000				
	2	240000	3	720000				
	3	240000	4	960000				
6	0	240000	1	240000	4,177,920	[32-34]	[31]	[1-30]
	1	240000	1.25	300000				
	2	240000	3	720000				
	3	240000	4	960000				
6.1	0	480000	1	480000	8,305,840		[32-34]	[1-31]
	1	480000	1.25	600000				
	2	480000	3	1440000				
	3	480000	4	1920000				
6.2	0	800000	1	800000	16,711,680			[1-34]
	1	800000	1.25	1000000				
	2	800000	3	2400000				
	3	800000	4	3200000				

Cuadro A.4: Información de la información de interés de las tablas A-1,A-2 y A-3, corregida para transmisiones en tiempo real.

No.	Format	Width	Height	MB/s@24	MB/s@60	MB/s@120
1	SQCIF	128	96	1152	2880	5760
2	QCIF	176	144	2376	5940	11880
3	QVGA	320	240	7200	18000	36000
4	525 SIF	352	240	7920	19800	39600
5	525 HHR	352	240	7920	19800	39600
6	CIF	352	288	9504	23760	47520
7	625 HHR	352	288	9504	23760	47520
8	VGA	640	480	28800	72000	144000
9	525 4SIF	704	480	31680	79200	158400
10	525 SD	704	480	31680	79200	158400
11	4CIF	704	576	38016	95040	190080
12	625 SD	704	576	38016	95040	190080
13	SVGA	800	600	45000	112500	225000
14	XGA	1024	768	73728	184320	368640
15	720p HD	1280	720	86400	216000	432000
16	4VGA	1280	960	115200	288000	576000
17	SXGA	1280	1024	122880	307200	614400
18	525 16SIF	1408	960	126720	316800	633600
19	16CIF	1408	1152	152064	380160	760320
20	4SVGA	1600	1200	180000	450000	900000
21	1080 HD	1920	1080	194400	486000	972000
22	2Kx1K	2048	1024	196608	491520	983040
23	2Kx1080	2048	1080	207360	518400	1036800
24	4XGA	2048	1536	294912	737280	1474560
25	16VGA	2560	1920	460800	1152000	2304000
26	3616x1536	3616	1536	520704	1301760	2603520
27	3672x1536	3672	1536	528768	1321920	2643840
28	3840x2160	3840	2160	777600	1944000	3888000
29	4Kx2K	4096	2048	786432	1966080	3932160
30	4096x2160	4096	2160	829440	2073600	4147200
31	4096x2304	4096	2304	884736	2211840	4423680
32	7680x4320	7680	4320	3110400	7776000	15552000
33	8192x4096	8192	4096	3145728	7864320	15728640
34	8192x4320	8192	4320	3317760	8294400	16588800

Cuadro A.3: Tabla enumerada de formatos de resolución

---

## **Apéndice B**

## **Anexo 2**

---

---

```

#!/bin/bash

#Configuración general
INTERFACE=""           # Nombre interfaz de red
TOTAL_DURATION=130
BURST_SIZE="32kbit"
LATENCY=400

#Configuración del Tramo A
DURATION_A=60
CONSTANT_RATE=15

#Configuración del Tramo B
DURATION_B=38
BASE_RATE=14
AMPLITUDE=1
FREQUENCY=0.10

#Configuración del Tramo C
DURATION_C=50
START_RATE=9

#Archivo CSV para almacenar los datos
CSV_FILE="network_behavior.log.csv"

# Función para aplicar el límite de ancho de banda y registrar en el CSV
apply_tc() {
    local rate=$1
    local timestamp=$(date -u +%s.%N)

    # Aplicar el límite de ancho de banda con 'tc'
    sudo tc qdisc replace dev "$INTERFACE" root tbf rate "${rate}mbit" burst "$BURST_SIZE" latency "${LATENCY}ms"

    # Registrar en el archivo CSV
    echo "$timestamp,$rate" >> "$CSV_FILE"

    # Mostrar en la terminal
    echo "[${timestamp}] - Ancho de banda: ${rate} Mbits/s"
}

# Función para eliminar la configuración de tc al final
cleanup() {
    sudo tc qdisc del dev "$INTERFACE" root
    echo "Limpieza completada."
}
trap cleanup EXIT

#Inicializar el archivo CSV yadir el encabezado
echo "timestamp,rate_mbps" > "$CSV_FILE"

#Tramo A - Conexión constante
echo "Iniciando Tramo A: "

for ((i = 0; i < DURATION_A; i++)); do
    apply_tc "$CONSTANT_RATE"
    sleep 1
done

#Tramo B - Comportamiento irregular
echo "Iniciando Tramo B: "
for ((i = 0; i < DURATION_B; i++)); do
    # Crear un patrón senoidal para generar el comportamiento cíclico reproducible
    # Fórmula: RATE = BASE_RATE + AMPLITUDE * sin(FREQUENCY * i * 2 * pi)
    RATE=$(echo "$BASE_RATE + $AMPLITUDE * s($FREQUENCY * $i * 2 * 3.14159)" | bc -l)
    RATE=$(printf "%.2f" "$RATE") # Formato a 2 decimales
    apply_tc "$RATE"
    sleep 1
done

#Tramo C - Reducción progresiva del bitrate
echo "Iniciando Tramo C: Reducción progresiva del bitrate hasta 0 durante ${DURATION_C} segundos..."
for ((i = 0; i < DURATION_C; i++)); do
    # Calcular la tasa actual basada en la reducción progresiva
    # Fórmula: RATE = START_RATE * (1 - i / DURATION_C)
    RATE=$(echo "$START_RATE * (1 - $i / $DURATION_C)" | bc -l)
    RATE=$(printf "%.2f" "$RATE") # Formato a 2 decimales
    apply_tc "$RATE"
    sleep 1
done

cleanup

```

---

Código B.1: Script bash utilizado para simular las condiciones de red.