PROYECTO FINAL DE INGENIERÍA

CONTRIBUCIÓN DE VIDEO DE ALTA CALIDAD SOBRE REDES NO GESTIONADAS PARA MÓVILES DE PRODUCCIÓN TELEVISIVA

Miceli, Nicolás – LU 101526
Ingeniería en Telecomunicaciones
Zunda Cornell, Santiago Andrés – LU 115614
Ingeniería en Telecomunicaciones

Tutor:

Ing. Esposito, Pablo Antonio - Universidad Argentina de la Empresa

2025

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

1. Resumen

Desde su concepción en la década del 1930 hasta hoy, la televisión y toda su cadena de producción y distribución han visto cambios y evoluciones tecnológicas que trajeron mejoras sustanciales en cuanto a la calidad y cantidad de las transmisiones, conforme las capacidades tecnológicas involucradas fueron incrementando.

Hacia finales del Siglo XX, en el mundo de las telecomunicaciones se empezó a gestar lo que se conoce como la "Convergencia IP", un proceso de integración de distintas tecnologías de las telecomunicaciones digitales en una plataforma común basada en protocolo de datos IP (*Internet Protocol*), permitiendo la intercomunicación de diferentes dispositivos en forma independiente de su tecnología subyacente. Esta Convergencia IP trajo consigo cambios sustanciales en la forma en que las personas se comunican, acceden a la información y consumen contenido.

Al contrastar el ritmo de evolución del proceso de Convergencia IP y de la tecnología empleada en televisión en la etapa de contribución (de contenido), donde prevalecen mecanismos para la producción y transmisión de índole analógica, que poseen una larga y probada trayectoria pero a un costo elevado y con limitaciones técnicas respecto a las nuevas demandas de los consumidores, consideramos que hay una oportunidad significativa para implementar tecnología moderna que permita realizar producciones de calidad comparable o superior, con una mayor flexibilidad, escalabilidad y a menor costo.

El presente trabajo se desarrolla como proyecto de reingeniería, en el marco del Proyecto Final de Ingeniería (PFI) para la carrera de Ingeniería en Telecomunicaciones de la Universidad Argentina de la Empresa (UADE), con el objetivo principal de desarrollar un esquema de servicio para la contribución de video de alta calidad sobre redes de datos no gestionadas para móviles de producción televisiva, con una significativa reducción de costos respecto a soluciones legadas.

La investigación realizada se desarrolla inicialmente brindando un marco teórico de las tecnologías, mecanismos y factores involucrados en la cadena productiva de televisión, para dar lugar luego a una propuesta tecnológica y económica resultante del análisis de los diversos factores actuantes. Se hace uso no solo de información y teoría para apoyar el análisis e hipótesis, sino también de experiencia en campo de primera mano, que reflejen propuestas realizables. Por último, se exponen las conclusiones vinculadas a la investigación, citadas a continuación.

La combinación de los diferentes componentes enumerados en la sección 6, de naturaleza modular, nos permite establecer flujos de trabajo para la realización de producciones remotas de contribución de video donde se maximiza la movilidad y el esquema de operación distribuido. Las señales de cámaras y fuentes remotas pueden ser enviados directamente por internet (y desde dispositivos móviles, por redes como LTE y 5G) hacia la nube, eliminando la necesidad de enlaces satelitales o enlaces fijos dedicados.

La baja latencia conjugada que brindan los componentes citados habilita la posibilidad de conmutar y producir en tiempo real, siempre y cuando las capacidades de procesamiento (ej. codificación/decodificación) sean suficientes.

Adicionalmente el ancho de banda que brinda redes como 5G y sus futuras iteraciones, permiten transmitir múltiples señales FHD o 4K en forma simultánea, facilitando producciones multicámara y de alta calidad.

La transición de un modelo de negocio intensivo en *CapEx* a uno flexible basado en *OpEx* representa el cambio más significativo en la economía de la producción televisiva en décadas. Este proyecto demuestra que dicha transformación no es meramente una opción, sino un imperativo estratégico impulsado por la necesidad de eficiencia, sostenibilidad y agilidad competitiva. La convergencia de las redes 5G no gestionadas y el protocolo de transporte SRT actúan como el catalizador tecnológico fundamental, haciendo que la producción remota de alta calidad sea no solo posible, sino económicamente viable a una escala sin precedentes.

El análisis de costos por escenarios revela que la adopción de flujos de trabajo REMI y equipamiento de menor costo puede reducir la inversión de capital en un orden de magnitud, pasando de millones de dólares para una OB-van tradicional a cientos de miles para una producción equivalente. Esta drástica disminución no solo optimiza los presupuestos, sino que, como se argumenta en la Introducción, "democratiza" el acceso a la producción de eventos en vivo, permitiendo la cobertura de un espectro mucho más amplio de contenido que antes era financieramente prohibitivo.

La prueba de campo del MVP, aunque es la implementación más básica, valida el principio fundamental: es posible lograr una contribución de video estable y de alta calidad sobre la internet pública utilizando herramientas accesibles. Esto confirma que la tecnología subyacente es sólida y escalable hacia los escenarios EFP y de grandes eventos más complejos.

2. Abstract

From its conception in the 1930s until today, television and its entire production and distribution chain have seen technological changes and evolutions that have brought substantial improvements in the quality and quantity of broadcasts, as the technological capabilities involved have increased.

Towards the end of the 20th century, the telecommunications world began to develop what is known as "IP Convergence", a process of integrating different digital telecommunications technologies into a common platform based on IP data protocol (Internet Protocol), enabling the intercommunication of different devices independently of their underlying technology. This IP Convergence brought about substantial changes in the way people communicate, access information, and consume content.

When contrasting the pace of evolution of the IP Convergence process and the technology used in television at the content contribution stage, where analog production and transmission mechanisms prevail, which have a long and proven track record but at a high cost and with technical limitations regarding new consumer demands, we consider that there is a significant opportunity to implement modern technology that allows for productions of comparable or higher quality, with greater flexibility, scalability and at lower cost.

This work is developed as a reengineering project, within the framework of the Final Engineering Project (PFI for its acronym in Spanish) for the Telecommunications Engineering degree at the *Universidad Argentina de la Empresa* (UADE), with the main objective of developing a service scheme for the contribution of high quality video over unmanaged data networks for mobile TV production, with a significant cost reduction when compared to legacy solutions.

The research initially provides a theoretical framework of the technologies, mechanisms, and factors involved in the television production line, leading to a technological and economic proposal resulting from the analysis of the various intervening factors. Not only

are information and theory used to support the analysis and hypotheses, but also first-hand field experience, reflecting feasible proposals. Finally, the conclusions related to the research are presented and cited here below.

The combination of the different components listed in section 6, which are modular in nature, allows us to establish workflows for remote video contribution productions that maximize mobility and distributed operation. Signals from cameras and remote sources can be sent directly over the internet (and from mobile devices, via networks such as LTE and 5G) to the cloud, eliminating the need for satellite links or dedicated fixed links.

The low latency provided by the aforementioned components enables real-time switching and production, as long as processing capabilities (e.g., encoding/decoding) are sufficient.

In addition, the bandwidth provided by networks such as 5G and its future iterations allows multiple FHD or 4K signals to be transmitted simultaneously, facilitating high-quality, multi-camera productions.

The transition from a CapEx-intensive business model to a flexible OpEx-based model represents the most notable change in the economics of television production in decades. This project demonstrates that such a transformation is not merely an option, but a strategic imperative driven by the need for efficiency, sustainability, and competitive agility. The convergence of unmanaged 5G networks and the SRT transport protocol acts as the fundamental technological catalyst, making high-quality remote production not only possible, but economically viable on an unprecedented scale.

Scenario-based cost analysis reveals that adopting REMI workflows and lower-cost equipment can reduce capital investment by an order of magnitude, from millions of dollars for a traditional OB-van to hundreds of thousands for an equivalent production. This dramatic decrease not only optimizes budgets but as presented in the Introduction, “democratizes” access to live event production, enabling coverage of a much broader spectrum of content that was previously financially prohibitive.

The MVP field test, although a basic implementation, validates the fundamental principle: it is possible to achieve stable, high-quality video contribution over the public internet using accessible tools. This confirms that the underlying technology is robust and scalable to more complex EFP and large event scenarios.

3. Contenidos

4. Introducción

La industria del *broadcasting* de televisión está atravesando un proceso de reformulación y de renovación impulsado no solo por los permanentes avances tecnológicos, sino también por las nuevas generaciones de consumidores, los tipos de contenidos audiovisuales, y la forma actual de producir/contribuir, distribuir y acceder a los mismos.

Desde la reciente pandemia de COVID-19, que vio alterados y acelerados diversos patrones de consumo de contenido digital, y que a su vez empujó una evolución en la conectividad fija y móvil por necesidad forzosa, también trajo consigo una mayor predisposición en la industria del *broadcasting* a montar producciones de transmisión digital (en especial producciones remotas) en reemplazo de las analógicas/híbridas legadas, que cuentan con probada trayectoria y uso predominante.

Adicionalmente, se observa la necesidad del mercado de cubrir una variedad de actividades y deportes más allá de los grandes eventos, sean por concurrencia o por televidentes, como podríamos enumerar el futsal de federaciones y ligas nacionales, hockey, ligas menores de futbol y básquet, rugby, vóley y su variante de playa, y categorías incipientes que anteriormente no se visibilizaban ni consumían, principalmente debido a los costos de producción remota que los volvía inviables en comparación con el presupuesto de recaudación potencial de dichos eventos. Todos estos factores de alguna forma han llevado a una ‘democratización’ del acceso a disciplinas y ligas no convencionales o de consumo masivo.

Entre los cambios de estos nuevos tipos de producción, y en el contexto de este proyecto, se puede mencionar la introducción de equipamiento de calidad *broadcast* en las etapas de producción y transporte, con un requerimiento de personal especializado más reducido, por ejemplo una OB-van (*Outside Broadcasting van*) o móvil de transmisión, cuyo presupuesto de equipamiento, personal y enlaces de transmisión tradicionales (analógico-digitales) presupone costos significativos para la realización de una transmisión.

Los principales retos que enfrentan las nuevas tecnologías digitales de producción y transmisión están vinculadas a la calidad y confiabilidad de las mismas, con la transmisión analógica-digital legada como base comparativa, así como también la disponibilidad tecnológica en toda la cadena (desde la captación hasta el consumidor), y la viabilidad de los flujos de producción.

Adicionalmente, existen cambios de paradigma en las formas de trabajo para los profesionales. Por ejemplo, un productor, que tradicionalmente se encuentra en un móvil o en una sala de dirección anexa a un estudio dirigiendo la transmisión, con las nuevas tecnologías digitales puede encontrarse ubicado a kilómetros de distancia del origen de la transmisión, inclusive en otro país o continente. Los servicios *online* brindados bajo la modalidad *Software- as-a-Service* (SaaS), o "en la nube", reducen significativamente los costos de equipamiento ya que solo se requiere una conexión a internet y una computadora o *tablet* para operar los *feeds*.

Por todo esto, para una implementación exitosa, consideramos que es fundamental el adecuado relevamiento tecnológico de los distintos y múltiples componentes que son parte de una cadena de transmisión, circunscripta a la geografía donde se desarrolla (ej. alcance, disponibilidad y calidad de redes de datos).

5. Descripción

5.1 Antecedentes y motivación

El sector de la televisión abierta (*broadcast*) ha dependido por muchos años de infraestructuras robustas, diseñadas específicamente con el propósito de garantizar la transmisión confiable de audio y vídeo de alta calidad desde ubicaciones remotas a instalaciones centrales de producción. Tradicionalmente, este proceso de contribución ha estado dominado por el uso de unidades móviles de transmisión (denominadas comúnmente como OB-van, *Outside Broadcast van*), equipadas con hardware específico como conmutadores (*switches*) de vídeo físicos, mezcladores de audio y enlaces de *uplink* por satélite, entre otros. Estos sistemas, aunque eficaces, se caracterizan por sus elevados costos de capital y operativos, una considerable complejidad logística y una limitada flexibilidad para adaptarse a los requisitos de producción que se encuentran hoy en rápida evolución.

En los últimos años, el panorama de las telecomunicaciones ha experimentado una transformación radical, impulsada por la proliferación de las redes móviles de alta velocidad y en especial la maduración de los protocolos de transporte basados en IP. La llegada de las redes móviles 5G, en particular, ha introducido un ancho de banda sin precedentes, una "ultra baja" latencia y una mayor confiabilidad, que las convierte en una alternativa atractiva a las soluciones tradicionales de satélite o enlaces dedicados para la contribución de la transmisión. Al mismo tiempo, el desarrollo de protocolos de transporte avanzados, como el SRT (*Secure Reliable Transport*) ha permitido la transmisión segura, de baja latencia y resiliente de vídeo de calidad profesional a través de redes no gestionadas, incluida la infraestructura móvil pública.

Esta convergencia, de la conectividad 5G y los protocolos de transporte IP modernos, presenta una oportunidad única para rediseñar los flujos de trabajo de contribución de la televisión. Aprovechando estas tecnologías, las emisoras pueden superar las limitaciones de las antiguas implementaciones de unidades móviles y enlaces satelitales dedicados, y adoptar

en su lugar soluciones ágiles, escalables y rentables que se ajustan mejor a las demandas y presupuestos de la producción de medios contemporánea.

5.2 Planteo de la problemática

A pesar de las fortalezas de las redes 5G y el transporte basado en IP, la transición de los flujos de producción con contribución tradicional hacia soluciones modernas e independientes de la red no está exenta de retos importantes. Las redes no gestionadas, como las redes 5G públicas, son intrínsecamente variables y carecen de las garantías determinísticas de calidad de servicio de los enlaces dedicados. La pérdida de paquetes, la fluctuación (*jitter*) y la variabilidad del ancho de banda pueden comprometer la integridad de las transmisiones de vídeo en directo, lo que supone un riesgo para la confiabilidad y calidad que exigen las emisoras broadcast.

Además, la integración de protocolos de transporte avanzados en los flujos de trabajo de transmisión, como SRT, RIST, NDI y otros, exige un conocimiento adicional tanto de la ingeniería de telecomunicaciones como del transporte de medios. SRT está diseñado para mitigar varios desafíos de las redes no gestionadas, pero al mismo tiempo introduce su propio conjunto de parámetros y compensaciones, como el tamaño del búfer, la gestión de la latencia y el cifrado de encriptación, que deben diseñarse cuidadosamente para lograr un rendimiento óptimo.

El problema central que se aborda en este proyecto es cómo diseñar, implementar y evaluar un flujo de trabajo de contribución para la televisión abierta que aproveche las redes móviles 5G y el protocolo SRT para el transporte de transmisiones de audio y video en vivo. La atención se centra estrictamente en contribución y en la capa de transporte, desde el punto de adquisición de la señal en el campo hasta la entrega de las transmisiones a la central de producción, dejando fuera del alcance la cadena de distribución al consumidor final.

Por último, es importante distinguir entre los conceptos de *broadcast* y *streaming*, ya que, si bien ambos involucran la transmisión de contenidos audiovisuales, presentan

diferencias técnicas, operativas y de propósito que afectan directamente el diseño de los flujos de trabajo de contribución. El *broadcast* televisivo tradicional se basa en la transmisión unidireccional de señales a una audiencia masiva mediante canales de transmisión gestionados y centralizados, como la televisión abierta o por cable. En este modelo, la señal es generada, procesada y transportada hasta un punto de emisión desde donde se distribuye de manera simultánea a todos los receptores, garantizando sincronización, calidad constante y cumplimiento de normativas regulatorias específicas. El flujo de contribución en broadcast, por lo tanto, exige alta confiabilidad, baja latencia y una robustez técnica capaz de sostener la calidad profesional requerida por la industria.

Por su parte, el *streaming* se basa en la entrega de contenido bajo demanda, típicamente a través de internet, donde cada usuario accede a la señal en momentos y condiciones personalizadas, y la transmisión se realiza en modo *unicast*, es decir, estableciendo un flujo individual para cada espectador. Si bien el *streaming* ofrece flexibilidad y personalización, suele tolerar mayores variaciones en calidad, latencia y continuidad, aspectos que serían inadmisibles en una transmisión *broadcast* en vivo. Por este motivo, el presente proyecto se orienta específicamente a los flujos de trabajo de contribución para televisión broadcast tradicional, priorizando la integridad, estabilidad y sincronización de la señal, y abordando los desafíos particulares que esto implica al utilizar redes no gestionadas como 5G en lugar de enlaces dedicados.

5.3 Objetivos del proyecto

El objetivo principal de este proyecto es rediseñar la etapa de contribución para un flujo de producción de televisión abierta o para un OTT, sustituyendo las tradicionales unidades móviles y el transporte por satélite por una solución que utilice redes móviles 5G y el protocolo SRT. Los objetivos específicos son los siguientes:

- Analizar las limitaciones de los flujos de trabajo de contribución legados, prestando especial atención a las restricciones técnicas y operativas impuestas por las soluciones de satélite y enlaces dedicados.

- Investigar las capacidades y limitaciones de las redes móviles 5G como medio de transporte para la contribución de transmisiones profesionales, incluyendo consideraciones de ancho de banda, latencia, confiabilidad y cobertura.
- Proporcionar un análisis técnico del protocolo SRT, incluyendo sus mecanismos de corrección de errores, gestión de la latencia, seguridad y adaptabilidad a condiciones de red variables.
- Diseñar e implementar un prototipo de flujo de trabajo de contribución que integre equipos de adquisición de campo, conectividad 5G y transporte basado en SRT, centrándose en el rendimiento y la confiabilidad en tiempo real.
- Evaluar el rendimiento del flujo de trabajo propuesto mediante pruebas empíricas, midiendo métricas clave como la latencia de extremo a extremo, la pérdida de paquetes, la fluctuación y la calidad general de vídeo/audio en condiciones de red variables.
- Identificar las mejores prácticas, las compensaciones de ingeniería y las posibles limitaciones en la implementación de SRT sobre 5G para la contribución de transmisiones broadcast, proporcionando recomendaciones prácticas para el futuro.
- Analizar un cambio de paradigma en la producción remota, migrando de modelos centralizados y dependientes de infraestructura (OB-vans, enlaces dedicados) a flujos de trabajo distribuidos, remotos y virtualizados, habilitados por SRT, las redes móviles de última generación y soluciones basadas en la nube.

5.4 Alcance y estructura del trabajo

Este proyecto final de Ingeniería se centrará en la etapa de contribución de la cadena de transmisión: el transporte de señales de audio y vídeo en directo desde lugares de producción remotos hasta las instalaciones centrales de transmisión.

El alcance abarcará lo siguiente:

- Adquisición en campo: la captura de señales de audio y vídeo en el lugar remoto, incluida la interfaz con codificadores y hardware de contribución.

- Capa de transporte: la transmisión de señales a través de redes móviles 5G públicas, utilizando el protocolo SRT para garantizar una entrega segura, confiable y de baja latencia.
- Recepción en la instalación central: la decodificación y el traspaso de las señales al entorno de producción, listas para su posterior procesamiento o distribución (lo cual queda fuera del alcance).
- La etapa de distribución al consumidor, la reproducción de contenidos y la entrega a los usuarios finales no se tratan en este trabajo. La atención se centra en los desafíos técnicos y las soluciones asociadas al transporte de señales de contribución a través de redes no gestionadas.

El proyecto se organiza de la siguiente manera:

- **Sección 1 (Capítulo 5): Descripción**
  Proporciona el contexto, la motivación, la exposición de la problemática elegida, los objetivos y el alcance del proyecto.

- **Sección 2 (Capítulo 6): Estado actual y estado del arte**
  Repasa los flujos de producción tradicionales y modernos, la evolución de los protocolos de transporte y el estado actual de la 5G y la SRT en la ingeniería de broadcast.

- **Sección 3 (Capítulo 7): Metodología y desarrollo**
  Detalla los costos, el diseño y la implementación del flujo de trabajo propuesto, incluyendo componentes del sistema, la integración de la 5G y la SRT, y la configuración experimental para la evaluación del rendimiento.

- **Sección 4 (Capítulo 8): Resultados**
  Presenta los resultados empíricos, incluidas las evaluaciones cuantitativas y cualitativas del rendimiento del flujo de trabajo propuesto en la sección 3.

- **Sección 5 (Capítulo 9): Conclusiones y observaciones**
  Sintetiza las ideas clave, analiza las ventajas e inconvenientes desde el punto de vista de la ingeniería y ofrece recomendaciones para las emisoras y los ingenieros que estén considerando realizar transiciones similares.

Debido a la naturaleza técnica del dominio tratado en este proyecto, y que es práctica habitual el uso de lenguaje, acrónimos y abreviaciones en idioma inglés, se incorpora un apartado al final del escrito (capítulo 10) con las distintas siglas empleadas. Adicionalmente, se resaltará a lo largo del texto en fuente formato '*itálica*' aquellas tecnologías y expresiones que por convención se expresan en inglés junto a una aclaración en español si contextualmente se considera necesario.

5.6 Impacto y relevancia previstos

La implementación exitosa de SRT sobre 5G para la contribución de transmisiones tiene el potencial de transformar la parte económica y logística de la contribución de medios en vivo. Al eliminar la necesidad de enlaces satelitales dedicados y reducir la dependencia de hardware especializado, las emisoras pueden lograr una mayor agilidad, escalabilidad y rentabilidad. El uso de redes móviles públicas democratiza el acceso a flujos de trabajo de contribución de alta calidad, lo que permite la cobertura de eventos en lugares que antes se consideraban poco prácticos o prohibitivos desde el punto de vista económico.

6. Antecedentes / Estado del arte

Nuestro análisis parte de la descripción de los diversos componentes partícipes en la compleja cadena de elementos necesarios para realizar transmisiones televisivas digitales de calidad *broadcast*, relevantes al estudio que realizaremos.

6.1 Supuestos preliminares

Con el objetivo de definir un alcance manejable y de referencia para el marco del Trabajo Final de Ingeniería, estaremos considerando los siguientes supuestos:

- La metodología de trabajo propuesta en este trabajo se enfocará en una producción de campo, por ejemplo de un evento deportivo que no sea un evento mayor, ya sea por concurrencia o por televidentes, los cuales usualmente no son cubiertos en transmisiones regulares de televisión deportiva debido la inviabilidad de realizar transmisiones de estas características, por un perfil de consumo de nicho.
  La magnitud y densidad de asistentes (y por ende de dispositivos de red) presentes en evento influye además en factores de calidad de transmisión de datos para redes inalámbricas, incluido 5G.

- Se plantea un escenario de disponibilidad y operatividad de redes móviles 5G que abarquen las zonas geográficas involucradas en la cobertura del lugar de desarrollo del evento deportivo, como también de la central del operador de la señal televisiva.

- En Argentina, dada sus características geográficas y su distribución poblacional predominantemente en grandes núcleos urbanos, la redes 5G se encuentran en expansión principalmente en dichos núcleos, lo cual puede implicar una limitante para llevar a cabo transmisiones utilizando el modelo operativo propuesto en este trabajo.
  A tal fin, es fundamental realizar trabajos de preproducción (frecuentes en la industria) para evaluar la factibilidad de realizar la transmisión con este medio de transmisión y equipamiento asociado.

6.2 Unidades móviles y esquema tradicional de transmisión

Como primer factor en la evaluación de este proyecto de reingeniería nos referiremos al arte previo, manifiesto en los métodos tradicionales de producción remota y transmisión televisiva, de los cuales luego buscaremos un reemplazo operativo equivalente que satisfaga los requerimientos necesarios para la producción remota.

Comenzaremos por las unidades móviles de producción televisiva, internacionalmente conocidas como "OB-vans" (*Outside Broadcast vans*), que han sido durante décadas el estándar de la industria para la cobertura profesional de eventos en vivo fuera del estudio. Su diseño y equipamiento permiten trasladar a campo la infraestructura técnica equivalente a la de un estudio de televisión, posibilitando la producción, mezcla, grabación y transmisión de señales audiovisuales de alta calidad desde cualquier ubicación.

6.2.1 Equipamiento y componentes

Una OB-van tradicional está equipada con una amplia variedad de sistemas y dispositivos especializados para garantizar la producción de calidad y transmisión en tiempo real bajo condiciones exigentes. Las OB-van varían en dimensiones, configuración y elementos en función del tipo de producción a realizar y el presupuesto asociado, pero podemos enumerar los siguientes componentes principales comunes a todas ellas:

- **Cámaras broadcast y controladores (CCU)**: Dependiendo del tamaño de producción, pueden emplear una multitud de cámaras de estudio de alta gama, conectadas a la unidad (ya sea por medios físicos como fibra óptica o cableado triaxial, o por medios inalámbricos como mochilas).
- **Mezcladora de producción (*video switcher*)**: Son equipos dedicados (hardware) para seleccionar, mezclar y aplicar efectos en tiempo real sobre las señales de video entrantes, permitiendo la producción multicámara.
- **Mezcladora de audio**: Consola de audio profesional, interconectada por redes dedicadas para la mezcla en vivo de múltiples fuentes.

- **"*Intercom*" y comunicación técnica**: Sistemas de intercomunicación para la coordinación del equipo técnico, que usualmente operan con matrices y paneles distribuidos.
- **Monitoreo de video y audio**: Matrices de monitores, "*multiviewers*" y monitores de audio para la supervisión por parte de los operadores.
- **Ruteo y distribución de señales**: Matrices de ruteo (ej. SDI/HD-SDI, etc.) para la gestión y distribución de señales dentro de la unidad móvil. También incluye sistemas conversores y generadores de señales de sincronismo.
- **Sistemas de grabación y reproducción**: Servidores de grabación multicanal, grabadoras tradicionales y sistemas para cámara lenta o repeticiones instantáneas.
- **Sistemas procesadores de video y audio**: Abarca una gama de equipos dedicados para realizar funciones especificas como corrección de color, inserción de gráficos, compresores de audio, etc.

6.2.2 Personal involucrado

Como se mencionó, en función del tipo de producción a realizar, la complejidad del evento y el presupuesto disponible, la operación de la OB-van puede involucrar una multitud de profesionales, con roles específicos y altamente especializados. Para una producción de gran envergadura, el personal clave incluye:

- **Director de televisión**: Responsable de la realización en tiempo real, selección de cámaras, coordinación artística y técnica general.
- **Productor**: Supervisa el contenido, los tiempos y la relación con el cliente o canal. Es el máximo responsable de la producción.
- **Director de Fotografía o Iluminador**: Especialmente en grandes eventos o producciones EFP, es el responsable de diseñar la estética visual, la iluminación de la escena o campo de juego y asegurar la consistencia y calidad de la imagen entre todas las cámaras.
- **Operador de *switcher* (Director técnico)**: Opera los cambios de cámara y efectos según las indicaciones del Director de televisión.

- **Ingeniero jefe (EIC, *Engineer in Charge*)**: Encargado del armado, mantenimiento y resolución de problemas técnicos de toda la infraestructura.
- **Técnicos de video (Operadores de CCU)**: Personal que realiza los ajustes finos de las cámaras (color, diafragma, etc.) para lograr una imagen uniforme entre todas las fuentes.
- **Operador de audio**: Mezcla y monitorea todas las fuentes de audio, instala micrófonos y gestiona el sistema de intercom.
- **Operador de replay y gráficos**: Gestionan las repeticiones instantáneas (*slow-motion*) y la inserción de gráficos en pantalla.
- **Asistentes de Cámara y Técnicos Adicionales**: Personal de apoyo fundamental que asiste a los camarógrafos, tiende cables (*grips*), y colabora en diversas tareas técnicas y de montaje.
- **Asistentes de producción y utileros**: Brindan apoyo en la logística, coordinación y necesidades generales de la producción en el campo.
- **Talentos (Presentadores, Comentaristas, Cronistas)**: El personal que aparece en cámara o provee la narración del evento. Suelen requerir monitoreo de video y audio específico (IFB - *Interruptible Foldback*).
- **Personal de Logística (Choferes, Coordinadores)**: Responsables del transporte seguro de la unidad móvil, el equipamiento y el personal hacia y desde la locación del evento.

6.2.3 Enlaces de subida de señal y factores adicionales

La transmisión de la señal principal (programa) desde la OB-van hacia el centro de producción o la emisora requiere de enlaces robustos y de alta capacidad. Tradicionalmente, los métodos más utilizados son:

- **Enlace satelital**: Utiliza antenas parabólicas motorizadas instaladas en el techo de la unidad móvil, capaces de transmitir señales en banda Ku o Ka a través de satélites geoestacionarios. Los enlaces satelitales pueden ser del tipo SNG (*Satellite New*

Gathering) o DSNG (*Digital Satellite News Gathering*), permitiendo la transmisión en tiempo real a grandes distancias, incluso en zonas sin infraestructura terrestre.

- **Enlace de microondas**: Para distancias más cortas, se utilizan enlaces punto a punto mediante microondas, considerados cuando se debe realizar la transmisión desde el sitio de producción a un centro de distribución cercano.
- **Fibra óptica**: Si la ubicación lo permitiese, se pueden utilizar enlaces de fibra óptica terrestre para el transporte de señal, ofreciendo mayor capacidad y baja latencia.
- **Redes dedicadas IP**: Las unidades que incorporan la posibilidad de transmisión IP pueden realizarlo por medio de redes privadas gestionadas.

Todos estos métodos requieren de diverso equipamiento asociado, que incluye amplificadores, codificadores/decodificadores de video, moduladores de señal, y el adecuado conectorizado de toda la cadena de transmisión (con vital importancia en las atenuaciones que puede introducir).

Por último, debemos también contemplar dentro del esquema de la unidad móvil las necesidades de energía y climatización adecuada, donde la presencia de equipamiento (hardware) y personal operativo hacen necesario un adecuado dimensionamiento y aprovisionamiento de generadores eléctricos, sistemas UPS, conexión a la red eléctrica, refrigeración para una operación segura, etc.

En conclusión, al enumerar todos los elementos que conforman una unidad móvil tradicional, el personal y el equipamiento dedicado para garantizar una producción remota de calidad, emergen factores como el alto costo (ya sea de capital, u operativo por contratación a un tercero), una logística y mantenimiento complejo (más aún si se consideran escenarios de traslado internacional), dificultades para operar en lugares de limitaciones físicas y de acceso, y limitada flexibilidad para adaptarse a los cambios de producción moderna.

6.3 Redes 5G / Redes no gestionadas

Las redes móviles de quinta generación, conocidas como "5G", representan un salto significativo en la tecnología de comunicación inalámbrica, y dadas las modernas prestaciones que incorpora, son una solución de características ideales para proyectos de transmisión de datos, y en particular de video, brindando una plataforma de red flexible, de alta calidad y con baja latencia.

6.3.1 Orígenes y evolución

A comienzos de este siglo empezó la ideación conceptual de lo que eventualmente serían las redes 5G, al reconocerse las limitaciones de las redes 4G, especialmente en términos de ancho de banda, latencia y la capacidad de conexión de múltiples dispositivos.

Con el advenimiento de nuevos perfiles de consumo de medios (ej. video a demanda, videollamadas, comunicaciones multipunto, realidad virtual/extendida, etc.), así como también la rápida evolución de lo que hoy llamamos IoT (*Internet of Things*), donde una multitud de dispositivos (que incluyen sensores, sistemas de automatización industrial, domótica, sistemas integrados, etc.) necesitan interconexión y flexibilidad para ser desplegados, pusieron en evidencia la necesidad de cubrir una demanda exponencial sobre las redes móviles.

Alrededor del 2010 comenzó el desarrollo formal de la tecnología 5G, con organismos como la UIT (Unión Internacional de Telecomunicaciones) y el 3GPP (*3rd Generation Partnership Project*) encabezando el proceso de estandarización. Entre 2012 y 2018, ambas organizaciones definieron los requisitos técnicos y publicaron normas claves (en particular las versiones 15 y 16 del 3GPP), que establecieron las características fundamentales del 5G: eMBB (*enhanced Mobile BroadBand*), URLLC (*Ultra-Reliable Low-Latency Communications*) y mMTC (*massive Machine-Type Communications*).

En 2019, Corea del Sur se convierte en el primer país en lanzar servicios comerciales de 5G, seguido por Estados Unidos, China y Europa. Entre 2020 y la actualidad las redes 5G se han expandido rápidamente, alcanzando zonas urbanas, suburbanas e incrementalmente

zonas rurales, dados los beneficios para los perfiles de uso previamente mencionados y la evolución tecnológica de antenas y dispositivos móviles. Según un informe de la consultora MarketsAndMarkets, se proyecta que el mercado global de servicios 5G crezca de 205 mil millones de dólares en 2023 a 497 mil millones de dólares para 2028.

En Argentina, el despliegue de la tecnología 5G comenzó en 2021 con pruebas iniciales por parte de Personal/Telecom Argentina. Los tres principales operadores (Claro, Movistar/Telefónica y Personal/Telecom Argentina) se aseguraron las bandas del espectro radioeléctrico de 3,5 GHz en una subasta crucial celebrada en Octubre de 2023, lo que permitió su despliegue en forma amplio. Movistar/Telefónica comenzó a ofrecer servicio 5G en 2024, y Claro en 2025.

Aunque el despliegue de 5G nacional está avanzado, aún sigue concentrado en los principales centros urbanos. La fase inicial de implementación se realizó principalmente en ciudades como Buenos Aires, Córdoba, Rosario y Mendoza, donde la densidad de población y la infraestructura de telecomunicaciones existente hacen que la expansión de la red sea más viable. La expansión de la red 5G está apalancada además por la Red Federal de Fibra Óptica (REFEFO), permitiendo una mayor interconexión nacional.

Figura 6.1: Diagrama de cobertura de la Red Federal de Fibra Óptica (REFEFO)

6.3.2 Características técnicas

Bandas de frecuencia:

Las redes 5G operan en un espectro mucho más amplio que las generaciones anteriores, utilizando tres bandas de frecuencia principales, las cuales proveen diferentes relaciones de velocidad de transferencia y distancia de cobertura. Las frecuencias utilizadas son:

- **Banda baja (<1 GHz)**: Similar al rango de frecuencia utilizado por redes 4G, con amplia cobertura, buena penetración en edificios, pero sus velocidades son relativamente bajas (5 a 250 Mbps).

- **Banda media (1-6 GHz)**: Es la más utilizada, equilibra la cobertura y la capacidad; incluye bandas muy utilizadas como la de 3,5 GHz (banda C) y permite buenas velocidades de transferencia (100 a 900 Mbps).

- **Banda alta (*mmWave*, 24-39 GHz+)**: Velocidades y capacidad muy altas, pero con alcance y penetración limitados, ideal para zonas urbanas densas y recintos cerrados.

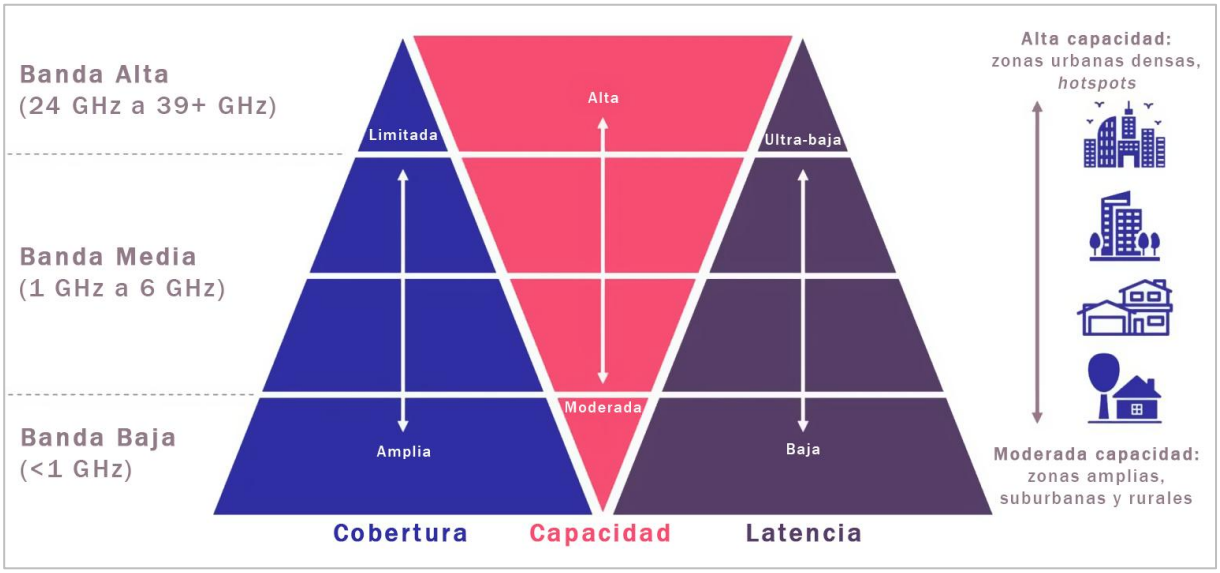


Figura 6.2: Esquema de bandas de frecuencia, y sus características principales

Latencia y rendimiento:

Latencia: El 5G puede alcanzar latencias de 1 milisegundo (ms), siendo una mejora exponencial con respecto a los 20-30 ms típicos del 4G. Esta latencia extremadamente baja es crítica para aplicaciones en tiempo real, como pueden ser una transmisión en vivo, juegos en la nube, una cirugía remota y el uso en vehículos autónomos.

Rendimiento: Las tasas máximas de transferencia de datos pueden alcanzar hasta 10 Gbps, con velocidades medias de uso superiores a las de 4G. Esto permite la transmisión fluida de vídeo en ultra alta definición (UHD) y permite aplicaciones que requieren un gran ancho de banda.

Arquitectura de red:

Aunque la arquitectura de red para redes 5G presenta una multitud de configuraciones, dada la naturaleza evolutiva que tuvo cada generación de red móvil y la tecnología implementada en cada geografía, los siguientes componentes clave son estructurales al armado de un esquema de red 5G:

- ***New Radio*** **(NR)**: La nueva interfaz de aire para 5G, admite bandas de espectro más amplias y tecnologías avanzadas como MIMO (por las siglas en ingles de “múltiples entradas, múltiples salidas”) masivo y formación de haces (llamado *"beamforming"* en inglés). La conjunción de ambas tecnologías permite multiplexado espacial de alta eficiencia, mejoras en la directividad y un excelente aprovechamiento de las antenas.

- ***5G Core Network*** **(5GC)**: Una arquitectura nativa de la nube y basada en servicios que permite una gestión flexible y escalable de las funciones de red, incluyendo la autenticación, la seguridad y la gestión de sesiones.

- **Red de acceso**: Abarca pequeñas células, "macro células" y sistemas de múltiples antenas distribuidas para proporcionar cobertura y capacidad.

- **Equipo de usuario (UE)**: Incluye teléfonos inteligentes (*smartphones*), dispositivos IoT y equipos de transmisión especializados.

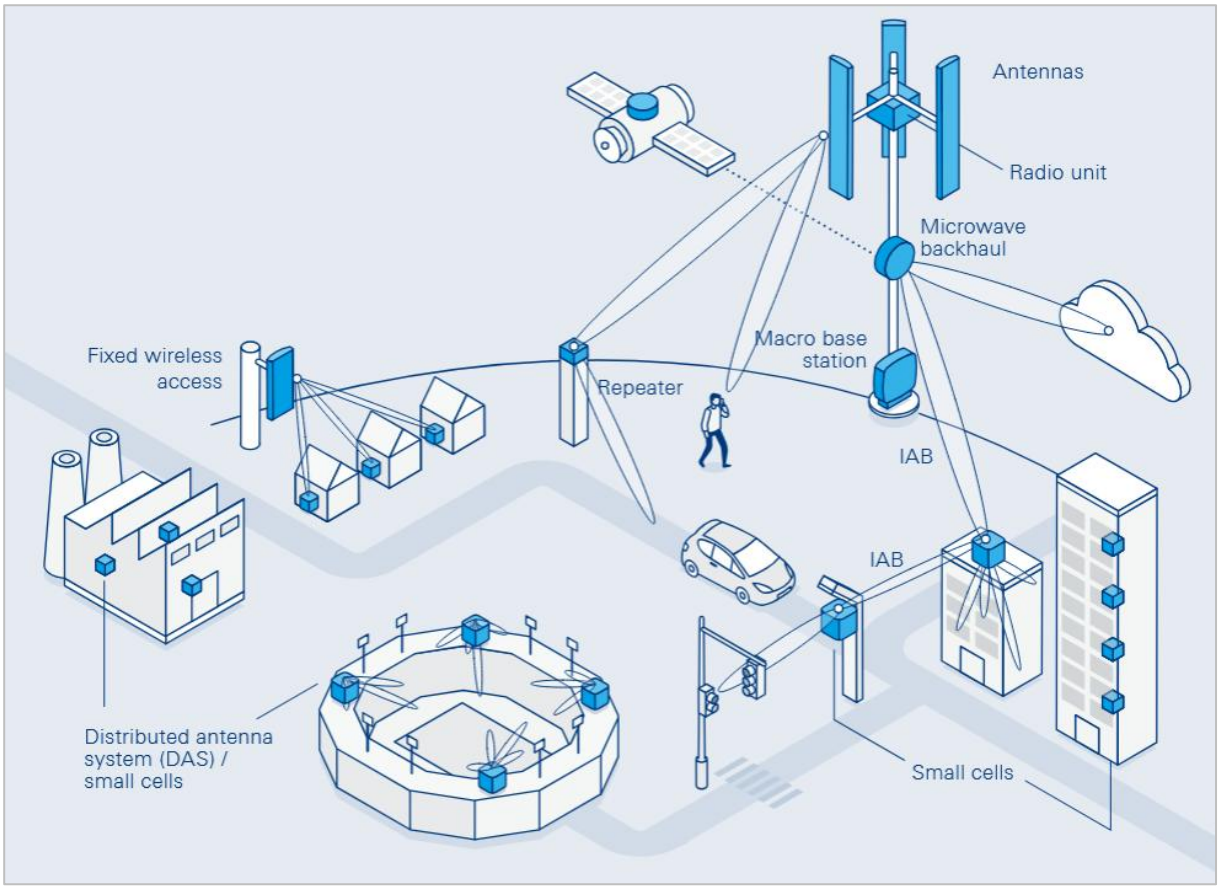


Figura 6.3: Esquema simplificado de infraestructura de red 5G, donde existe una multiplicidad de pequeñas antenas que proveen mayor directividad y ancho de banda.

Innovaciones claves relevantes:

Podemos destacar una serie de innovaciones relevantes que traen las redes de quinta generación respecto a las redes legadas:

- **Segmentación de red (*'network slicing'*)**: Permite a los operadores crear múltiples redes virtuales en una única infraestructura física, cada una de ellas optimizada para casos de uso específicos (por ejemplo, alto ancho de banda para vídeo, baja latencia para control operacional crítico, etc.).

- **Computación de borde**: Aproxima el procesamiento y el almacenamiento de datos al usuario (en lugar de concentrarlo en centro de cómputo), lo que reduce la latencia y mejora el rendimiento de las aplicaciones en tiempo real.

- **MIMO masivo y formación de haces (*"beamforming"*)**: Son técnicas que utilizan grandes conjuntos de antenas para aumentar la capacidad, la eficiencia espectral y la calidad de la señal.

- **Densidad y capacidad de dispositivos**: 5G puede admitir hasta un millón de dispositivos por kilómetro cuadrado, lo que permite implementaciones de IoT densas y aplicaciones para ciudades inteligentes (*"smart city"*).

Beneficios:

Enfocados en el contexto de uso para transmisiones de video y broadcast, podemos destacar los siguientes factores determinantes en la elección de esta tecnología:

- **Gran ancho de banda disponible**: Las redes 5G pueden admitir múltiples transmisiones de video de alta definición, transmisiones "4K" y "8K", múltiples *streams* de audio, juegos en la nube y experiencias inmersivas (como realidad virtual y realidad aumentada).

- **“Ultra-baja” latencia**: Permite la contribución de video en tiempo real, la producción remota y aplicaciones interactivas y bidireccionales.

- **Movilidad y flexibilidad**: Permite a los operadores de transmisión desplegar unidades móviles (ej. OB-van) en cualquier ubicación con cobertura 5G, eliminando la necesidad de enlaces satelitales o enlaces dedicados.

- **Escalabilidad**: La función de segmentación de red (*network slicing*) y la compatibilidad con un gran número de dispositivos conectados en forma simultánea permiten escalar las operaciones de transmisión para eventos masivos o para equipos de producción distribuidos.

- **Costo-eficiente**: Si la zona geográfica está cubierta, reduce la dependencia en costosos enlaces dedicados, enlaces de fibra o enlaces satelitales, bajando los costos operativos para transmisión remota.

- **Eficiencia energética**: La redes 5G fueron diseñadas para reducir el impacto energético requerido, reduciendo no solo los costos de operación sino también ayudando a alcanzar metas de sostenibilidad energética.

6.3.3 5G y redes no gestionadas

Un factor clave que evaluamos al plantear el modelo de trabajo presentado en este trabajo es la naturaleza de las redes no gestionadas, que se refiere a la presencia de segmentos de red o dispositivos de los cuales no se tiene una supervisión centralizada, con políticas de seguridad estandarizadas, o una gestión de la configuración homogénea. Esto es algo cada vez más frecuente para redes 5G, debido a que su naturaleza es descentralizada, se hace amplio uso de tecnología definida por software (conocidas en inglés como SDN, *software-defined networks*) y la virtualización.

El uso de redes no gestionadas para transmisiones de índole profesional trae aparejadas una serie de consecuencias que deben ser consideradas y, en la medida de lo posible, activamente mitigadas. Entre ellas, y en el contexto del trabajo, podemos destacar:

- Riesgos de ciberseguridad: El gran número de dispositivos conectados aumenta los posibles puntos de entrada para las amenazas de ciberseguridad, especialmente si los dispositivos no se gestionan o protegen adecuadamente.
  Adicionalmente, la dependencia que posee 5G en componentes distribuidos y definidos por software dificulta la aplicación de una política de seguridad uniforme, lo que aumenta el riesgo y la superficie de ataque para un actor de seguridad malicioso.
  Amenazas comunes como el espionaje, los ataques de denegación de servicio (DoS), las redes fraudulentas y las vulnerabilidades de la cadena de transmisión se incrementan y acentúan en entornos no gestionados. Por su parte, muchos dispositivos IoT (*Internet of Things*) carecen de una autenticación sólida y se actualizan con poca frecuencia, lo que los convierte en objetivos fáciles.

- Otro factor clave a evaluar es la calidad de servicio (conocida también como QoS, *Quality of Service*), que nos garantice el ancho de banda y latencia necesarias para llevar a cabo la transmisión. En redes no gestionadas, sin un manejo centralizado, la QoS puede ser impredecible, con riesgos de congestión de red, pérdida de paquetes y picos de latencia, lo que resulta especialmente problemático para las transmisiones en tiempo real.
  Por otro lado, si se utiliza la funcionalidad 5G de segmentación de red ('*network slicing*') anteriormente mencionada, puede traer aparejado que los segmentos mal gestionados no puedan cumplir los parámetros de QoS previstos, afectando a la experiencia del usuario final.

Con las consideraciones previas planteadas, las siguientes estrategias pueden ayudar a mitigar dichos riesgos:

- El monitoreo centralizado, implementados a través de plataformas unificadas para el monitoreo en tiempo real de las transmisiones realizadas, y que permita también la aplicación de políticas y configuraciones.

- Utilización de autenticación y cifrado de encriptación sólidos, aplicando una seguridad robusta para todos los dispositivos y datos en tránsito. Al trabajar en conjunto con el protocolo SRT (descrito más adelante), haremos uso de cifrados AES que proveen la seguridad necesaria para evitar espionaje o un uso indebido de los *streams* transmitidos.

- Mantener auditorías de sistemas periódicas, auditando continuamente las configuraciones y aplicando parches y actualizaciones de seguridad.

- La evaluación de seguridad de la cadena de transmisión, examinando a los proveedores utilizados y supervisando las vulnerabilidades, comunicaciones, etc.

Las redes 5G son consideradas una pieza fundamental en la próxima generación de transmisión broadcast digital y producción de medios. Su extensa historia de investigación y desarrollo, y el panorama de evolución futura, con el advenimiento de redes 5.5G y 6G, y evoluciones de la generación actual como el "5G Broadcast", nos brindan un panorama de la rápida transformación que estas tecnologías brindan en forma transversal a diversas industrias y aplicaciones.

El nivel y rapidez de adopción global, las altas velocidades que ofrece, la “ultra baja” latencia y la posibilidad de conexión masiva de dispositivos abren las puertas a nuevos paradigmas de producción de medios, desde su creación hasta su distribución y consumo.

La naturaleza no gestionada de las redes 5G y el uso intensivo de software impone nuevos desafíos para las emisoras, los ingenieros y los operadores de transmisiones, factores que deben ser considerados en el trabajo de preproducción, junto con el correcto análisis de disponibilidad en la ubicación geográfica donde se realizará la producción.

Dada la creciente adopción y evolución de esta tecnología y sus prestaciones técnicas, la consideramos un factor esencial y viable para la operatoria planteada en este trabajo.

6.4 Protocolo de transferencia SRT – *Secure Reliable Transport*

A medida que la industria del *broadcasting* televisivo se mueve de flujos de trabajo analógicos/híbridos hacia digitales completos, se hizo evidente la necesidad de contar con protocolos de red de transmisión de datos confiables, de baja latencia y que provean seguridad en el transporte de los *streams* de video.

El protocolo SRT (*Secure Reliable Transport*, por sus siglas en inglés) surgió como una alternativa para el uso en redes no gestionadas (impredecibles, a diferencia de enlaces dedicados). SRT opera en la capa de transporte del modelo OSI y es agnóstico al contenido del paquete, lo que significa que puede transportar cualquier códec de video/audio, cualquier resolución o velocidad de fotogramas. Esta flexibilidad permite su uso en una amplia gama de aplicaciones de transmisión y *streaming*.

Figura 6.4: Isologo del protocolo SRT

El protocolo fue originalmente desarrollado en el año 2013 por Haivision, una compañía con sede en Montreal, Canadá. Sin embargo, hasta 2017 permaneció de uso interno en la compañía, hasta su liberación en el año 2017, cuando deciden publicarlo con código abierto. Al mismo tiempo crearon la "Alianza SRT", con el objetivo de colaborar con los distintos actores de la industria para promover su adopción, uso y evolución. Estas acciones llevaron a una rápida adopción por parte de marcas principales como Microsoft, Avid, Telestream, Sony, AWS, etc. La alianza hoy cuenta con más de 500 entidades miembro y la adopción del protocolo va en crecimiento.

El desarrollo de SRT tuvo por objetivo resolver la problemática de transportar video de alta calidad, con baja latencia sobre la internet pública. Recordemos que internet presenta una serie de desafíos para este caso de uso, como pérdidas de paquetes, *jitter* y fluctuaciones

de ancho de banda, los cuales impactan directamente en la viabilidad y confiabilidad de un *stream* de video por este medio de conexión.

Durante su concepción, los protocolos existentes eran RTMP (*Real-Time Messaging Protocol*) y HLS (*HTTP Live Streaming*), los cuales no eran considerados lo suficientemente veloces o robustos para utilizar en contribución de video profesional en vivo.

6.4.1 Características principales

A nivel técnico, el protocolo SRT fue diseñado para *streams* de video y audio en tiempo real con baja latencia, y sus principales características incluyen:

- **Transporte basado en UDP**: El protocolo SRT opera con paquetes IP tipo UDP (*User Datagram Protocol*), los cuales a diferencia de paquetes TCP (*Transport Control Protocol*), no garantizan la entrega ni el secuenciamiento ordenado de paquetes en destino. Esto permite una transmisión de datos más rápida, al mismo tiempo que SRT tolera cierta pérdida de paquetes a favor de una menor latencia.

- **Recuperación de errores**: El protocolo compensa la falta de garantía de entrega de UDP mediante su propio algoritmo llamado ARQ (*Automatic Repeat reQuest*), que detecta los paquetes perdidos y solicita su retransmisión dentro de una ventana de latencia configurable. Adicionalmente, en forma opcional, se puede utilizar FEC (*Forward Error Correction*) para reconstruir los paquetes de datos perdidos sin necesidad de retransmisión.

- **Buffer para fluctuaciones por *Jitter***: SRT utiliza un buffer configurable para absorber la fluctuación de la red y darle tiempo al protocolo para que se recuperen los paquetes perdidos, buscando balancear confiabilidad y latencia.

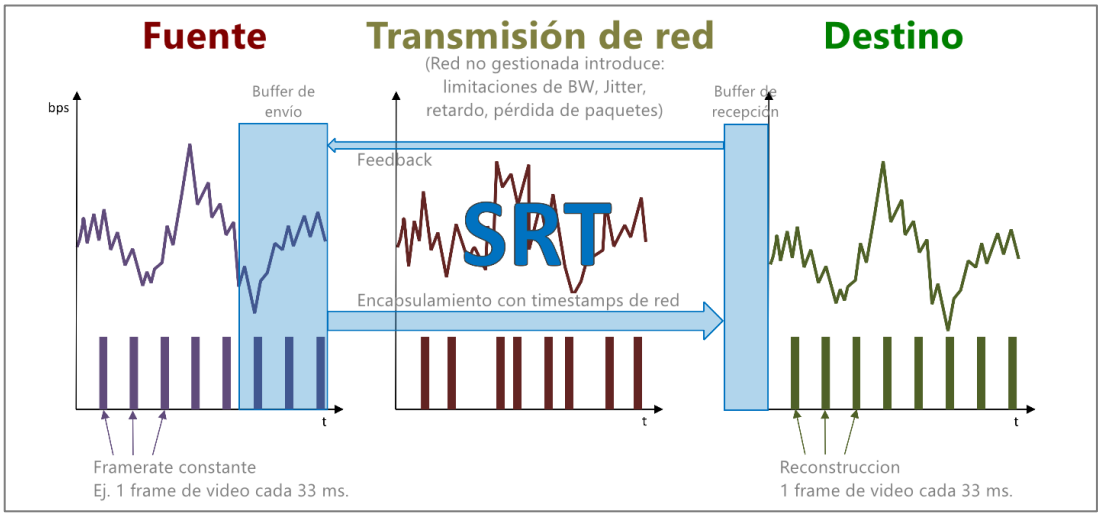


Figura 6.5: Esquema de buffers para compensación de fluctuaciones

- **Cifrado criptográfico**: SRT admite cifrado extremo a extremo mediante encriptación AES-128, AES-192 o AES-256, lo que garantiza que las transmisiones de video estén protegidas de ser interceptadas y manipuladas.

- **Gestión de conexiones**: El protocolo incluye funciones de *handshake*, gestión de sesiones, y NAT (*Network Address Translation*) transversal, y soporte para atravesar *firewalls*, volviéndolo ideal para entornos de redes complejos y de múltiples tramos.

- **Multiplexación**: SRT puede admitir múltiples transmisiones a través de una sola conexión, lo que aumenta la eficiencia en los flujos de trabajo con múltiples canales.

6.4.2 Beneficios

Analizando las ventajas de utilizar el protocolo SRT respecto a otros protocolos de transmisión como RTMP, podemos destacar las siguientes:

- **Baja latencia**: SRT está diseñado para una latencia por debajo de un segundo, haciéndolo ideal para transmisiones en vivo y aplicaciones en tiempo real. Esto supone una mejora significativa respecto a protocolos como RTMP y HLS que frecuentemente introducen varios segundos de retraso en las señales transportadas.

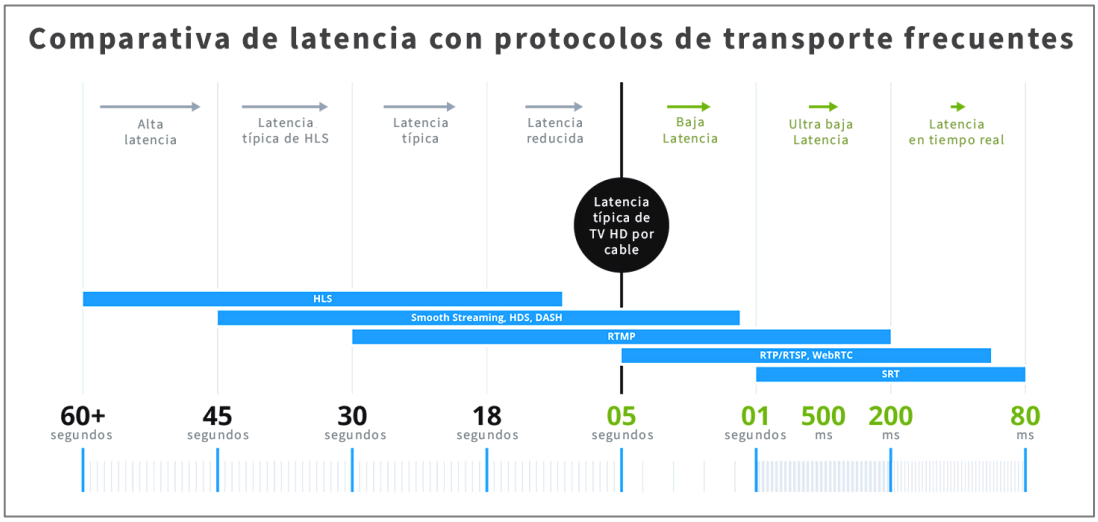


Figura 6.6: Comparativa de latencias con otros protocolos comunes de transporte

- **Alta confiabilidad en redes no gestionadas**: SRT se adapta dinámicamente a las condiciones cambiantes de la red, compensando la pérdida de paquetes, la fluctuación y las variaciones de ancho de banda. Esto nos garantiza video fluido y sin interrupciones incluso cuando las condiciones de conexión no son ideales, como la Internet pública o redes móviles 5G.

- **Calidad de video superior**: Los mecanismos avanzados de corrección y recuperación de errores evitan o minimizan los errores de imagen, que se producen típicamente cuando hay pérdida de paquetes, y mantienen una alta calidad de video, incluso cuando las condiciones de la red se degradan.

- **Alta seguridad**: Con el cifrado criptográfico AES integrado, SRT proporciona una protección sólida contra el espionaje y la manipulación, algo que otros protocolos como RTMP básico carecen.

- **Tasa de bits (*bitrate*) adaptativa y ajustes en tiempo real**: SRT puede ajustar la velocidad de retransmisión y los tamaños de buffer en tiempo real, optimizando el rendimiento en función de las condiciones actuales de la red.

- **Código abierto e interoperable**: La naturaleza de código abierto de SRT fomenta su adopción generalizada, con mejoras impulsadas por la comunidad e integración en una amplia gama de soluciones de hardware y software.

- **Funciones de transmisión profesional**: SRT admite transmisiones punto a punto, *feeds* de contribución seguros y flujos de trabajo para producción remota confiables, lo que lo hace muy adecuado para entornos de transmisión profesional.

6.4.3 Limitaciones y desventajas

Aunque SRT ofrece diversas ventajas como vimos anteriormente, es importante considerar sus limitaciones y posibles desventajas:

- **Posibilidad de aumento de la latencia**: Los mecanismos de corrección de errores de SRT, como la retransmisión de paquetes perdidos, pueden introducir una latencia adicional, especialmente en redes con alto nivel de pérdidas o alta fluctuación/*jitter*. Se trata de un balance entre confiabilidad y rendimiento en tiempo real.

- **Mayores requisitos computacionales**: Las funciones avanzadas del protocolo (cifrado criptográfico, corrección de errores, *bitrate* adaptativo) requieren un mayor procesamiento que protocolos más simples, lo cual en implementaciones a gran escala puede implicar que sea necesario el uso de hardware moderno, potencialmente de mayor costo.

- **Complejidad en la configuración y el mantenimiento**: SRT es más complejo de configurar y mantener que los protocolos legados, ya que requiere un ajuste cuidadoso de parámetros como los *buffers* de latencia y la recuperación de paquetes perdidos.

- **Utilización del ancho de banda**: Dependiendo la calidad de los tramos que conformen la red de datos que se utilizará, la retransmisión de paquetes perdidos puede aumentar el uso del ancho de banda, especialmente en enlaces inestables o de poco ancho de banda.

- **Problemas de compatibilidad**: No todos los equipos legados de transmisión *broadcast* son compatibles con SRT de forma nativa, lo que puede requerir actualizaciones (de hardware o software) para habilitar su operación.

- **Curva de aprendizaje**: Existe una curva de aprendizaje para los profesionales de la transmisión acostumbrados a protocolos más antiguos, aunque esto se ve mitigado por la creciente documentación y el soporte abierto de la comunidad.

6.4.4 SRT y código abierto

Una característica definitoria de SRT es su condición de código abierto. El código fuente del protocolo se encuentra abiertamente disponible en GitHub (repositorio de código fuente online) bajo la licencia Mozilla Public License 2.0 (MPLv2.0), lo que permite a cualquiera utilizar, modificar y distribuir el software, siempre que las modificaciones también se pongan a disposición bajo la misma licencia.

El proyecto SRT es mantenido por Haivision, pero existe también una amplia comunidad de miembros colaboradores y la Alianza SRT. Este enfoque colaborativo garantiza que SRT evolucione en respuesta a las necesidades del mundo real y se beneficie de una amplia gama de conocimientos especializados. El modelo de código abierto acelera la innovación, fomenta la interoperabilidad y proporciona transparencia: los usuarios pueden auditar el código para comprobar su seguridad y confiabilidad, lo que es especialmente importante para las aplicaciones de transmisión de misión crítica. Adicionalmente, y fundamental para la ecuación de costos de las producciones televisivas es que no requiere pago de licencias por su utilización.

La extensibilidad de SRT es otra ventaja. Los desarrolladores pueden crear complementos o modificar el protocolo para adaptarlo a requisitos específicos, como protocolos de cifrado de uso especial, o la inspección de paquetes. El ecosistema de código abierto también ha dado lugar al desarrollo de herramientas, software complementario y soluciones de monitoreo de terceros, lo que amplía aún más las capacidades de SRT.

6.4.5 Adopción en el mundo real e impacto en la industria

El lanzamiento del código abierto de SRT y la colaboración de la industria impulsaron una rápida adopción. En la edición 2022 del reporte anual que realiza Haivision sobre la evolución del *broadcast* IP, SRT alcanzó el primer puesto de protocolos de transporte utilizado por la industria para video, con una fuerte tasa de crecimiento y progresiva integración en codificadores y decodificadores de transmisión profesionales, y plataformas basadas en la nube.

Las principales emisoras a nivel mundial utilizan SRT para la retransmisión de eventos en directo, la producción remota, la emisión basada en la nube y la transmisión directa al consumidor. Su flexibilidad le permite sustituir enlaces de alto costo vía satélite o fibra óptica, habilitar la producción remota y móvil a través de 5G, y admitir flujos de trabajo basados en la nube.

6.4.6 Arquitectura

A nivel técnico, SRT es un protocolo de transporte de datos, que se trata de una evolución de un protocolo previo llamado UDT (“*UDP-based Data Transfer Protocol*”), diseñado para transferencia de datos de alta performance, con especial foco en transporte de video. Como observamos en la figura 6.7, la arquitectura de SRT se sitúa entre la capa de aplicación y la de transporte del modelo TCP/IP, utilizando tráfico UDP como base debido a su bajo *overhead*, y flexibilidad en la gestión de conexiones.

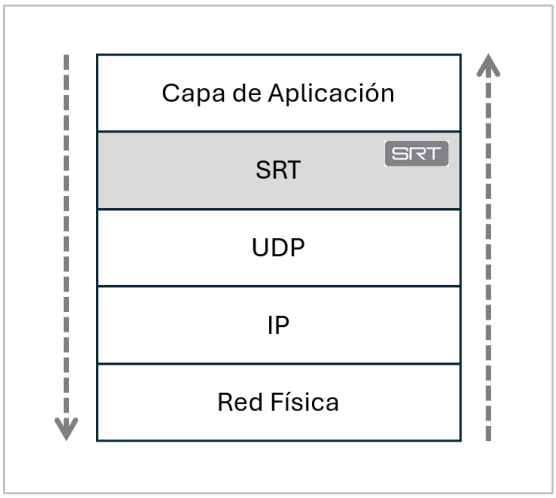


Figura 6.7: Ubicación de SRT en la pila de protocolos del modelo TCP/IP, entre las capas de aplicación y de transporte UDP/IP

A priori y en comparación con el tráfico TCP, UDP no maneja velocidades de transmisión del tráfico (conocido como *throttling*), lo cual permite velocidades de transferencia mayores. UDP además no mantiene control de sesiones (se lo denomina *stateless*), lo cual facilita una recuperación mas rápida ante desconexiones momentáneas.

Sin embargo, estos beneficios tienen como contraparte que UDP carece de las características de confiabilidad necesarias para un *stream* de calidad, y es aquí donde SRT introduce sus mejoras críticas. Entre ellas destacamos:

- **Retransmisión de paquetes (ARQ)**: Por medio de un mecanismo denominado ARQ (*Automatic Repeat reQuest*), cuando se detecta una pérdida de paquete, SRT solicita automáticamente la retransmisión del dato faltante. A diferencia de TCP, estas retransmisiones son selectivas y optimizadas para medios en tiempo real.

- ***Buffer* de latencia configurable**: SRT mantiene un *buffer* de paquetes recientemente transmitidos, manteniéndolos listos para una posible retransmisión. El tamaño de este buffer (y por ende de la latencia total) puede configurarse para lograr el balance entre confiabilidad y retardo en el *stream*.

- **Corrección de errores con FEC**: Algunas implementaciones de SRT incluyen la posibilidad de utilizar FEC (*Forward Error Correction*) o codificación de canal, una técnica frecuente en telecomunicaciones para controlar errores en transmisiones sobre canales poco confiables o con ruido. Esto habilita un margen de pérdida de paquetes que puede suceder sin requerir retransmisión, reduciendo aún más la latencia.

- **Manejo activo de la ventana de paquetes**: Las transmisiones de paquetes TCP pueden verse afectadas por una mala implementación del flujo de control (conocida como SWS, "*silly window syndrome"*), que deriva en paquetes de tamaño inadecuado y enlentecimiento del procesado (problema que se compensa con el denominado "algoritmo de Nagle").
  SRT en cambio no hace combinaciones de paquetes pequeños dentro de paquetes grandes, evitando la necesidad de aplicar el algoritmo mencionado y previniendo de este modo latencia adicional que pudiera impactar aplicaciones en tiempo real.

SRT incluye además características que facilitan su implementación en entornos de red complejos, con distintos modelos de conexión adaptables a diferentes topologías y restricciones de red:

- **Modelo "*Caller/Listener*" (Cliente/Servidor)**: Utilizado típicamente cuando uno de los extremos tiene dirección IP pública y puede recibir conexiones entrantes. En este esquema, SRT se configura con un extremo con rol de "*listener*" (servidor, tanto *on-premises* como en la nube) que espera conexiones entrantes, y el otro extremo como "*caller*" (cliente) que típicamente será el emisor y que se encarga de la conexión hacia el "*listener*". Ver figura 6.8.
  Una ventaja de SRT en este modelo, es que el "*listener*" puede recibir múltiples conexiones entrantes en un mismo puerto, simplificando su implementación.

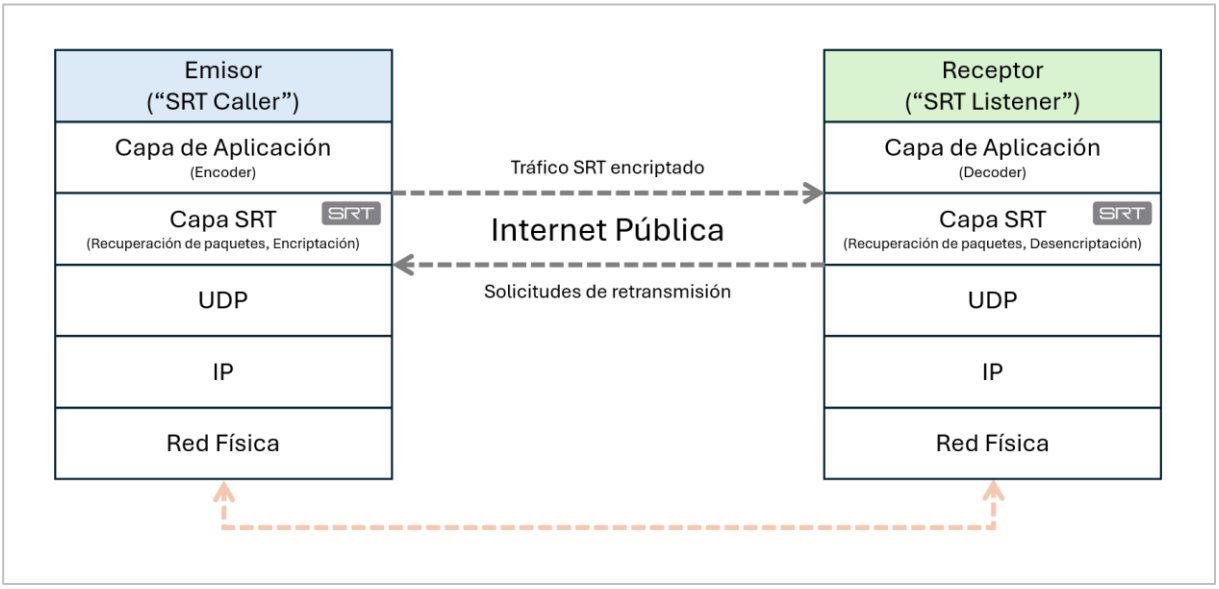


Figura 6.8: Modelo "*caller/listener*" (cliente/servidor) de SRT

- **Modelo "*Rendesvouz*" (*Peer-to-peer*)**: En este caso, ambos extremos SRT intentan conectarse entre sí simultáneamente, negociando el establecimiento de sesión y facilitando la conexión a través de escenarios que integran NAT (traducción de direcciones IP) o firewalls restrictivos.
  Al configurar este modo, los dispositivos operan como pares, buscando establecer el canal de transmisión sin necesidad de configurar, por ejemplo, reglas de *forwarding* de trafico de red. SRT logra este objetivo utilizando el mismo numero de puerto en ambos extremos, de tal modo que para un firewall *stateful* (los más frecuentes, que inspeccionan de paquetes y realizan seguimiento de las conexiones, gestionando una tabla interna de conexiones), el tráfico externo que recibe con destino interno lo interpreta como trafico aceptable y autoriza su ingreso.
  Debido a que en la mayoría de los escenarios, los dispositivos 'de borde' (firewalls y routers) transfieren desde direcciones IP de la red LAN (privada) hacia IPs de la red pública, se aplica NAT/PAT en las direcciones, lo cual altera el puerto origen al convertir las direcciones. Por este motivo, el modo "rendezvous" es de uso infrecuente, siendo más practico configurar reglas de "*port forwarding*".

- Una vez establecida la conexión, SRT es *full-duplex*, permitiendo el envío y recepción simultánea de datos en ambos sentidos.

Por último, mencionaremos una característica de SRT denominada “StreamID”, que básicamente incorpora metadatos al *stream*, los cuales pueden tener usos múltiples y arbitrarios, como ser ruteo, autenticación o identificación del contenido. Esto deriva en un mejor aprovechamiento de la arquitectura. Un caso de uso es el mencionado anteriormente, donde en un misma IP/puerto podemos recibir múltiples *streams*, y por medio de StreamIDs diferentes, los podemos luego discriminar.

6.4.7 Seguridad informática y SRT

La seguridad es un aspecto fundamental y un componente central en el diseño del protocolo SRT. Sin embargo, requiere atención a distintos factores involucrados en su correcta configuración. No gestionar adecuadamente los aspectos de seguridad informática en una transmisión de video IP (mediante SRT u otros protocolos) puede exponer a la organización a una serie de riesgos críticos con impacto directo en la operación y el negocio.

Para ejemplificar, una mala gestión de las claves de cifrado (PSK) o el uso de claves débiles puede permitir que atacantes intercepten y descifren el contenido transmitido, facilitando el acceso no autorizado a material exclusivo, la piratería de contenidos y la filtración de información sensible.

Adicionalmente, la falta de autenticación robusta y la exposición de puertos “*listener*” sin la protección adecuada (ej. firewalls, listas de control de acceso) convierten a los equipos en objetivos vulnerables a ataques de denegación de servicio (DDoS) o intentos de acceso no autorizado, lo que puede provocar la interrupción de transmisiones en vivo, pérdida de audiencia y daños reputacionales.

Al operar además con software, dispositivos *software-defined*, y hardware con firmware que acepte actualizaciones, es fundamental seguir las buenas prácticas de seguridad informática (ej. uso de cifrado fuerte, autenticación adecuada, protección de puertos de red, actualizaciones regulares, monitoreo activo, etc.) para evitar que una solución tecnológica eficiente puede transformarse en un vector de riesgo operativo, legal y reputacional para los operadores.

A nivel técnico, SRT incorpora las siguientes características para establecer su marco de seguridad, esquematizado en la figura 6.9:

- **Cifrado de encriptación** AES: SRT soporta cifrado AES-128, AES-192 o AES-256, protegiendo los datos efectivos (llamado comúnmente *payload*) de los paquetes.
- **Clave pre-compartida**: El cifrado AES se basa en el uso de una clave pre-compartida (PSK, *pre-shared key*) que debe ser configurada en ambos extremos antes de establecer la conexión. Es fundamental utilizar claves robustas y gestionarlas en forma adecuada y segura, evitando su exposición o reutilización excesiva.
- **Intercambio seguro de claves**: Durante el establecimiento de la conexión, las claves utilizadas para el cifrado son intercambiadas en forma segura.
- **Aislamiento de *stream***: Cada *stream* SRT puede tener su propia configuración independiente de cifrado, lo cual permite un control más granular de la seguridad.

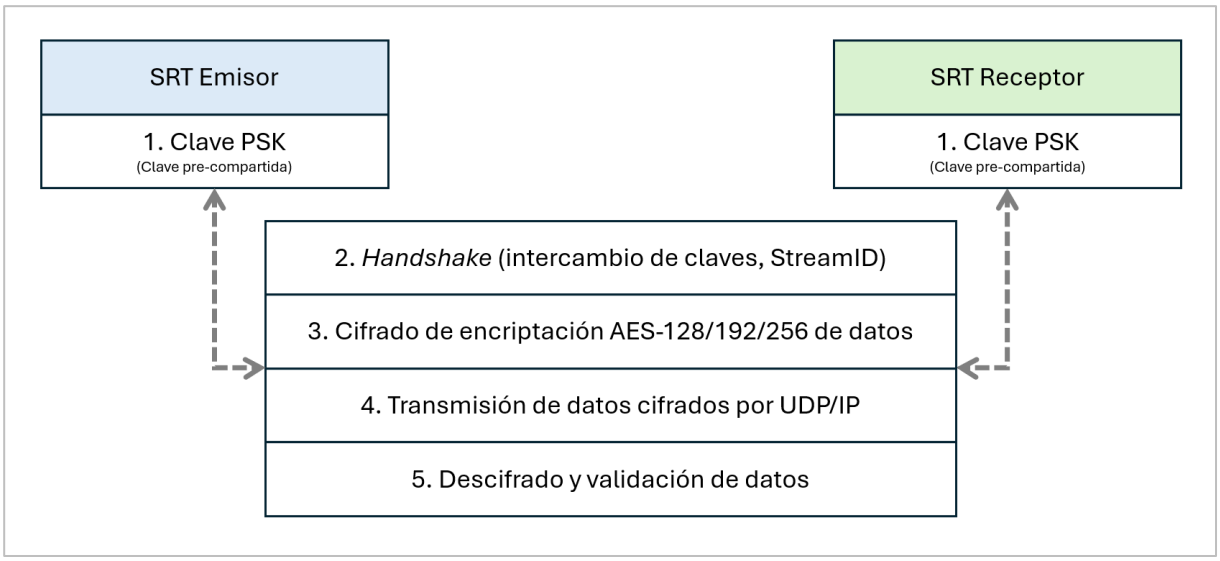


Figura 6.9: Secuencia de seguridad en flujo SRT. Ambos extremos utilizan la misma clave, realizan el handshake y transmiten datos cifrados. El "StreamID" puede incluir metadatos para autenticación adicional.

Debemos tener presentes tres factores que se desprenden de estas características. La autenticación se basa en la posesión de la clave PSK. Si un *endpoint* no tiene la clave correcta, la conexión no se establece. Sin embargo, SRT no implementa autenticación basada en certificados ni mecanismos avanzados como TLS, por lo que la seguridad depende principalmente de la gestión de las claves.

El uso de claves estáticas puede ser un riesgo si no se rotan periódicamente o si se comparten entre múltiples *endpoints*. La distribución segura de claves es crítica, especialmente en entornos con muchos dispositivos o personal rotativo.

Considerar además la exposición de puertos de red. Los *endpoints* configurados como "*listener*" y expuestos a internet pueden ser blanco de ataques de denegación de servicio (DDoS) o intentos de acceso no autorizado. Una recomendación es proteger estos *endpoints* con firewalls, listas de control de acceso y, si es posible, mecanismos para mitigación de DDoS.

6.4.8 Parámetros de configuración SRT

La correcta configuración de los parámetros de protocolo SRT es necesaria para garantizar transmisiones de calidad, confiabilidad y seguridad sobre redes IP. A continuación, se describen algunos conceptos y parámetros principales a considerar.

- **Modelo de consumo de ancho de banda**
  Como vimos, SRT utiliza UDP como base e incorpora mecanismos de control, corrección de errores y cifrado, lo que trae aparejado una sobrecarga (llamado usualmente *overhead*) por encima del *bitrate* nominal del contenido. Este *overhead* es necesario para gestionar retransmisiones, paquetes de control y señalización, y debe ser considerado al dimensionar el ancho de banda del enlace de red (ver esquema en la figura 6.10).

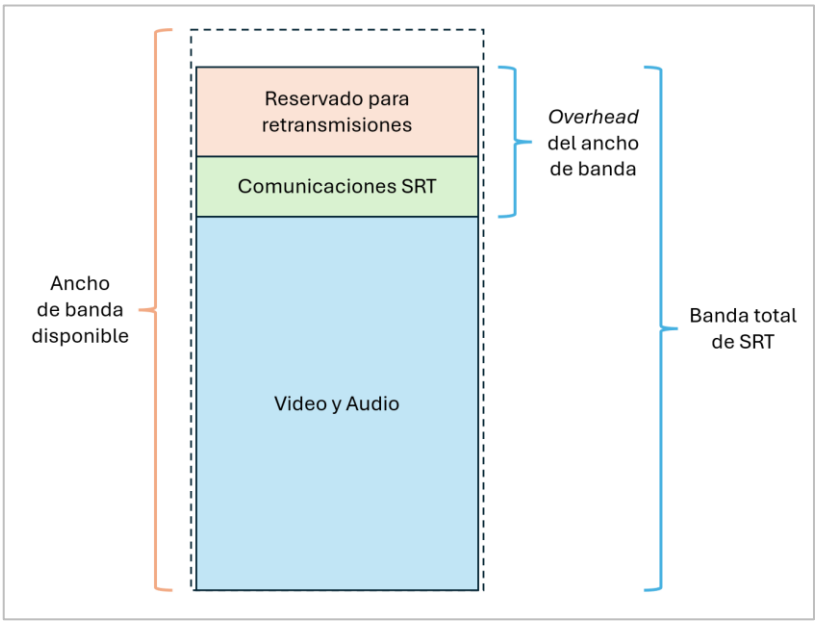


Figura 6.10: Esquema de sobrecarga (*overhead*) del ancho de banda

Por documentación y en forma empírica, se recomienda reservar típicamente un *overhead* del 25% sobre el *bitrate* del *stream* para garantizar su calidad, especialmente en redes que puedan presentar inestabilidad o pérdidas significativas de paquetes.
Para ejemplificar, si el *bitrate* de video es 1000 kbps y el del audio es 128 kbps, la suma nos da 1128 kbps. Al aplicarle el *overhead* del 25% (1128 kbps * 1,25 = 1410 kbps), nos da como resultado que debemos reservar al menos 1,41 Mbps para ese *stream*.

Nótese que al transmitir varios *streams* SRT simultáneamente hacia un mismo punto (ej. a un servidor o a la nube), el consumo de ancho de banda es acumulado. Cada conexión gestiona su propio *overhead* y retransmisiones, por lo cual el ancho de banda total requerido es la suma de todos los *streams* individuales, cada uno con su margen de *overhead* aplicado (formula i).

$$\boldsymbol{BW_{Total}} = \sum(BW_i + Overhead_i) = \sum(BW_i * 1{,}25) \qquad \text{(i)}$$

El protocolo SRT incorpora un parámetro llamado “maxbw” que se utiliza para

definir el ancho de banda máximo permitido para la transmisión (medido en bytes por segundo). Su valor por defecto es "-1", que implica ancho de banda máximo ilimitado hasta 1 Gbps. Es recomendado configurarlo con al menos el doble de *bitrate* del *stream* para absorber retransmisiones y picos de tráfico.

- ***Round-trip time* (RTT), ventana de retransmisión y multiplicador**
  Como expusimos previamente en la arquitectura del protocolo, SRT incorpora el mecanismo ARQ (*Automatic Repeat reQuest*) para la recuperación de paquetes perdidos. En este contexto el parámetro clave es la ventana de retransmisión, que determina cuánto tiempo el emisor retiene los paquetes en buffer para permitir su retransmisión si el receptor notifica pérdidas.

  La ventana de retransmisión debe ser suficiente para cubrir el tiempo de ida y vuelta (RTT) de la red multiplicado por un margen de seguridad. La recomendación estándar es configurar la latencia de SRT (y, por ende, de la ventana de retransmisión) en aproximadamente 4 veces el RTT, medido entre emisor y receptor. A modo de ejemplo, si el RTT es de 50 ms, la ventana de retransmisión recomendada a configurar sería de 200 ms.

  La documentación oficial de SRT sugiere que, en redes con mayor pérdida de paquetes, puede ser necesario aumentar el multiplicador de RTT, para asegurar la recuperación de los paquetes. Por ejemplo, con tasas de pérdida superior al 3%, se recomienda un multiplicador de 6 o más.

Tasa perdida de paquetes (%)	**<= 1**	**<= 3**	**<= 7**	**<= 10**	**<= 12**	**<= 21**	**<= 25**	**<= 27**	**<= 30**	**<= 40**
Multiplicador RTT Mínimo	3	4	6	8	8	10	13	14	14	30
BW Overhead	1	4	9	15	20	38	46	50	61	97
Latencia mínima (para RTT <= 20 ms)	60	80	120	160	160	200	260	280	280	600

Tabla I: Recomendación de Haivision de multiplicadores de RTT recomendados (valores conservadores) en función de la tasa de pérdida de paquetes

Cabe resaltar que un multiplicador mayor incrementa la latencia total, pero mejora la robustez ante perdidas, motivo por el cual es necesario balancear necesidades.
En el contexto analizado, cuando hablamos de redes no gestionadas como 5G pública, debemos considerar que el RTT puede fluctuar (*jitter*) y los paquetes de control e incluso las retransmisiones pueden perderse si el *buffer* configurado no está adecuadamente dimensionado.

- **Métodos de cifrado de encriptación**
  Por último, como explicamos en la sección 6.4.7, es necesario configurar el cifrado de encriptación extremo a extremo que se utilizará, considerando el soporte de SRT para cifrado con AES-128, 192 o 256. Aunque el cifrado es opcional para la operación, por lo motivos expuestos es fundamental considerarlo en aplicaciones de producción, más aún cuando se operan *streams* sobre redes abiertas.
  Para su utilización, se requieren dos parámetros a configurar: la "*passphrase*" compartida entre emisor y receptor, y la longitud de la clave a utilizar (16, 24 o 32 bytes para AES-128, 192 o 256 respectivamente).

6.4.9 Evaluación de costos indirectos asociados a la implementación de SRT

Como vimos en el apartado 6.4.4 ("SRT y código abierto"), aunque SRT es de código abierto y no tiene costo de licencia, su adopción implica una serie de costos indirectos que debemos considerar en el análisis de viabilidad.

- **Capacitación y formación**: El personal técnico debe formarse en la configuración, operación y resolución de problemas de SRT (incluyendo la gestión de claves, *buffers*, y modos de conexión), adicional al conocimiento de *networking* IP necesario para el trabajo con redes de datos. Esto implica tiempo y recursos en capacitación, y una curva de aprendizaje a considerar.

- **Integración y compatibilidad**: La integración de SRT con sistemas existentes/legados (codificadores/decodificadores, *switchers*, MAM, *playout*) puede requerir desarrollos de interfaces o software intermediario (*middleware*), pruebas de interoperabilidad y ajustes de configuración, lo que demanda recursos especializados de ingeniería.

- **Mantenimiento y soporte**: Es necesario destinar recursos para el monitoreo, actualización y soporte de la infraestructura SRT, incluyendo la gestión de incidentes y una eventual actualización de software/firmware, etc.

- **Infraestructura de red**: Aunque SRT optimiza el uso de redes (gestionadas y no gestionadas), puede ser necesario invertir en enlaces de mayor capacidad, equipamiento de borde más robusto, redundancia de conectividad, y hacer un manejo activo de la calidad de servicio (QoS) con los proveedores, para garantizar el nivel de servicio requerido.
  En el contexto de nuestro análisis, confirmamos que los proveedores de servicio 5G locales ofrecen servicios especializados para empresas (con costos adicionales), proveyendo una QoS determinada con fines específicos, mientras se opera en sus redes generales.

- **Gestión de seguridad y *compliance***: Como expusimos en la sección 6.4.7 (“Seguridad informática y SRT”), la administración de claves, la configuración de equipos de borde (*firewalls, VPNs*, etc.) y la realización de auditorías de seguridad pueden requerir recursos adicionales, con perfiles especializados, particularmente en entornos regulados.

- **Cambio de procesos y documentación**: La adopción de SRT puede requerir la actualización de manuales, procedimientos y documentación interna, así como la adaptación de flujos de trabajo existentes.

- **Soporte**: Si se utilizan soluciones de código abierto, puede ser necesario destinar recursos internos para soporte o considerar contratos de soporte comercial para garantizar tiempos de respuesta adecuados.

6.5 Teléfonos móviles celulares con soporte de redes 5G

La telefonía celular ha visto en los últimos 20 años, con la introducción de los llamados "*smartphones*" ('teléfonos inteligentes'), y hasta el día de hoy, una permanente evolución que integra en cada nueva generación de dispositivos una variedad de tecnologías a disposición de los usuarios.

Cada elemento componente de los teléfonos celulares incorpora las últimas novedades, que incluyen pantallas de mayor resolución y velocidad de refresco, calidad y colores; procesadores de mayor velocidad, eficiencia, consumo y prestaciones; baterías de mayor duración y carga más rápida; mejoras en conectividad física (ej. USB-C) e inalámbrica (ej. Wi-Fi 6/7, LTE/5G, antenas de mayor directividad y eficiencia, soporte de nuevos protocolos de transmisión).

Las cámaras de fotografía y video que incorporan merecen un apartado especial ya que son de creciente resolución, prestaciones y con conjuntos ópticos que incluso rivalizan hoy en día a cámaras de uso profesional (ej. capacidad de grabar video en resolución 4K, en 120 fps o cuadros por segundo, *zoom* óptico, estabilización mecánica de lente, lentes gran angular o telefoto, etc.).

La capacidad de procesamiento con que cuentan los dispositivos actuales, años atrás podía requerir de habitaciones enteras de equipamiento dedicado, y hoy está a nuestro alcance y uso en forma inmediata y portátil. Adicionalmente cuentan con sistemas operativos que permiten mucha flexibilidad en su programación e incorporación de software dedicado, APIs, protocolos y control.

Al integrar en un mismo dispositivo altas capacidades de captura de imagen y excelentes prestaciones de procesamiento de datos (hardware y software trabajando en conjunto), se logra unificar en un único dispositivo las prestaciones que en escenarios tradicionales legados eran cubiertas por múltiples equipos (ej. cámara, codificador de señal, encriptación, transferencia de señal, etc.).

Con relación a la conectividad, los dispositivos actuales cuentan con una variedad de módems (modulador/demodulador) y antenas especiales para soportar un abanico de tipos de redes inalámbricas masificadas, como bluetooth, redes wifi, redes móviles EDGE, 3G, LTE y 5G. La llegada del soporte de redes 5G abre las puertas al uso y transmisión de datos que permita *streams* de video múltiple, en zonas geográficas con mayor densidad de personas (y dispositivos móviles) operando en simultaneo.

Por otra parte, si consideramos las limitaciones emergentes, en particular para trabajo en campo con el rol que buscamos en este trabajo, consideramos que dos puntos claves a resolver son la alimentación eléctrica y la captura de sonido.

La batería interna de los celulares está diseñada para uso regular diario, donde no se pretende estar con filmación continua a altas resoluciones o tasas de captura, considerando que el teléfono celular también realiza el procesamiento de la señal y en el esquema que planteamos también la conversión, encriptación y transmisión de datos. Por estos factores, es necesario que el dispositivo cuente con alimentación permanente externa, con alguna de las múltiples opciones de alimentación externa disponibles en el mercado y de fácil acceso por utilizar conectividad masificada como es USB-C por ejemplo.

La otra limitante que consideramos es la captura de audio, ya que los micrófonos que incorporan los celulares, aunque muy efectivos y que también cuentan con procesamiento para mejorar la calidad de sonido, están orientados al uso personal o en campo cercano al dispositivo. Para lograr captar audio ambiente de un evento como el que planteamos en este trabajo, de índole deportiva, sería necesario incorporar un micrófono externo. Esto puede traer desafíos dependiendo de la tecnología a emplear, ya que si realizamos todo el procesamiento del *stream* (audio + video) en el celular, se deben aprovisionar conectividad para el micrófono y para la alimentación externa mencionada anteriormente.

Por todos estos factores y capacidades, así como también por la consideración de las limitantes que presentan, nos llevan a considerar a los teléfonos celulares modernos como una plataforma ideal para las pruebas móviles propuestas de transmisión que planteamos.

6.6 Codificación H.265/HEVC

En la optimización del transporte de señales de video de alta calidad en forma digital es necesario evaluar el caudal de información que transmitiremos, y consecuentemente el requerimiento y disponibilidad de ancho de banda. Cuando trabajamos con enlaces dedicados, existe una cierta garantía sobre los parámetros de conectividad (ancho de banda, latencia, etc.) que en redes no gestionadas como 5G presentan incertezas y desafíos. Los codec de video (ya sea por software o hardware) comprimen y descomprimen las señales de video digital, buscando un balance entre calidad de imagen y capacidad de reducir la cantidad de información a transmitir.

El estándar *High Efficiency Video Coding* (HEVC), también conocido como H.265 o MPEG-H Parte 2, representa una de las evoluciones mas significativas de compresión de video digital desde la aparición del H.264 (en 2003). Publicado en 2013 por el consorcio ISO/IEC MPEG y ITU-T VCEG, H.265 fue diseñado para responder a la creciente demanda de video de alta resolución (FHD, 4K, 8K) y a la necesidad de reducir el consumo de ancho de banda en aplicaciones de transmisión, almacenamiento y contribución profesional.

Actualmente su extensa adopción y compatibilidad lo hacen un candidato ideal para el esquema planteado en el proyecto.

6.6.1 Aspectos técnicos

El codec H.265/HEVC utiliza una arquitectura híbrida basada en bloques, similar en principio a H.264, pero con avances que mejoran la eficiencia y la calidad visual:

- **Unidades de árbol de codificación (CTU)**: El bloque básico de procesamiento en HEVC es la CTU, que puede ser de hasta 64x64 píxeles (frente a los 16x16 de H.264). Estas CTU pueden subdividirse en forma recursiva en bloques mas pequeños, permitiendo una adaptación flexible al contenido de la imagen.
- **Predicción Intra/Inter**: HEVC ofrece 35 modelos de predicción intra (contra 9 de H.264) lo que permite modelar con mayor precisión bordes y texturas. La predicción inter utiliza estimación y compensación de movimiento mejoradas, optimizando la reutilización de información de cuadros previos.

- **Transformación y Cuantización**: Soporta tamaños de transformada de 4x4 hasta 32x32 PUs (unidades de predicción), empleando DCT y DST (transformadas discretas de Coseno y Seno). La cuantización reduce aún mas el tamaño de los datos.
- **Codificación de Entropía**: Utiliza una tecnología (denominada CABAC) que incrementa la eficiencia y permite procesamiento paralelo a través de codificación adaptativa al contexto.
- **Filtros de mejora de imagen**: Incluye filtros de reducción de bloques en la imagen (*deblocking*) y de errores de observación (*artifacts*).
- **Procesamiento paralelo**: Incorpora mecanismos y funcionalidades (*slices, tiles*, y *WPP, wavefront parallel processing*) que permiten aprovechar microprocesadores multinúcleo, reduciendo la latencia. Este punto es fundamental, ya que H.265 tiene una carga computacional mayor a su antecesor.
  El procesamiento paralelo junto a funciones avanzadas de predicción de movimiento permiten que la codificación/decodificación sean más rápidas, brindando baja latencia para producciones remotas como las que estamos buscando.
- **Perfiles, niveles y *tiers***: H.265 incorpora una serie de perfiles de compresión con distintos usos y objetivos. Los perfiles tienen como objetivo definir un conjunto de herramientas de codificación que se pueden utilizar para crear un flujo de bits que se adecúe a ese perfil. Entre los parámetros que tipifica se encuentran por ejemplo: tasa de bits (*bitrate*), tamaño de cuadro, requerimientos de procesamiento, etc. Existen perfiles para uso en consumo general, como también perfiles para aplicaciones profesionales.

Efectivamente, el codec H.265 logra entre un 35 y un 50% de mejora de eficiencia de compresión respecto a H.264 para la misma calidad visual, especialmente en altas resoluciones (4K, 8K). Esto se traduce directamente en la posibilidad de transmitir video de calidad profesional a la mitad del *bitrate* requerido por su antecesor, o bien mejorar la calidad visual manteniendo el mismo ancho de banda.

Si evaluamos el nivel de compatibilidad que brinda el codec, H.265 es ampliamente soportado por hardware y software profesional disponibles en el mercado, así como también

provee flexibilidad y compatibilidad al operar con los protocolos de transporte IP como SRT, RIST y Zixi.

6.6.2 Requerimientos de *bitrate* por resolución

La siguiente tabla muestra en forma resumida distintos requerimientos de *bitrate* para realizar transmisiones de video de alta calidad sobre enlaces IP, incluyendo la comparativa entre H.264 y H.265. Los valores ponen en evidencia la significativa mejora que trae los algoritmos incorporados en H.265 y su idoneidad para uso en el escenario que proponemos.

Resolución	H.264 *bitrate* (Mbps)	H.265 *bitrate* (Mbps)	Ahorro de ancho de banda (%)	Calidad comparativa
720p	8	4	50	Equivalente
FHD (1080p)	16	8	50	Equivalente o superior
4K (2160p)	32	15	53	Equivalente o superior

Tabla II: Comparativa de requerimientos de *bitrate* por codec y por resolución

6.6.3 Integración de H.265 y redes 5G

La alta eficiencia del codec H.265 es ideal para su utilización con protocolos de trasporte como SRT, y también para conjugarlo con el uso sobre redes 5G donde el ancho de banda puede ser variable. Tanto SRT como 5G ofrecen baja latencia en sus operaciones, mientras que H.265 reduce la cantidad de datos a transmitir, habilitando la posibilidad de realizar producción remota televisiva. Existen casos documentados de uso de coberturas de eventos en vivo, permitiendo a la emisora operar con equipos mas pequeños, flexibles, sin necesidad de enlaces satelitales o enlaces terrestres dedicados.

En contraparte, H.265 es mas sensible a errores de transmisión debido a su alta compresión. Para mitigar esto, se utilizan protocolos como SRT, técnicas de corrección de errores y codificación adaptativa.

La mayor complejidad del algoritmo de H.265 requiere a su vez una mayor carga computacional, por lo cual es necesario hardware especializado para la aceleración del

procesamiento y, por ende, para mantener una baja latencia en aplicaciones en vivo. Existen codificadores de H.265 tanto en hardware como en software, brindando una amplia oferta y escalabilidad de soluciones que pueden desarrollarse incorporando esta tecnología.

6.7 Soluciones de *switching* de video basadas en la nube

El surgimiento y adopción de *switchers* de video basados en la nube representa una de las transformaciones mas significativas y disruptivas en la producción audiovisual profesional, permitiendo realizar la conmutación, mezcla y procesamiento de señales de video en tiempo real, pero a diferencia de los *switchers* tradicionales basados en hardware, toda la lógica y el procesamiento se ejecutan en infraestructura virtualizada y distribuida en la nube. Esta evolución, enmarcada dentro de las soluciones actuales "*software defined*", responde a la necesidad de mayor flexibilidad, escalabilidad y eficiencia operativa para entornos de producción remota y distribuida, especialmente para el contexto de flujos de trabajo sobre redes IP.

6.7.1 Beneficios

Entre las principales ventajas de las soluciones de conmutación de video basadas en la nube, podemos destacar las siguientes:

- **Flexibilidad y escalabilidad**: La virtualización permite escalar recursos de manera instantánea, adaptando la infraestructura a las dimensiones de la producción sin necesidad de hardware dedicado, volviéndola ideal para coberturas de eventos variables o múltiples producciones en simultaneo.

- **Operación remota y colaborativa**: Los operadores pueden acceder y controlar el *switcher* desde cualquier lugar físico que cuenta con la conectividad suficiente, facilitando la colaboración de equipos distribuidos, ya sea local, regional o globalmente. Esto reduce costos de traslado, y permite aprovechar talento especializado sin restricciones geográficas.

- **Integración de flujos de trabajo IP**: Los *switcher* en la nube se integran nativamente a los flujos IP, permitiendo la contribución y distribución de señales a través de redes estándar, eliminando la dependencia de infraestructura especializada, como el SDI tradicional. Protocolos como SMPTE ST 2110 y NDI aseguran la

interoperabilidad y baja latencia.

- **Reducción de costos y tiempo de implementación**: Al eliminar la necesidad de hardware físico y permitir modelos de pago por uso/contratación, los costos de capital y operación (*CapEx/OpEx*) se reducen drásticamente. Además, la puesta en marcha de nuevos canales o producciones puede realizarse en días en lugar de meses.

- **Resiliencia y redundancia**: La infraestructura en la nube permite lo que se denomina "*failover*" (el conmutado o 'salto' automático y sin intervención a infraestructura redundante en caso de una falla) de forma automática, así como también provee balanceo de carga y recuperación ante fallas, garantizando una alta disponibilidad, incluso ante problemas de red o caídas de servidores. En el contexto de las producciones televisivas, es crítico contar con mecanismos de redundancia ya que no es aceptable la interrupción de una transmisión en el caso de un evento en vivo.

6.7.2 Desafíos técnicos y soluciones

Las soluciones de *switchers* de video en la nube presentan desafíos particulares que deben considerarse para una producción equivalente a la tradicional en calidad y prestaciones.

El primer factor es la variabilidad de la red de datos, donde la latencia y el *throughput* pueden fluctuar, más aún cuando nos referimos a redes no gestionadas como 5G. Para mitigar el impacto, se emplean técnicas de adaptación del *bitrate*, buffers dinámicos y el monitoreo en tiempo real con métricas como RSRP, RSRQ y SNR/SINR.

Por otro lado, ante redes donde pueden producirse errores de transmisión, es fundamental la integración y soporte de protocolos de transporte resilientes como SRT o RIST, que proveen corrección de errores, recuperación de paquetes perdidos y cifrado criptográfico nativo.

La seguridad y protección de las señales transmitidas es un factor clave, sobre todo al trabajar en redes públicas, por lo cual se debe implementar cifrado de extremo a extremo, y una autenticación robusta.

Por último, y como enumeramos en la sección anterior, la redundancia es un requerimiento fundamental para evitar interrupciones en la producción, por lo cual es frecuente el uso de múltiples enlaces de datos (5G), *fallback* a redes alternativas (LTE o Wifi), y el uso de "*edge computing*" para procesamiento local y reducción de latencia.

6.7.3 Arquitectura

Entre las principales ventajas de las soluciones de conmutación de video basadas en la nube, podemos destacar las siguientes:

- Una interfaz de usuario (*front end*) generalmente accesible vía web o aplicación dedicada, que permite a los operadores gestionar distintas fuentes de video, transiciones, efectos y salidas desde cualquier ubicación con acceso a internet.

- La capa de aplicación provee interfaces programables (APIs, por su sigla en inglés) y servicios de contribución, procesamiento y distribución de señales, integrando herramientas de terceros como gráficos, mezcla de audio y grabación.

- La infraestructura está virtualizada, por lo cual los recursos de procesamiento, almacenamiento y red son virtuales y asignados dinámicamente según la demanda requerida, aprovechando la elasticidad de las soluciones en la nube.

- Motores de procesamiento multimedia dedicados realizan tareas de codificación/decodificación, mezcla y conmutación en tiempo real.

- El almacenamiento en la nube permite la grabación, archivado y reproducción a demanda de los contenidos.

Principios y especificaciones técnicas:

- **Virtualización y *pooling* de recursos**: Permite ejecutar múltiples instancias de *switchers* sobre la misma infraestructura, optimizando costos y la escalabilidad.

- **Arquitectura orientada a servicios (SOA)**: Componentes modulares que se comunican por interfaces programables (APIs) facilitan la integración con otros servicios y herramientas de *broadcast*.
- **Escalabilidad elástica**: Capacidad de aumentar o reducir recursos según la carga de trabajo (ej. cantidad de señales, complejidad de la producción, etc.).
- ***Multi-tenancy***: Soporte para múltiples usuarios o producciones simultaneas, con separación lógica de recursos y datos.
- **Diseño centrado en red IP**: Todo el transporte de video y control se realiza sobre redes IP utilizando protocolos de baja latencia como SRT, NDI, RTMP, WebRTC, etc.
- **Salidas de video**: Permite "*multiview*", RTMP, NDI, grabación ISO, streaming a múltiples destinos, brindando gran flexibilidad e interconexión de flujos de trabajo.
- Audio con mezcla multicanal, *audio-follow-video* e integración con fuentes externas.
- Puede brindan soporte para servicios adicionales como intercom virtualizado.
- **Control**: interfaz web, APIs para automatización, integración con superficies físicas de control opcionales.
- **Seguridad y control de acceso**: Implementación de cifrado criptográfico, autenticación y control de roles para proteger tanto el contenido como las operaciones.
- Redundancia: *failover* multi-región, *backup* de *streams*, auto-reconexión.

6.7.4 Modelos de Costos y Proveedores de Referencia

La transición a soluciones de producción en la nube se fundamenta en un cambio del modelo económico de CapEx a OpEx. En lugar de adquirir hardware costoso, las productoras pagan por el uso de software y recursos de computación (almacenamiento, procesamiento, etc.), en modalidades IaaS (*Infrastructure-as-a-Service*), PaaS (*Platform-as-a-Service*) y SaaS (*Software-as-a-Service*), cada uno con modelos de licenciamiento diferentes. A continuación, se detallan los costos de dos de las soluciones más relevantes del mercado:

- **"vMix"**: Esta solución basada en software para PC ofrece un modelo híbrido. Permite una compra de licencia perpetua (CapEx) o una suscripción mensual (OpEx). vMix es reconocido por su robustez y por incluir una gran cantidad de funcionalidades, incluyendo *switching*, gráficos, grabación y repetición instantánea.

    - **Licencia Perpetua (CapEx)**: La licencia vMix PRO, que incluye hasta 1000 entradas, resolución 4K, 8 canales de repetición instantánea (replay) y 4 salidas SRT, tiene un costo de U$S 1.200 por un pago único.

    - **Suscripción (OpEx)**: vMix ofrece la opción vMix MAX, que brinda todas las funcionalidades de la versión PRO por una suscripción mensual de U$S 50.

- **"LiveU Studio"**: A diferencia de vMix, LiveU Studio es una plataforma 100% nativa de la nube (*cloud-native*), operada enteramente desde un navegador web. Su modelo de negocio es puramente OPEX, basado en el uso y la escala de la producción.

    - **Modelo Basado en Uso (OpEx)**: LiveU Studio no publica una lista de precios fija, ya que su costo se adapta a las necesidades del evento (cantidad de entradas, horas de uso, destinos de *streaming*, etc.). Ofrecen un modelo PAYG (*Pay-as-you-go*), que permite a las productoras pagar únicamente por los eventos que producen.

    - **Funcionalidades**: La plataforma incluye herramientas de nivel profesional como *switching* con precisión de cuadro, repetición instantánea multiángulo, grabación aislada de cámaras (ISO) y mezcla de audio, todo gestionado desde la nube.

- Otros *players* del mercado incluyen: TVU Networks, Vizrt, AWS Elemental MediaLive, Microsoft Azure Media Services, Google Cloud Media Solutions, Haivision Hub Media Platform, etc.

7. Metodología y Desarrollo

La presente sección establece el marco metodológico para el análisis de costos y la evaluación de las arquitecturas de producción televisiva que son el foco de este trabajo. Se deconstruye el modelo económico tradicionalmente empleado, intensivo en capital, para luego proponer, cuantificar y validar empíricamente alternativas basadas en tecnologías IP. El objetivo es demostrar cómo estas nuevas tecnologías modifican radicalmente la estructura de costos y la viabilidad operativa de las producciones en exteriores, conectando directamente con la problemática y los objetivos planteados en las secciones anteriores de este documento.

7.1 Marco teórico y de costeo

7.1.1 El modelo económico de Producción: de CapEx a OpEx

El análisis se fundamenta en un modelo de costeo detallado que considera la inversión de capital (CapEx) y los gastos operativos (OpEx). Los valores monetarios presentados son estimaciones de mercado basadas en precios de listas de fabricantes, cotizaciones de integradores de soluciones para la industria, y datos públicos, actualizados a la fecha de este análisis. La variabilidad inherente a estos costos se debe a factores como negociaciones y descuentos comerciales, paquetes de hardware, y la elección entre modelos de distinta gama dentro de cada marca especializada.

El modelo de negocio de la radiodifusión se ha sostenido históricamente sobre una producción intensiva en CapEx. Esta realidad se materializa en la OB-van (unidad móvil de televisión) que como vimos en la sección anterior de este documento, se trata de un vehículo que posee la totalidad del equipamiento necesario para una producción televisiva en vivo y/o remota. La principal debilidad financiera de esta arquitectura radica en su bajo índice de utilización de activos. En forma genérica, este índice es una medida de la eficiencia con la que una empresa hace uso de sus activos para generar ingresos. Una sola OB-van, cuyo valor puede ascender a varios millones de dólares, puede permanecer inactiva la mayor parte del año, generando un costo de oportunidad considerable y un retorno de inversión (ROI) sumamente lento. Esta inversión se deprecia rápidamente debido a la permanente evolución

tecnológica e incurre en costos constantes de mantenimiento, logísticos y de almacenamiento, incluso cuando no está en uso.

La respuesta de la industria ha sido la adopción de la producción remota (denominada en la industria como REMI, por las siglas en inglés de *Remote integration model*), que traslada la inversión de CapEx a OpEx. En este modelo, el equipamiento de producción mas costoso se centraliza en un lugar específico denominado *hub*, mientras que en el lugar del evento se mantienen solo las cámaras y los codificadores. Esta arquitectura optimiza el uso de activos, ya que los mezcladores y servidores del *hub* pueden ser utilizados para múltiples producciones, lo que incrementa su ROI y reduce los costos de despliegue y de personal involucrado en la producción. El cambio a un modelo de OpEx se vuelve aún mas notorio con las soluciones basadas en la nube, donde la inversión en hardware de procesamiento y conectividad es prácticamente nula, reemplazándose por un gasto recurrente por servicio o evento.

7.1.2 Fundamentos de producción televisiva en campo

Para estructurar el análisis de costos en forma relevante, es fundamental considerar la clasificación adoptada en forma estándar por la industria. A partir de la obra de Herbert Zettl "Television Production Handbook", cuya primera edición data de 1961, utilizaremos la definición de una distinción clave sobre las modalidades de producción en exteriores que informará nuestro análisis.

- **ENG (*Electronic News Gathering* – Captura electrónica de noticias)**: Se define por la inmediatez y la agilidad. Su propósito es responder a eventos imprevistos, donde la rapidez de despliegue y transmisión es prioritaria sobre la complejidad de la producción.
- **EFP (*Electronic Field Production* – Producción electrónica en campo)**: Se caracteriza por luna planificación detallada y una calidad de producción comparable a la de un estudio. Zettl la describe como una metodología que toma la movilidad en la ENG y le añade el cuidado y control de calidad del estudio.

- Este trabajo ampliará la categoría EFP para incluir "**Grandes Eventos**", que representan la máxima escala en complejidad y equipamiento de producciones en campo.

Estos tres escenarios (ENG, EFP y Grande eventos) servirán como base para el análisis comparativo de costos que se presenta en la sección de Resultados (capítulo 8).

7.1.3 Nuevos modelos de producción remota

Con el surgimiento y progresiva adopción de tecnologías digitales habilitantes, como las expuestas en el capítulo 6, incluyendo códecs de alta capacidad, protocolos de transporte eficientes, redes móviles de alta velocidad y creciente calidad, se dan las condiciones necesarias para una transformación profunda en los modelos de producción audiovisual que permiten la transición de esquemas tradicionales, centralizados y dependientes de infraestructura costosa, hacia flujos de trabajo distribuidos, remotos y virtualizados. Este nuevo paradigma, conocido en la industria como REMI (*Remote Integration Model*) o "producción *at-home*" redefine la manera en que se captan, procesan y distribuyen los contenidos, especialmente en eventos en vivo en el ámbito de broadcast y eventos de gran escala.

En el modelo tradicional, la producción de eventos en vivo requiere el despliegue de unidades móviles (OB-vans), grandes equipos técnicos y una logística compleja para trasladar tanto personal como equipamiento especializado al lugar del evento. Todo el procesamiento, incluida la mezcla de señales, la inserción de gráficos, la gestión de cámaras y la dirección, se realiza in situ, lo que implica altos costos operativos, limitaciones de escalabilidad y un considerable esfuerzo logístico. La llegada de las tecnologías digitales mencionadas, incluido el avance de la Convergencia IP, las redes móviles de alta velocidad y baja latencia, los códecs eficientes para altas resoluciones como H.265, los protocolos modernos como SRT, y la gran capacidad de cómputo portátil (fundamental para el procesamiento de imagen, sonido y video) han permitido que solo el equipamiento esencial (ej. cámaras, codificadores, mochilas y personal técnico mínimo) permanezcan en el sitio del evento, mientras que la mayor parte de la producción se centraliza en un *hub*, un centro de operaciones/control o incluso en la nube.

De manera pragmática, y como breve anécdota, al comenzar este trabajo en 2023, uno de los casos de éxito que evidenciaba este modelo incipiente fue durante la pandemia de COVID-19, cuando la industria se vio forzada a buscar alternativas para mantener la continuidad de las transmisiones en un contexto de restricciones sanitarias y distanciamiento social. En EE. UU., durante la producción televisiva de la edición 2020 del "*draft*" de la liga de futbol americano NFL, ante la imposibilidad de reunir a jugadores, entrenadores y equipos de producción en un solo lugar, la liga optó por un enfoque completamente remoto: se distribuyeron kits de producción a los jugadores prospecto, quienes transmitieron desde sus hogares utilizando *smartphones* y conexiones de banda ancha. La contribución de todos estos *feeds* se realizó hacia una solución ad-hoc armada en la nube, transportando señales con el protocolo SRT como tecnología subyacente habilitante. En simultaneo, había *feeds* adicionales de presentadores y de los propios equipos de la NFL. La agregación de la contribución, el *switching* y la realización de la señal final se llevó a cabo en centros de operación remotos, y mediante plataformas en la nube (Amazon Web Services). Adicionalmente, y sin limitarse a *feeds* de video, incorporaron participación interactiva del público y de los responsables de la liga. La transmisión resultó de alta calidad, con récords de audiencia y una experiencia inédita, lograda sin la presencia física de los equipos de producción en un solo lugar.

Este modelo, que sirvió de inspiración para este trabajo, no solo demostró la viabilidad técnica y operativa, sino que evidenció ventajas económicas y estratégicas significativas al reducir el equipamiento en campo, los costos de transporte, alojamiento y logística. Al mismo tiempo se incrementa la flexibilidad para cubrir múltiples eventos en paralelo desde un mismo centro de producción. Esta centralización tiene el beneficio agregado que se puede hacer uso de los mejores talentos técnicos y creativos (que no necesariamente tienen que estar físicamente presentes en el centro de producción), optimizando recursos y mejorando la calidad del producto final.

La adopción de SRT sobre 5G en particular fue replicada en otros contextos, como en la cobertura de eventos deportivos por parte de NBCUniversal, y por el Grupo RBS (Brasil) que han consolidado operaciones de *playout* y *master control* en *hubs* centralizados, conectando sedes remotas a través de SRT sobre enlaces IP, incluso con uso de redes públicas y privadas 5G. Esto, manifiestan, ha reducido la dependencia en enlaces satelitales o fibra

dedicada, garantizando la resiliencia y calidad de transmisión incluso en entornos de conectividad variable.

En resumen, los nuevos modelos de producción que habilitan tecnologías como SRT, la actual (y futura) oferta de redes móviles públicas de alta velocidad y las tecnologías digitales subyacentes representan un cambio de paradigma para la industria audiovisual: la producción se descentraliza, se virtualiza y se vuelve mas ágil, escalable y eficiente. Los casos de éxito en eventos de alto perfil y las encuestas anuales realizadas a la industria por Haivision reflejan en sus resultados que esta transformación no solo es posible, sino que ya está redefiniendo los estándares de producción en vivo a nivel global. Por este motivo, consideramos que es de vital importancia para el planteo de reingeniería local una concurrente adopción de estas tecnologías como factor clave para asegurar su evolución y competitividad.

7.2 Metodología de prueba en campo: Producto mínimo viable (MVP)

Para validar empíricamente la viabilidad de la tecnología central de este proyecto de reingeniería, se define un protocolo de prueba en campo. Este MVP, refiriéndonos a una versión mínima del flujo de trabajo y componentes que permitan la validación de la viabilidad de la solución propuesta, se enfoca en el escenario más básico y ágil: una transmisión ENG utilizando un teléfono celular inteligente (*smartphone*). El objetivo es demostrar que, incluso con la configuración mas simple, la combinación de 5G y SRT puede ofrecer un enlace de contribución robusto y de calidad profesional.

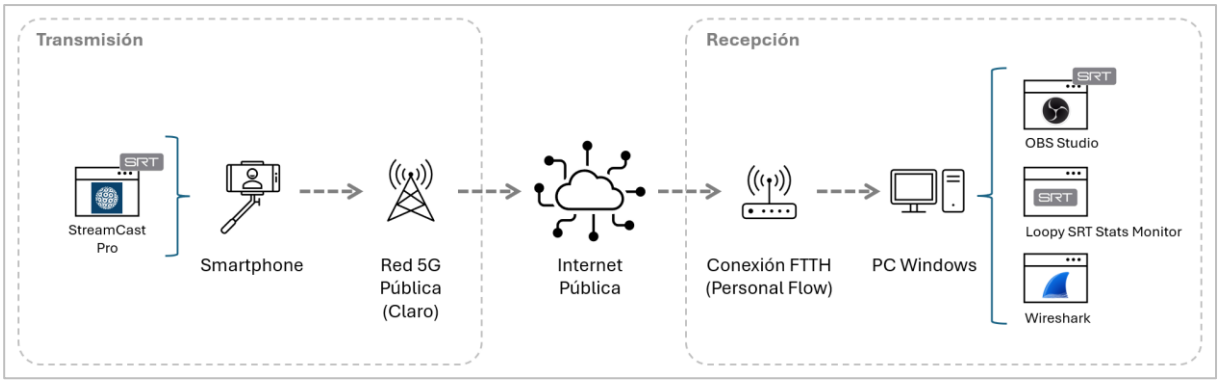


Figura 7.1: Esquema de prueba en campo tipo MVP

7.2.1 Arquitectura y configuración

Se considerarán los siguientes elementos para estructurar la prueba de campo:

- **Conectividad**:
  - Transmisor: Red Claro Argentina LTE 5G.
  - Receptor: Conexión a internet para la PC por medio de Personal Flow, fibra óptica (FTTH), 600 Mbps simétricos. Se configura el modem para realizar *port-forwarding* hacia la PC destino dentro de la LAN.
    IP publica: 181.86.62.183, puerto destino seleccionado: 49994.

- **Transmisor:**
    - Se utilizará un *smartphone* Apple iPhone 15 Pro con iOS versión 26.0.1 (Modelo MTQW3LL/A, con soporte y servicio de red 5G).
    - Se utilizarán las aplicaciones:
        - "Field Test" (parte del sistema operativo iOS).
        - Aplicación gratuita "Network Ping Lite". Disponible en: https://apps.apple.com/app/network-ping-lite/id289967115
        - Aplicación gratuita "StreamCast Pro". Disponible en: https://apps.apple.com/app/streamcast-pro/id6443880300
    - Configuración de la conexión de transmisión en "StreamCast Pro" (figura 7.2):
        - Configuración SRT:
            - URL: `srt://181.86.62.183:49994`
              La IP publica y puerto corresponden a la dirección destino del receptor de la transmisión SRT (PC).
            - Modo: Caller
            - Latencia: 2000 ms.
              Se aplicará la regla empírica recomendada por la industria de establecer la latencia 4 veces el *Round Trip Time* (RTT) de la red (tiempo de ida y vuelta), medido previamente.
        - Configuración de video:
            - Resolución de video: 1920 x 1080 (Full HD)
            - *Frame rate*: 30 cuadros por segundo
            - *Bitrate*: 1500 kbps
            - Codificación: HEVC
            - Perfil HEVC: Principal (*Main*)
            - Se activa codificación por hardware (GPU)

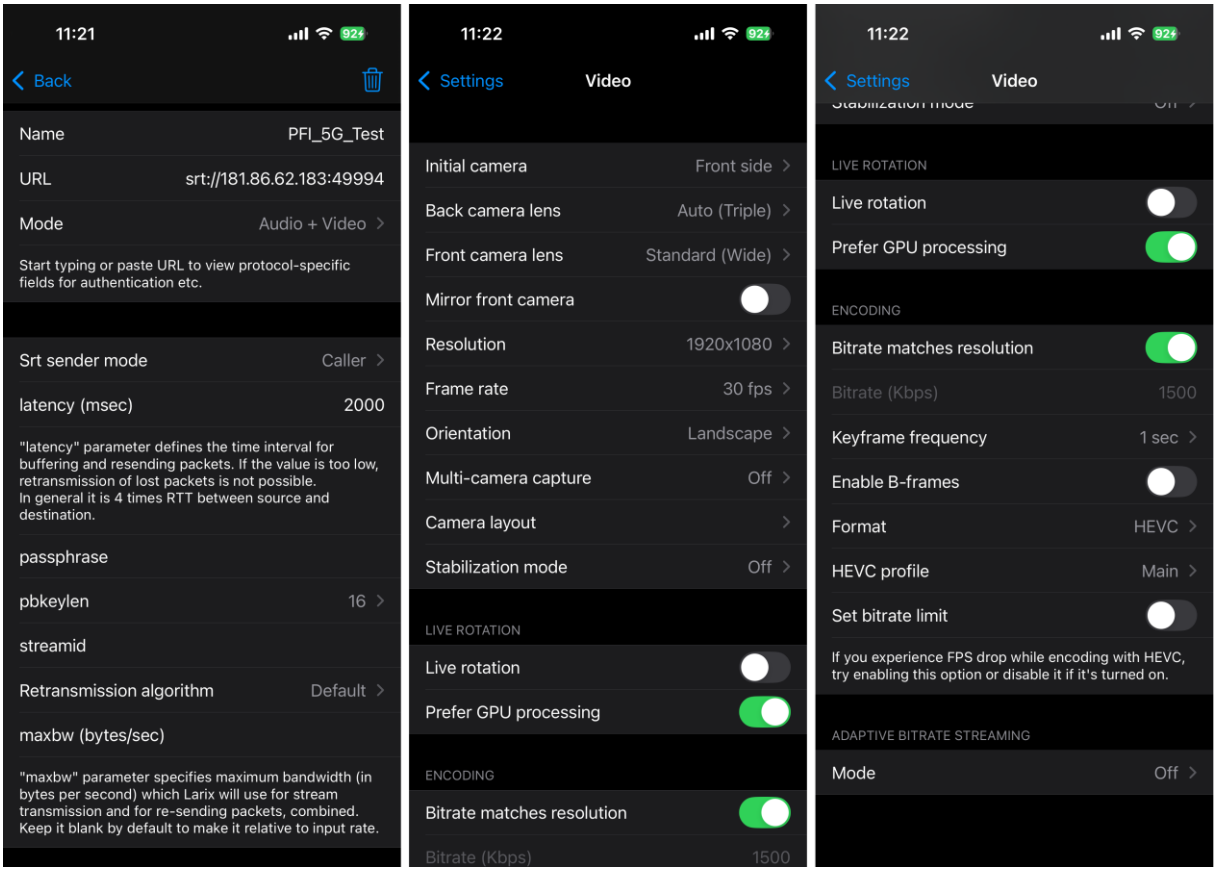


Figura 7.2: Configuración de transmisión SRT y video en “StreamCast Pro”.

- **Receptor**:
  - Se utilizará una computadora (PC) con:
    - Sistema operativo “Microsoft Windows 11 Pro” versión 10.0.26100.6584
    - Aplicación gratuita “OBS Studio” versión 32.0.1
      Disponible en: https://obsproject.com/
  - Configuración de la conexión de recepción en “OBS Studio” (figura 7.3):
    - Se incorpora una fuente tipo “Media source” con la siguiente entrada (*input*):
      `srt://0.0.0.0:49994?mode=listener&latency=2000000&timeout=5000000`
    - Se activa decodificación por hardware (GPU).

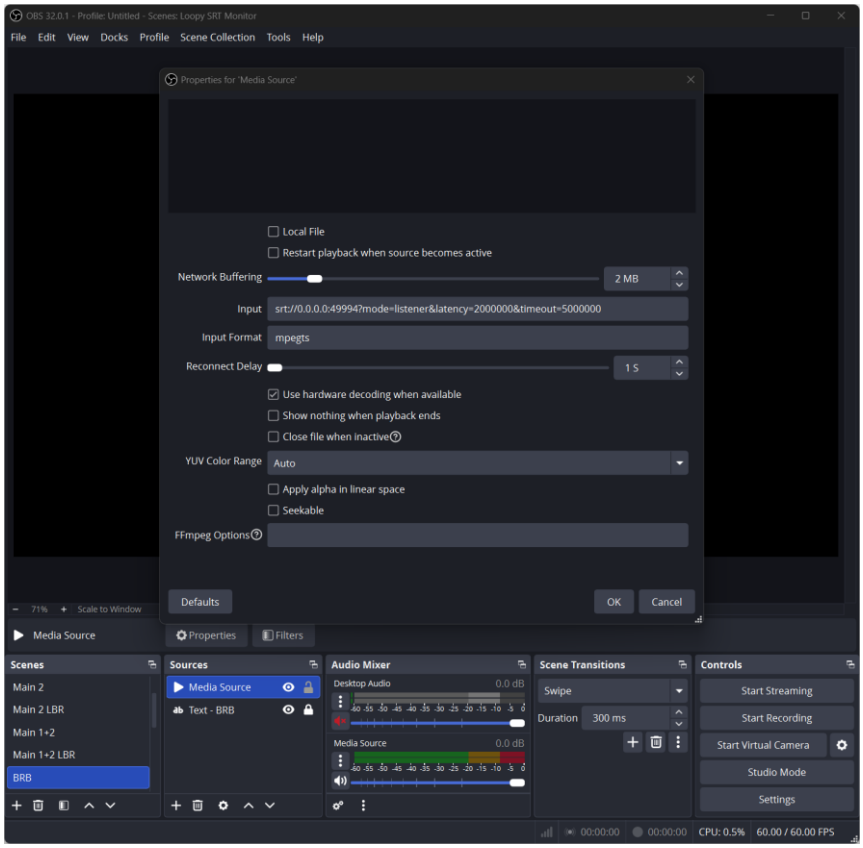


Figura 7.3: Configuración de recepción en “OBS Studio”.

- **Monitoreo**:
  - Aplicación gratuita “Loopy SRT Stats Monitor” versión 1.1.3
    Disponible en: https://github.com/loopy750/SRT-Stats-Monitor
  - Aplicación gratuita “Wireshark” versión 4.4.9
    Disponible en: https://www.wireshark.org/
  - Python (gratuito) versión 3.12.9, con las bibliotecas pandas (v2.3.3), matplotlib (v3.10.6) y numpy (2.3.3) para procesar y graficar las métricas obtenidas a partir de la captura de paquetes de Wireshark.
    Disponible en: https://www.python.org/

  - Configuración específica para captura de trafico de red con “Wireshark”:
    Para maximizar la correcta detección del tráfico SRT, se recomienda:
    - Deshabilitar el protocolo "UDT" (protocolo en el que SRT está originalmente basado) para evitar posibles fallos de detección por

parte del capturador. Esto se realiza desde el menú "Analyze" > "Enabled Protocols" y se deseleccionan aquellos protocolos que no deseamos sean considerados.

- Adicionalmente, en la sección Avanzada de "Edit" > "Preferences", se recomienda habilitar (configurándolo como "TRUE") el parámetro llamado "udp.try_heurisic_first", debido al modo que Wireshark interpreta el vínculo entre paquetes UDP y la operatoria de SRT (vinculado a un término de “Wireshark” denominado "sub-dissectors").
- Se activa la captura de trafico de red sobre la interfaz de red conectada a internet.

7.2.2 Protocolo de ejecución

Se ejecutará la prueba de campo mediante el siguiente protocolo:

1) **Preproducción (relevamiento de sitio)**: En la ubicación de la prueba, se medirá la calidad instantánea de señal 5G que tiene el *smartphone*., con métricas de RSRP, RSRQ y SINR. Se medirá la latencia a la IP del receptor y se calculará el RTT.

2) **Configuración**: Se ajustarán el parámetro de latencia en "StreamCast Pro" según la medición del RTT.

3) **Transmisión y captura**: Se realizará una transmisión continua de 5 minutos de video, capturando en el receptor el flujo de paquetes SRT con "Wireshark" y métricas adicionales con "Loopy SRT Stats Monitor". Se registrará también la calidad de video subjetiva.

Posterior a la captura de paquetes de red SRT, realizaremos un análisis para extraer cuatro métricas principales que ayuden a apoyar la hipótesis:

- **Latencia**: Es el tiempo que tarda un paquete de datos en viajar desde el origen hasta el destino. En contextos de *streaming*, una baja latencia es crítica porque permite que el video se reproduzca de manera continua y sin interrupciones. La latencia se mide en segundos o milisegundos, y cualquier incremento puede resultar en retrasos en el video o en la comunicación.
- ***Jitter*** (fluctuación): Mide la variabilidad en la latencia de los paquetes en una conexión. Es decir, aunque los paquetes viajen rápido, pueden llegar en intervalos de tiempo muy diferentes, lo que puede causar problemas en la continuidad de la reproducción de video. Un *jitter* alto puede resultar en saltos o distorsiones en la visualización del video.

- **Paquetes perdidos**: Es la cantidad de paquetes de datos que no se reciben en el destino. Esto puede suceder por congestión de red, errores de transmisión o problemas en el hardware. La pérdida de paquetes puede ser crítica para el rendimiento del *streaming*, ya que puede afectar la calidad del video y causar interrupciones.
- ***Throughput*** (rendimiento): Es la cantidad efectiva de datos transmitidos en un periodo de tiempo específico. Se calcula como la cantidad de datos entrantes dividida por el tiempo que tardó en llegar. Un *throughput* alto indica una conexión eficiente, mientras que un *throughput* bajo puede resultar en *buffering* y caídas de calidad en el video.

Por último, con el objetivo de extraer, calcular y graficar dichas métricas a partir de la captura de paquetes de red realizada con “Wireshark”, aplicaremos a la bajada de datos en formato de separación con comas (.CSV) un *script* hecho en Python, que procese eficientemente las miles de entradas de datos capturadas. Se incorpora a continuación el código fuente Python del script desarrollado.

```
# ----------------------------
# Script para calcular y generar graficas de Latencia, Jitter, Perdida de paquetes y
#  Throughput a partir de una captura de paquetes de Wireshark, exportada a CSV bajo
#  el nombre de archivo "Captura_WS_PFI.csv".
#
# Nota: Los nombres de columna en ingles corresponden al archivo CSV de Wireshark.
#  Las variables en castellano son agregadas como calculo.
#
# Autores:        Miceli, Nicolas (LU 101526); Zunda Cornell, Santiago Andres (LU 115614)
# Version/Fecha:  v4 - 2025-09-12


import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import re

# Carga de archivo CSV
df = pd.read_csv('Captura_WS_PFI.csv', delimiter=',', decimal='.', dtype={'Time': str,
'Length': float})

# Convierte la columna Time (Tiempo (segundos)) a float
df['Time'] = df['Time'].str.replace(',', '.').astype(float)

# Ordena por tiempo
df = df.sort_values(by='Time').reset_index(drop=True)

# Calcula latencia (diferencia entre tiempos de paquetes consecutivos)
```

```
df['Latencia'] = df['Time'].diff()

# Calcula jitter (diferencia entre valores de latencia consecutivos)
df['Jitter'] = df['Latencia'].diff()

# Extrae numeros de secuencia de paquetes de la columna 'Info' si estan disponibles
df['NumSec'] = df['Info'].str.extract(r'seqno: (\d+)').astype(float)

# Calcular perdida de paquetes verificando huecos en los numeros de secuencia
df['PerdidaPaquetes'] = df['NumSec'].diff().apply(lambda x: x - 1 if pd.notnull(x) and x >
1 else 0).fillna(0)

# Calcula Throughput (bytes por segundo)
# Asegura que Lenght y Latencia no tengan ceros antes de aplicar la division
df['Throughput'] = df['Length'] / df['Latencia'].replace({0: np.nan})
df['Throughput'] = df['Throughput'].fillna(0)

# Grafica las metricas
fig, axs = plt.subplots(4, 1, figsize=(12, 16), sharex=True)

axs[0].plot(df['Time'], df['Latencia'], label='Latencia', color='blue')
axs[0].set_ylabel('Latencia (s)')
axs[0].set_title('Latencia')
axs[0].grid(True)
axs[0].legend()

axs[1].plot(df['Time'], df['Jitter'], label='Jitter', color='orange')
axs[1].set_ylabel('Jitter (s)')
axs[1].set_title('Jitter')
axs[1].grid(True)
axs[1].legend()

axs[2].plot(df['Time'], df['PerdidaPaquetes'], label='Perdida de Paquetes', color='red')
axs[2].set_ylabel('Paquetes Perdidos')
axs[2].set_title('Perdida de Paquetes')
axs[2].grid(True)
axs[2].legend()

axs[3].plot(df['Time'], df['Throughput'], label='Throughput', color='green')
axs[3].set_ylabel('Throughput (Bytes/s)')
axs[3].set_xlabel('Tiempo (s)')
axs[3].set_title('Throughput')
axs[3].grid(True)
axs[3].legend()

# Muestra y guarda graficos como archivo .PNG
plt.tight_layout()
plt.savefig('Graficos.png', format='png')
plt.show()
plt.close()
# ----------------------------
```

8. Resultados

Esta sección presenta los resultados del análisis de costos y el marco para la evaluación de la prueba de campo, proporcionando una visión cuantitativa y cualitativa de las arquitecturas de producción estudiadas.

8.1 Desglose de costos de producción por escenario

A continuación, se presentan los resultados del análisis de costos, divididos en los tres escenarios de producción definidos en la metodología.

8.1.1 Escenario de Noticiero (ENG)

Este análisis compara el MVP basado en la utilización de un *smartphone* con una solución profesional ENG tradicional.

Categoría de Equipamiento	**Escenario Mínimo Viable (*Smartphone*)**	**Escenario Profesional (Cámara ENG)**
Cámara	Smartphone de alta gama. Costo: U$S 1.000 – 1.500	Cámara ENG profesional (ej. Sony PXW-Z450). Costo: Aprox. U$S 20.000
Codificador/Transmisión	App “StreamCast Pro” (iOS) (gratuita)	Mochila de transmisión (Dejero EnGo, LiveU LU800) Costo: U$S 15.000 – 23.000
Accesorios	Baterías externas, *gimbal*, soportes	Baterías, media (almacenamiento), trípode ligero.
Costo Total Aproximado (CapEx)	**U$S 1.000 – 2.000**	**U$S 35.000 – 48.000**

Tabla III: Comparativa de costos de capital para escenarios de producción tipo ENG

8.1.2 Escenario de Producciones Pequeñas (EFP – 4 cámaras)

La siguiente tabla compara los costos de una producción EFP de 4 cámaras, desglosando la inversión entre un modelo de producción on-site y uno remoto (REMI), y a su vez subdividimos cada uno entre alternativas de alta gama (denominada "premium") y económicas.

Categoría de Gasto	**Opción On-site (Premium)**	**Opción On-site (Económica)**	**Opción Remota (Premium)**	**Opción Remota (Económica)**
Cadenas de Cámara (x4)	Sony HDC-3500. U$S 120k – 180k c/u	Blackmagic URSA G2. U$S 8k – 15k c/u	Sony HDC-3500. U$S 120k – 180k c/u	Blackmagic URSA G2. U$S 8k – 15k c/u
Lentes (x4)	Lentes de estudio (ej. Canon KJ22ex). U$S 30k – 40k c/u	Lentes 22x usados. U$S 3k – 6k c/u	Lentes de estudio. U$S 30k – 40k c/u	Lentes 22x usados. U$S 3k – 6k c/u
Switcher y *Replay*	Ross Carbonite / EVS XT-3. U$S 150k - 250k	Blackmagic ATEM / vMix. U$S 10k – 30k	No incluído (en *hub*)	No incluido (en la nube)
Audio y Comunicaciones	Lawo / Riedel. U$S 100k – 300k	Behringer / Hollyland. U$S 5k – 20k	No incluído (en *hub*)	No incluido (en la nube)
Transmisión	Satélite/Fibra. (OpEx muy alto, U$S ~1.000/hora)	Codificadores y 5G. U$S 10k – 30k	Mochilas LiveU/Dejero. U$S 20k – 80k	Codificadores Kiloview. U$S 8k – 30k
Total estimado (CAPEX)	**U$S 900.000 – 1.800.000**	**U$S 70.000 – 180.000**	**U$S 650.000 – 1.300.000**	**U$S 50.000 – 150.000**

Tabla IV: Comparativa de costos de capital para escenarios de producción tipo EFP

8.1.3 Escenario de Grandes Eventos (10 cámaras)

La siguiente tabla extiende el análisis a una producción de gran escala, donde los costos de capital se magnifican.

Categoría de Gasto	**Opción On-site (Premium)**	**Opción On-site (Económica)**	**Opción Remota (Premium)**	**Opción Remota (Económica)**
Cadenas de Cámara (x10)	Sony/Grass Valley. U$S 180k – 250k c/u	Blackmagic URSA G2. U$S 8k – 15k c/u	Sony/Grass Valley. U$S 180k – 250k c/u	Blackmagic URSA G2. U$S 8k – 15k c/u
Lentes (x10)	Lentes de gran zoom (ej. Canon HJ40x). U$S 40k – 100k c/u	Lentes 22x usados. U$S 3k – 6k c/u	Lentes de gran zoom. U$S 40k – 100k c/u	Lentes 22x usados. U$S 3k – 6k c/u
Switcher y *Replay*	Ross Acuity / EVS XT-VIA. U$S 400k – 1.5M	Blackmagic ATEM / vMix. U$S 20k – 50k	No incluido (en hub).	No incluido (en la nube)
Audio y Comunicaciones	Lawo / Riedel. U$S 200k – 700k	Behringer / Hollyland. U$S 20k – 50k	Lawo / Riedel (campo). U$S 50k – 200k	Behringer / Hollyland (campo). U$S 10k – 50k
Transmisión	Satélite/Fibra. (OpEx muy alto, U$S ~1.000/hora)	Codificadores y transmisor 5G. U$S 20k – 50k	Mochilas LiveU/Dejero. U$S 200k – 500k	Codificadores Kiloview. U$S 20k – 80k
Total estimado (CAPEX)	**U$S 3.000.000 – 6.000.000+**	**U$S 250.000 – 500.000**	**U$S 2.500.000 – 5.000.000**	**U$S 180.000 – 400.000**

Tabla V: Comparativa de costos de capital para escenarios de producción tipo Gran Evento

8.1.4 Análisis económico-financiero comparativo entre modelos

En línea con el planteo de la sección 7.1.3, se plantea debajo un análisis comparativo de los principales factores económicos y operativos entre el modelo de producción tradicional y el modelo Remoto/REMI, basado en los datos recabados.

Factor económico-operativo	**Modelo Tradicional**	**Modelo Remoto/REMI**	**Impacto y ventaja del nuevo modelo**
Costo por evento	Alto (U$S 10.000 – 100.000+ según la escala)	Bajo (U$S 1.000 – 10.000, según recursos)	Reducción significativa de los costos operativos totales
Tiempo de despliegue	Horas/días/semanas	Horas	Mayor agilidad y capacidad de respuesta ante cambios e imprevistos
Escalabilidad de eventos	Limitada por disponibilidad de las OB-van y equipos	Alta, centralizando recursos y virtualizando producción	Posibilidad de cubrir más eventos en paralelo, incluso en diferentes ubicaciones geográficas
Costo de infraestructura	Muy alto (CapEx/OpEx en móviles y equipos)	Bajo (Uso de servicios en la nube y posibilidad de equipos "*off the shelf*")	Menor inversión inicial, costos variables y adaptables al volumen de producción
Flexibilidad y movilidad	Baja	Muy Alta	Adaptación a diferentes tipos de eventos y ubicaciones, incluso con recursos mínimos en campo
Consumo energético	Alto (móviles, generadores, equipos)	Bajo (equipos y baterías portátiles, servicios en la nube eficientes)	Reducción de los costos asociados a energía y transporte, y del impacto ambiental
Personal en sitio	10 – 30 personas	2 – 5 personas	Menor necesidad de traslados y viáticos; optimización de recursos humanos
Utilización de recursos	Baja (Equipos ociosos entre eventos)	Alta (recursos compartidos y virtualizados)	Mejor aprovechamiento de infraestructura y personal, mayor retorno de la inversión

Tabla VI: Comparativa de modelos de operación tradicional y modelo remoto/REMI

8.2 Resultados y Análisis de la Prueba de Campo (MVP)

En esta sección se presentarán los datos cuantitativos y cualitativos obtenidos de la ejecución de la prueba de campo descrita en la metodología propuesta (sección 7.2).

8.2.1 Resultados de la Preproducción (relevamiento de sitio)

La prueba sobre red 5G se realiza con el dispositivo móvil descrito, desde una posición estacionaria, en un ambiente interior, en las coordenadas geográficas -34.5724 (latitud) y -58.4912 (longitud), correspondientes al barrio de Villa Urquiza, CABA, en altura al nivel de la calle. La prueba se realiza un sábado, alrededor de las 9 AM.

En la figura 8.1 se presenta el mapa actual de cobertura del servicio 5G del proveedor utilizado en la zona geográfica de prueba (Proveedor de servicio: Claro Argentina).

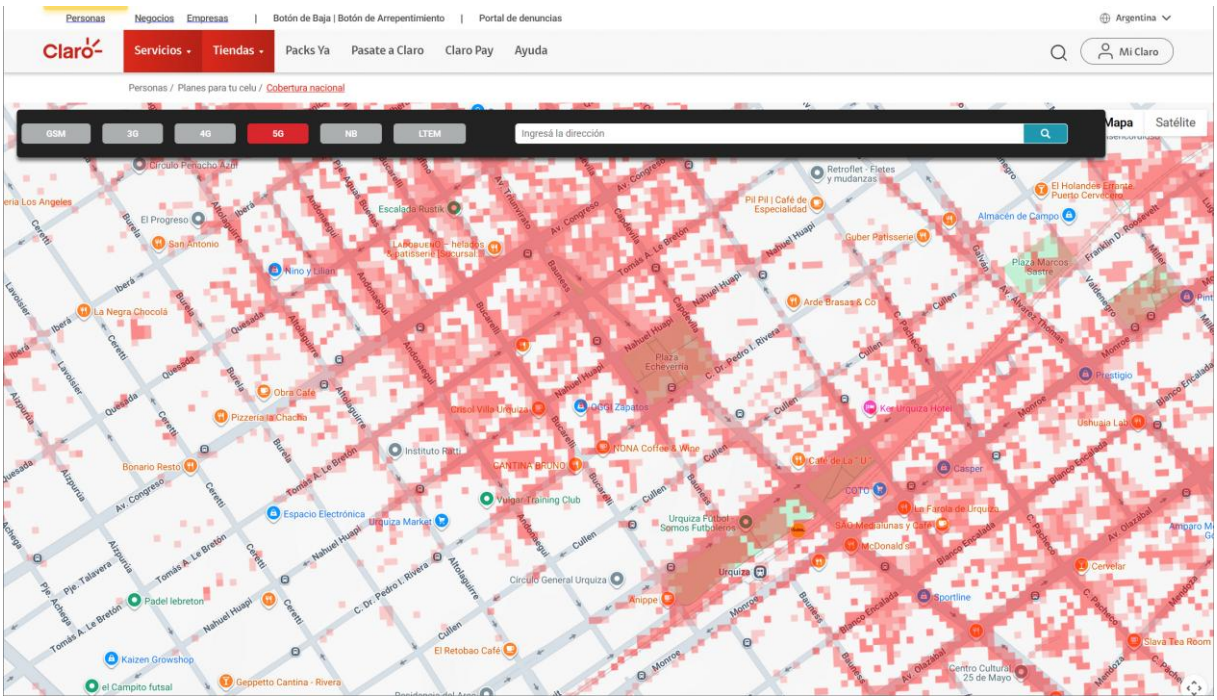


Figura 8.1: Mapa de cobertura de 5G para Claro Argentina en las coordenadas de realización de las pruebas.

Utilizando el modo "Field Test" de iOS (modo que es parte del sistema operativo, utilizado para medición en campo de señal de red móvil), realizamos una medición de la calidad de señal para referencia, enfocándonos en los siguientes valores:

- **Intensidad de Señal (RSRP por sus siglas en inglés)**: Indica cuán fuerte es la señal cuando llega al dispositivo móvil. Este parámetro se mide en dBm (decibeles de miliwatt). Los valores normalmente oscilan entre -50 dBm (excelente) y -120 dBm (muy débil).
- **Calidad de la Señal de Referencia Recibida (RSRQ por sus siglas en inglés)**: Indica cuán 'limpia' o clara es la señal. Este parámetro se mide en dB (decibeles), con un rango usual entre 0 dB y -20 dB. Cuanto más cercano a cero, mejor será la calidad de la señal.
- **Relación Señal a Interferencia y Ruido (SINR por sus siglas en inglés)**: Muestra cuánta interferencia o ruido está afectando la señal. Números más altos son positivos, con valores por encima de 20 dB considerados excelentes.

Se obtienen las siguientes métricas instantáneas de señal 5G en la ubicación descrita (Figura 8.2):

- RSRP: **-106 dBm** (Valor pobre)
- RSRQ: **-15 dB** (Valor promedio)
- SINR 0 (comunicación principal): **-1.3 dB** (Valor pobre)

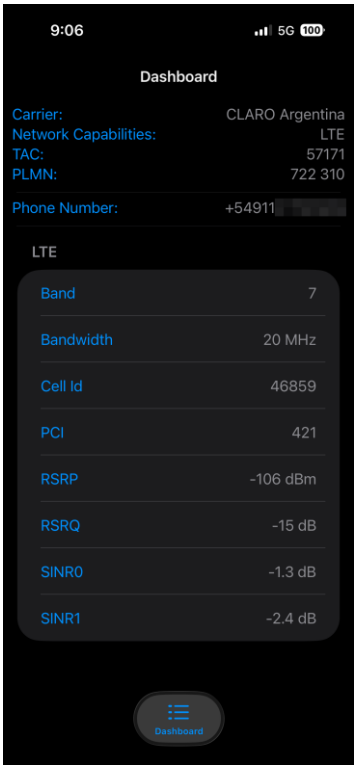


Figura 8.2: Resultados de medición de calidad de señal 5G con “Field Test” de iOS, al momento de realización de la prueba de campo

Adicionalmente, utilizando la aplicación móvil “Network Ping Lite”, se realiza una prueba de latencia por medio del habitual comando de red “ping”, entre el transmisor (*smartphone*) y el receptor (PC). Los resultados devuelven valores **por debajo de los 50 ms** (figura 8.3).

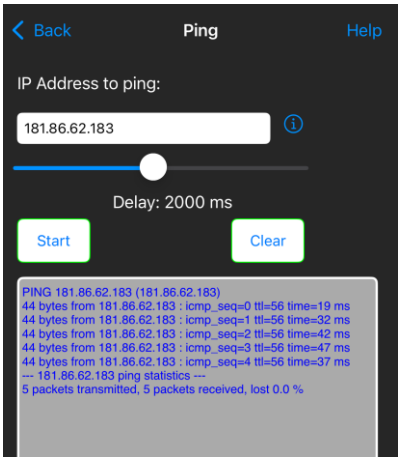


Figura 8.3: Resultados de prueba “ping” entre transmisor (smartphone) y receptor (PC)

Con estos valores, observamos que las condiciones de conectividad 5G empleadas no son las óptimas para transmisión a nivel de señal 5G, evitando de este modo escenarios ideales en la realización de la prueba de campo. Esto nos permite reflejar condiciones reales de transmisión, la cuales difícilmente sean las ideales dada la variabilidad que puede involucrar una producción del estilo propuesto, así como también las limitaciones actuales de la cobertura y calidad del servicio nacional 5G.

Por otro lado, la baja latencia, incluso al atravesar una multiplicidad de redes no gestionadas que conforman los tramos de red entre transmisor y receptor, muestran que aun en condiciones adversas, la conectividad a alta velocidad puede llevarse a cabo con éxito.

8.2.1 Resultados de la transmisión y captura

Desde el dispositivo móvil, luego de haber configurado la conectividad 5G y registrado la medición de calidad de señal, se procede a iniciar la transmisión de video SRT hacia la computadora receptora. Se transmite durante 5 minutos continuos, capturando en el receptor el tráfico de red con “Wireshark” para ser analizado con posterioridad en sus métricas principales. Una vez completada la transmisión y la captura, se exportan los datos a formato CSV y se corre el código Python detallado en la sección 7.2.2 para calcular y extraer gráficas de las métricas relevantes (figura 8.5).

Por su parte, “Loopy SRT Stats Monitor” funciona como complemento para “OBS Studio” con dos funciones, proveer métricas cumulativas ‘en vivo’ de alto nivel sobre el *stream* y al mismo tiempo funciona como conmutador automático en caso que la señal se pierda, mostrando una gráfica o video a elección, de tal modo que la audiencia no se desconecte ante una transmisión sin señal fuente. Del monitoreo “en vivo” realizado con la aplicación durante la recepción proveniente del transmisor con 5G, se desprenden los valores de la figura 8.4.

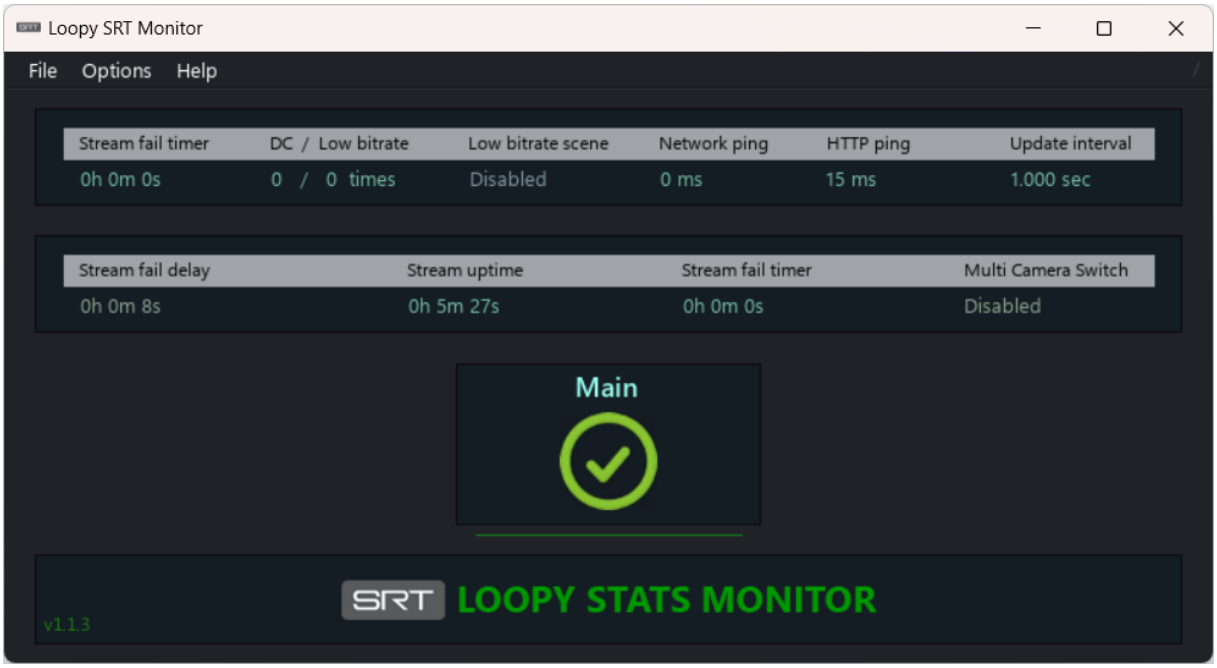


Figura 8.4: Métricas capturadas ‘en vivo’ por “Loopy SRT Stats Monitor”

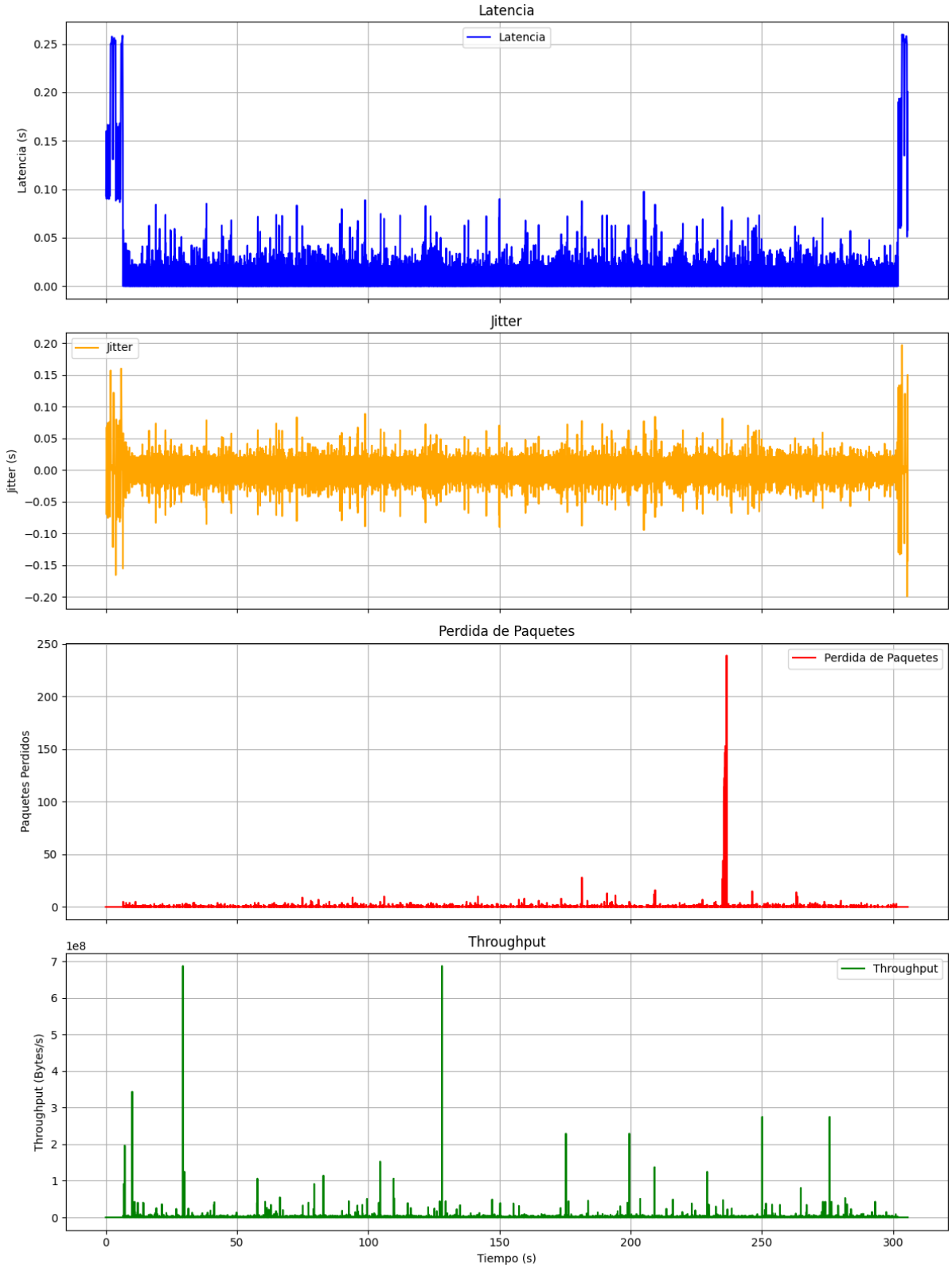


Figura 8.5: Métricas obtenidos de los principales indicadores de performance de transmisión

8.2.3 Interpretación y Análisis de Resultados

El análisis de los datos y gráficas obtenidas nos permite evaluar la eficacia del protocolo SRT en condiciones de uso reales en campo.

Por un lado, las métricas 'en vivo' obtenidas por "Loopy SRT Stats Monitor" nos indican que, durante el periodo de transmisión, se logró una baja latencia, no hubo caídas de *bitrate* por debajo del umbral de baja calidad, y la transmisión se mantuvo sin fallas ni cortes.

Al evaluar las métricas extraídas a partir de la captura de paquetes con "Wireshark" y analizadas en Python, se observa:

- Que aproximadamente el segundo 6 de comenzada la captura con "Wireshark", se inicia la recepción del video por SRT, y continúa hasta alrededor del segundo 300 (~5 minutos), cuando finaliza. Entre estos márgenes, observamos valores en una banda vertical acotada muy aceptable (bajos valores de latencia y *jitter*), que vuelven viable una transmisión estable.
- Alrededor de los segundos 230-240, se presenta un incremento momentáneo de la pérdida de paquetes. Sin embargo, durante la reproducción del video y audio no se percibió ningún corte o alteración al flujo. La aplicación de mecanismos del protocolo SRT como ARQ (*Automatic Repeat reQuest*), observables en el detalle de los paquetes capturados, demuestra las capacidades dinámicas del protocolo respecto a la retransmisión y tolerancia a interrupciones de corta duración, según varía la congestión de la red.
- El *throughput* mantiene una base relativamente constante, factor idealmente necesario para la estabilidad de la transmisión, y presenta picos esporádicos. Como indicamos previamente, el ancho de banda disponible y las capacidades del protocolo permitieron que, incluso durante los picos, la operatoria se mantenga estable, sin alteraciones perceptibles al video y audio.
- Por último, evaluando los límites de cada parámetro en comparación con los valores promedio de operación, vemos que hay margen para incrementar la calidad del video transmitido (resolución, fps, etc.) sin impactar el flujo.

9. Conclusiones y Observaciones

Como corolario de este trabajo, realizaremos a continuación un análisis FODA (Fortalezas, Oportunidades, Debilidades y Amenazas) respecto al escenario propuesto de contribución de video de alta calidad sobre redes no gestionadas, con enfoque directo en la propuesta de utilizar los elementos descritos en las secciones previas. Finalmente realizaremos una serie de conclusiones y observaciones emergentes.

9.1. Análisis FODA

Fortalezas	Oportunidades
• Reducción de costos de capital y operativos • Alta eficiencia y calidad de transmisión • Seguridad y resiliencia • Flexibilidad y escalabilidad • Adopción y soporte abierto • Movilidad y rapidez de despliegue	• Expansión en la cobertura de 5G • Tendencia en la producción remota y en la nube • Nuevos modelos de negocio y servicios • Integración con tecnologías emergentes • Demanda de contenidos en vivo y a demanda
Debilidades	**Amenazas**
• Dependencia de la cobertura y calidad de las redes no gestionadas • Complejidad de configuración y operación • Integración con equipamiento legado • Requerimientos de hardware específico • Gestión de energía y autonomía	• Problemas de conectividad y calidad de servicio • Cobertura y disponibilidad actualmente limitada de 5G • Riesgos de seguridad y ciberataques • Evolución tecnológica acelerada • Competencia de plataformas OTT y nuevos actores • Regulaciones y políticas cambiantes

Tabla VII: Análisis FODA (Fortalezas, Oportunidades, Debilidades y Amenazas)

9.1.1 Fortalezas

- **Reducción de costos de capital y operativos (CapEx/OpEx)**: El uso de tecnologías modernas y eficientes como el codec H.265, el protocolo de transporte SRT combinado con redes de alta velocidad (no gestionadas) como 5G eliminan la necesidad de costosas OB-vans, enlaces satelitales y equipamiento dedicado, permitiendo operar con hardware más accesible y flexible.
- **Alta eficiencia y calidad de transmisión**: Las tecnologías seleccionadas fueron diseñadas para aplicación de video en tiempo real, con soporte de baja latencia y alta calidad, incluso sobre redes no gestionadas, con tolerancia a cierto margen de pérdidas sin degradación perceptible.
- **Seguridad y resiliencia**: SRT incorpora cifrado criptográfico AES de 128/256 bits y mecanismos de corrección de errores, lo que asegura la integridad y confiabilidad de la señal.
- **Flexibilidad y escalabilidad**: La solución propuesta permite desplegar producciones en campo en potencialmente cualquier lugar con cobertura 5G, lo que facilita el acceso a eventos y ubicaciones previamente inaccesibles o muy costosas.
- **Adopción y soporte abierto**: SRT es un protocolo de código abierto y de creciente adopción en la industria, lo que facilita la interoperabilidad y la integración con múltiples plataformas, sistemas y fabricantes.
- **Movilidad y rapidez de despliegue**: El equipamiento necesario es portátil y puede instalarse rápidamente, reduciendo tiempos de armado, preparación y logística frente a la OB-van tradicional.

9.1.2 Oportunidades

- **Expansión en la cobertura de 5G**: El despliegue acelerado de redes 5G en zonas urbanas y en cercanía de eventos masivos habilita nuevos escenarios de contribución móvil y remota.
- **Tendencia en la producción remota y en la nube**: La industria del broadcast está migrando hacia flujos de producción virtualizados y en la nube, donde tecnologías como SRT y redes no gestionadas son habilitadoras clave. A su vez, estos nuevos

paradigmas de producción traen consigo bajas en los costos y productos más accesibles.

- **Nuevos modelos de negocio y servicios**: La reducción de barreras técnicas y de costos permite a productoras pequeñas (y por ende a las difusoras) acceder a coberturas de eventos grandes (antes reservadas para los principales jugadores del mercado) y chicos (que no eran cubiertos por limitaciones de presupuesto). Esto diversifica la oferta y el alcance.
- **Integración con tecnologías emergentes**: Las tecnologías propuestas optimizan la eficiencia y la calidad de producción, y al mismo tiempo abren la posibilidad de ser integradas con futuras nuevas tecnologías en cada espacio (ej. redes 6G, nuevas versiones de los códecs y protocolos, etc.).
- **Demanda de contenidos en vivo y a demanda**: El crecimiento del consumo de video en tiempo real y la necesidad de coberturas ágiles favorecen la adopción de soluciones móviles y flexibles.

9.1.3 Debilidades

- **Dependencia de la cobertura y calidad de las redes no gestionadas**: El sistema será tan robusto como la red sobre la que se operará. En zonas de cobertura deficiente o saturadas, la calidad puede verse comprometida.
- **Complejidad de configuración y operación**: La adopción de estas nuevas tecnologías requieren un bagaje de conocimiento actualizado y configuraciones adecuadas (ej. para SRT, los *buffers*, latencia y cifrado de encriptación) para optimizar el desempeño. Esto trae aparejado una demanda de capacitación técnica adicional.
- **Integración con equipamiento legado**: Algunos sistemas y flujos de producción tradicional pueden no ser compatibles con los códecs y protocolos digitales modernos, requiriendo de adaptadores o hardware especial para lograr su integración.
- **Requerimientos de hardware específico**: La codificación/decodificación eficiente de video (especialmente para códecs como H.265, que ya vimos es de alto requerimiento computacional) o el manejo de SRT pueden requerir hardware dedicado o aceleradoras, lo cual puede limitar la portabilidad según la implementación que se realice.

- **Gestión de energía y autonomía**: Aunque el equipamiento es mas portátil, una producción de múltiples horas en campo va a requerir de baterías y fuentes confiables, lo que puede seguir siendo un desafío en eventos extensos o ubicaciones remotas.

9.1.4 Amenazas

- **Problemas de conectividad y calidad de servicio**: Al operar con redes no gestionadas como 5G, pueden verse afectadas por obstrucciones físicas que afectan la estabilidad de conexión, así como también problemas de latencia al operar sobre múltiples tramos de redes de distinta naturaleza.
- **Cobertura y disponibilidad actualmente limitada de 5G**: A pesar del rápido crecimiento que se observa en áreas urbanas de Argentina, evidenciado en los mapas de cobertura de las distintas operadoras telefónicas, 5G aún no está disponible universalmente, especialmente en el interior o zonas rurales. Esto puede limitar la viabilidad de la solución, dependiendo de los tipos de evento a cubrir.
- **Riesgos de seguridad y ciberataques**: Aunque operemos con un protocolo que aplica cifrado, en toda plataforma digital existe el riesgo a vulnerabilidades de infraestructura, y más aun al operar sobre redes públicas, exponiéndose a posibles intercepciones o ataques.
- **Evolución tecnológica acelerada**: La rápida velocidad con la que suceden nuevos desarrollos a nivel de protocolos, estándares y soluciones puede volver obsoleta una inversión en poco tiempo, lo que exige una adecuada selección de tecnologías para maximizar interoperabilidad, y requiriendo actualización permanente.
- **Competencia de plataformas OTT y nuevos actores**: El avance de servicios de *streaming* y entrada de nuevos competidores puede desplazar al modelo de operación tradicional si no se adaptan rápidamente, tanto a tecnologías como a consumidores.
- **Regulaciones y políticas cambiantes**: Los cambios de asignación del espectro radioeléctrico, cambios en políticas públicas, en licencias de las tecnologías empleadas, y los requisitos técnicos pueden imponer barreras o requerir adaptaciones costosas, con el riesgo adicional de seleccionar tecnologías que puedan quedar rápidamente obsoletas o reemplazas.

9.2. Conclusiones

La combinación de los diferentes componentes enumerados en la sección 6, de naturaleza modular, nos permite establecer flujos de trabajo para la realización de producciones remotas de contribución de video donde se maximiza la movilidad y el esquema de operación distribuido. Las señales de cámaras y fuentes remotas pueden ser enviados directamente por internet (y desde dispositivos móviles, por redes como LTE y 5G) hacia la nube, eliminando la necesidad de enlaces satelitales o enlaces fijos dedicados.

La baja latencia conjugada que brindan los componentes citados habilita la posibilidad de conmutar y producir en tiempo real, siempre y cuando las capacidades de procesamiento (ej. codificación/decodificación) sean suficientes.

Adicionalmente el ancho de banda que brinda redes como 5G y sus futuras iteraciones, permiten transmitir múltiples señales HD o 4K en forma simultánea, facilitando producciones multicámara y de alta calidad.

La transición de un modelo de negocio intensivo en CapEx a uno flexible basado en OpEx representa el cambio más significativo en la economía de la producción televisiva en décadas. Este proyecto ha demostrado que dicha transformación no es meramente una opción, sino un imperativo estratégico impulsado por la necesidad de eficiencia, sostenibilidad y agilidad competitiva. La convergencia de las redes 5G no gestionadas y el protocolo de transporte SRT actúa como el catalizador tecnológico fundamental, haciendo que la producción remota de alta calidad sea no solo posible, sino económicamente viable a una escala sin precedentes.

El análisis de costos por escenarios revela que la adopción de flujos de trabajo REMI y equipamiento de menor costo puede reducir la inversión de capital en un orden de magnitud, pasando de millones de dólares para una OB-van tradicional a cientos de miles para una producción equivalente. Esta drástica disminución no solo optimiza los presupuestos, sino que, como se argumenta en la introducción, "democratiza" el acceso a la producción de

eventos en vivo, permitiendo la cobertura de un espectro mucho más amplio de contenido que antes era financieramente prohibitivo.

La prueba de campo del MVP, aunque es la implementación más básica, valida el principio fundamental: es posible lograr una contribución de video estable y de alta calidad sobre la internet pública utilizando herramientas accesibles. Esto confirma que la tecnología subyacente es sólida y escalable hacia los escenarios EFP y de grandes eventos más complejos.

9.3. La transformación del Ingeniero de *Broadcast*

Una de las conclusiones más profundas de este análisis es que la adopción de estas tecnologías redefine fundamentalmente el perfil del ingeniero de *broadcast*. El conjunto de habilidades tradicional, centrado en la electrónica, el flujo de señales de banda base y el mantenimiento de hardware específico, ya no es suficiente. La industria está experimentando una convergencia crítica entre la ingeniería de broadcast y la ingeniería de telecomunicaciones y la tecnología de información (IT).

El ingeniero moderno debe ser un profesional híbrido, con una profunda competencia en redes IP, incluyendo configuración de switches, gestión de subredes, ciberseguridad y, crucialmente, un entendimiento de las complejidades de las redes inalámbricas y los protocolos de transporte de medios. Debe ser capaz de diagnosticar problemas no en una placa de circuito con un osciloscopio, sino en una traza de red con un analizador de tráfico de red, entendiendo conceptos como *jitter*, pérdida de paquetes y latencia de extremo a extremo. Este cambio representa la evolución del rol desde un especialista en electrónica a un experto en telecomunicaciones y sistemas de información. Esta transformación no es una consecuencia secundaria, sino un requisito indispensable para el éxito del nuevo paradigma de producción.

10. Glosario de acrónimos y abreviaturas

El siguiente listado presenta las abreviaciones y acrónimos empleados en el desarrollo del presente trabajo.

1080p	Resolución de imagen de 1080 líneas horizontales (1920 x 1080 píxeles)
3GPP	*3rd Generation Partnership Project*
4G	Redes móviles de cuarta generación
4K	Resolución de imagen de aproximadamente 4000 píxeles horizontales (3840 x 2160 píxeles)
5G	Redes móviles de quinta generación
5GC	*5G Core Network*
6G	Redes móviles de sexta generación
720p	Resolución de imagen de 720 líneas horizontales (1280 x 720 píxeles)
8K	Resolución de imagen de aproximadamente 8000 píxeles horizontales (7680 x 4320 píxeles)
AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
ARQ	*Automatic Repeat reQuest*
AWS	*Amazon Web Services*
bps	*bits per second*
CABAC	*Context-Adaptive Binary Arithmetic Coding*
CapEx	*Capital Expenditure*
CCU	*Camera Control Unit*
CTU	*Coding Tree Unit*
DAS	*Distributed Antenna System*
DCT	*Discrete Cosine Transform*
DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*
DSNG	*Digital Satellite News Gathering*
DST	*Discrete Sine Transform*
EDGE	*Enhanced Data rates for GSM Evolution*
EFP	*Electronic Field Production*
EIC	*Engineer In Charge*

eMBB	*enhanced Mobile BroadBand*
ENG	*Electronic News Gathering*
FEC	*Forward Error Correction*
fps	frames per second
GSM	*Global System for Mobile Communications*
HD	*High Definition*
HEVC	*High Efficiency Video Codec*
HLS	*HTTP Live Streaming*
HTTP	*Hypertext Transfer Protocol*
IaaS	*Infrastructure-as-a-Service*
IAB	*Integrated Access/Backhaul*
IEC	*International Electrotechnical Commission*
IFB	*Interruptible Foldback*
IoT	*Internet of Things*
IP	*Internet Protocol*
IT	*Information Technology*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
ITU-T	*International Telecommunication Union Telecommunication Standardization Sector*
kbps	*kilobits per second*
LTE	*Long-Term Evolution*
Mbps	*Megabits per second*
MIMO	*Multiple-Input, Multiple-Output*
mMTC	*massive Machine-Type Communications*
mmWave	*millimeter Wave*
MPEG	*Moving Picture Experts Group*
MPL	*Mozilla Public License*
MPV	*Minimum Viable Product*
NAT / PAT	*Network Address Translation / Port Address Translation*
NDI	*Network Device Interface*
NR	*New Radio*
OB-van	*Outside Broadcast van*
OpEx	*Operational Expenditure*

PaaS	*Platform-as-a-Service*
PFI	Proyecto Final de Ingeniería
QoS	*Quality of Service*
REFEFO	Red Federal de Fibra Óptica (Argentina)
REMI	*Remote Integration Model*
RIST	*Reliable Internet Stream Transport*
ROI	*Return On Investment*
RSRP	*Reference Signal Received Power*
RSRQ	*Reference Signal Received Quality*
RTMP	*Real-Time Messaging Protocol*
RTT	*Round Trip Time*
SaaS	*Software-as-a-Service*
SDI	*Serial digital interface*
SDN	*Software Defined Network(s)*
SINR	*Signal to Interference plus Noise Ratio*
SMPTE	*Society of Motion Picture and Television Engineers*
SNG	*Satellite New Gathering*
SNR	*Signal to Noise Ratio*
SOA	*Service-Oriented Architecture*
SRT	*Secure Reliable Transport*
SWS	*Silly window syndrome* (TCP/IP)
UDP	*User Datagram Protocol*
UDT	*UDP-based Data Transfer Protocol*
UE	*User Equipment*
UHD	*Ultra High Definition*
UIT	Unión Internacional de Telecomunicaciones (también ITU)
UPS	*Uninterruptible Power Supply*
URLLC	*Ultra-Reliable Low-Latency Communications*
USB	*Universal Serial Bus*
USB-C	*Universal Serial Bus* Tipo C
VCEG	*Video Coding Experts Group*
WebRTC	*Web Real-Time Communication*
WPP	*Wavefront Parallel Processing*

11. Bibliografía

11.1. Libros

- THE INSTITUTION OF ENGINEERING AND TECHNOLOGY. *Digital Television Fundamentals*. Stevenage (Reino Unido): 2023. Capítulo 7.2, Video compression, pp. 235-264. Capítulo 8, pp. 315-327. Capítulo 9, pp. Digital video broadcasting primer pp.335-347. ISBN-13: 9781785612510.
  Disponible en: https://digital-library.theiet.org/doi/book/10.1049/pbte075e
- TOZER, E.P.J., editor. *Broadcast Engineer's Reference Book*. Burlington (EE.UU.): Elsevier Inc., 2004. Capítulo 5, Outside Broadcast Systems and Hardware, pp. 669-722. ISBN 0240519086.
- ZETTL, Herbert. *Manual de producción de televisión*. 10a ed. DF (México): Cengage Learning, 2010. Capítulo 13, "Switcheo" o edición instantánea, pp. 268-282. Capítulo 18, Producción de campo y grandes remotos. pp. 382-417. ISBN-13: 9786075193885.

11.2. Páginas Web

- *ARGENTINA.GOB.AR* [en línea]. © 2024 [consulta: 18 julio 2025]. Disponible en: https://www.argentina.gob.ar/jefatura/innovacion-publica/telecomunicaciones-y-conectividad/conectar/que-es-la-red-federal-de .
- *CLARO ARGENTINA* [en línea]. © 2025 [consulta: 19 septiembre 2025]. Disponible en: https://www.claro.com.ar/personas/planes-prepago-pospago/cobertura .
- *DGTL INFRA* [en línea]. © 2024 [consulta: 15 febrero 2025]. Disponible en: https://dgtlinfra.com/5g-spectrum-explained/ .
- *ENACOM* [en línea]. © 2021 [consulta: 12 febrero 2025]. Disponible en: https://www.enacom.gob.ar/institucional/enacom-convoca-a-pruebas-de-tecnologia-de-5g-en-argentina_n3038 .
- *ENACOM* [en línea]. © 2022 [consulta: 12 febrero 2025]. Disponible en: https://www.enacom.gob.ar/institucional/un-paso-decisivo-para-la-llegada-del-5g-a-la-argentina_n4236 .

- *ENACOM* [en línea]. © 2023 [consulta: 12 febrero 2025]. Disponible en: https://www.enacom.gob.ar/institucional/el-estado-argentino-licito-la-banda-5g-por-mas-de-875-millones-de-dolares_n4578 .
- *GITHUB* [en línea]. © 2025 [consulta: 22 noviembre 2024]. Disponible en: https://github.com/Haivision/srt .
- *HAIVISION* [en línea]. © 2025 [consulta: 6 octubre 2025]. Disponible en: https://doc.haivision.com/SRT/1.5.4/Haivision/ .
- *TVTECHNOLOGY* [en línea]. © 2025 [consulta: 6 octubre 2025]. Disponible en: https://www.tvtechnology.com/opinion/srt-in-practice-enabling-smarter-scalable-live-broadcast-infrastructure .
- *ROHDE & SCHWARZ* [en línea]. © 2025 [consulta: 16 febrero 2025]. Disponible en: https://www.rohde-schwarz.com/lat/soluciones/wireless-communications-testing/mobile-network-infrastructure-testing/infografia-creando-la-base-de-la-red-5g_255101.html .

11.3. Recursos electrónicos monográficos

- HAIVISION. *Broadcast Transformation Report 2025* [en línea]. © 2025. [consulta: 13 septiembre 2025]. Disponible en: https://www.haivision.com/white-papers/broadcast-ip-transformation-reports/ .
- HAIVISION. *White Paper: SRT Protocol Technical Overview* [en línea]. © 2018. [consulta: 12 octubre 2022]. Disponible en: https://www.haivision.com/white-papers/srt-protocol-technical-overview/ .
- HAIVISION. *White Paper: RMTP vs. SRT. Comparing Latency and Maximum Bandwidth* [en línea]. © 2018. [consulta: 14 octubre 2022]. Disponible en: https://www.haivision.com/white-papers/srt-versus-rtmp/ .
- HAIVISION. *White Paper: Unlocking the Potential of 5G for Live Broadcast Production* [en línea]. © 2022. [consulta: 14 octubre 2022]. Disponible en: https://www.haivision.com/white-papers/unlocking-the-potential-of-5g-for-live-broadcast-production/ .
- MARKETSANDMARKETS RESEARCH PRIVATE LTD. *5G Services Market Size, Share, Growth Analysis, Industry* [en línea]. © 2025. [consulta: 16 febrero 2025].

Código de reporte TC 6596. Disponible en: https://www.marketsandmarkets.com/Market-Reports/5g-services-market-226908556.html .

- SHARABAYKO, M.P y SHARABAYKO, M.A. *The SRT Protocol, draft-sharabayko-srt-01* [en línea]. IETF, 2021. [consulta: 20 noviembre 2024]. Disponible en: https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01 .
- SRT ALLIANCE. *The SRT Deployment Guide* [en línea]. © 2023. [consulta: 14 noviembre 2024]. Disponible en: https://www.srtalliance.org/srt-deployment-guide/ .
- SRT ALLIANCE. *Live Video Transport: The Race for Cloud Leadership* [en línea]. © 2025. [consulta: 13 septiembre 2025]. Disponible en: https://www.srtalliance.org/the-race-for-cloud-leadership/ .
- TELEFONAKTIEBOLAGET LM ERICSSON (ERICSSON). *The state of 5G in Latin America: a closer look* [en línea]. © 2025. [consulta: 15 febrero 2025]. Disponible en: https://www.ericsson.com/en/reports-and-papers/mobility-report/closer-look/latin-america .

11.4. Otros Soportes

- TVU NETWORKS. *La tecnología remota: un camino para ampliar el interés en deportes diversos* [en línea]. Disponible en: https://www.youtube.com/watch?v=YNwvdrxUx04 .

12. Anexos

12.1. Especificación del protocolo SRT “Internet Draft 01” – 7 de Septiembre de 2021

Disponible en: https://datatracker.ietf.org/doc/html/draft-sharabayko-srt-01

(sólo disponible en inglés)

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 March 2022

M.P. Sharabayko
M.A. Sharabayko
Haivision Network Video, GmbH
J. Dube
Haivision Systems, Inc.
JS. Kim
JW. Kim
SK Telecom Co., Ltd.
7 September 2021

The SRT Protocol
draft-sharabayko-srt-01

Abstract

This document specifies Secure Reliable Transport (SRT) protocol. SRT is a user-level protocol over User Datagram Protocol and provides reliability and security optimized for low latency live video streaming, as well as generic bulk data transfer. For this, SRT introduces control packet extension, improved flow control, enhanced congestion control and a mechanism for data encryption.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 March 2022.

Copyright Notice

Table of Contents

1. Introduction

1.1. Motivation

The demand for live video streaming has been increasing steadily for many years. With the emergence of cloud technologies, many video processing pipeline components have transitioned from on-premises appliances to software running on cloud instances. While real-time streaming over TCP-based protocols like RTMP [RTMP] is possible at low bitrates and on a small scale, the exponential growth of the streaming market has created a need for more powerful solutions.

To improve scalability on the delivery side, content delivery networks (CDNs) at one point transitioned to segmentation-based technologies like HLS (HTTP Live Streaming) [RFC8216] and DASH (Dynamic Adaptive Streaming over HTTP) [ISO23009]. This move increased the end-to-end latency of live streaming to over few tens of seconds, which makes it unattractive for specific use cases where real-time is important. Over time, the industry optimized these delivery methods, bringing the latency down to few seconds.

While the delivery side scaled up, improvements to video transcoding became a necessity. Viewers watch video streams on a variety of different devices, connected over different types of networks. Since upload bandwidth from on-premises locations is often limited, video transcoding moved to the cloud.

RTMP became the de facto standard for contribution over the public Internet. But there are limitations for the payload to be transmitted, since RTMP as a media specific protocol only supports two audio channels and a restricted set of audio and video codecs, lacking support for newer formats such as HEVC [H.265], VP9 [VP9], or

AV1 [AV1].

Since RTMP, HLS and DASH rely on TCP, these protocols can only guarantee acceptable reliability over connections with low RTTs, and can not use the bandwidth of network connections to their full extent due to limitations imposed by congestion control. Notably, QUIC [RFC9000] has been designed to address these problems with HTTP-based delivery protocols in HTTP/3 [I-D.ietf-quic-http]. Like QUIC, SRT [SRTSRC] uses UDP instead of the TCP transport protocol, but assures more reliable delivery using Automatic Repeat Request (ARQ), packet acknowledgments, end-to-end latency management, etc.

1.2. Secure Reliable Transport Protocol

Low latency video transmissions across reliable (usually local) IP based networks typically take the form of MPEG-TS [ISO13818-1] unicast or multicast streams using the UDP/RTP protocol, where any packet loss can be mitigated by enabling forward error correction (FEC). Achieving the same low latency between sites in different cities, countries or even continents is more challenging. While it is possible with satellite links or dedicated MPLS [RFC3031] networks, these are expensive solutions. The use of public Internet connectivity, while less expensive, imposes significant bandwidth overhead to achieve the necessary level of packet loss recovery. Introducing selective packet retransmission (reliable UDP) to recover from packet loss removes those limitations.

Derived from the UDP-based Data Transfer (UDT) protocol [GHG04b], SRT is a user-level protocol that retains most of the core concepts and mechanisms while introducing several refinements and enhancements, including control packet modifications, improved flow control for handling live streaming, enhanced congestion control, and a mechanism for encrypting packets.

SRT is a transport protocol that enables the secure, reliable transport of data across unpredictable networks, such as the Internet. While any data type can be transferred via SRT, it is ideal for low latency (sub-second) video streaming. SRT provides improved bandwidth utilization compared to RTMP, allowing much higher contribution bitrates over long distance connections.

As packets are streamed from source to destination, SRT detects and adapts to the real-time network conditions between the two endpoints, and helps compensate for jitter and bandwidth fluctuations due to congestion over noisy networks. Its error recovery mechanism minimizes the packet loss typical of Internet connections.

To achieve low latency streaming, SRT had to address timing issues. The characteristics of a stream from a source network are completely changed by transmission over the public Internet, which introduces delays, jitter, and packet loss. This, in turn, leads to problems with decoding, as the audio and video decoders do not receive packets at the expected times. The use of large buffers helps, but latency is increased. SRT includes a mechanism to keep a constant end-to-end latency, thus recreating the signal characteristics on the receiver side, and reducing the need for buffering.

Like TCP, SRT employs a listener/caller model. The data flow is bi-directional and independent of the connection initiation - either the sender or receiver can operate as listener or caller to initiate a connection. The protocol provides an internal multiplexing mechanism, allowing multiple SRT connections to share the same UDP port, providing access control functionality to identify the caller on the listener side.

Supporting forward error correction (FEC) and selective packet retransmission (ARQ), SRT provides the flexibility to use either of the two mechanisms or both combined, allowing for use cases ranging from the lowest possible latency to the highest possible reliability.

SRT maintains the ability for fast file transfers introduced in UDT, and adds support for AES encryption.

2. Terms and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

SRT: The Secure Reliable Transport protocol described by this document.

PRNG: Pseudo-Random Number Generator.

3. Packet Structure

SRT packets are transmitted as UDP payload [RFC0768]. Every UDP packet carrying SRT traffic contains an SRT header immediately after the UDP header (Figure 1).

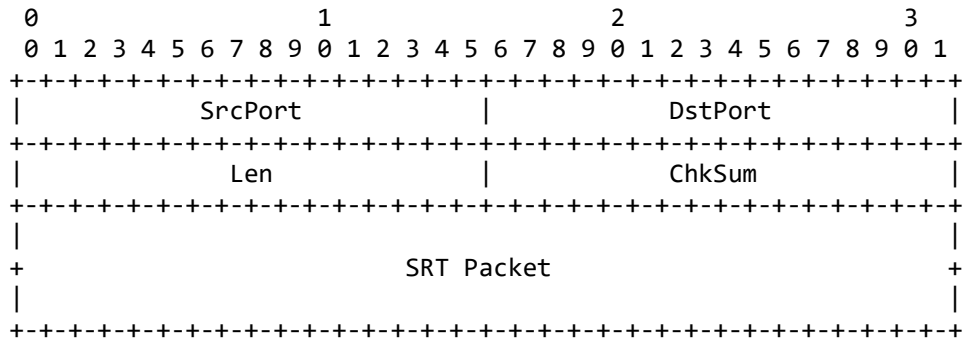


Figure 1: SRT packet as UDP payload

SRT has two types of packets distinguished by the Packet Type Flag: data packet and control packet.

The structure of the SRT packet is shown in Figure 2.

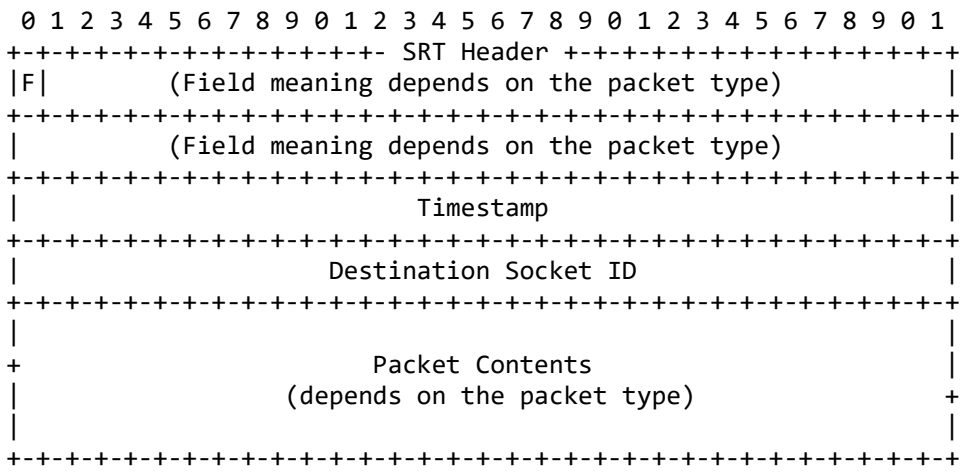


Figure 2: SRT packet structure

F: 1 bit. Packet Type Flag. The control packet has this flag set to "1". The data packet has this flag set to "0".

Timestamp: 32 bits. The timestamp of the packet, in microseconds. The value is relative to the time the SRT connection was established. Depending on the transmission mode (Section 4.2), the field stores the packet send time or the packet origin time.

Destination Socket ID: 32 bits. A fixed-width field providing the SRT socket ID to which a packet should be dispatched. The field may have the special value "0" when the packet is a connection request.

3.1. Data Packets

The structure of the SRT data packet is shown in Figure 3.

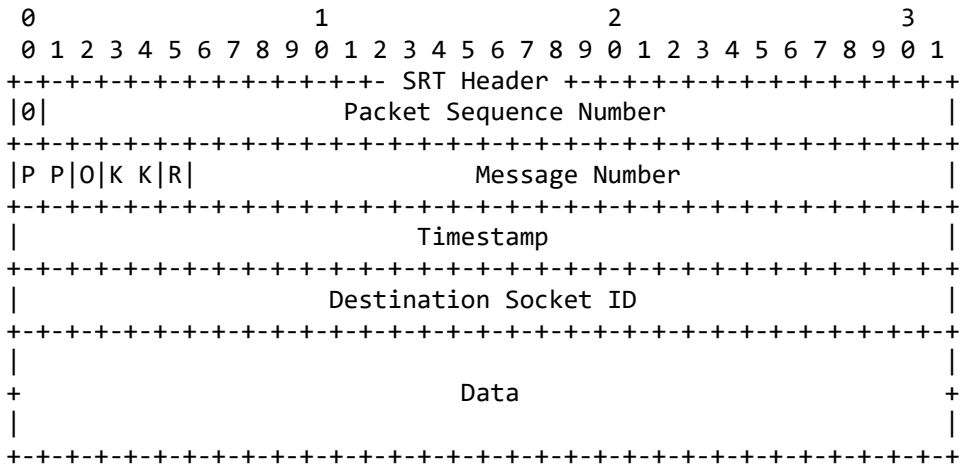


Figure 3: Data packet structure

Packet Sequence Number: 31 bits. The sequential number of the data packet.

PP: 2 bits. Packet Position Flag. This field indicates the position of the data packet in the message. The value "10b" (binary) means the first packet of the message. "00b" indicates a packet in the middle. "01b" designates the last packet. If a single data packet forms the whole message, the value is "11b".

O: 1 bit. Order Flag. Indicates whether the message should be delivered by the receiver in order (1) or not (0). Certain restrictions apply depending on the data transmission mode used (Section 4.2).

KK: 2 bits. Key-based Encryption Flag. The flag bits indicate whether or not data is encrypted. The value "00b" (binary) means data is not encrypted. "01b" indicates that data is encrypted with an even key, and "10b" is used for odd key encryption. Refer to Section 6. The value "11b" is only used in control packets.

R: 1 bit. Retransmitted Packet Flag. This flag is clear when a packet is transmitted the first time. The flag is set to "1" when a packet is retransmitted.

Message Number: 26 bits. The sequential number of consecutive data packets that form a message (see PP field).

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Data: variable length. The payload of the data packet. The length of the data is the remaining length of the UDP packet.

3.2. Control Packets

An SRT control packet has the following structure.

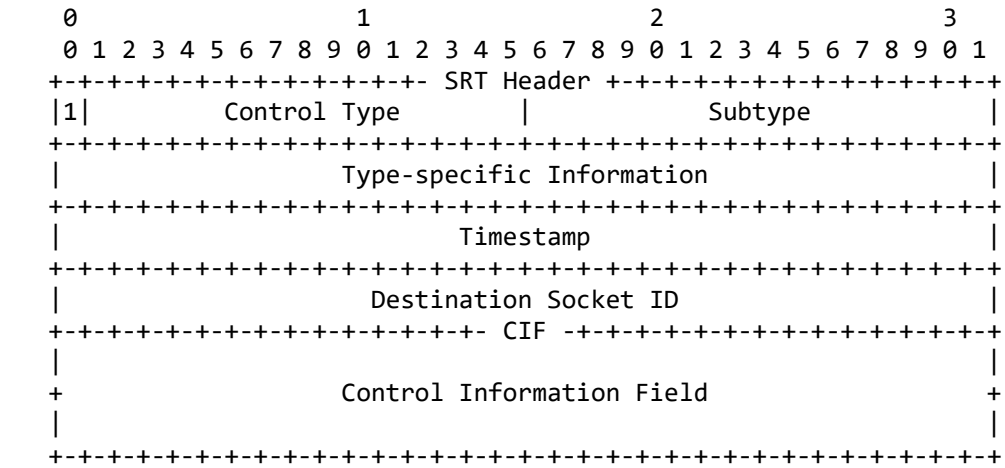


Figure 4: Control packet structure

Control Type: 15 bits. Control Packet Type. The use of these bits is determined by the control packet type definition. See Table 1.

Subtype: 16 bits. This field specifies an additional subtype for specific packets. See Table 1.

Type-specific Information: 32 bits. The use of this field depends on the particular control packet type. Handshake packets do not use this field.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Control Information Field (CIF): variable length. The use of this field is defined by the Control Type field of the control packet.

The types of SRT control packets are shown in Table 1. The value "0x7FFF" is reserved for a user-defined type.

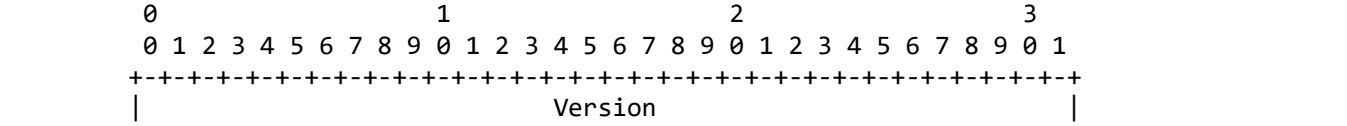
Packet Type	Control Type	Subtype	Section
HANDSHAKE	0x0000	0x0	Section 3.2.1
KEEPALIVE	0x0001	0x0	Section 3.2.3
ACK	0x0002	0x0	Section 3.2.4
NAK (Loss Report)	0x0003	0x0	Section 3.2.5
Congestion Warning	0x0004	0x0	Section 3.2.6
SHUTDOWN	0x0005	0x0	Section 3.2.7
ACKACK	0x0006	0x0	Section 3.2.8
DROPREQ	0x0007	0x0	Section 3.2.9
PEERERROR	0x0008	0x0	Section 3.2.10
User-Defined Type	0x7FFF	-	N/A

Table 1: SRT control packet types

3.2.1. Handshake

Handshake control packets (Control Type = 0x0000) are used to exchange peer configurations, to agree on connection parameters, and to establish a connection.

The Control Information Field (CIF) of a handshake control packet is shown in Figure 5.



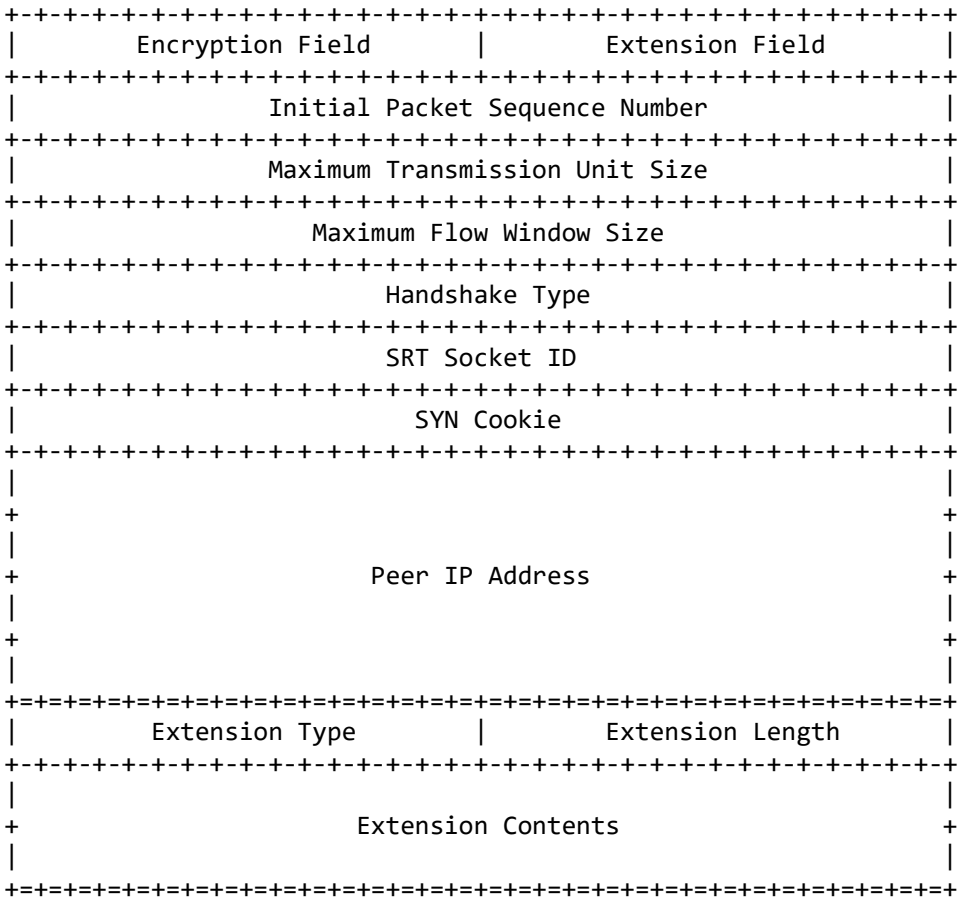


Figure 5: Handshake packet structure

Version: 32 bits. A base protocol version number. Currently used values are 4 and 5. Values greater than 5 are reserved for future use.

Encryption Field: 16 bits. Block cipher family and key size. The values of this field are described in Table 2. The default value is AES-128.

Value	Cipher Family and Key Size
0	No Encryption Advertised
2	AES-128
3	AES-192
4	AES-256

Table 2: Handshake Encryption Field values

Extension Field: 16 bits. This field is message specific extension

related to Handshake Type field. The value MUST be set to 0 except for the following cases. (1) If the handshake control packet is the INDUCTION message, this field is sent back by the Listener. (2) In the case of a CONCLUSION message, this field value should contain a combination of Extension Type values. For more details, see Section 4.3.1.

Bitmask	Flag
0x00000001	HSREQ
0x00000002	KMREQ
0x00000004	CONFIG

Table 3: Handshake Extension Flags

Initial Packet Sequence Number: 32 bits. The sequence number of the very first data packet to be sent.

Maximum Transmission Unit Size: 32 bits. This value is typically set to 1500, which is the default Maximum Transmission Unit (MTU) size for Ethernet, but can be less.

Maximum Flow Window Size: 32 bits. The value of this field is the maximum number of data packets allowed to be "in flight" (i.e. the number of sent packets for which an ACK control packet has not yet been received).

Handshake Type: 32 bits. This field indicates the handshake packet type. The possible values are described in Table 4. For more details refer to Section 4.3.

Value	Handshake Type
0xFFFFFFFD	DONE
0xFFFFFFFE	AGREEMENT
0xFFFFFFFF	CONCLUSION
0x00000000	WAVEHAND
0x00000001	INDUCTION

Table 4: Handshake Type

SRT Socket ID: 32 bits. This field holds the ID of the source SRT socket from which a handshake packet is issued.

SYN Cookie: 32 bits. Randomized value for processing a handshake.

The value of this field is specified by the handshake message type. See Section 4.3.

Peer IP Address: 128 bits. IPv4 or IPv6 address of the packet's sender. The value consists of four 32-bit fields. In the case of IPv4 addresses, fields 2, 3 and 4 are filled with zeroes.

Extension Type: 16 bits. The value of this field is used to process an integrated handshake. Each extension can have a pair of request and response types.

Value	Extension Type	HS Extension Flag
1	SRT_CMD_HSREQ	HSREQ
2	SRT_CMD_HSRSP	HSREQ
3	SRT_CMD_KMREQ	KMREQ
4	SRT_CMD_KMRSP	KMREQ
5	SRT_CMD_SID	CONFIG
6	SRT_CMD_CONGESTION	CONFIG
7	SRT_CMD_FILTER	CONFIG
8	SRT_CMD_GROUP	CONFIG

Table 5: Handshake Extension Type values

Extension Length: 16 bits. The length of the Extension Contents field in four-byte blocks.

Extension Contents: variable length. The payload of the extension.

3.2.1.1. Handshake Extension Message

In a Handshake Extension, the value of the Extension Field of the handshake control packet is defined as 1 for a Handshake Extension request (SRT_CMD_HSREQ in Table 5), and 2 for a Handshake Extension response (SRT_CMD_HSRSP in Table 5).

The Extension Contents field of a Handshake Extension Message is structured as follows:

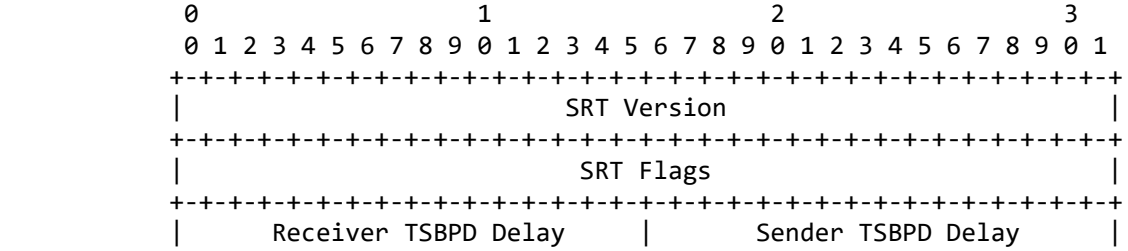


Figure 6: Handshake Extension Message structure

SRT Version: 32 bits. SRT library version MUST be formed as major * 0x10000 + minor * 0x100 + patch.

SRT Flags: 32 bits. SRT configuration flags (see Section 3.2.1.1.1).

Receiver TSBPD Delay: 16 bits. Timestamp-Based Packet Delivery (TSBPD) Delay of the receiver. Refer to Section 4.5.

Sender TSBPD Delay: 16 bits. TSBPD of the sender. Refer to Section 4.5.

3.2.1.1.1. Handshake Extension Message Flags

Bitmask	Flag
0x00000001	TSBPDSND
0x00000002	TSBPDRCV
0x00000004	CRYPT
0x00000008	TLPKTDROP
0x00000010	PERIODICNAK
0x00000020	REXMITFLG
0x00000040	STREAM
0x00000080	PACKET_FILTER

Table 6: Handshake Extension Message Flags

* TSBPDSND flag defines if the TSBPD mechanism (Section 4.5) will be used for sending.

* TSBPDRCV flag defines if the TSBPD mechanism (Section 4.5) will be used for receiving.

* CRYPT flag MUST be set. It is a legacy flag that indicates the party understands KK field of the SRT Packet (Figure 3).

* TLPKTDROP flag should be set if too-late packet drop mechanism will be used during transmission. See Section 4.6.

* PERIODICNAK flag set indicates the peer will send periodic NAK packets. See Section 4.8.2.

* REXMITFLG flag MUST be set. It is a legacy flag that indicates

the peer understands the R field of the SRT DATA Packet (Figure 3).

* STREAM flag identifies the transmission mode (Section 4.2) to be used in the connection. If the flag is set, the buffer mode (Section 4.2.2) is used. Otherwise, the message mode (Section 4.2.1) is used.

* PACKET_FILTER flag indicates if the peer supports packet filter.

3.2.1.2. Key Material Extension Message

If an encrypted connection is being established, the Key Material (KM) is first transmitted as a Handshake Extension message. This extension is not supplied for unprotected connections. The purpose of the extension is to let peers exchange and negotiate encryption-related information to be used to encrypt and decrypt the payload of the stream.

The extension can be supplied with the Handshake Extension Type field set to either SRT_CMD_KMREQ or SRT_CMD_HSRSP (see Table 5 in Section 3.2.1). For more details refer to Section 4.3.

The KM message is placed in the Extension Contents. See Section 3.2.2 for the structure of the KM message.

3.2.1.3. Stream ID Extension Message

The Stream ID handshake extension message can be used to identify the stream content. The Stream ID value can be free-form, but there is also a recommended convention that can be used to achieve interoperability.

The Stream ID handshake extension message has SRT_CMD_SID extension type (see Table 5. The extension contents are a sequence of UTF-8 characters. The maximum allowed size of the StreamID extension is 512 bytes.

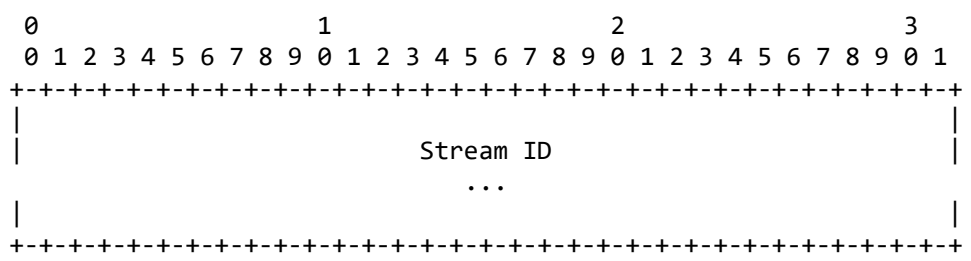


Figure 7: Stream ID Extension Message

The Extension Contents field holds a sequence of UTF-8 characters (see Figure 7). The maximum allowed size of the StreamID extension is 512 bytes. The actual size is determined by the Extension Length field (Figure 5), which defines the length in four byte blocks. If the actual payload is less than the declared length, the remaining bytes are set to zeros.

The content is stored as 32-bit little endian words.

3.2.1.4. Group Membership Extension

The Group Membership handshake extension is reserved for the future and is going to be used to allow multipath SRT connections.

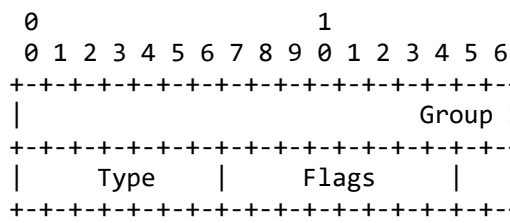


Figure 8: Group Membership Extension Message

GroupID: 32 bits. The identifier of a group whose members include the sender socket that is making a connection. The target socket that is interpreting GroupID SHOULD belong to the corresponding group on the target side. If such a group does not exist, the target socket MAY create it.

Type: 8 bits. Group type, as per SRT_GTYPE_ enumeration:

* 0: undefined group type,

* 1: broadcast group type,

* 2: main/backup group type,

* 3: balancing group type,

* 4: multicast group type (reserved for future use).

Flags: 8 bits. Special flags mostly reserved for the future. See Figure 9.

Weight: 16 bits. Special value with interpretation depending on the Type field value:

* Not used with broadcast group type,

* Defines the link priority for main/backup group type,

* Not yet defined for any other cases (reserved for future use).

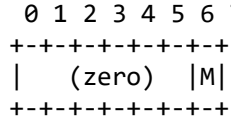


Figure 9: Group Membership Extension Flags

M: 1 bit. When set, defines synchronization on message numbers, otherwise transmission is synchronized on sequence numbers.

3.2.2. Key Material

The purpose of the Key Material Message is to let peers exchange encryption-related information to be used to encrypt and decrypt the payload of the stream.

This message can be supplied in two possible ways:

* as a Handshake Extension (see Section 3.2.1.2)

* in the Content Information Field of the User-Defined control packet (described below).

When the Key Material is transmitted as a control packet, the Control Type field of the SRT packet header is set to User-Defined Type (see Table 1), the Subtype field of the header is set to SRT_CMD_KMREQ for key-refresh request and SRT_CMD_KMRSP for key-refresh response (Table 5). The KM Refresh mechanism is described in Section 6.1.6.

The structure of the Key Material message is illustrated in Figure 10.

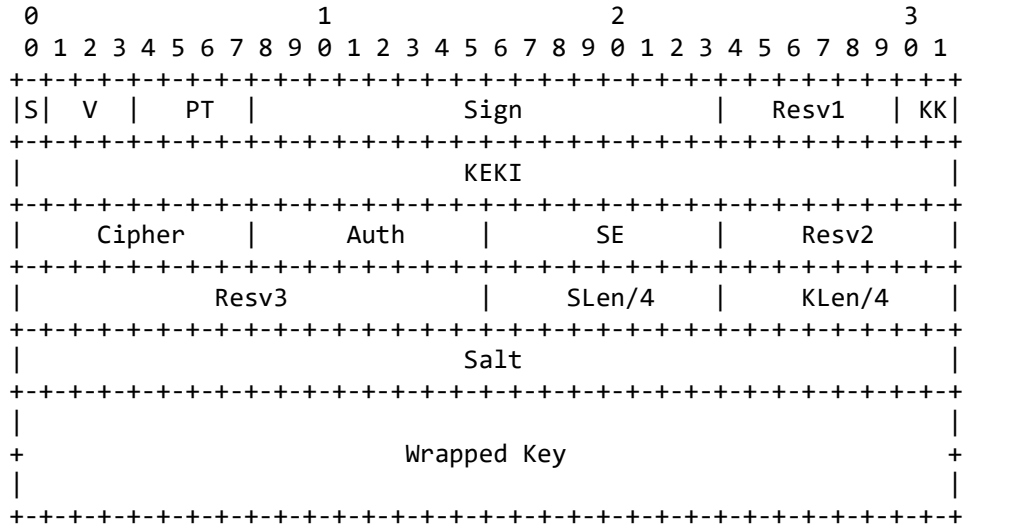


Figure 10: Key Material Message structure

S: 1 bit, value = {0}. This is a fixed-width field that is reserved for future usage.

Version (V): 3 bits, value = {1}. This is a fixed-width field that indicates the SRT version:

* 1: Initial version.

Packet Type (PT): 4 bits, value = {2}. This is a fixed-width field that indicates the Packet Type:

* 0: Reserved

```
      *  1: Media Stream Message (MSmsg)

      *  2: Keying Material Message (KMmsg)

      *  7: Reserved to discriminate MPEG-TS packet (0x47=sync byte).

   Sign: 16 bits, value = {0x2029}.  This is a fixed-width field that
      contains the signature 'HAI' encoded as a PnP Vendor ID [PNPID]
      (in big-endian order).

   Resv1: 6 bits, value = {0}.  This is a fixed-width field reserved for
      flag extension or other usage.

   Key-based Encryption (KK): 2 bits.  This is a fixed-width field that
      indicates which SEKs (odd and/or even) are provided in the
      extension:

      *  00b: No SEK is provided (invalid extension format);

      *  01b: Even key is provided;

      *  10b: Odd key is provided;

      *  11b: Both even and odd keys are provided.

   Key Encryption Key Index (KEKI): 32 bits, value = {0}.  This is a
      fixed-width field for specifying the KEK index (big-endian order)
      was used to wrap (and optionally authenticate) the SEK(s).  The
      value 0 is used to indicate the default key of the current stream.
      Other values are reserved for the possible use of a key management
      system in the future to retrieve a cryptographic context.

      *  0: Default stream associated key (stream/system default)

      *  1..255: Reserved for manually indexed keys.

   Cipher: 8 bits, value = {0..2}.  This is a fixed-width field for
      specifying encryption cipher and mode:

      *  0: None or KEKI indexed crypto context

      *  2: AES-CTR [SP800-38A].

   Authentication (Auth): 8 bits, value = {0}.  This is a fixed-width
      field for specifying a message authentication code algorithm:

      *  0: None or KEKI indexed crypto context.

   Stream Encapsulation (SE): 8 bits, value = {2}.  This is a fixed-
      width field for describing the stream encapsulation:

      *  0: Unspecified or KEKI indexed crypto context

      *  1: MPEG-TS/UDP

      *  2: MPEG-TS/SRT.
```

Resv2: 8 bits, value = {0}. This is a fixed-width field reserved for future use.

Resv3: 16 bits, value = {0}. This is a fixed-width field reserved for future use.

SLen/4: 8 bits, value = {4}. This is a fixed-width field for specifying salt length SLen in bytes divided by 4. Can be zero if no salt/IV present. The only valid length of salt defined is 128 bits.

KLen/4: 8 bits, value = {4,6,8}. This is a fixed-width field for specifying SEK length in bytes divided by 4. Size of one key even if two keys present. MUST match the key size specified in the Encryption Field of the handshake packet Table 2.

Salt (SLen): SLen * 8 bits, value = { }. This is a variable-width field that complements the keying material by specifying a salt key.

Wrap: (64 + n * KLen * 8) bits, value = { }. This is a variable-width field for specifying Wrapped key(s), where n = (KK + 1)/2 and the size of the wrap field is ((n * KLen) + 8) bytes.

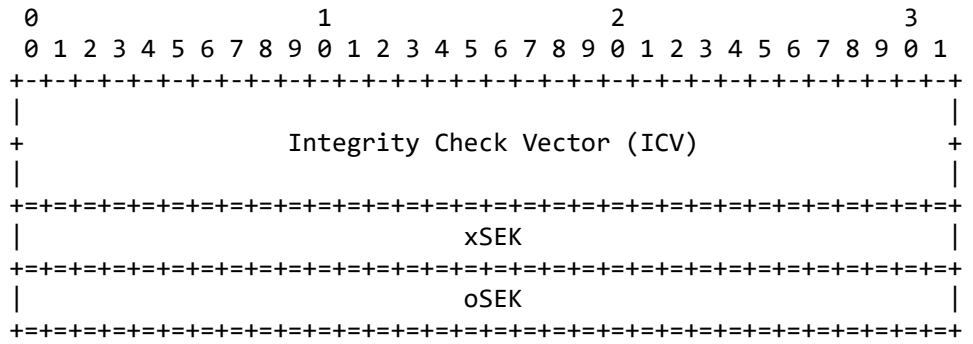


Figure 11: Unwrapped key structure

ICV: 64 bits. 64-bit Integrity Check Vector(AES key wrap integrity). This field is used to detect if the keys were unwrapped properly. If the KEK in hand is invalid, validation fails and unwrapped keys are discarded.

xSEK: variable width. This field identifies an odd or even SEK. If only one key is present, the bit set in the KK field tells which SEK is provided. If both keys are present, then this field is eSEK (even key) and it is followed by odd key oSEK. The length of this field is calculated as KLen * 8.

oSEK: variable width. This field with the odd key is present only when the message carries the two SEKs (identified by he KK field).

3.2.3. Keep-Alive

Keep-alive control packets are sent after a certain timeout from the

last time any packet (Control or Data) was sent. The purpose of this control packet is to notify the peer to keep the connection open when no data exchange is taking place.

The default timeout for a keep-alive packet to be sent is 1 second.

An SRT keep-alive packet is formatted as follows:

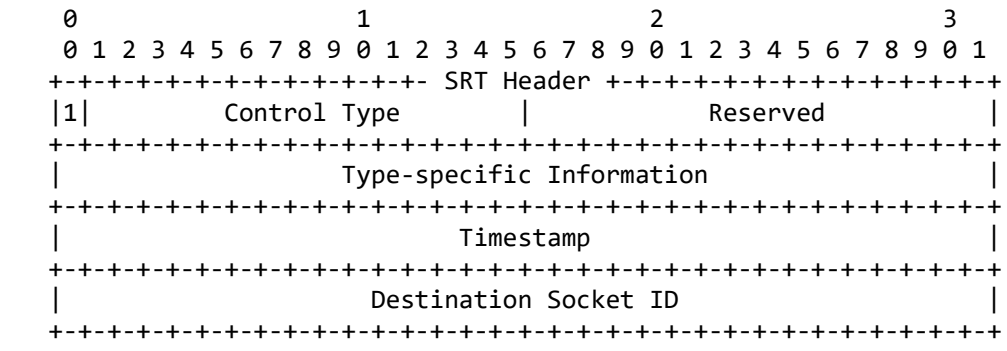


Figure 12: Keep-Alive control packet

Packet Type: 1 bit, value = 1. The packet type value of a keep-alive control packet is "1".

Control Type: 15 bits, value = KEEPALIVE{0x0001}. The control type value of a keep-alive control packet is "1".

Reserved: 16 bits, value = 0. This is a fixed-width field reserved for future use.

Type-specific Information. This field is reserved for future definition.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Keep-alive controls packet do not contain Control Information Field (CIF).

3.2.4. ACK (Acknowledgment)

Acknowledgment (ACK) control packets are used to provide the delivery status of data packets. By acknowledging the reception of data packets up to the acknowledged packet sequence number, the receiver notifies the sender that all prior packets were received or, in the case of live streaming (Section 4.2, Section 7.1), preceding missing packets (if any) were dropped as too late to be delivered (Section 4.6).

ACK packets may also carry some additional information from the receiver like the estimates of RTT, RTT variance, link capacity, receiving speed, etc. The CIF portion of the ACK control packet is expanded as follows:

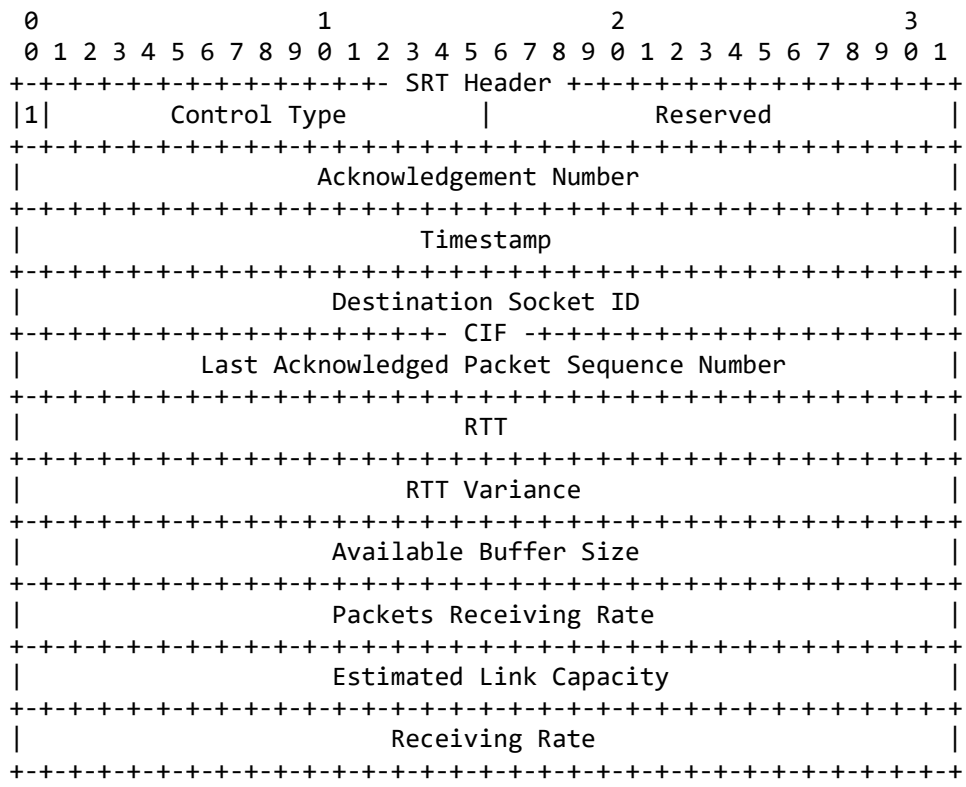


Figure 13: ACK control packet

Packet Type: 1 bit, value = 1. The packet type value of an ACK control packet is "1".

Control Type: 15 bits, value = ACK{0x0002}. The control type value of an ACK control packet is "2".

Reserved: 16 bits, value = 0. This is a fixed-width field reserved for future use.

Acknowledgement Number: 32 bits. This field contains the sequential number of the full acknowledgment packet starting from 1.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Last Acknowledged Packet Sequence Number: 32 bits. This field contains the sequence number of the last data packet being acknowledged plus one. In other words, if it the sequence number of the first unacknowledged packet.

RTT: 32 bits. RTT value, in microseconds, estimated by the receiver based on the previous ACK/ACKACK packet pair exchange.

RTT Variance: 32 bits. The variance of the RTT estimate, in microseconds.

Available Buffer Size: 32 bits. Available size of the receiver's

buffer, in packets.

Packets Receiving Rate: 32 bits. The rate at which packets are being received, in packets per second.

Estimated Link Capacity: 32 bits. Estimated bandwidth of the link, in packets per second.

Receiving Rate: 32 bits. Estimated receiving rate, in bytes per second.

There are several types of ACK packets:

* A Full ACK control packet is sent every 10 ms and has all the fields of Figure 13.

* A Light ACK control packet includes only the Last Acknowledged Packet Sequence Number field. The Type-specific Information field should be set to 0.

* A Small ACK includes the fields up to and including the Available Buffer Size field. The Type-specific Information field should be set to 0.

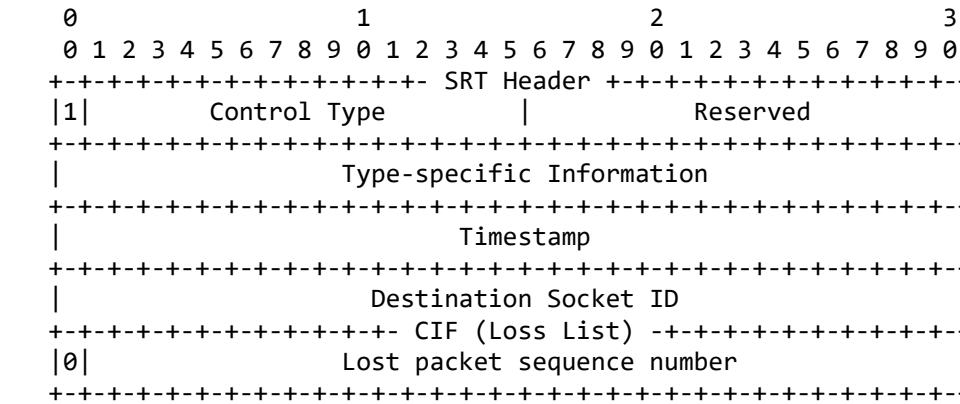
The sender only acknowledges the receipt of Full ACK packets (see Section 3.2.8).

The Light ACK and Small ACK packets are used in cases when the receiver should acknowledge received data packets more often than every 10 ms. This is usually needed at high data rates. It is up to the receiver to decide the condition and the type of ACK packet to send (Light or Small). The recommendation is to send a Light ACK for every 64 packets received.

3.2.5. NAK (Negative Acknowledgement or Loss Report)

Negative acknowledgment (NAK) control packets are used to signal failed data packet deliveries. The receiver notifies the sender about lost data packets by sending a NAK packet that contains a list of sequence numbers for those lost packets.

An SRT NAK packet is formatted as follows:



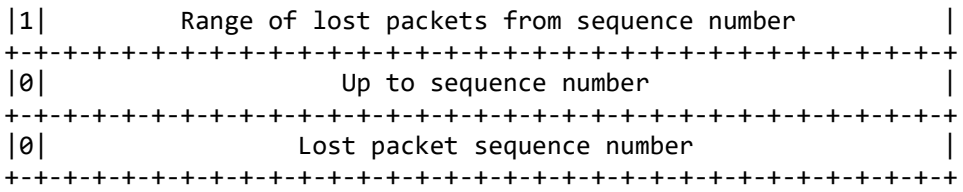


Figure 14: NAK control packet

Packet Type: 1 bit, value = 1. The packet type value of a NAK control packet is "1".

Control Type: 15 bits, value = NAK{0x0003}. The control type value of a NAK control packet is "3".

Reserved: 16 bits, value = 0. This is a fixed-width field reserved for future use.

Type-specific Information: 32 bits. This field is reserved for future definition.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Control Information Field (CIF). A single value or a range of lost packets sequence numbers. See packet sequence number coding in Appendix A.

3.2.6. Congestion Warning

The Congestion Warning control packet is reserved for future use. Its purpose is to allow a receiver to signal a sender that there is congestion happening at the receiving side. The expected behaviour is that upon receiving this packet the sender slows down its sending rate by increasing the minimum inter-packet sending interval by a discrete value (posited to be 12.5%).

Note that the conditions for a receiver to issue this type of packet are not yet defined.

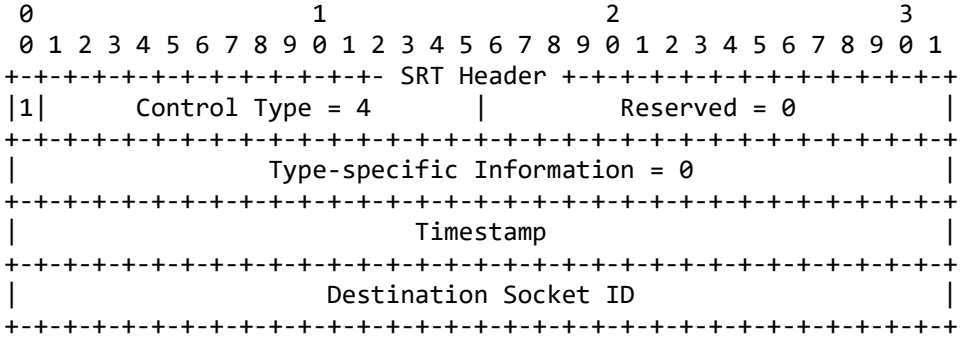


Figure 15: Congestion Warning control packet

Packet Type: 1 bit, value = 1. The packet type value of a Congestion

Warning control packet is "1".

Control Type: 15 bits, value = 4. The control type value of a Congestion Warning control packet is "4".

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Type-specific Information. This field is reserved for future definition.

3.2.7. Shutdown

Shutdown control packets are used to initiate the closing of an SRT connection.

An SRT shutdown control packet is formatted as follows:

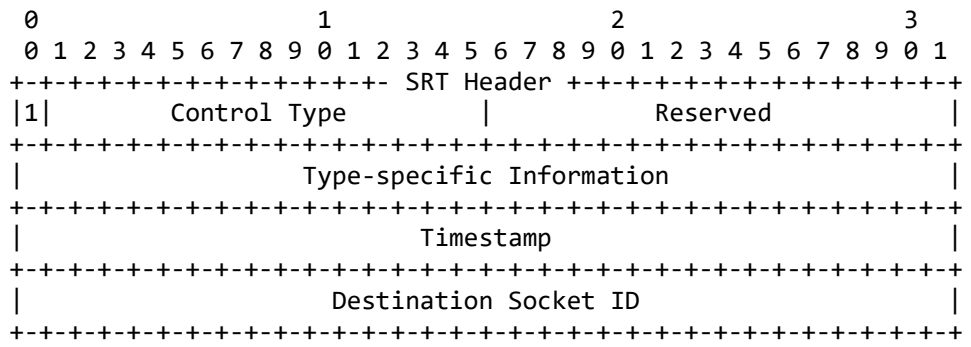


Figure 16: Shutdown control packet

Packet Type: 1 bit, value = 1. The packet type value of a shutdown control packet is "1".

Control Type: 15 bits, value = SHUTDOWN{0x0005}. The control type value of a shutdown control packet is "5".

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

Type-specific Information. This field is reserved for future definition.

Shutdown control packets do not contain Control Information Field (CIF).

3.2.8. ACKACK (Acknowledgement of Acknowledgement)

Acknowledgement of Acknowledgement (ACKACK) control packets are sent to acknowledge the reception of a Full ACK and used in the calculation of the round-trip time by the SRT receiver.

An SRT ACKACK control packet is formatted as follows:

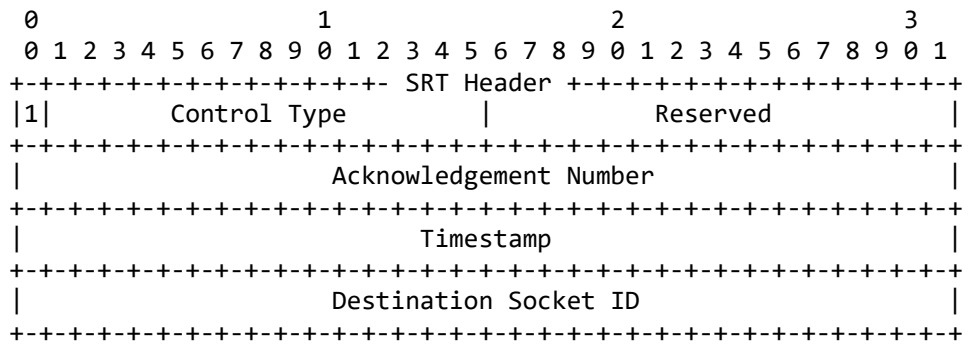


Figure 17: ACKACK control packet

Packet Type: 1 bit, value = 1. The packet type value of an ACKACK control packet is "1".

Control Type: 15 bits, value = ACKACK{0x0006}. The control type value of an ACKACK control packet is "6".

Acknowledgement Number. This field contains the Acknowledgement Number of the full ACK packet the reception of which is being acknowledged by this ACKACK packet.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

ACKACK control packets do not contain Control Information Field (CIF).

3.2.9. Message Drop Request

A Message Drop Request control packet is sent by the sender to the receiver when a retransmission of an unacknowledged packet (forming a whole or a part of a message) which is not present in the sender's buffer is requested. This may happen, for example, when a TTL parameter (passed in the sending function) triggers a timeout for retransmitting one or more lost packets which constitute parts of a message, causing these packets to be removed from the sender's buffer.

The sender notifies the receiver that it must not wait for retransmission of this message. Note that a Message Drop Request control packet is not sent if the Too Late Packet Drop mechanism (Section 4.6) causes the sender to drop a message, as in this case the receiver is expected to drop it anyway.

A Message Drop Request contains the message number and corresponding range of packet sequence numbers which form the whole message. If the sender does not already have in its buffer the specific packet or packets for which retransmission was requested, then it is unable to restore the message number. In this case the Message Number field must be set to zero, and the receiver should drop packets in the provided packet sequence number range.

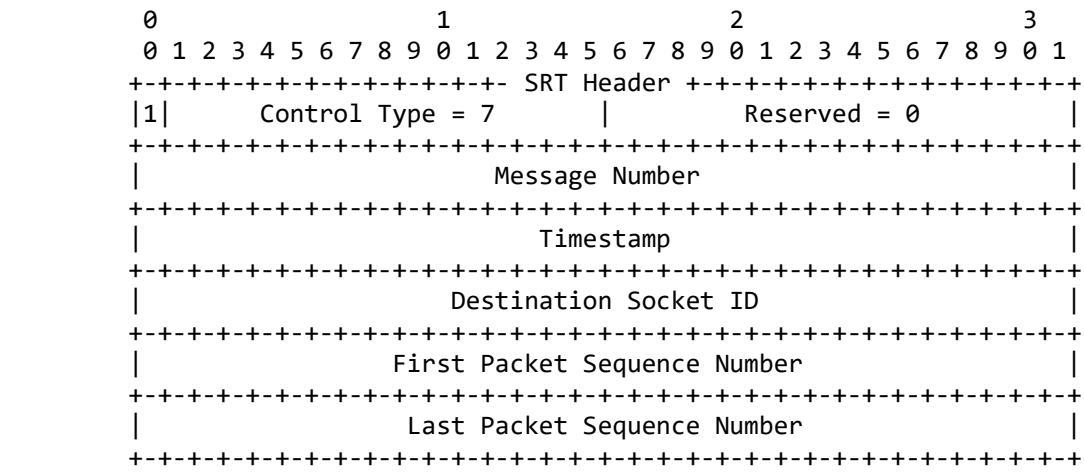


Figure 18: Drop Request control packet

Packet Type: 1 bit, value = 1. The packet type value of a Drop Request control packet is "1".

Control Type: 15 bits, value = 7. The control type value of a Drop Request control packet is "7".

Message Number: 32 bits. The identifying number of the message requested to be dropped. See the Message Number field in Section 3.1.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

First Packet Sequence Number: 32 bits. The sequence number of the first packet in the message.

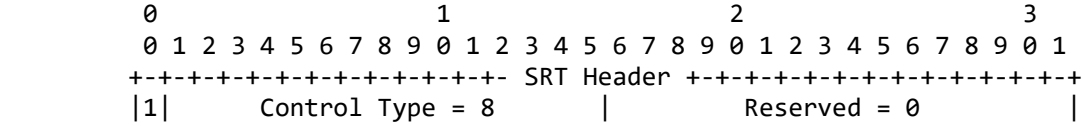
Last Packet Sequence Number: 32 bits. The sequence number of the last packet in the message.

3.2.10. Peer Error

The Peer Error control packet is sent by a receiver when a processing error (e.g. write to disk failure) occurs. This informs the sender of the situation and unblocks it from waiting for further responses from the receiver.

The sender receiving this type of control packet must unblock any sending operation in progress.

NOTE: This control packet is only used if the File Transfer Congestion Control (Section 5.2) is enabled.



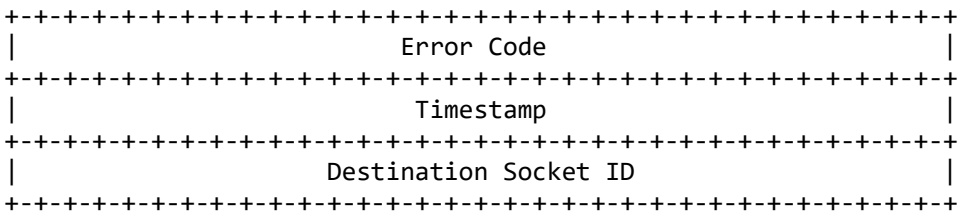


Figure 19: Peer Error control packet

Packet Type: 1 bit, value = 1. The packet type value of a Peer Error control packet is "1".

Control Type: 15 bits, value = 8. The control type value of a Peer Error control packet is "8".

Error Code: 32 bits. Peer error code. At the moment the only value defined is 4000 - file system error.

Timestamp: 32 bits. See Section 3.

Destination Socket ID: 32 bits. See Section 3.

4. SRT Data Transmission and Control

This section describes key concepts related to the handling of control and data packets during the transmission process.

After the handshake and exchange of capabilities is completed, packet data can be sent and received over the established connection. To fully utilize the features of low latency and error recovery provided by SRT, the sender and receiver must handle control packets, timers, and buffers for the connection as specified in this section.

4.1. Stream Multiplexing

Multiple SRT sockets may share the same UDP socket so that the packets received to this UDP socket will be correctly dispatched to those SRT sockets they are currently destined.

During the handshake, the parties exchange their SRT Socket IDs. These IDs are then used in the Destination Socket ID field of every control and data packet (see Section 3).

4.2. Data Transmission Modes

There are two data transmission modes supported by SRT: message mode (Section 4.2.1) and buffer mode (Section 4.2.2). These are the modes originally defined in the UDT protocol [GHG04b].

As SRT has been mainly designed for live video and audio streaming, its main and default transmission mode is message mode with certain settings applied (Section 7.1).

Besides live streaming, SRT maintains the ability for fast file transfers introduced in UDT (Section 7.2). The usage of both message

and buffer modes is possible in this case.

Best practices and configuration tips for both use cases can be found in Section 7.

4.2.1. Message Mode

When the STREAM flag of the handshake Extension Message Section 3.2.1.1 is set to 0, the protocol operates in Message mode, characterized as follows:

* Every packet has its own Packet Sequence Number.

* One or several consecutive SRT data packets can form a message.

* All the packets belonging to the same message have a similar message number set in the Message Number field.

The first packet of a message has the first bit of the Packet Position Flags (Section 3.1) set to 1. The last packet of the message has the second bit of the Packet Position Flags set to 1. Thus, a PP equal to "11b" indicates a packet that forms the whole message. A PP equal to "00b" indicates a packet that belongs to the inner part of the message.

The concept of the message in SRT comes from UDT [GHG04b]. In this mode, a single sending instruction passes exactly one piece of data that has boundaries (a message). This message may span multiple UDP packets and multiple SRT data packets. The only size limitation is that it shall fit as a whole in the buffers of the sender and the receiver. Although internally all operations (e.g., ACK, NAK) on data packets are performed independently, an application must send and receive the whole message. Until the message is complete (all packets are received) the application will not be allowed to read it.

When the Order Flag of a data packet is set to 1, this imposes a sequential reading order on messages. An Order Flag set to 0 allows an application to read messages that are already fully available, before any preceding messages that may have some packets missing.

4.2.2. Buffer Mode

Buffer mode is negotiated during the handshake by setting the STREAM flag of the handshake Extension Message Flags (Section 3.2.1.1.1) to 1.

In this mode, consecutive packets form one continuous stream that can be read with portions of any size.

4.3. Handshake Messages

SRT is a connection-oriented protocol. It embraces the concepts of "connection" and "session". The UDP system protocol is used by SRT for sending data and control packets.

An SRT connection is characterized by the fact that it is:

```
*  first engaged by a handshake process,

*  maintained as long as any packets are being exchanged in a timely
   manner, and

*  considered closed when a party receives the appropriate close
   command from its peer (connection closed by the foreign host), or
   when it receives no packets at all for some predefined time
   (connection broken on timeout).

SRT supports two connection configurations:

1.  Caller-Listener, where one side waits for the other to initiate a
    connection;

2.  Rendezvous, where both sides attempt to initiate a connection.

The handshake is performed between two parties: "Initiator" and
"Responder" in the following order:

*  Initiator starts an extended SRT handshake process and sends
   appropriate SRT extended handshake requests.

*  Responder expects the SRT extended handshake requests to be sent
   by the Initiator and sends SRT extended handshake responses back.

There are three basic types of SRT handshake extensions that are
exchanged in the handshake:

*  Handshake Extension Message exchanges the basic SRT information;

*  Key Material Exchange exchanges the wrapped stream encryption key
   (used only if an encryption is requested).

*  Stream ID extension exchanges some stream-specific information
   that can be used by the application to identify an incoming stream
   connection.

The Initiator and Responder roles are assigned depending on the
connection mode.

For Caller-Listener connections: the Caller is the Initiator, the
Listener is the Responder.  For Rendezvous connections: the Initiator
and Responder roles are assigned based on the initial data
interchange during the handshake.

The Handshake Type field in the Handshake Structure (see Figure 5)
indicates the handshake message type.

Caller-Listener handshake exchange has the following order of
Handshake Types:

1.  Caller to Listener: INDUCTION

2.  Listener to Caller: INDUCTION (reports cookie)
```

3. Caller to Listener: CONCLUSION (uses previously returned cookie)

4. Listener to Caller: CONCLUSION (confirms connection established).

Rendezvous handshake exchange has the following order of Handshake Types:

1. After starting the connection: WAVEAHAND

2. After receiving the above message from the peer: CONCLUSION

3. After receiving the above message from the peer: AGREEMENT.

When a connection process has failed before either party can send the CONCLUSION handshake, the Handshake Type field will contain the appropriate error value for the rejected connection. See the list of error codes in Table 7.

Code	Error	Description
1000	REJ_UNKNOWN	Unknown reason
1001	REJ_SYSTEM	System function error
1002	REJ_PEER	Rejected by peer
1003	REJ_RESOURCE	Resource allocation problem
1004	REJ_ROGUE	incorrect data in handshake
1005	REJ_BACKLOG	listener's backlog exceeded
1006	REJ_IPE	internal program error
1007	REJ_CLOSE	socket is closing
1008	REJ_VERSION	peer is older version than agent's min
1009	REJ_RDVCOOKIE	rendezvous cookie collision
1010	REJ_BADSECRET	wrong password
1011	REJ_UNSECURE	password required or unexpected
1012	REJ_MESSAGEAPI	Stream flag collision
1013	REJ_CONGESTION	incompatible congestion-controller type
1014	REJ_FILTER	incompatible packet filter
1015	REJ_GROUP	incompatible group

Table 7: Handshake Rejection Reason codes

The specification of the cipher family and block size is decided by the data Sender. When the transmission is bidirectional, this value MUST be agreed upon at the outset because when both are set the Responder wins. For Caller-Listener connections it is reasonable to set this value on the Listener only. In the case of Rendezvous the only reasonable approach is to decide upon the correct value from the different sources and to set it on both parties (note that *AES-128* is the default).

4.3.1. Caller-Listener Handshake

This section describes the handshaking process where a Listener is waiting for an incoming Handshake request on a bound UDP port from a Caller. The process has two phases: induction and conclusion.

4.3.1.1. The Induction Phase

The INDUCTION phase serves only to set a cookie on the Listener so that it doesn't allocate resources, thus mitigating a potential DoS attack that might be perpetrated by flooding the Listener with handshake commands.

The Caller begins by sending the INDUCTION handshake which contains the following significant fields:

* Version: MUST always be 4
* Encryption Field: 0
* Extension Field: 2
* Handshake Type: INDUCTION
* SRT Socket ID: SRT Socket ID of the Caller
* SYN Cookie: 0.

The Destination Socket ID of the SRT packet header in this message is 0, which is interpreted as a connection request.

The handshake version number is set to 4 in this initial handshake. This is due to the initial design of SRT that was to be compliant with the UDT protocol [GHG04b] on which it is based.

The Listener responds with the following:

* Version: 5
* Encryption Field: Advertised cipher family and block size
* Extension Field: SRT magic code 0x4A17
* Handshake Type: INDUCTION
* SRT Socket ID: Socket ID of the Listener

* SYN Cookie: a cookie that is crafted based on host, port and current time with 1 minute accuracy to avoid SYN flooding attack [RFC4987].

At this point the Listener still does not know if the Caller is SRT or UDT, and it responds with the same set of values regardless of whether the Caller is SRT or UDT.

If the party is SRT, it does interpret the values in Version and Extension Field. If it receives the value 5 in Version, it understands that it comes from an SRT party, so it knows that it should prepare the proper handshake messages phase. It also checks the following:

* whether the Extension Flags contains the magic value 0x4A17; otherwise the connection is rejected. This is a contingency for the case where someone who, in an attempt to extend UDT independently, increases the Version value to 5 and tries to test it against SRT;

* whether the Encryption Flags contain a non-zero value, which is interpreted as an advertised cipher family and block size.

A legacy UDT party completely ignores the values reported in Version and Handshake Type. It is, however, interested in the SYN Cookie value, as this must be passed to the next phase. It does interpret these fields, but only in the "conclusion" message.

4.3.1.2. The Conclusion Phase

Once the Caller gets the SYN cookie from the Listener, it sends the CONCLUSION handshake to the Listener.

The following values are set by the compliant Caller:

* Version: 5

* Handshake Type: CONCLUSION

* SRT Socket ID: Socket ID of the Caller

* SYN Cookie: the cookie previously received in the induction phase

* Encryption Flags: advertised cipher family and block size

* Extension Flags: a set of flags that define the extensions provided in the handshake

* The Destination Socket ID in this message is the socket ID that was previously received in the induction phase in the SRT Socket ID field of the handshake structure.

The Listener responds with the same values shown above, without the cookie (which is not needed here), as well as the extensions for HS Version 5 (which will probably be exactly the same).

There is not any "negotiation" here. If the values passed in the handshake are in any way not acceptable by the other side, the connection will be rejected. The only case when the Listener can have precedence over the Caller is the advertised Cipher Family and Block Size (see Table 2) in the Encryption Field of the Handshake.

The value for latency is always agreed to be the greater of those reported by each party.

4.3.2. Rendezvous Handshake

The Rendezvous process uses a state machine. It is slightly different from UDT Rendezvous handshake [GHG04b], although it is still based on the same message request types.

Both parties start with WAVEAHAND and use the Version value of 5. Legacy Version 4 clients do not look at the Version value, whereas Version 5 clients can detect version 5. The parties only continue with the Version 5 Rendezvous process when Version is set to 5 for both. Otherwise the process continues exclusively according to Version 4 rules [GHG04b].

With Version 5 Rendezvous, both parties create a cookie for a process called the "cookie contest". This is necessary for the assignment of Initiator and Responder roles. Each party generates a cookie value (a 32-bit number) based on the host, port, and current time with 1 minute accuracy. This value is scrambled using an MD5 sum calculation. The cookie values are then compared with one another.

Since it is impossible to have two sockets on the same machine bound to the same NIC and port and operating independently, it is virtually impossible that the parties will generate identical cookies. However, this situation may occur if an application tries to "connect to itself" - that is, either connects to a local IP address, when the socket is bound to INADDR_ANY, or to the same IP address to which the socket was bound. If the cookies are identical (for any reason), the connection will not be made until new, unique cookies are generated (after a delay of up to one minute). In the case of an application "connecting to itself", the cookies will always be identical, and so the connection will never be established.

When one party's cookie value is greater than its peer's, it wins the cookie contest and becomes Initiator (the other party becomes the Responder).

At this point there are two possible "handshake flows": serial and parallel.

4.3.2.1. Serial Handshake Flow

In the serial handshake flow, one party is always first, and the other follows. That is, while both parties are repeatedly sending WAVEAHAND messages, at some point one party - let's say Alice - will find she has received a WAVEAHAND message before she can send her next one, so she sends a CONCLUSION message in response. Meantime,

Bob (Alice's peer) has missed Alice's WAVEAHAND messages, so that Alice's CONCLUSION is the first message Bob has received from her.

This process can be described easily as a series of exchanges between the first and following parties (Alice and Bob, respectively):

1. Initially, both parties are in the waving state. Alice sends a handshake message to Bob:

   * Version: 5

   * Type: Extension field: 0, Encryption field: advertised "PBKEYLEN"

   * Handshake Type: WAVEAHAND

   * SRT Socket ID: Alice's socket ID

   * SYN Cookie: Created based on host/port and current time.

   While Alice does not yet know if she is sending this message to a Version 4 or Version 5 peer, the values from these fields would not be interpreted by the Version 4 peer when the Handshake Type is WAVEAHAND.

2. Bob receives Alice's WAVEAHAND message, switches to the "attention" state. Since Bob now knows Alice's cookie, he performs a "cookie contest" (compares both cookie values). If Bob's cookie is greater than Alice's, he will become the Initiator. Otherwise, he will become the Responder.

   The resolution of the Handshake Role (Initiator or Responder) is essential for further processing.

   Then Bob responds:

   * Version: 5

   * Extension field: appropriate flags if Initiator, otherwise 0

   * Encryption field: advertised PBKEYLEN

   * Handshake Type: CONCLUSION.

   If Bob is the Initiator and encryption is on, he will use either his own cipher family and block size or the one received from Alice (if she has advertised those values).

3. Alice receives Bob's CONCLUSION message. While at this point she also performs the "cookie contest", the outcome will be the same. She switches to the "fine" state, and sends:

   * Version: 5

   * Appropriate extension flags and encryption flags

* Handshake Type: CONCLUSION.

Both parties always send extension flags at this point, which will contain HSREQ if the message comes from an Initiator, or HSRSP if it comes from a Responder. If the Initiator has received a previous message from the Responder containing an advertised cipher family and block size in the encryption flags field, it will be used as the key length for key generation sent next in the KMREQ extension.

4. Bob receives Alice's CONCLUSION message, and then does one of the following (depending on Bob's role):

   * If Bob is the Initiator (Alice's message contains HSRSP), he:

     - switches to the "connected" state, and

     - sends Alice a message with Handshake Type AGREEMENT, but containing no SRT extensions (Extension Flags field should be 0).

   * If Bob is the Responder (Alice's message contains HSREQ), he:

     - switches to "initiated" state,

     - sends Alice a message with Handshake Type CONCLUSION that also contains extensions with HSRSP, and

     - awaits a confirmation from Alice that she is also connected (preferably by AGREEMENT message).

5. Alice receives the above message, enters into the "connected" state, and then does one of the following (depending on Alice's role):

   * If Alice is the Initiator (received CONCLUSION with HSRSP), she sends Bob a message with Handshake Type = AGREEMENT.

   * If Alice is the Responder, the received message has Handshake Type AGREEMENT and in response she does nothing.

6. At this point, if Bob was an Initiator, he is connected already. If he was a Responder, he should receive the above AGREEMENT message, after which he switches to the "connected" state. In the case where the UDP packet with the agreement message gets lost, Bob will still enter the "connected" state once he receives anything else from Alice. If Bob is going to send, however, he has to continue sending the same CONCLUSION until he gets the confirmation from Alice.

4.3.2.2. Parallel Handshake Flow

The chances of the parallel handshake flow are very low, but still it may occur if the handshake messages with WAVEAHAND are sent and received by both peers at precisely the same time.

The resulting flow is very much like Bob's behaviour in the serial handshake flow, but for both parties. Alice and Bob will go through the same state transitions:

Waving -> Attention -> Initiated -> Connected

In the Attention state they know each other's cookies, so they can assign roles. In contrast to serial flows, which are mostly based on request-response cycles, here everything happens completely asynchronously: the state switches upon reception of a particular handshake message with appropriate contents (the Initiator MUST attach the HSREQ extension, and Responder MUST attach the "HSRSP" extension).

Here is how the parallel handshake flow works, based on roles and states:

(1) Initiator

1. Waving

   * Receives WAVEAHAND message,

   * Switches to Attention,

   * Sends CONCLUSION + HSREQ.

2. Attention

   Receives CONCLUSION message which

   * either contains no extensions, then switches to Initiated, still sends CONCLUSION + HSREQ; or

   * contains "HSRSP" extension, then switches to Connected, sends AGREEMENT.

3. Initiated

   Receives CONCLUSION message, which

   * either contains no extensions, then REMAINS IN THIS STATE, still sends CONCLUSION + HSREQ; or

   * contains "HSRSP" extension, then switches to Connected, sends AGREEMENT.

4. Connected

   May receive CONCLUSION and respond with AGREEMENT, but normally by now it should already have received payload packets.

(2) Responder

1. Waving

```
      *  Receives WAVEAHAND message,

      *  Switches to Attention,

      *  Sends CONCLUSION message (with no extensions).

   2. Attention

      *  Receives CONCLUSION message with HSREQ.  This message might
         contain no extensions, in which case the party SHALL simply
         send the empty CONCLUSION message, as before, and remain in
         this state.

      *  Switches to Initiated and sends CONCLUSION message with HSRSP.

   3. Initiated

      Receives:

      *  CONCLUSION message with HSREQ, then responds with CONCLUSION
         with HSRSP and remains in this state;

      *  AGREEMENT message, then responds with AGREEMENT and switches
         to Connected;

      *  Payload packet, then responds with AGREEMENT and switches to
         Connected.

   4. Connected

      Is not expecting to receive any handshake messages anymore.  The
      AGREEMENT message is always sent only once or per every final
      CONCLUSION message.

Note that any of these packets may be missing, and the sending party
will never become aware.  The missing packet problem is resolved this
way:

   1. If the Responder misses the CONCLUSION + HSREQ message, it simply
      continues sending empty CONCLUSION messages.  Only upon reception
      of CONCLUSION + HSREQ it does respond with CONCLUSION + HSRSP.

   2. If the Initiator misses the CONCLUSION + HSRSP response from the
      Responder, it continues sending CONCLUSION + HSREQ.  The
      Responder MUST always respond with CONCLUSION + HSRSP when the
      Initiator sends CONCLUSION + HSREQ, even if it has already
      received and interpreted it.

   3. When the Initiator switches to the Connected state it responds
      with a AGREEMENT message, which may be missed by the Responder.
      Nonetheless, the Initiator may start sending data packets because
      it considers itself connected - it does not know that the
      Responder has not yet switched to the Connected state.  Therefore
      it is exceptionally allowed that when the Responder is in the
      Initiated state and receives a data packet (or any control packet
      that is normally sent only between connected parties) over this
```

connection, it may switch to the Connected state just as if it had received a AGREEMENT message.

4. If the the Initiator has already switched to the Connected state it will not bother the Responder with any more handshake messages. But the Responder may be completely unaware of that (having missed the AGREEMENT message from the Initiator). Therefore it does not exit the connecting state, which means that it continues sending CONCLUSION + HSRSP messages until it receives any packet that will make it switch to the Connected state (normally AGREEMENT). Only then does it exit the connecting state and the application can start transmission.

4.4. SRT Buffer Latency

The SRT sender and receiver have buffers to store packets.

On the sender, latency is the time that SRT holds a packet to give it a chance to be delivered successfully while maintaining the rate of the sender at the receiver. If an acknowledgment (ACK) is missing or late for more than the configured latency, the packet is dropped from the sender buffer. A packet can be retransmitted as long as it remains in the buffer for the duration of the latency window. On the receiver, packets are delivered to an application from a buffer after the latency interval has passed. This helps to recover from potential packet losses. See Section 4.5, Section 4.6 for details.

Latency is a value, in milliseconds, that can cover the time to transmit hundreds or even thousands of packets at high bitrate. Latency can be thought of as a window that slides over time, during which a number of activities take place, such as the reporting of acknowledged packets (ACKs) (Section 4.8.1) and unacknowledged packets (NAKs) (Section 4.8.2).

Latency is configured through the exchange of capabilities during the extended handshake process between initiator and responder. The Handshake Extension Message (Section 3.2.1.1) has TSBPD delay information, in milliseconds, from the SRT receiver and sender. The latency for a connection will be established as the maximum value of latencies proposed by the initiator and responder.

4.5. Timestamp-Based Packet Delivery

The goal of the SRT Timestamp-Based Packet Delivery (TSBPD) mechanism is to reproduce the output of the sending application (e.g., encoder) at the input of the receiving application (e.g., decoder) in the case of live streaming (Section 4.2, Section 7.1). It attempts to reproduce the timing of packets committed by the sending application to the SRT sender. This allows packets to be scheduled for delivery by the SRT receiver, making them ready to be read by the receiving application (see Figure 20).

The SRT receiver, using the timestamp of the SRT data packet header, delivers packets to a receiving application with a fixed minimum delay from the time the packet was scheduled for sending on the SRT sender side. Basically, the sender timestamp in the received packet

is adjusted to the receiver's local time (compensating for the time drift or different time zones) before releasing the packet to the application. Packets can be withheld by the SRT receiver for a configured receiver delay. A higher delay can accommodate a larger uniform packet drop rate, or a larger packet burst drop. Packets received after their "play time" are dropped if the Too-Late Packet Drop feature is enabled (Section 4.6). For example, in the case of live video streaming, TSBPD and Too-Late Packet Drop mechanisms allow to intentionally drop those packets that were lost and have no chance to be retransmitted before their play time. Thus, SRT provides a fixed end-to-end latency of the stream.

The packet timestamp, in microseconds, is relative to the SRT connection creation time. Packets are inserted based on the sequence number in the header field. The origin time, in microseconds, of the packet is already sampled when a packet is first submitted by the application to the SRT sender unless explicitly provided. The TSBPD feature uses this time to stamp the packet for first transmission and any subsequent retransmission. This timestamp and the configured SRT latency (Section 4.4) control the recovery buffer size and the instant that packets are delivered at the destination (the aforementioned "play time" which is decided by adding the timestamp to the configured latency).

It is worth mentioning that the use of the packet sending time to stamp the packets is inappropriate for the TSBPD feature, since a new time (current sending time) is used for retransmitted packets, putting them out of order when inserted at their proper place in the stream.

Figure 20 illustrates the key latency points during the packet transmission with the TSBPD feature enabled.

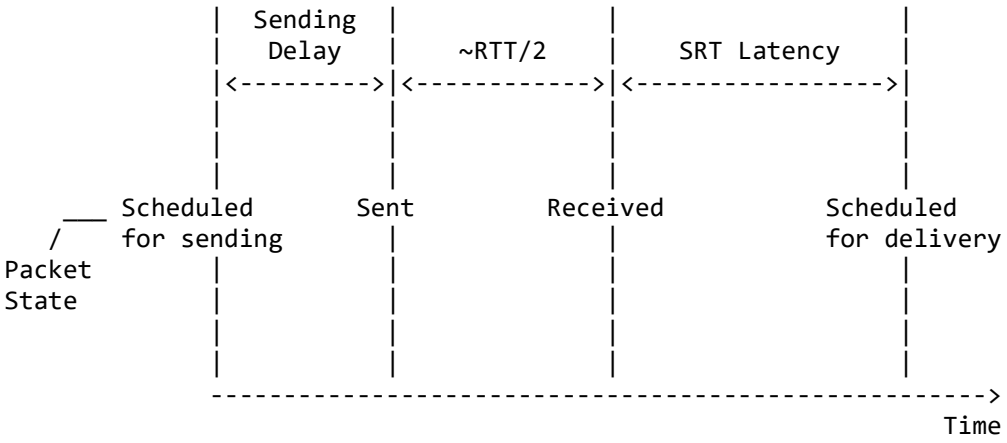


Figure 20: Key latency points during the packet transmission

The main packet states shown in Figure 20 are the following:

* "Scheduled for sending": the packet is committed by the sending application, stamped and ready to be sent;

* "Sent": the packet is passed to the UDP socket and sent;

* "Received": the packet is received and read from the UDP socket;

* "Scheduled for delivery": the packet is scheduled for the delivery and ready to be read by the receiving application.

It is worth noting that the round-trip time (RTT) of an SRT link may vary in time. However the actual end-to-end latency on the link becomes fixed and is approximately equal to (RTT_0/2 + SRT Latency) once the SRT handshake exchange happens, where RTT_0 is the actual value of the round-trip time during the SRT handshake exchange (the value of the round-trip time once the SRT connection has been established).

The value of sending delay depends on the hardware performance. Usually it is relatively small (several microseconds) in contrast to RTT_0/2 and SRT latency which are measured in milliseconds.

4.5.1. Packet Delivery Time

Packet delivery time is the moment, estimated by the receiver, when a packet should be delivered to the upstream application. The calculation of packet delivery time (PktTsbpdTime) is performed upon receiving a data packet according to the following formula:

PktTsbpdTime = TsbpdTimeBase + PKT_TIMESTAMP + TsbpdDelay + Drift

where

* TsbpdTimeBase is the time base that reflects the time difference between local clock of the receiver and the clock used by the sender to timestamp packets being sent (see Section 4.5.1.1);

* PKT_TIMESTAMP is the data packet timestamp, in microseconds;

* TsbpdDelay is the receiver's buffer delay (or receiver's buffer latency, or SRT Latency). This is the time, in milliseconds, that SRT holds a packet from the moment it has been received till the time it should be delivered to the upstream application;

* Drift is the time drift used to adjust the fluctuations between sender and receiver clock, in microseconds.

SRT Latency (TsbpdDelay) should be a buffer time large enough to cover the unexpectedly extended RTT time, and the time needed to retransmit the lost packet. The value of minimum TsbpdDelay is negotiated during the SRT handshake exchange and is equal to 120 milliseconds. The recommended value of TsbpdDelay is 3-4 times RTT.

It is worth noting that TsbpdDelay limits the number of packet retransmissions to a certain extent making it impossible to retransmit packets endlessly. This is important for the case of live streaming (Section 4.2, Section 7.1).

4.5.1.1. TSBPD Time Base Calculation

The initial value of TSBPD time base (TsbpdTimeBase) is calculated at the moment of the second handshake request is received as follows:

TsbpdTimeBase = T_NOW - HSREQ_TIMESTAMP

where T_NOW is the current time according to the receiver clock; HSREQ_TIMESTAMP is the handshake packet timestamp, in microseconds.

The value of TsbpdTimeBase is approximately equal to the initial one-way delay of the link RTT_0/2, where RTT_0 is the actual value of the round-trip time during the SRT handshake exchange.

During the transmission process, the value of TSBPD time base may be adjusted in two cases:

1. During the TSBPD wrapping period. The TSBPD wrapping period happens every 01:11:35 hours. This time corresponds to the maximum timestamp value of a packet (MAX_TIMESTAMP). MAX_TIMESTAMP is equal to 0xFFFFFFFF, or the maximum value of 32-bit unsigned integer, in microseconds (Section 3). The TSBPD wrapping period starts 30 seconds before reaching the maximum timestamp value of a packet and ends once the packet with timestamp within (30, 60) seconds interval is delivered (read from the buffer). The updated value of TsbpdTimeBase will be recalculated as follows:

   TsbpdTimeBase = TsbpdTimeBase + MAX_TIMESTAMP + 1

2. By drift tracer. See Section 4.7 for details.

4.6. Too-Late Packet Drop

The Too-Late Packet Drop (TLPKTDROP) mechanism allows the sender to drop packets that have no chance to be delivered in time, and allows the receiver to skip missing packets that have not been delivered in time. The timeout of dropping a packet is based on the TSBPD mechanism (Section 4.5).

When the TLPKTDROP mechanism is enabled, a packet is considered "too late" to be delivered and may be dropped by the sender if the packet timestamp is older than TLPKTDROP_THRESHOLD.

TLPKTDROP_THRESHOLD is related to SRT latency (Section 4.4). For the Too-Late Packet Drop mechanism to function effectively, it is recommended that a value higher than the SRT latency is used. This will allow the SRT receiver to drop missing packets first while the sender drops packets if a proper response is not received from the peer in time (e.g., due to severe congestion). The recommended threshold value is 1.25 times the SRT latency value.

Note that the SRT sender keeps packets for at least 1 second in case the latency is not high enough for a large RTT (that is, if TLPKTDROP_THRESHOLD is less than 1 second).

When enabled on the receiver, the receiver drops packets that have

not been delivered or retransmitted in time, and delivers the subsequent packets to the application when it is their time to play.

In pseudo-code, the algorithm of reading from the receiver buffer is the following:

```
<CODE BEGINS>
pos = 0;  /* Current receiver buffer position */
i = 0;    /* Position of the next available in the receiver buffer
             packet relatively to the current buffer position pos */

while(True) {
    // Get the position i of the next available packet
    // in the receiver buffer
    i = next_avail();
    // Calculate packet delivery time PktTsbpdTime
    // for the next available packet
    PktTsbpdTime = delivery_time(i);

    if T_NOW < PktTsbpdTime:
        continue;

    Drop packets which buffer position number is less than i;

    Deliver packet with the buffer position i;

    pos = i + 1;
}
<CODE ENDS>
```

where T_NOW is the current time according to the receiver clock.

When a receiver encounters the situation where the next packet to be played was not successfully received from the sender, the receiver will "skip" this packet and send a fake ACK packet (Section 4.8.1). To the sender, this fake ACK is a real ACK, and so it just behaves as if the packet had been received. This facilitates the synchronization between SRT sender and receiver. The fact that a packet was skipped remains unknown by the sender. It is recommended that skipped packets are recorded in the statistics on the SRT receiver.

The TLPKTDROP mechanism can be turned off to always ensure a clean delivery. However, a lost packet can simply pause a delivery for some longer, potentially undefined time, and cause even worse tearing for the player. Setting SRT latency higher will help much more in the event that TLPKTDROP causes packet drops too often.

4.7. Drift Management

When the sender enters "connected" status it tells the application there is a socket interface that is transmitter-ready. At this point the application can start sending data packets. It adds packets to the SRT sender's buffer at a certain input rate, from which they are transmitted to the receiver at scheduled times.

A synchronized time is required to keep proper sender/receiver buffer levels, taking into account the time zone and round-trip time (up to 2 seconds for satellite links). Considering addition/subtraction round-off, and possibly unsynchronized system times, an agreed-upon time base drifts by a few microseconds every minute. The drift may accumulate over many days to a point where the sender or receiver buffers will overflow or deplete, seriously affecting the quality of the video. SRT has a time management mechanism to compensate for this drift.

When a packet is received, SRT determines the difference between the time it was expected and its timestamp. The timestamp is calculated on the receiver side. The RTT tells the receiver how much time it was supposed to take. SRT maintains a reference between the time at the leading edge of the send buffer's latency window and the corresponding time on the receiver (the present time). This allows to convert packet timestamp to the local receiver time. Based on this time, various events (packet delivery, etc.) can be scheduled.

The receiver samples time drift data and periodically calculates a packet timestamp correction factor, which is applied to each data packet received by adjusting the inter-packet interval. When a packet is received it is not given right away to the application. As time advances, the receiver knows the expected time for any missing or dropped packet, and can use this information to fill any "holes" in the receive queue with another packet (see Section 4.5).

It is worth noting that the period of sampling time drift data is based on a number of packets rather than time duration to ensure enough samples, independently of the media stream packet rate. The effect of network jitter on the estimated time drift is attenuated by using a large number of samples. The actual time drift being very slow (affecting a stream only after many hours) does not require a fast reaction.

The receiver uses local time to be able to schedule events -- to determine, for example, if it is time to deliver a certain packet right away. The timestamps in the packets themselves are just references to the beginning of the session. When a packet is received (with a timestamp from the sender), the receiver makes a reference to the beginning of the session to recalculate its timestamp. The start time is derived from the local time at the moment that the session is connected. A packet timestamp equals "now" minus "StartTime", where the latter is the point in time when the socket was created.

4.8. Acknowledgement and Lost Packet Handling

To enable the Automatic Repeat reQuest of data packet retransmissions, a sender stores all sent data packets in its buffer.

The SRT receiver periodically sends acknowledgments (ACKs) for the received data packets so that the SRT sender can remove the acknowledged packets from its buffer (Section 4.8.1). Once the acknowledged packets are removed, their retransmission is no longer possible and presumably not needed.

Upon receiving the full acknowledgment (ACK) control packet, the SRT sender SHOULD acknowledge its reception to the receiver by sending an ACKACK control packet with the sequence number of the full ACK packet being acknowledged.

The SRT receiver also sends NAK control packets to notify the sender about the missing packets (Section 4.8.2). The sending of a NAK packet can be triggered immediately after a gap in sequence numbers of data packets is detected. In addition, a Periodic NAK report mechanism can be used to send NAK reports periodically. The NAK packet in that case will list all the packets that the receiver considers being lost up to the moment the Periodic NAK report is sent.

Upon reception of the NAK packet, the SRT sender prioritizes retransmissions of lost packets over the regular data packets to be transmitted for the first time.

The retransmission of the missing packet is repeated until the receiver acknowledges its receipt, or if both peers agree to drop this packet (Section 4.6).

4.8.1. Packet Acknowledgement (ACKs, ACKACKs)

At certain intervals (see below), the SRT receiver sends an acknowledgment (ACK) that causes the acknowledged packets to be removed from the SRT sender's buffer.

An ACK control packet contains the sequence number of the packet immediately following the latest in the list of received packets. Where no packet loss has occurred up to the packet with sequence number n, an ACK would include the sequence number (n + 1).

An ACK (from a receiver) will trigger the transmission of an ACKACK (by the sender), with almost no delay. The time it takes for an ACK to be sent and an ACKACK to be received is the RTT. The ACKACK tells the receiver to stop sending the ACK position because the sender already knows it. Otherwise, ACKs (with outdated information) would continue to be sent regularly. Similarly, if the sender does not receive an ACK, it does not stop transmitting.

There are two conditions for sending an acknowledgment. A full ACK is based on a timer of 10 milliseconds (the ACK period or synchronization time interval SYN). For high bitrate transmissions, a "light ACK" can be sent, which is an ACK for a sequence of packets. In a 10 milliseconds interval, there are often so many packets being sent and received that the ACK position on the sender does not advance quickly enough. To mitigate this, after 64 packets (even if the ACK period has not fully elapsed) the receiver sends a light ACK. A light ACK is a shorter ACK (SRT header and one 32-bit field). It does not trigger an ACKACK.

When a receiver encounters the situation where the next packet to be played was not successfully received from the sender, it will "skip" this packet (see Section 4.6) and send a fake ACK. To the sender,

this fake ACK is a real ACK, and so it just behaves as if the packet had been received. This facilitates the synchronization between SRT sender and receiver. The fact that a packet was skipped remains unknown by the sender. Skipped packets are recorded in the statistics on the SRT receiver.

4.8.2. Packet Retransmission (NAKs)

The SRT receiver sends NAK control packets to notify the sender about the missing packets. The NAK packet sending can be triggered immediately after a gap in sequence numbers of data packets is detected.

Upon reception of the NAK packet, the SRT sender prioritizes retransmissions of lost packets over the regular data packets to be transmitted for the first time.

The SRT sender maintains a list of lost packets (loss list) that is built from NAK reports. When scheduling packet transmission, it looks to see if a packet in the loss list has priority and sends it if so. Otherwise, it sends the next packet scheduled for the first transmission list. Note that when a packet is transmitted, it stays in the buffer in case it is not received by the SRT receiver.

NAK packets are processed to fill in the loss list. As the latency window advances and packets are dropped from the sending queue, a check is performed to see if any of the dropped or resent packets are in the loss list, to determine if they can be removed from there as well so that they are not retransmitted unnecessarily.

There is a counter for the packets that are resent. If there is no ACK for a packet, it will stay in the loss list and can be resent more than once. Packets in the loss list are prioritized.

If packets in the loss list continue to block the send queue, at some point this will cause the send queue to fill. When the send queue is full, the sender will begin to drop packets without even sending them the first time. An encoder (or other application) may continue to provide packets, but there's no place for them, so they will end up being thrown away.

This condition where packets are unsent does not happen often. There is a maximum number of packets held in the send buffer based on the configured latency. Older packets that have no chance to be retransmitted and played in time are dropped, making room for newer real-time packets produced by the sending application. See Section 4.5, Section 4.6 for details.

In addition to the regular NAKs, the Periodic NAK report mechanism can be used to send NAK reports periodically. The NAK packet in that case will have all the packets that the receiver considers being lost at the time of sending the Periodic NAK report.

SRT Periodic NAK reports are sent with a period of (RTT + 4 * RTTVar) / 2 (so called NAKInterval), with a 20 milliseconds floor, where RTT and RTTVar are defined in Section 4.10. A NAK control packet

contains a compressed list of the lost packets. Therefore, only lost packets are retransmitted. By using NAKInterval for the NAK reports period, it may happen that lost packets are retransmitted more than once, but it helps maintain low latency in the case where NAK packets are lost.

An ACKACK tells the receiver to stop sending the ACK position because the sender already knows it. Otherwise, ACKs (with outdated information) would continue to be sent regularly.

An ACK serves as a ping, with a corresponding ACKACK pong, to measure RTT. The time it takes for an ACK to be sent and an ACKACK to be received is the RTT. Each ACK has a number. A corresponding ACKACK has that same number. The receiver keeps a list of all ACKs in a queue to match them. Unlike a full ACK, which contains the current RTT and several other values in the Control Information Field (CIF) (Section 3.2.4), a light ACK just contains the sequence number. All control messages are sent directly and processed upon reception, but ACKACK processing time is negligible (the time this takes is included in the round-trip time).

4.9. Bidirectional Transmission Queues

Once an SRT connection is established, both peers can send data packets simultaneously.

4.10. Round-Trip Time Estimation

Round-trip time (RTT) in SRT is estimated during the transmission of data packets based on a difference in time between an ACK packet is sent out and a corresponding ACKACK packet is received back by the SRT receiver.

An ACK sent by the receiver triggers an ACKACK from the sender with minimal processing delay. The ACKACK response is expected to arrive at the receiver roughly one RTT after the corresponding ACK was sent.

The SRT receiver records the time when an ACK is sent out. The ACK carries a unique sequence number (independent of the data packet sequence number). The corresponding ACKACK also carries the same sequence number. Upon receiving the ACKACK, SRT calculates the RTT by comparing the difference between the ACKACK arrival time and the ACK departure time. In the following formula, RTT is the current value that the receiver maintains and rtt is the recent value that was just calculated from an ACK/ACKACK pair:

```
RTT = 7/8 * RTT + 1/8 * rtt
```

RTT variance (RTTVar) is obtained as follows:

```
RTTVar = 3/4 * RTTVar + 1/4 * abs(RTT - rtt)
```

where abs() means an absolute value.

Both RTT and RTTVar are measured in microseconds. The initial value of RTT is 100 milliseconds, RTTVar is 50 milliseconds.

The round-trip time (RTT) calculated by the receiver as well as the RTT variance (RTTVar) are sent with the next full acknowledgement packet (see Section 3.2.4). Note that the first ACK in an SRT session might contain an initial RTT value of 100 milliseconds, because the early calculations may not be precise.

The sender always gets the RTT from the receiver. It does not have an analog to the ACK/ACKACK mechanism, i.e. it can not send a message that guarantees an immediate return without processing. Upon an ACK reception, the SRT sender updates its own RTT and RTTVar values using the same formulas as above, in which case rtt is the most recent value it receives, i.e., carried by an incoming ACK.

Note that an SRT socket can both send and receive data packets. RTT and RTTVar are updated by the socket based on algorithms for the sender (using ACK packets) and for the receiver (using ACK/ACKACK pairs). When an SRT socket receives data, it updates its local RTT and RTTVar, which can be used for its own sender as well.

5. SRT Packet Pacing and Congestion Control

SRT provides certain mechanisms for exchanging feedback on the state of packet transmission between sender and receiver. Every 10 milliseconds the receiving side sends acknowledgement (ACK) packets (Section 3.2.4) to the sender that include the latest values of RTT, RTT variance, available buffer size, receiving rate, and estimated link capacity. Similarly, NAK packets (Section 3.2.5) from the receiver inform the sender of any packet loss during the transmission, triggering an appropriate response. These mechanisms provide a solid background for the integration of various congestion control algorithms in the SRT protocol.

As SRT is designed both for live streaming and file transmission (Section 4.2), there are two groups of congestion control algorithms defined in SRT: Live Congestion Control (LiveCC), and File Transfer Congestion Control (FileCC).

5.1. SRT Packet Pacing and Live Congestion Control (LiveCC)

To ensure smooth video playback on a receiving peer during live streaming, SRT must control the sender's buffer level to prevent overfill and depletion. The pacing control module is designed to send packets as fast as they are submitted by a video application while maintaining a relatively stable buffer level. While this looks like a simple problem, the details of the Automatic Repeat Request (ARQ) behaviour between input and output of the SRT sender add some complexity.

SRT needs a certain amount of bandwidth overhead in order to have space for the sender to insert packets for retransmission with minimum impact on the output rate of the main packet transmission.

This balance is achieved by adjusting the maximum allowed bandwidth MAX_BW (Section 5.1.1) which limits the bandwidth usage by SRT. The MAX_BW value is used by the Live Congestion Control (LiveCC) module

to calculate the minimum interval between consecutive sent packets PKT_SND_PERIOD. In principle, the space between packets determines where retransmissions can be inserted, and the overhead represents the available margin. There is an empiric calculation that defines the interval, in microseconds, between two packets to give a certain bitrate. It is a function of the average packet payload (which includes video, audio, etc.) and the configured maximum bandwidth (MAX_BW). See Section 5.1.2 for details.

In the case of live streaming, the sender is allowed to drop packets that cannot be delivered in time (Section 4.6).

The combination of pacing control and Live Congestion Control (LiveCC), based on the input rate and an overhead for packets retransmission, helps avoid congestion during fluctuations of the source bitrate.

During live streaming over highly variable networks, fairness can be achieved by controlling the bitrate of the source encoder at the input of the SRT sender. SRT sender can provide a variety of network related statistics, such as RTT estimate, packet loss level, the number of packets dropped, etc., to the encoder which can be used for making decisions and adjusting the bitrate in real time.

5.1.1. Configuring Maximum Bandwidth

There are several ways of configuring maximum bandwidth (MAX_BW):

1. MAXBW_SET mode: Set the value explicitly.

   The recommended default value is 1 Gbps. The default value is set only for live streaming.

   Note that this static setting is not well-suited to a variable input, like when you change the bitrate on an encoder. Each time the input bitrate is configured on the encoder, MAX_BW should also be reconfigured.

2. INPUTBW_SET mode: Set the SRT sender's input rate (INPUT_BW) and overhead (OVERHEAD).

   In this mode, SRT calculates the maximum bandwidth as follows:

   MAX_BW = INPUT_BW * (1 + OVERHEAD /100)

   Note that INPUTBW_SET mode reduces to the MAXBW_SET mode and the same restrictions apply.

3. INPUTBW_ESTIMATED mode: Measure the SRT sender's input rate internally and set the overhead (OVERHEAD).

   In this mode, SRT adjusts the value of maximum bandwidth each time it gets the updated estimate of the input rate EST_INPUT_BW:

   MAX_BW = EST_INPUT_BW * (1 + OVERHEAD /100)

Note that the units of MAX_BW, INPUT_BW, and EST_INPUT_BW are bytes per second. OVERHEAD is defined in %.

INPUTBW_ESTIMATED mode is recommended for setting the maximum bandwidth (MAX_BW) as it follows the fluctuations in SRT sender's input rate. However, there are certain considerations that should be taken into account.

In INPUTBW_SET mode, SRT takes as an input the rate that had been configured as the expected output rate of an encoder (in terms of bitrate for the packets including audio and overhead). But it is normal for an encoder to occasionally overshoot. At low bitrate, sometimes an encoder can be too optimistic and will output more bits than expected. Under these conditions, SRT packets would not go out fast enough because the configured bandwidth limitation would be too low.

This is mitigated by calculating the bitrate internally (INPUTBW_ESTIMATED mode). SRT examines the packets being submitted and calculates the input rate as a moving average. However, this introduces a bit of a delay based on the content. It also means that if an encoder encounters black screens or still frames, this would dramatically lower the bitrate being measured, which would in turn reduce the SRT output rate. And then, when the video picks up again, the input rate rises sharply. SRT would not start up again fast enough on output because of the time it takes to measure the speed. Packets might be accumulated in the SRT's sender buffer and delayed as a result, causing them to arrive too late at the decoder, and possible drops by the receiver.

The following table shows a summary of the bandwidth configuration modes and the variables that need to be set (v) or ignored (-):

Mode / Variable	MAX_BW	INPUT_BW	OVERHEAD
MAXBW_SET	v	-	-
INPUTBW_SET	-	v	v
INPUTBW_ESTIMATED	-	-	v

5.1.2. SRT's Default LiveCC Algorithm

The main goal of the SRT's default LiveCC algorithm is to adjust the minimum allowed packet sending period PKT_SND_PERIOD (and, as a result, the maximum allowed sending rate) during transmission based on the average packet payload size (AvgPayloadSize) and maximum bandwidth (MAX_BW).

On the sender side, there are three events that the LiveCC algorithm reacts to: (1) sending a data packet, (2) receiving an acknowledgement (ACK) packet, and (3) a timeout event as described below.

(1) On sending a data packet (either original or retransmitted), update the value of average packet payload size (AvgPayloadSize):

AvgPayloadSize = 7/8 * AvgPayloadSize + 1/8 * PacketPayloadSize

where PacketPayloadSize is the payload size of a sent data packet, in bytes; the initial value of AvgPayloadSize is equal to the maximum allowed packet payload size, which cannot be larger than 1456 bytes.

(2) On an acknowledgement (ACK) packet reception:

Step 1. Calculate SRT packet size (PktSize) as the sum of average payload size (AvgPayloadSize) and SRT header size (Section 3), in bytes.

Step 2. Calculate the minimum allowed packet sending period (PKT_SND_PERIOD) as:

PKT_SND_PERIOD = PktSize * 1000000 / MAX_BW

where MAX_BW is the configured maximum bandwidth which limits the bandwidth usage by SRT, in bytes per second; PKT_SND_PERIOD is measured in microseconds.

(3) On a retransmission timeout (RTO) event, follow the same steps as described in method (1) above.

RTO is the amount of time within which an acknowledgement is expected after a data packet is sent out. If there is no ACK after this amount of time has elapsed, a timeout event is triggered. Since SRT only acknowledges every SYN time (Section 4.8.1), the value of retransmission timeout is defined as follows:

RTO = RTT + 4 * RTTVar + 2 * SYN

where RTT is the round-trip time estimate, in microseconds, and RTTVar is the variance of RTT estimate, in microseconds, reported by the receiver and smoothed at the sender side (see Section 3.2.4, Section 4.10). Here and throughout the current section, smoothing means applying an exponentially weighted moving average (EWMA).

Continuous timeout should increase the RTO value. In SRT, a counter (RexmitCount) is used to track the number of continuous timeouts:

RTO = RexmitCount * (RTT + 4 * RTTVar + 2 * SYN) + SYN

On the receiver side, when a loss report is sent, the sending interval of periodic NAK reports (Section 4.8.2) is updated as follows:

NAKInterval = max((RTT + 4 * RTTVar) / 2, 20000)

where RTT and RTTVar are receiver's estimates (see Section 3.2.4, Section 4.10). The minimum value of NAKInterval is set to 20 milliseconds in order to avoid sending periodic NAK reports too often under low latency conditions.

5.2. File Transfer Congestion Control (FileCC)

For file transfer (Section 4.2), any known congestion control

algorithm like CUBIC [RFC8312] or BBR [BBR] can be applied, including SRT's default FileCC algorithm described below.

5.2.1. SRT's Default FileCC Algorithm

SRT's default FileCC algorithm is a modified version of the UDT native congestion control algorithm [GuAnAO], [GHG04b] designed for a bulk data transfer over networks with a large bandwidth-delay product (BDP). It is a hybrid Additive Increase Multiplicative Decrease (AIMD) algorithm, hence it adjusts both congestion window size (CWND_SIZE) and packet sending period (PKT_SND_PERIOD). The units of measurement for CWND_SIZE and PKT_SND_PERIOD are packets and microseconds, respectively.

The algorithm controls sending rate by tuning the packet sending period (i.e. how often packets are sent out). The sending rate is increased upon receipt of an acknowledgement (ACK), and decreased when receiving a loss report (negative acknowledgement, or NAK). Only full ACKs, not light ACKs (Section 4.8.1), trigger an increase in the sending rate.

SRT congestion control has two phases: "Slow Start" and "Congestion Avoidance". In the slow start phase the congestion control module probes the network to determine available bandwidth and the target sending rate for the next (operational) phase, which is congestion avoidance. In this phase, if there is no congestion detected via loss reports, the sending rate is gradually increased. Conversely, if a network congestion is detected, the algorithm decreases the sending rate to reduce subsequent packet loss. The slow start phase runs exactly once at the beginning of a connection, and stops when a packet loss occurs, when the congestion window size reaches its maximum value, or on a timeout event.

The detailed algorithm behaviour at both phases is described in Section 5.2.1.1 and Section 5.2.1.2, respectively.

As with LiveCC, SRT's default FileCC algorithm reacts to three events: (1) sending a data packet, (2) receiving an acknowledgement (ACK) packet, and (3) a timeout event. These are described below as they apply to the congestion control phases.

5.2.1.1. Slow Start

During the slow start phase, the packet sending period PKT_SND_PERIOD is kept at 1 microsecond in order to send packets as fast as possible, but not at an infinite rate. The initial value of the congestion window size (CWND_SIZE) is set to 16 packets. CWND_SIZE has an upper threshold, which is the maximum allowed congestion window size (MAX_CWND_SIZE), so that even if there is no packet loss, the slow start phase has to stop at a certain point. The threshold can be set to the maximum receiver buffer size (12 MB).

(1) On an acknowledgement (ACK) packet reception:

Step 1. If the interval since the last time the sending rate was either increased or kept (LastRCTime) is less than RC_INTERVAL:

```
a.  Keep the sending rate at the same level;

b.  Stop.

<CODE BEGINS>
if (currTime - LastRCTime < RC_INTERVAL)
{
    Keep the sending rate at the same level;
    Stop;
}
<CODE ENDS>

where currTime is the current time, in microseconds; LastRCTime is
the last time the sending rate was either increased, or kept, in
microseconds.

Step 2.  Update the value of LastRCTime to the current time:

LastRCTime = currTime

Step 3.  The size of congestion window CWND_SIZE is increased by the
difference in sequence numbers of the data packet being acknowledged
ACK_SEQNO and the last acknowledged data packet LAST_ACK_SEQNO:

CWND_SIZE += ACK_SEQNO - LAST_ACK_SEQNO

Step 4.  The sequence number of the last acknowledged data packet
LAST_ACK_SEQNO is updated as follows:

LAST_ACK_SEQNO = ACK_SEQNO

Step 5.  If the congestion window size CWND_SIZE calculated at Step 3
is greater than the upper threshold MAX_CWND_SIZE, slow start phase
ends.  Set the packet sending period PKT_SND_PERIOD as follows:

<CODE BEGINS>
if (RECEIVING_RATE > 0)
    PKT_SND_PERIOD = 1000000 / RECEIVING_RATE;
else
    PKT_SND_PERIOD = CWND_SIZE / (RTT + RC_INTERVAL);
<CODE ENDS>

where

*  RECEIVING_RATE is the rate at which packets are being received, in
   packets per second, reported by the receiver and smoothed at the
   sender side (see Section 3.2.4, Section 5.2.1.3);

*  RTT is the round-trip time estimate, in microseconds, reported by
   the receiver and smoothed at the sender side (see Section 3.2.4,
   Section 4.10);

*  RC_INTERVAL is the fixed rate control interval, in microseconds.
   RC_INTERVAL of SRT is SYN, or synchronization time interval, which
   is 0.01 second.  An ACK in SRT is sent every fixed time interval.
```

```
      The maximum and default ACK time interval is SYN.  See
      Section 4.8.1 for details.

   (2) On a loss report (NAK) packet reception:

   *  Slow start phase ends;

   *  Set the packet sending period PKT_SND_PERIOD as described in Step
      5 of section (1) above.

   (3) On a retransmission timeout (RTO) event:

   *  Slow start phase ends;

   *  Set the packet sending period PKT_SND_PERIOD as described in Step
      5 of section (1) above.
```

5.2.1.2. Congestion Avoidance

```
   Once the slow start phase ends, the algorithm enters the congestion
   avoidance phase and behaves as described below.

   (1) On an acknowledgement (ACK) packet reception:

   Step 1.  If the interval since the last time the sending rate was
   either increased or kept (LastRCTime) is less than RC_INTERVAL:

   a.  Keep the sending rate at the same level;

   b.  Stop.

   <CODE BEGINS>
   if (currTime - LastRCTime < RC_INTERVAL)
   {
       Keep the sending rate at the same level;
       Stop;
   }
   <CODE ENDS>

   where currTime is the current time, in microseconds; LastRCTime is
   the last time the sending rate was either increased, or kept, in
   microseconds.

   Step 2.  Update the value of LastRCTime to the current time:

   LastRCTime = currTime

   Step 3.  Set the congestion window size to:

   CWND_SIZE = RECEIVING_RATE * (RTT + RC_INTERVAL) / 1000000 + 16

   Step 4.  If there is packet loss reported by the receiver
   (bLoss=True):

   a.  Keep the value of PKT_SND_PERIOD at the same level;
```

```
b.  Set the value of bLoss to False;

c.  Stop.

bLoss flag is equal to True if a packet loss has happened since the
last sending rate increase.  Initial value: False.

Step 5.  If there is no packet loss reported by the receiver
(bLoss=False), calculate PKT_SND_PERIOD as follows:

<CODE BEGINS>
inc = 0;

lossBandwidth = 2 * (1000000 / LastDecPeriod);
linkCapacity = min(lossBandwidth, EST_LINK_CAPACITY);
B = linkCapacity - 1000000 / PKT_SND_PERIOD;

if ((PKT_SND_PERIOD > LastDecPeriod) && ((linkCapacity / 9) < B))
    B = linkCapacity / 9;
if (B <= 0)
    inc = 1 / S;
else
{
    inc = pow(10.0, ceil(log10(B * S * 8))) * 0.0000015 / S;
    inc = max(inc, 1 / S);
}

PKT_SND_PERIOD = (PKT_SND_PERIOD * RC_INTERVAL) /
                  (PKT_SND_PERIOD * inc + RC_INTERVAL);
<CODE ENDS>

where

*  LastDecPeriod is the value of PKT_SND_PERIOD right before the last
   sending rate decrease has happened (on a loss report (NAK) packet
   reception), in microseconds.  The initial value of LastDecPeriod
   is set to 1 microsecond;

*  EST_LINK_CAPACITY is the estimated link capacity reported by the
   receiver within an ACK packet and smoothed at the sender side
   (Section 5.2.1.3), in packets per second;

*  B is the estimated available bandwidth, in packets per second;

*  S is the SRT packet size (in terms of IP payload) in bytes.  SRT
   treats 1500 bytes as a standard packet size.

A detailed explanation of the formulas used to calculate the increase
in sending rate can be found in [GuAnAO].  UDT's available bandwidth
estimation has been modified to take into account the bandwidth
registered at the moment of packet loss, since the estimated link
capacity reported by the receiver may overestimate the actual link
capacity significantly.

Step 6.  If the value of maximum bandwidth MAX_BW defined in
Section 5.1 is set, limit the value of PKT_SND_PERIOD to the minimum
```

```
allowed period, if necessary:

<CODE BEGINS>
if (MAX_BW)
    MIN_PERIOD = 1000000 / (MAX_BW / S);

    if (PKT_SND_PERIOD < MIN_PERIOD)
        PKT_SND_PERIOD = MIN_PERIOD;
<CODE ENDS>

Note that in the case of file transmission the the maximum allowed
bandwidth (MAX_BW) for SRT can be defined.  This limits the minimum
possible interval between packets sent.  Only the usage of MAXBW_SET
mode is possible (Section 5.1.1).  In contrast with live streaming,
there is no default value set for MAX_BW, and the transmission rate
is not limited if not set explicitly.

(2) On a loss report (NAK) packet reception:

Step 1.  Set the value of flag bLoss equal to True.

Step 2.  If the current loss ratio estimated by the sender is less
than 2%:

a.  Keep the sending rate at the same level;

b.  Update the value of LastDecPeriod:

LastDecPeriod = PKT_SND_PERIOD

c.  Stop.

This modification has been introduced to increase the algorithm
tolerance to a random packet loss specific for public networks, but
not related to the absence of available bandwidth.

Step 3.  If sequence number of a packet being reported as lost is
greater than the largest sequence number has been sent so far
(LastDecSeq), i.e. this NAK starts a new congestion period:

a.  Set the value of LastDecPeriod to the current packet sending
period PKT_SND_PERIOD;

b.  Increase the value of packet sending period:

PKT_SND_PERIOD = 1.03 * PKT_SND_PERIOD

c.  Update AvgNAKNum:

AvgNAKNum = 0.97 * AvgNAKNum + 0.03 * NAKCount

d.  Reset NAKCount and DecCount values to 1;

e.  Record the current largest sent sequence number LastDecSeq;

f.  Compute DecRandom to a random (uniform distribution) number
```

between 1 and AvgNAKNum. If DecRandom < 1: DecRandom = 1;

g. Stop;

where

* AvgNAKNum is the average number of NAKs during a congestion period. Initial value: 0;

* NAKCount is the number of NAKs received so far in the current congestion period. Initial value: 0;

* DecCount means the number of times that the sending rate has been decreased during the congestion period. Initial value: 0;

* DecRandom is a random number used to decide if the rate should be decreased or not for the following NAKs (not the first one) during the congestion period. DecRandom is a random number between 1 and the average number of NAKs per congestion period (AvgNAKNum).

Congestion period is defined as the time between two NAKs in which the biggest lost packet sequence number carried in the NAK is greater than the LastDecSeq.

The coefficients used in the formulas above have been slightly modified to reduce the amount by which the sending rate decreases.

Step 4. If DecCount <= 5, and NAKCount == DecCount * DecRandom:

a. Update SND period: SND = 1.03 * SND;

b. Increase DecCount and NAKCount by 1;

c. Record the current largest sent sequence number (LastDecSeq).

5.2.1.3. Link Capacity and Receiving Rate Estimation

Estimates of link capacity and receiving rate, in packets/bytes per second, are calculated at the receiver side during file transmission (Section 4.2). It is worth noting that the receiving rate estimate, while available during the entire data transmission period, is used only during the slow start phase of the congestion control algorithm (Section 5.2.1.1). The latest estimate obtained before the end of the slow start period is used by the sender as a reference maximum speed to continue data transmission without further congestion. Link capacity is estimated all the time and used primarily (as well as packet loss ratio and other protocol statistics) for sending rate adjustments during the transmission process.

As each data packet arrives, the receiver records the time delta with respect to the arrival of the previous data packet, which is used to estimate bandwidth and receiving speed (delivery rate). This and other control information is communicated to the sender by means of acknowledgment (ACK) packets sent every 10 milliseconds. At the sender side, upon receiving a new value, an exponentially weighted moving average (EWMA) is applied to update the latest estimate

maintained at the sender side.

It is important to note that for bandwidth estimation only data probing packets are taken into account, while all data packets (both data and data probing) are used for estimating receiving speed. Data probing refers to the use of the packet pairs technique, whereby pairs of probing packets are sent to a server back-to-back, thus making it possible to measure the minimum interval in receiving consecutive packets.

The detailed description of models used to estimate link capacity and receiving rate can be found in [GuAnAO], [GHG04b].

6. Encryption

This section describes the encryption mechanism that protects the payload of SRT streams. Based on standard cryptographic algorithms, the mechanism allows an efficient stream cipher with a key establishment method.

6.1. Overview

SRT implements encryption using AES [AES] in counter mode (AES-CTR) [SP800-38A] with a short-lived key to encrypt and decrypt the media stream. The AES-CTR cipher is suitable for continuous stream encryption that permits decryption from any point, without access to start of the stream (random access), and for the same reason tolerates packet loss. It also offers strong confidentiality when the counter is managed properly.

6.1.1. Encryption Scope

SRT encrypts only the payload of SRT data packets (Section 3.1), while the header is left unencrypted. The unencrypted header contains the Packet Sequence Number field used to keep the synchronization of the cipher counter between the encrypting sender and the decrypting receiver. No constraints apply to the payload of SRT data packets as no padding of the payload is required by counter mode ciphers.

6.1.2. AES Counter

The counter for AES-CTR is the size of the cipher's block, i.e. 128 bits. It is derived from a 128-bit sequence consisting of

* a block counter in the least significant 16 bits which counts the blocks in a packet;

* a packet index, based on the packet sequence number in the SRT header, in the next 32 bits;

* eighty zeroed bits.

The upper 112 bits of this sequence are XORed with an Initialization Vector (IV) to produce a unique counter for each crypto block. The IV is derived from the Salt provided in the Keying Material

(Section 3.2.2):

IV = MSB(112, Salt): Most significant 112 bits of the salt.

6.1.3. Stream Encrypting Key (SEK)

The key used for AES-CTR encryption is called the "Stream Encrypting Key" (SEK). It is used for up to 2^25 packets with further rekeying. The short-lived SEK is generated by the sender using a pseudo-random number generator (PRNG), and transmitted within the stream, wrapped with another longer-term key, the Key Encrypting Key (KEK), using a known AES key wrap protocol.

For connection-oriented transport such as SRT, there is no need to periodically transmit the short-lived key since no additional party can join a stream in progress. The keying material is transmitted within the connection handshake packets, and for a short period when rekeying occurs.

6.1.4. Key Encrypting Key (KEK)

The Key Encrypting Key (KEK) is derived from a secret (passphrase) shared between the sender and the receiver. The KEK provides access to the Stream Encrypting Key, which in turn provides access to the protected payload of SRT data packets. The KEK has to be at least as long as the SEK.

The KEK is generated by a password-based key generation function (PBKDF2) [RFC8018], using the passphrase, a number of iterations (2048), a keyed-hash (HMAC-SHA1) [RFC2104], and a key length value (KLen). The PBKDF2 function hashes the passphrase to make a long string, by repetition or padding. The number of iterations is based on how much time can be given to the process without it becoming disruptive.

6.1.5. Key Material Exchange

The KEK is used to generate a wrap [RFC3394] that is put in a key material (KM) message by the initiator of a connection (i.e. caller in caller-listener handshake and initiator in the rendezvous handshake, see Section 4.3) to send to the responder (listener). The KM message contains the key length, the salt (one of the arguments provided to the PBKDF2 function), the protocol being used (e.g. AES-256) and the AES counter (which will eventually change, see Section 6.1.6).

On the other side, the responder attempts to decode the wrap to obtain the Stream Encrypting Key. In the protocol for the wrap there is a padding, which is a known template, so the responder knows from the KM that it has the right KEK to decode the SEK. The SEK (generated and transmitted by the initiator) is random, and cannot be known in advance. The KEK formula is calculated on both sides, with the difference that the responder gets the key length (KLen) from the initiator via the key material (KM). It is the initiator who decides on the configured length. The responder obtains it from the material sent by the initiator.

The responder returns the same KM message to show that it has the same information as the initiator, and that the encoded material will be decrypted. If the responder does not return this status, this means that it does not have the SEK. All incoming encrypted packets received by the responder will be lost (undecrypted). Even if they are transmitted successfully, the receiver will be unable to decrypt them, and so packets will be dropped. All data packets coming from responder will be unencrypted.

6.1.6. KM Refresh

The short lived SEK is regenerated for cryptographic reasons when a pre-determined number of packets has been encrypted. The KM refresh period is determined by the implementation. The receiver knows which SEK (odd or even) was used to encrypt the packet by means of the KK field of the SRT Data Packet (Section 3.1).

There are two variables used to determine the KM Refresh timing:

* KM Refresh Period specifies the number of packets to be sent before switching to the new SEK.

* KM Pre-Announcement Period specifies when a new key is announced in a number of packets before key switchover. The same value is used to determine when to decommission the old key after switchover.

The recommended KM Refresh Period is after 2^25 packets encrypted with the same SEK are sent. The recommended KM Pre-Announcement Period is 4000 packets (i.e. a new key is generated, wrapped, and sent at 2^25 minus 4000 packets; the old key is decommissioned at 2^25 plus 4000 packets).

Even and odd keys are alternated during transmission the following way. The packets with the earlier key #1 (let it be the odd key) will continue to be sent. The receiver will receive the new key #2 (even), then decrypt and unwrap it. The receiver will reply to the sender if it is able to understand. Once the sender gets to the 2^25th packet using the odd key (key #1), it will then start to send packets with the even key (key #2), knowing that the receiver has what it needs to decrypt them. This happens transparently, from one packet to the next. At 2^25 plus 4000 packets the first key will be decommissioned automatically.

Both keys live in parallel for two times the Pre-Announcement Period (e.g. 4000 packets before the key switch, and 4000 packets after). This is to allow for packet retransmission. It is possible for packets with the older key to arrive at the receiver a bit late. Each packet contains a description of which key it requires, so the receiver will still have the ability to decrypt it.

6.2. Encryption Process

6.2.1. Generating the Stream Encrypting Key

On the sending side SEK, Salt and KEK are generated in the following way:

```
SEK  = PRNG(KLen)
Salt = PRNG(128)
KEK = PBKDF2(passphrase, LSB(64,Salt), Iter, KLen)
```

where

* PBKDF2 is the PKCS#5 Password Based Key Derivation Function [RFC8018];

* passphrase is the pre-shared passphrase;

* Salt is a field of the KM message;

* LSB(n, v) is the function taking n least significant bits of v;

* Iter=2048 defines the number of iterations for PBKDF2;

* KLen is a field of the KM message.

```
Wrap = AESkw(KEK, SEK)
```

where AESkw(KEK, SEK) is the key wrapping function [RFC3394].

6.2.2. Encrypting the Payload

The encryption of the payload of the SRT data packet is done with AES-CTR

```
EncryptedPayload = AES_CTR_Encrypt(SEK, IV, UnencryptedPayload)
```

where the Initialization Vector (IV) is derived as

```
IV = (MSB(112, Salt) << 2) XOR (PktSeqNo)
```

PktSeqNo is the value of the Packet Sequence Number field of the SRT data packet.

6.3. Decryption Process

6.3.1. Restoring the Stream Encrypting Key

For the receiver to be able to decrypt the incoming stream it has to know the stream encrypting key (SEK) used by the sender. The receiver MUST know the passphrase used by the sender. The remaining information can be extracted from the Keying Material message.

The Keying Material message contains the AES-wrapped [RFC3394] SEK used by the encoder. The Key-Encryption Key (KEK) required to unwrap the SEK is calculated as:

```
KEK = PBKDF2(passphrase, LSB(64,Salt), Iter, KLen)
```

where

* PBKDF2 is the PKCS#5 Password Based Key Derivation Function [RFC8018];

* passphrase is the pre-shared passphrase;

* Salt is a field of the KM message;

* LSB(n, v) is the function taking n least significant bits of v;

* Iter=2048 defines the number of iterations for PBKDF2;

* KLen is a field of the KM message.

SEK = AESkuw(KEK, Wrap)

where AESkuw(KEK, Wrap) is the key unwrapping function.

6.3.2. Decrypting the Payload

The decryption of the payload of the SRT data packet is done with AES-CTR

DecryptedPayload = AES_CTR_Encrypt(SEK, IV, EncryptedPayload)

where the Initialization Vector (IV) is derived as

IV = (MSB(112, Salt) << 2) XOR (PktSeqNo)

PktSeqNo is the value of the Packet Sequence Number field of the SRT data packet.

7. Best Practices and Configuration Tips for Data Transmission via SRT

7.1. Live Streaming

This section describes real world examples of live audio/video streaming and the current consensus on maintaining the compatibility between SRT implementations by different vendors. It is meant as guidance for developers to write applications compatible with existing SRT implementations.

The term "live streaming" refers to MPEG-TS style continuous data transmission with latency management. Live streaming based on segmentation and transmission of files like in HLS protocol [RFC8216] is not part of this use case.

The default SRT data transmission mode for continuous live streaming is message mode (Section 4.2.1) with certain settings applied as described below:

* Only data packets with their Packet Position Flag (PP) field set to "11b" are allowed, meaning a single data packet forms exactly one message (Section 3.1).

* Timestamp-Based Packet Delivery (TSBPD) (Section 4.5) and Too-Late

Packet Drop (TLPKTDROP) (Section 4.6) mechanisms must be enabled.

* Live Congestion Control (LiveCC) (Section 5.1) must be used.

* Periodic NAK reports (Section 4.8.2) must be enabled.

* The Order Flag (Section 3.1) needs special attention. In the case of live streaming, it is set to 0 allowing out of order delivery of a packet. However, in this use case the Order Flag has to be ignored by the receiver. As TSBPD is enabled, the receiver will still deliver packets in order, but based on the timestamps. In the case of a packet arriving too late and skipped by the TLPKTDROP mechanism, the order of delivery is still maintained except for potential sequence discontinuity.

  This method has grown historically and is the current common standard for live streaming across different SRT implementations. A change or variation of the settings will break compatibility between two parties.

This combination of settings allows live streaming with a constant latency (Section 4.4). The receiving end will not "fall behind" in time by waiting for missing packets. However, data integrity might not be ensured if packets or retransmitted packets do not arrive within the expected time frame. Audio or video interruption can occur, but the overall latency is maintained and does not increase over time whenever packets are missing.

7.2. File Transmission

This section describes the use case of file transmission and provides configuration examples.

The usage of both message and buffer modes (Section 4.2) is possible in this case. For both modes, Timestamp-Based Packet Delivery (TSBPD) (Section 4.5) and Too-Late Packet Drop (TLPKTDROP) (Section 4.6) mechanisms must be turned off, while File Transfer Congestion Control (FileCC) (Section 5.2) must be enabled.

When TSBPD is disabled, each packet gets timestamped with the time it is sent by the SRT sender. A packet being sent for the first time will have a timestamp different from that of a corresponding retransmitted packet. In contrast to the live streaming case, the timing of packets' delivery, when sending files, is not critical. The most important thing is data integrity. Therefore the TLPKTDROP mechanism must be disabled in this case. No data is allowed to be dropped, because this will result in corrupted files with missing data. The retransmission of missing packets has to happen until the packets are finally acknowledged by the SRT receiver.

The File Transfer Congestion Control (FileCC) mechanism will take care of using the available link bandwidth for maximum transfer speed.

7.2.1. File Transmission in Buffer Mode

The original UDT protocol [GHG04b] used buffer mode (Section 4.2.2) to send files, and the same is possible in SRT. This mode was designed to transmit one file per connection. For a single file transmission, a socket is opened, a file is transmitted, and then the socket is closed. This procedure is repeated for each subsequent single file, as the receiver cannot distinguish between two files in a continuous data stream.

Buffer mode is not suitable for the transmission of many small files since for every file a new connection has to be established. To initiate a new connection, at least two round-trip times (RTTs) for the handshake exchange are required (Section 4.3).

It is also important to note that the SRT protocol does not add any information to the data being transmitted. The file name or any auxiliary information can be declared separately by the sending application, e.g., in the form of a Stream ID Extension Message (Section 3.2.1.3).

7.2.2. File Transmission in Message Mode

If message mode (Section 4.2.1) is used for the file transmission, the application should either segment the file into several messages, or use one message per file. The size of an individual message plays an important role on the receiving side since the size of the receiver buffer should be large enough to store at least a single message entirely.

In the case of file transfer in message mode, the file name, segmentation rules, or any auxiliary information can be specified separately by both sending and receiving applications. The SRT protocol does not provide a specific way of doing this. It could be done by setting the file name, etc., in the very first message of a message sequence, followed by the file itself.

When designing an application for SRT file transfer, it is also important to be aware of the delivery order of the received messages. This can be set by the Order Flag as described in Section 3.1.

8. Security Considerations

SRT provides confidentiality of the payload using stream cipher and a pre-shared private key as specified in Section 6. The security can be compromised if the pre-shared passphrase is known to the attacker.

On the protocol control level, SRT does not encrypt packet headers. Therefore it has some vulnerabilities similar to TCP [RFC6528]:

* A peer tells a counterpart its public IP during the handshake that is visible to any attacker.

* An attacker may potentially count the number of SRT processes behind a Network Address Translator (NAT) by establishing multiple SRT connections and tracking the ranges of SRT Socket IDs. If a random Socket ID is generated for the first connection, subsequent connections may get consecutive SRT Socket IDs. Assuming one

system runs only one SRT process, for example, then an attacker can estimate the number of systems behind a NAT.

* Similarly, the possibility of attack depends on the implementation of the initial sequence number (ISN) generation. If an ISN is not generated randomly for each connection, an attacker may potentially count the number of systems behind a Network Address Translator (NAT) by establishing a number of SRT connections and identifying the number of different sequence number "spaces", given that no SRT packet headers are encrypted.

* An eavesdropper can hijack existing connections only if it steals the IP and port of one of the parties. If some stream addresses an existing SRT receiver by its SRT socket ID, IP, and port number, but arrives from a different IP or port, the SRT receiver ignores it.

* SRT has a certain protection from DoS attacks, see Section 4.3.

There are some important considerations regarding the encryption feature of SRT:

* The SEK must be changed at an appropriate refresh interval to avoid the risk associated with the use of security keys over a long period of time.

* The shared secret for KEK generation must be carefully configured by a security officer responsible for security policies, enforcing encryption, and limiting key size selection.

9. IANA Considerations

This document makes no requests of the IANA.

Contributors

This specification is based on the SRT Protocol Technical Overview [SRTTO] written by Jean Dube and Steve Matthews.

In alphabetical order, the contributors to the pre-IETF SRT project and specification at Haivision are: Marc Cymontkowski, Roman Diouskine, Jean Dube, Mikolaj Malecki, Steve Matthews, Maria Sharabayko, Maxim Sharabayko, Adam Yellen.

The contributors to this specification at SK Telecom are Jeongseok Kim and Joonwoong Kim.

It is worth acknowledging also the contribution of the following people in this document: Justus Rogmann.

We cannot list all the contributors to the open-sourced implementation of SRT on GitHub. But we appreciate the help, contribution, integrations and feedback of the SRT and SRT Alliance community.

Acknowledgments

The basis of the SRT protocol and its implementation was the UDP-based Data Transfer Protocol [GHG04b]. The authors thank Yunhong Gu and Robert Grossman, the authors of the UDP-based Data Transfer Protocol [GHG04b].

TODO acknowledge.

References

Normative References

[GHG04b] Gu, Y., Hong, X., and R.L. Grossman, "Experiences in Design and Implementation of a High Performance Transport Protocol", DOI 10.1109/SC.2004.24, December 2004, <https://doi.org/10.1109/SC.2004.24>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Informative References

[AES] National Institute of Standards and Technology, "FIPS Pub 197: Advanced Encryption Standard (AES)", November 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[AV1] Rivaz, P.d. and J. Haughton, "AV1 Bitstream & Decoding Process Specification", September 2021, <https://aomediacodec.github.io/av1-spec/av1-spec.pdf>.

[BBR] Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., and V. Jacobson, "BBR: Congestion-Based Congestion Control", ACM Queue, vol. 14 , October 2016.

[GuAnAO] Gu, Y., Hong, X., and R.L. Grossman, "An Analysis of AIMD Algorithm with Decreasing Increases", Proceedings of the 1st International Workshop on Networks for Grid Applications (GridNets '04) , October 2004.

[H.265] International Telecommunications Union, "H.265 : High efficiency video coding", ITU-T Recommendation H.265, 2019.

[I-D.ietf-quic-http] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <https://www.ietf.org/archive/id/draft-ietf-quic-http-34.txt>.

[ISO13818-1]
ISO, "Information technology - Generic coding of moving pictures and associated audio information: Systems", ISO/IEC 13818-1, September 2021.

[ISO23009] ISO, "Information technology - Dynamic adaptive streaming over HTTP (DASH)", ISO/IEC 23009:2019, September 2021.

[PNPID] "PNP ID AND ACPI ID REGISTRY", September 2021, <https://uefi.org/PNP_ACPI_Registry>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007, <https://www.rfc-editor.org/info/rfc4987>.

[RFC6528] Gont, F. and S. Bellovin, "Defending against Sequence Number Attacks", RFC 6528, DOI 10.17487/RFC6528, February 2012, <https://www.rfc-editor.org/info/rfc6528>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming", RFC 8216, DOI 10.17487/RFC8216, August 2017, <https://www.rfc-editor.org/info/rfc8216>.

[RFC8312] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/info/rfc8312>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RTMP] "Real-Time Messaging Protocol", September 2021, <https://www.adobe.com/devnet/rtmp.html>.

[SP800-38A] Dworkin, M., "Recommendation for Block Cipher Modes of Operation", December 2001.

[SRTSRC] "SRT fully functional reference implementation", September 2021, <https://github.com/Haivision/srt>.

[SRTTO] Dube, J. and S. Matthews, "SRT Protocol Technical Overview", December 2019.

[VP9] WebM, "VP9 Video Codec", September 2021, <https://www.webmproject.org/vp9>.

Appendix A. Packet Sequence List Coding

For any single packet sequence number, it uses the original sequence number in the field. The first bit MUST start with "0".

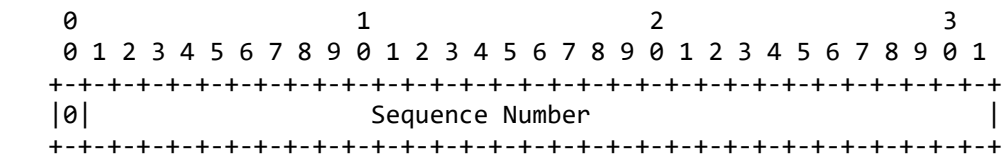


Figure 21: Single sequence numbers coding

For any consecutive packet sequence numbers that the difference between the last and first is more than 1, only record the first (a) and the the last (b) sequence numbers in the list field, and modify the the first bit of a to "1".

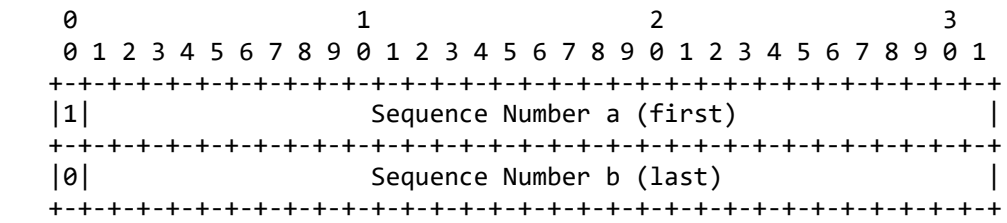


Figure 22: Range of sequence numbers coding

Appendix B. SRT Access Control

One type of information that can be interchanged when a connection is being established in SRT is the Stream ID, which can be used in a caller-listener connection layout. This is a string of maximum 512 characters set on the caller side. It can be retrieved at the listener side on the newly accepted connection.

SRT listener can notify an upstream application about the connection attempt when a HS conclusion arrives, exposing the contents of the Stream ID extension message. Based on this information, the

application can accept or reject the connection, select the desired data stream, or set an appropriate passphrase for the connection.

The Stream ID value can be used as free-form, but there is a recommended convention so that all SRT users speak the same language. The intent of the convention is to:

* promote readability and consistency among free-form names,

* interpret some typical data in the key-value style.

B.1. General Syntax

This recommended syntax starts with the characters known as an executable specification in POSIX: "#!".

The next character defines the format used for the following key-value pair syntax. At the moment, there is only one supported syntax identified by ":" and described below.

Everything that comes after a syntax identifier is further referenced as the content of the Stream ID.

The content starts with a ":" or "{" character identifying its format:

<spanx style="verb">:</spanx> comma-separated key-value pairs with no nesting,

<spanx style="verb">{</spanx> a nested block with one or several key-value pairs that must end with a "}" character. Nesting means that multiple level brace-enclosed parts are allowed.

The form of the key-value pair is

key1=value1,key2=value2,...

B.2. Standard Keys

Beside the general syntax, there are several top-level keys treated as standard keys. All single letter key definitions, including those not listed in this section, are reserved for future use. Users can additionally use custom key definitions with user_* or companyname_* prefixes, where user and companyname are to be replaced with an actual user or company name.

The existing key values MUST NOT be extended, and MUST NOT differ from those described in this section.

The following keys are standard:

* u: User Name, or authorization name, that is expected to control which password should be used for the connection. The application should interpret it to distinguish which user should be used by the listener party to set up the password.

* r: Resource Name identifies the name of the resource and facilitates selection should the listener party be able to serve multiple resources.

* h: Host Name identifies the hostname of the resource. For example, to request a stream with the URI somehost.com/videos/querry.php?vid=366 the hostname field should have somehost.com, and the resource name can have videos/querry.php?vid=366 or simply 366. Note that this is still a key to be specified explicitly. Support tools that apply simplifications and URI extraction are expected to insert only the host portion of the URI here.

* s: Session ID is a temporary resource identifier negotiated with the server, used just for verification. This is a one-shot identifier, invalidated after the first use. The expected usage is when details for the resource and authorization are negotiated over a separate connection first, and then the session ID is used here alone.

* t: Type specifies the purpose of the connection. Several standard types are defined:

  - stream (default, if not specified): for exchanging the user-specified payload for an application-defined purpose,

  - file: for transmitting a file where r is the filename,

  - auth: for exchanging sensible data. The r value states its purpose. No specific possible values for that are known so far (for future use).

* m: Mode expected for this connection:

  - request (default): the caller wants to receive the stream data,

  - publish: the caller wants to send the stream data,

  - bidirectional: bidirectional data exchange is expected.

Note that "m" is not required in the case where Stream ID is not used to distinguish authorization or resources, and the caller is expected to send the data. This is only for cases where the listener can handle various purposes of the connection and is therefore required to know what the caller is attempting to do.

B.3. Examples

The example content of the Stream ID is the following:

```
#!::u=admin,r=bluesbrothers1_hi
```

It specifies the username and the resource name of the stream to be served to the caller.

The next example specifies that the file is expected to be transmitted from the caller to the listener and its name is

```
   results.csv:

   #!::u=johnny,t=file,m=publish,r=results.csv
```

Appendix C. Changelog

C.1. Since draft-sharabayko-mops-srt-00

* Improved and extended the description of "Encryption" section.

* Improved and extended the description of "Round-Trip Time Estimation" section.

* Extended the description of "Handshake" section with "Stream ID Extension Message", "Group Membership Extension" subsections.

* Extended "Handshake Messages" section with the detailed description of handshake procedure.

* Improved "Key Material" section description.

* Changed packet structure formatting for "Packet Structure" section.

* Did minor additions to the "Acknowledgement and Lost Packet Handling" section.

* Fixed broken links.

* Extended the list of references.

C.2. Since draft-sharabayko-mops-srt-01

* Extended "Congestion Control" section with the detailed description of SRT packet pacing for both live streaming and file transmission cases.

* Improved "Group Membership Extension" section.

* Reworked "Security Consideration" section.

* Added missing control packets: Drop Request, Peer Error, Congestion Warning.

* Improved "Data Transmission Modes" section as well as added "Best Practices and Configuration Tips for Data Transmission via SRT" section describing the use cases of live streaming and file transmission via SRT.

* Changed the workgroup from "MOPS" to "Network Working Group".

* Changed the intended status of the document from "Standards Track" to "Informational".

* Overall corrections throughout the document: fixed lists, punctuation, etc.

C.3. Since draft-sharabayko-srt-00

* Message Drop Request control packet: added note about possible zero-valued message number.

* Corrected an error in the formula for NAKInterval: changed min to max.

* Added a note in "Best Practices and Configuration Tips for Data Transmission via SRT" section that Periodic NAK reports must be enabled in the case of live streaming.

* Introduced the value of TLPKTDROP_THRESHOLD for Too-Late Packet Drop mechanism.

* Improved the description of general syntax for SRT Access Control.

* Updated the list of contributors.

* Overall corrections throughout the document.

Authors' Addresses

Maxim Sharabayko
Haivision Network Video, GmbH

Email: maxsharabayko@haivision.com

Maria Sharabayko
Haivision Network Video, GmbH

Email: msharabayko@haivision.com

Jean Dube
Haivision Systems, Inc.

Email: jdube@haivision.com

Jeongseok Kim
SK Telecom Co., Ltd.

Email: jeongseok.kim@sk.com

Joonwoong Kim
SK Telecom Co., Ltd.

Email: joonwoong.kim@sk.com