Secure Reliable Transport (SRT) Protocol

Edición 2024

Agenda

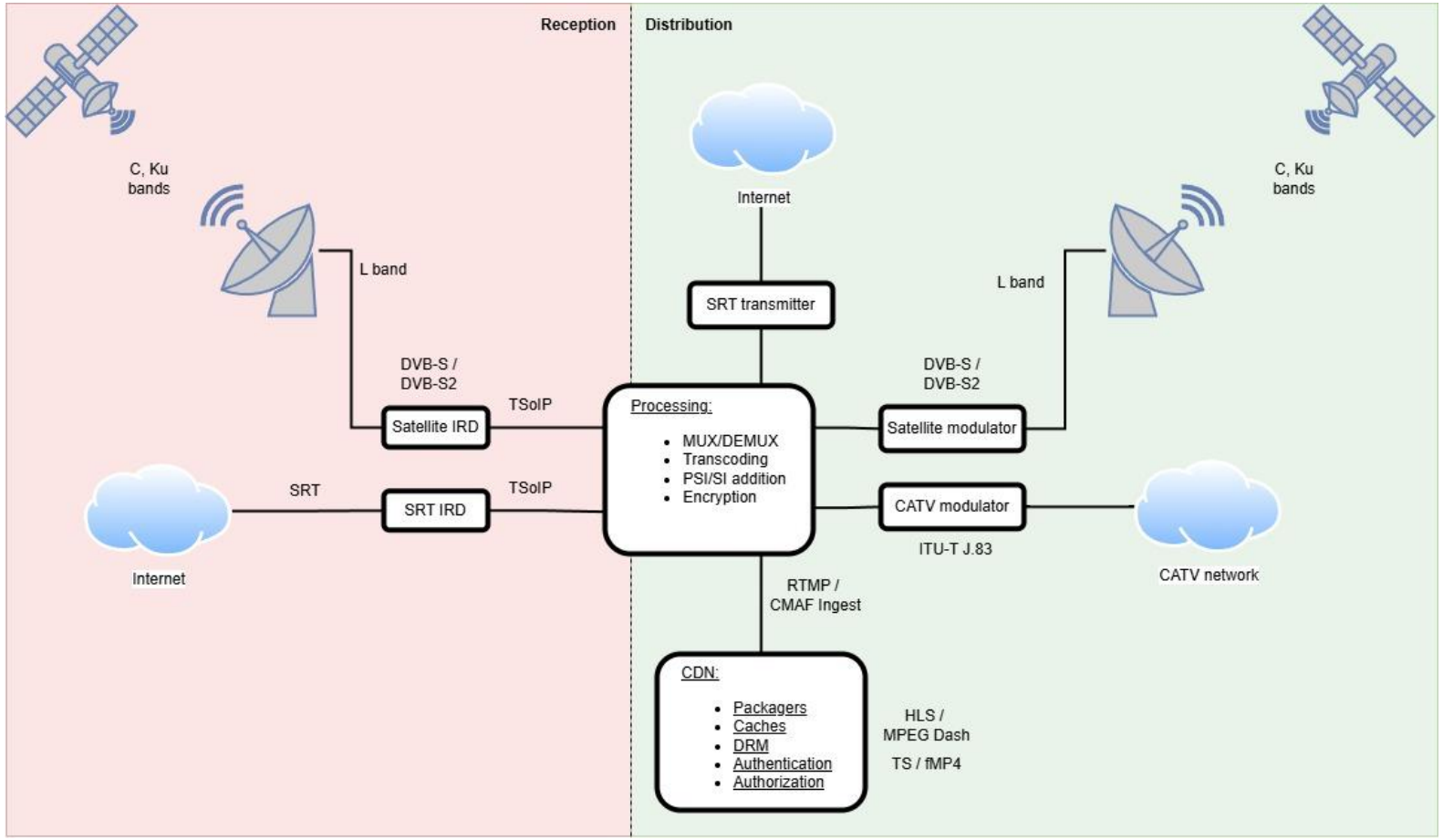
Clase 1: Teórico

- Introducción y motivación
- Estructura y tipos de paquetes
- Modos de conexión y transmisión
- Métodos de control

Clase 2: Práctico

- Repaso de teórico
- Demo de transmisión con SRT
- Análisis de paquetes

Contexto


Reception
Distribution
C, Ku bands
L band
DVB-S / DVB-S2
TSoIP
Satellite IRD
SRT
SRT IRD
TSoIP
Internet
Processing:
MUX/DEMUX
Transcoding
PSI/SI addition
Encryption
Internet
SRT transmitter
DVB-S / DVB-S2
Satellite modulator
L band
C, Ku bands
CATV modulator
ITU-T J.83
CATV network
RTMP / CMAF Ingest
CDN:
Packagers
Caches
DRM
Authentication
Authorization
HLS / MPEG Dash
TS / fMP4

Motivación

¿Qué es SRT?

SRT es un protocolo de **transporte de video** diseñado en base a algunas características:

- **Baja latencia** en la entrega de video
- Transmisión **fiable** del contenido
- Adaptación a las **condiciones de la red**
- **Seguridad** (AES)

SRT se enfoca en la **transmisión desde el origen, no en la entrega** al usuario final.

Motivación

¿Por qué SRT?

¿Por qué SRT y no HAS?

¿Por qué SRT y no RTMP?

Motivación

¿Por qué SRT?

¿Por qué SRT y no HAS?

- Mejora latencia para real time
- El destinatario no requiere adaptar calidad
- Uso de internet pública en vez de CDN

¿Por qué SRT y no RTMP?

Motivación

¿Por qué SRT?

¿Por qué SRT y no HAS?

- Mejora latencia para real time
- El destinatario no requiere adaptar calidad
- Uso de internet pública en vez de CDN

¿Por qué SRT y no RTMP?

- UDP en lugar de TCP
  - Control se hace en capa 5
  - No se deriva a TCP
- Mejora latencia
- Más soporte de codecs

Motivación

¿Cuándo SRT?

Cámaras de noticieros o en eventos deportivos	Transmisión de señales a operadores	Video en eventos corporativos
Cámaras de streamers	Ingestas con transcoding en la nube	Otros

El protocolo

Transmisiones de **video de baja latencia** en redes IP locales confiables suelen usar **MPEG-TS con UDP/RTP** y corrección de errores (FEC).

Alcanzar la misma baja latencia en **largas distancias** entre diferentes ciudades, países o continentes **es más difícil y costoso** (satélites o MPLS dedicado).

Las redes públicas de **Internet** requieren **más ancho de banda para la recuperación de pérdida de paquetes**, pero con retransmisión selectiva (UDP fiable) SRT elimina estas limitaciones.

El protocolo

SRT introduce mejoras como **control de flujo**, **control de congestión** y **cifrado**.

Ideal para transmisión de video de **baja latencia (*sub-second*)** y mejor uso del ancho de banda comparado con RTMP.

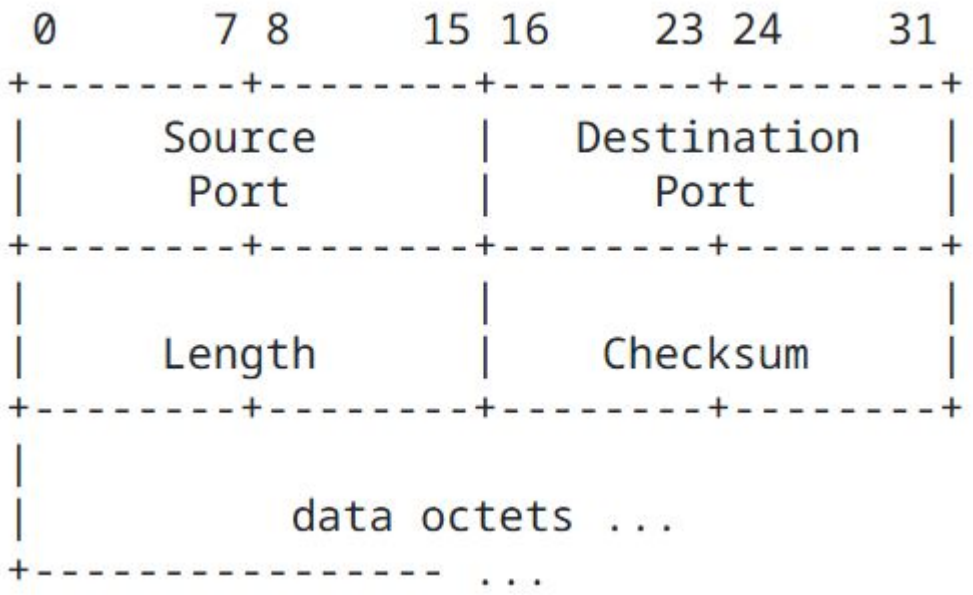
Detecta condiciones de la red en tiempo real, compensando el jitter y las fluctuaciones de ancho de banda, minimizando la pérdida de paquetes.

Mantiene la **latencia constante** y reduce la necesidad de buffers grandes.

Modelo bidireccional de llamador/escucha con multiplexación de conexiones sobre un mismo puerto UDP.

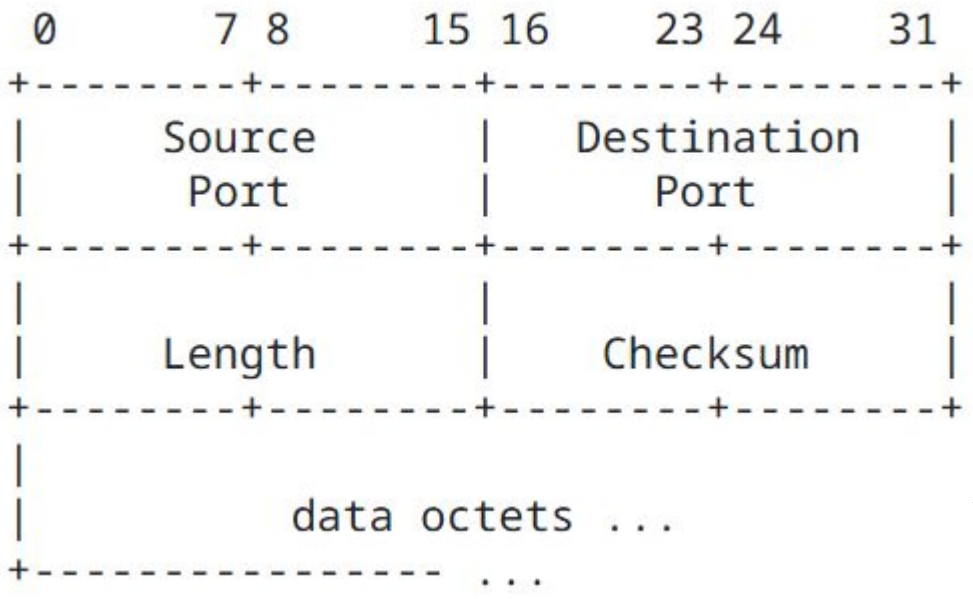
Soporte para FEC, retransmisión selectiva (ARQ), y cifrado AES, adaptándose a diferentes casos de uso según latencia o fiabilidad.

Estructura de un paquete UDP



RFC-768: User Datagram Protocol

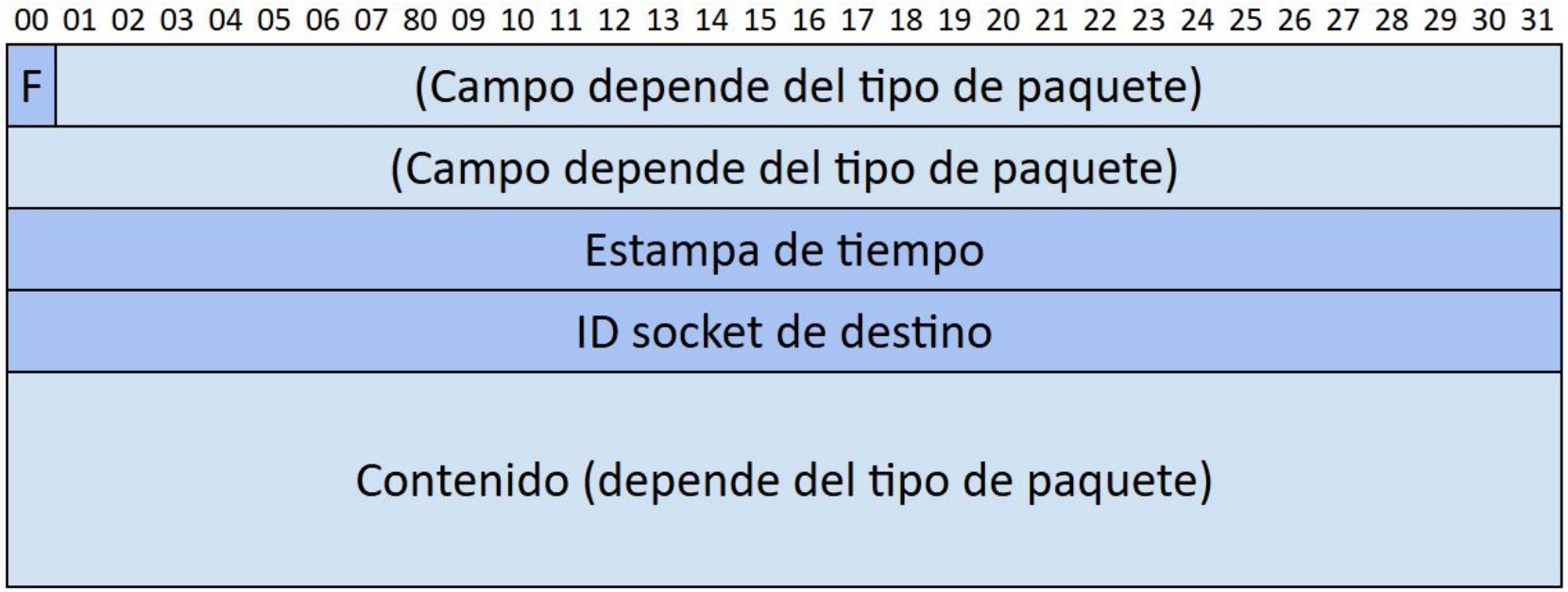
Estructura de un paquete UDP



Paquete SRT

RFC-768: User Datagram Protocol

Estructura de un paquete SRT



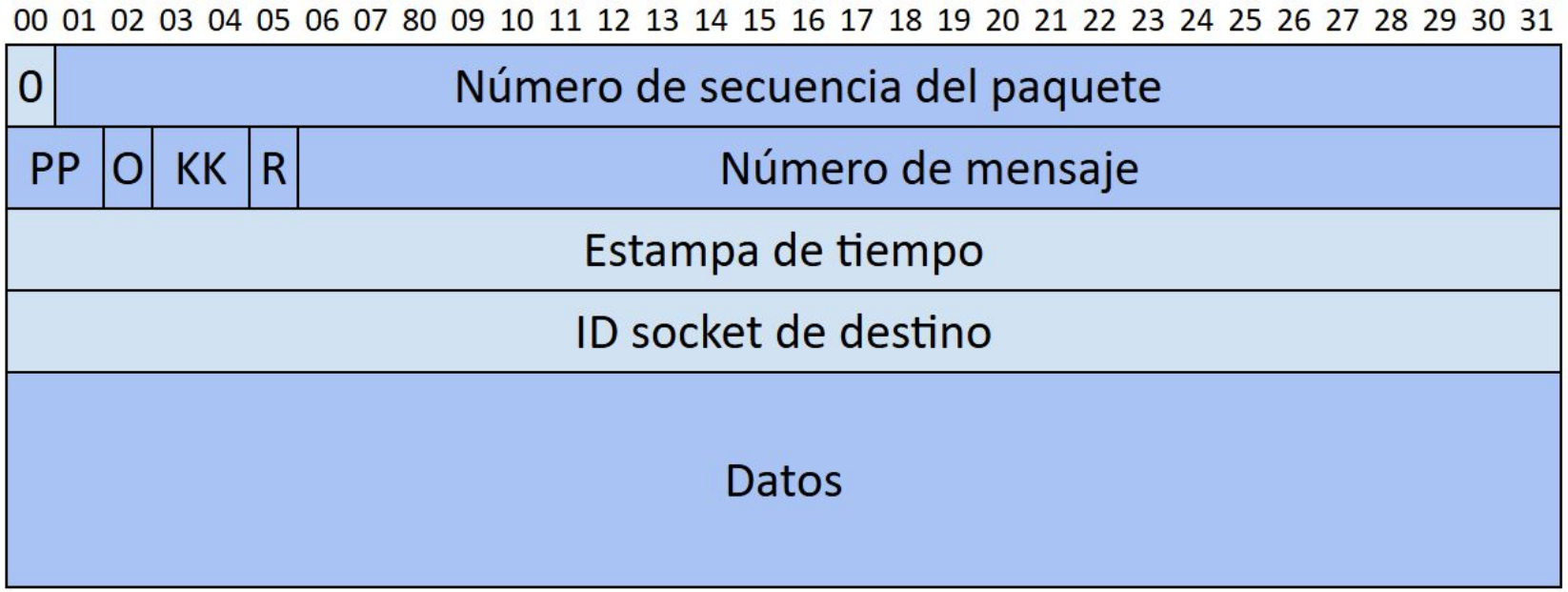
F: 1 bit. Flag de tipo de paquete

- 0: Paquete de datos
- 1: Paquete de control

Estampa de tiempo: 32 bits. En microsegundos, puede indicar tiempo de envío o de origen, según modo de transmisión

ID socket de destino: 32 bits. Vale “0” en connection request

Paquetes de datos



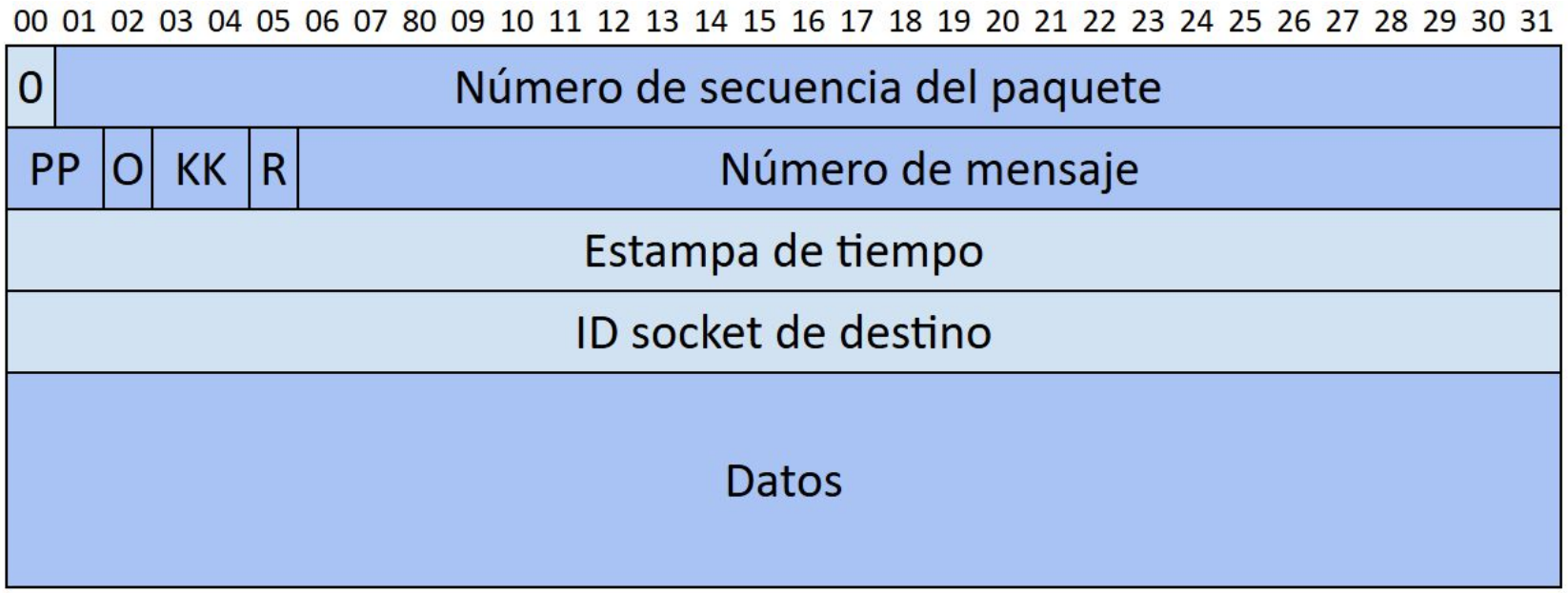
Número de secuencia del paquete: 31 bits.

PP: 2 bits. Packet position flag, indica si el paquete está al principio, al final, o en el medio del mensaje, o si es el único.

O: 1 bit. Order flag, indica si importa el orden del mensaje.

R: 1 bit. Retransmitted flag, indica si es retransmitido.

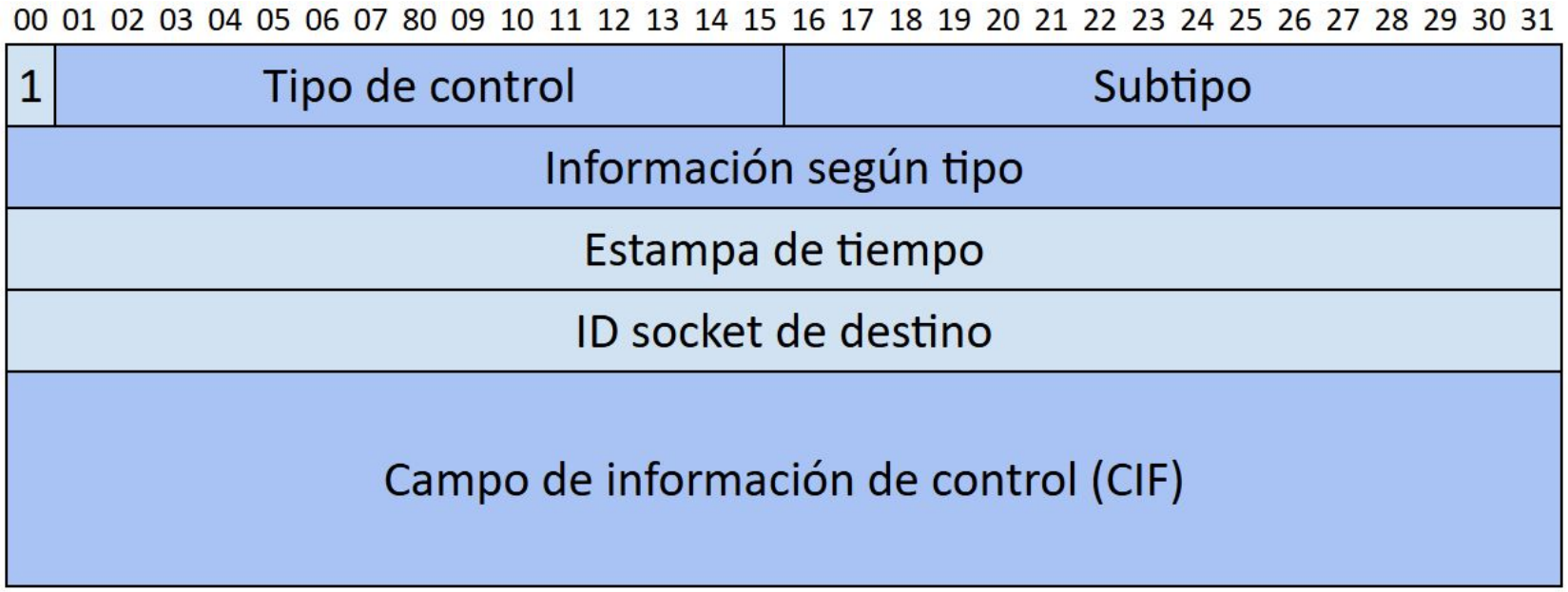
Paquetes de datos



Número de mensaje: 26 bits. Secuencia del mensaje al que pertenece el paquete.

Datos: largo variable. Contiene el payload de datos (en caso del video, el TS). El largo máximo es el permitido por el paquete UDP.

Paquetes de control



Tipo de control: 15 bits. Definiciones más adelante.

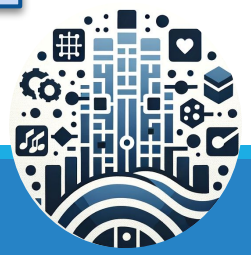
Subtipo: 16 bits. Subtipo para paquetes específicos.

Información de tipo: 32 bit. Uso de este campo depende del tipo.

CIF: largo variable. Uso de este campo depende del tipo.

Tipos de paquetes de control

HANDSHAKE	KEEPALIVE	ACK
NAK (Loss Report)	Congestion Warning	SHUTDOWN
ACKACK	DROPREQ	PEERERROR



Paquetes de control tipo *HANDSHAKE*

Versión: actualmente 4 o 5.

Encriptado: algoritmo y tamaño de clave. Por defecto es AES-128.

Extensión: Relativo al tipo de handshake. Es 0 a excepción de algunos casos.

Secuencia inicial: Valor del 1er paquete.

MTU: Típicamente 1500 para Ethernet, pero puede ser menor.

Ventana: Máximo número de paquetes que pueden estar “en viaje”.

Paquetes de control tipo *HANDSHAKE*

Tipo de Handshake: Puede tener los valores “DONE”, “AGREEMENT”, “CONCLUSION”, “WAVEHAND” o “INDUCTION”. Dependen del modo.

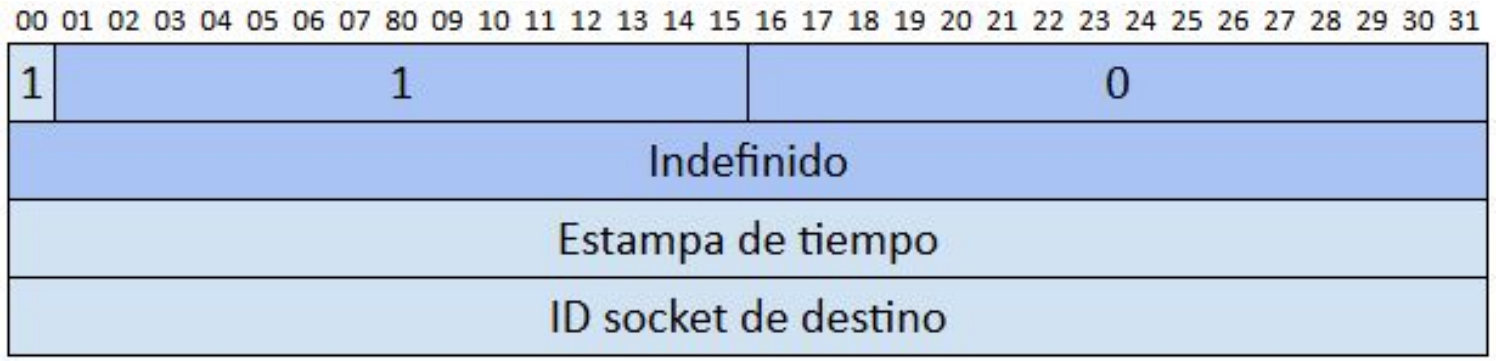
ID socket e **IP**: dirección de origen.

SYN Cookie: valor aleatorio para procesar el handshake.

Tipo de extensión: HS, KM o CONFIG.

Largo de extensión: tamaño del campo de contenido de extensión.

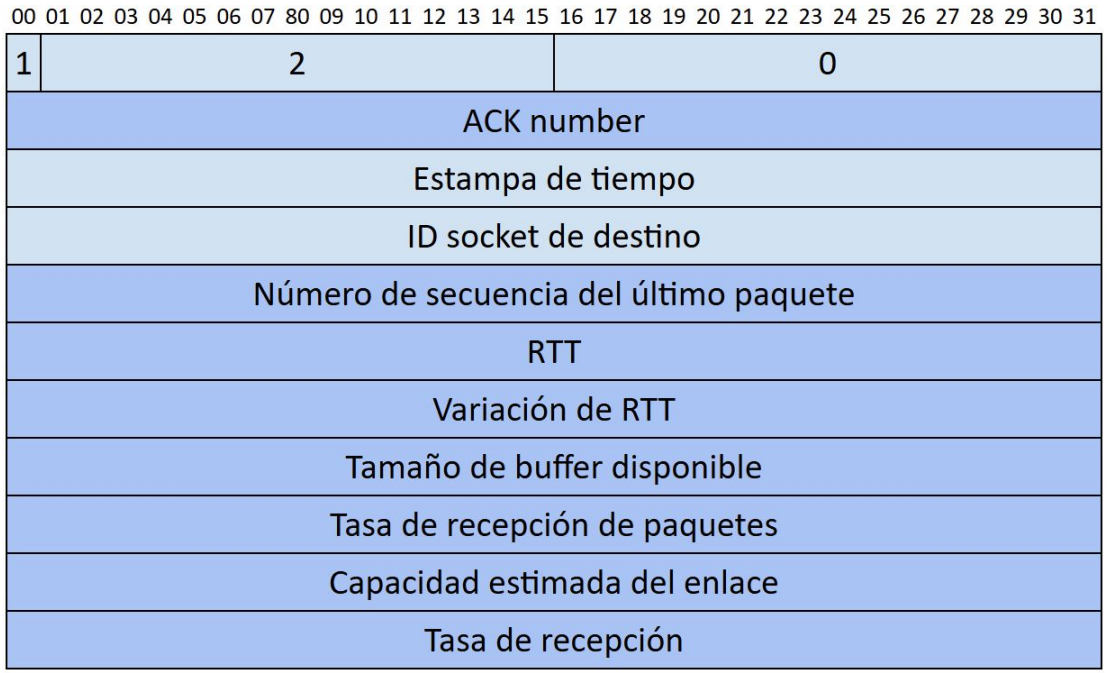
Paquetes de control tipo *KEEP-ALIVE*



Solo incluye los campos básicos del encabezado de control, sin información en el campo CIF

Packet type se envía en 1

Paquetes de control tipo *ACK/NAK*



Paquete ACK

Paquete NAK

Modos de transmisión de datos

Modo se define con una flag del campo de extensión del paquete de HANDSHAKE. Hay dos modos:

Message Mode	Buffer Mode
Agnóstico del contenido	Agnóstico del contenido
Real-time o non-real-time	Non-real-time
Un mensaje puede ser formado con uno o varios paquetes SRT consecutivos	Usado para abrir una conexión, enviar un único archivo de datos y cerrar la conexión
Live Mode: sub-set de message mode para transmitir en vivo	
Live Mode tiene manejo de latencia	

SRT: orientado a conexión

SRT usa los conceptos de **conexión** y **sesión** para resolver en capa 5 lo que RTMP deja a TCP

Conexión definida por tres partes:

1. Se establece por el proceso de handshake
2. Se mantiene la sesión mientras se intercambian paquetes
3. Se cierra por pedido de una parte o por cierto tiempo sin transmisión

Existen dos configuraciones para el establecimiento de la sesión:

Caller-Listener	**Rendezvous**
Una parte espera que la otra inicie la conexión	Ambas partes intentan iniciar la conexión

Handshake Caller-Listener

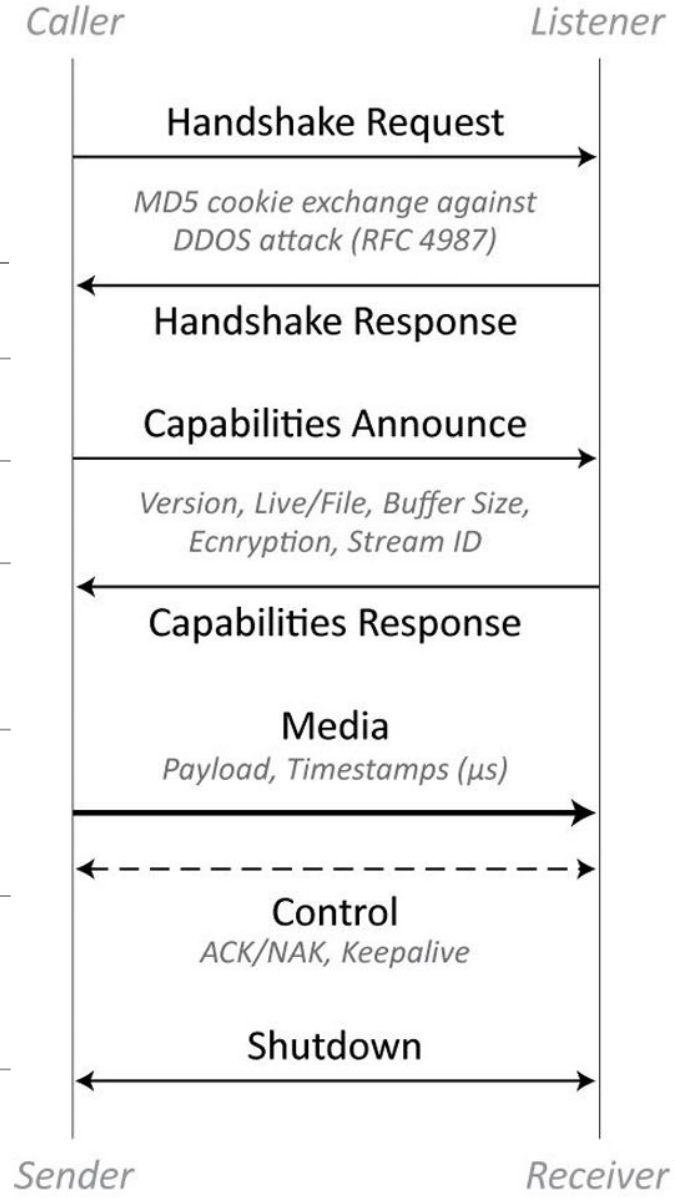
El **Caller es quien inicia la conexión** con un handshake tipo INDUCTION.

El **Listener responde** el pedido inicial, negociando versión de HS.

El Caller valida la versión y negocia parámetros de la comunicación, como latencia, tamaños de buffer, encriptado y Stream ID.

Si las partes proponen valores distintos de latencia, se establecer el tiempo que sea mayor.

El Caller puede ser tanto Sender como Receiver. Es decir, la sesión puede ser iniciada tanto por quien envía el video como quien lo pide.

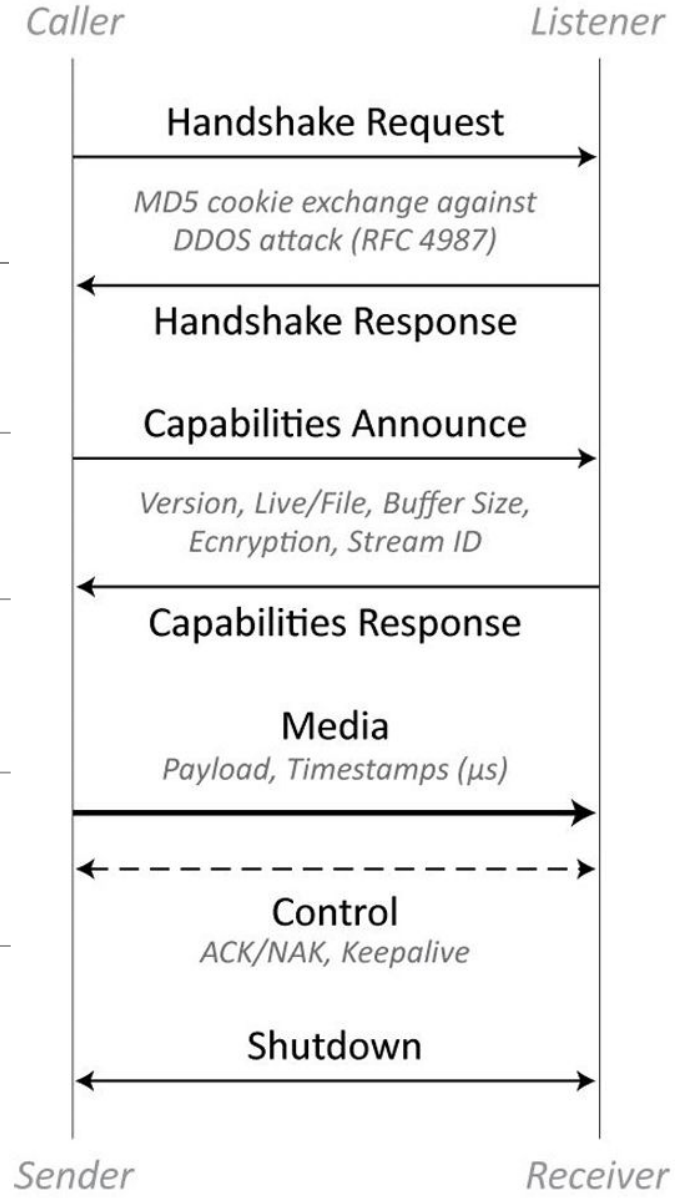


Handshake Caller-Listener

Luego de confirmar los parámetros, se comienza la transmisión de paquetes de datos, típicamente video.

Se intercambian paquetes ACK y NAK para control de transmisión, así como Keepalive para control de la sesión.

Para cerrar la sesión se envía un Handshake de tipo CONCLUSION, a lo que la otra parte responde con otro paquete del mismo tipo.

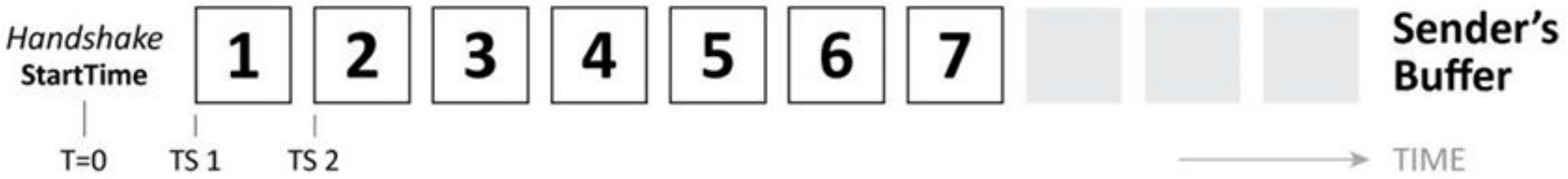


SRT Buffer Latency

El Sender de SRT mantiene un buffer para **almacenar temporalmente** los paquetes enviados en caso de que alguno requiera ser retransmitido.

Si la recepción del ACK/NAK tiene una latencia mayor al buffer del Sender, ya no va a tener el paquete cuando tenga que transmitirlo.

Por eso la ventana del buffer tiene que ser, al menos, la latencia esperada en la red.



Idea de funcionamiento

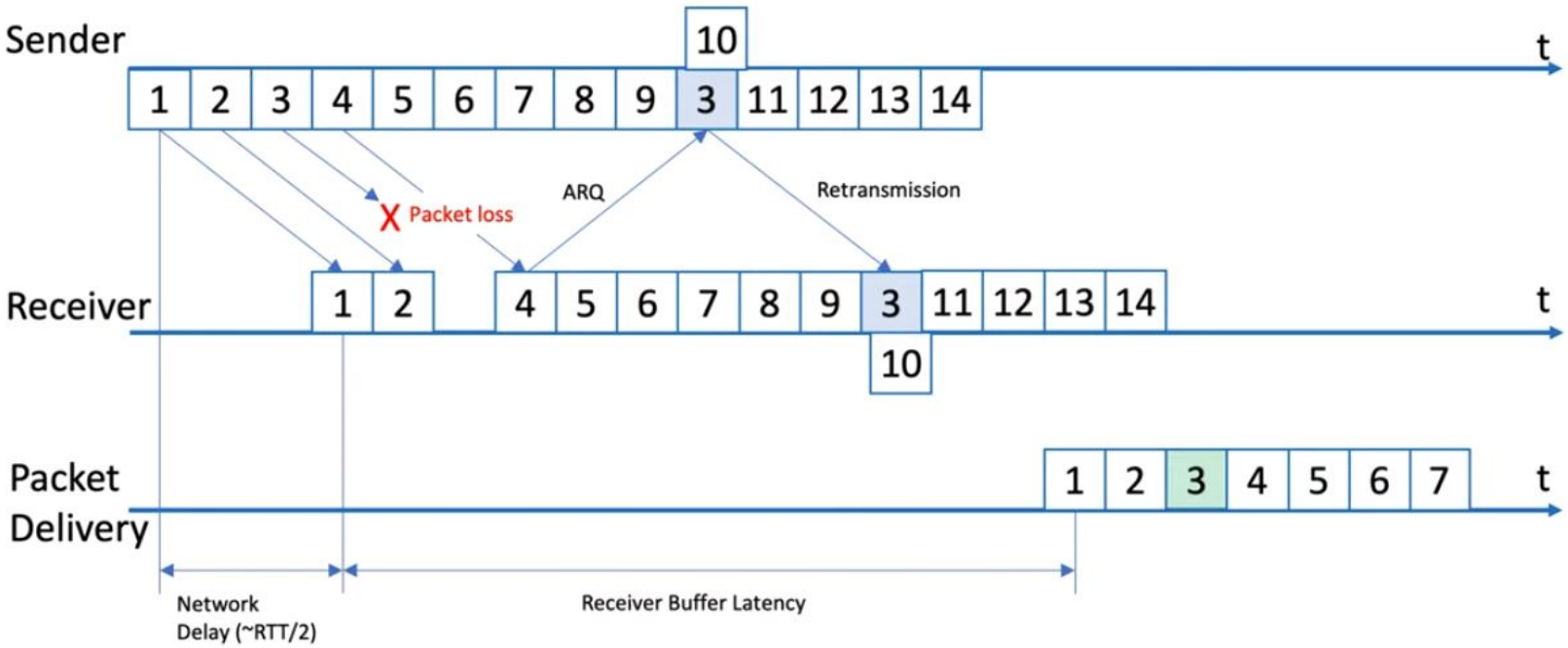
Tomado de Alex Converse, SF Video Technology [Online]. Disponible: https://www.youtube.com/watch?v=m1FABQtAjgU&t=369s

Timestamp-Based Packet Delivery

Objetivo: reproducir la salida en el timing propuesto a la entrada

Para esto, **el tiempo de delivery a la aplicación no es igual al tiempo de recepción**, porque no siempre podrían reproducirse

El receptor usa el timestamp del encabezado para **retener los paquetes** con un delay que compensa variaciones



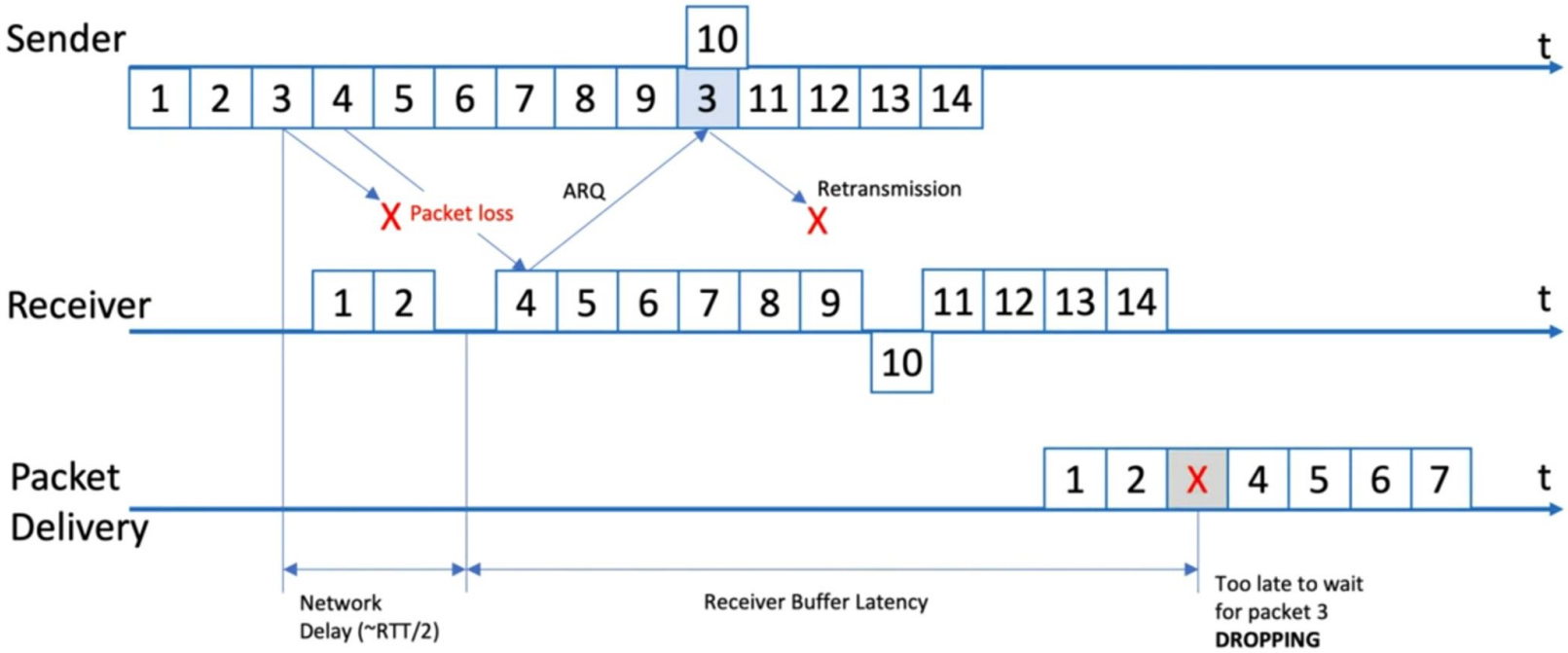
Too-Late Packet Drop

Mecanismo para complementar el TSBPD

Si se alcanza el tiempo de delivery pero todavía no se recibió la retransmisión del paquete, SRT contempla que se **pueda descartar ese paquete**

Se ejecuta cuando el timestamp del paquete se atrasa un 125% de la latencia

Se anulan demás reintentos y se hace el delivery



Drift Management

Se requiere **sincronismo entre transmisor y receptor** para mantener buffers consistentes, entre ellos y con la latencia, por hacerlo en base a timestamps.

SRT contempla un mecanismo para compensar las variaciones que se acumulan entren ambos.

Cuando se recibe un paquete, el receptor SRT calcula la diferencia entre el tiempo esperado del paquete y el timestamp. El RTT determina lo que debería demorar el envío, con lo cual SRT mantiene una referencia con respecto al transmisor.

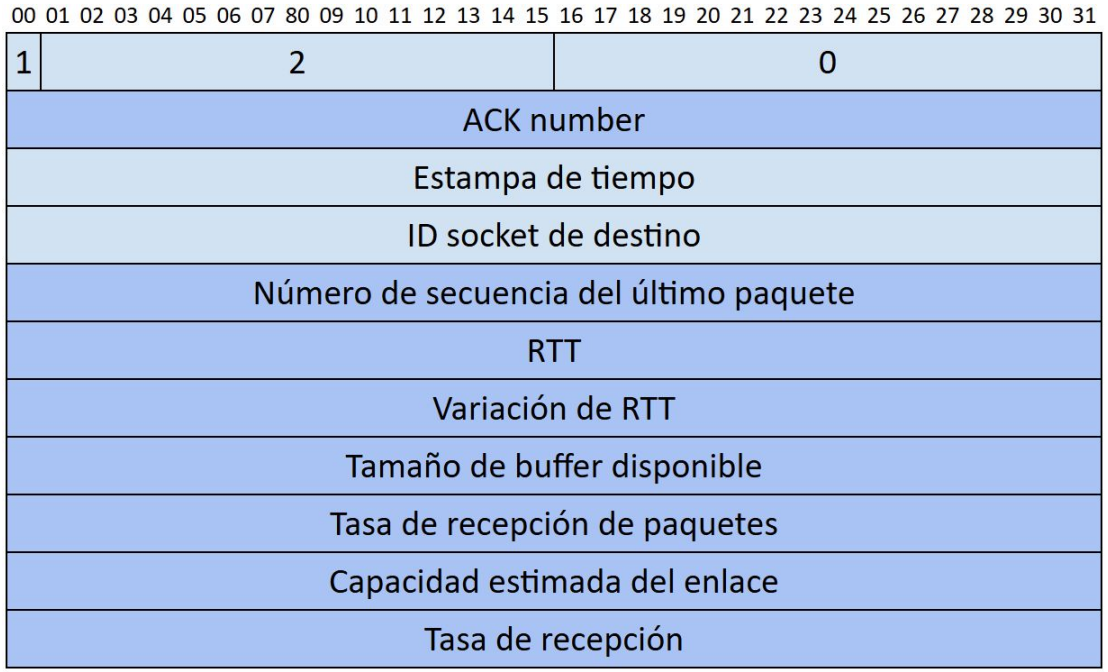
El receptor calcula periódicamente un factor de corrección que se aplica a cada paquete e interpreta los huecos que debe llenar antes de entregarlo a la aplicación aguas arriba.

Recuperación de paquetes

SRT define dos modos de recuperación de información ante pérdidas en la red:

Automatic Repeat reQuest (ARQ)	**Forward Error Correction (FEC)**
Reenvío de información ante NAK	Información adicional para corrección
Más eficiente ante menos pérdidas	Agrega overhead pero corrige más rápido

Acknowledgement



Paquete ACK

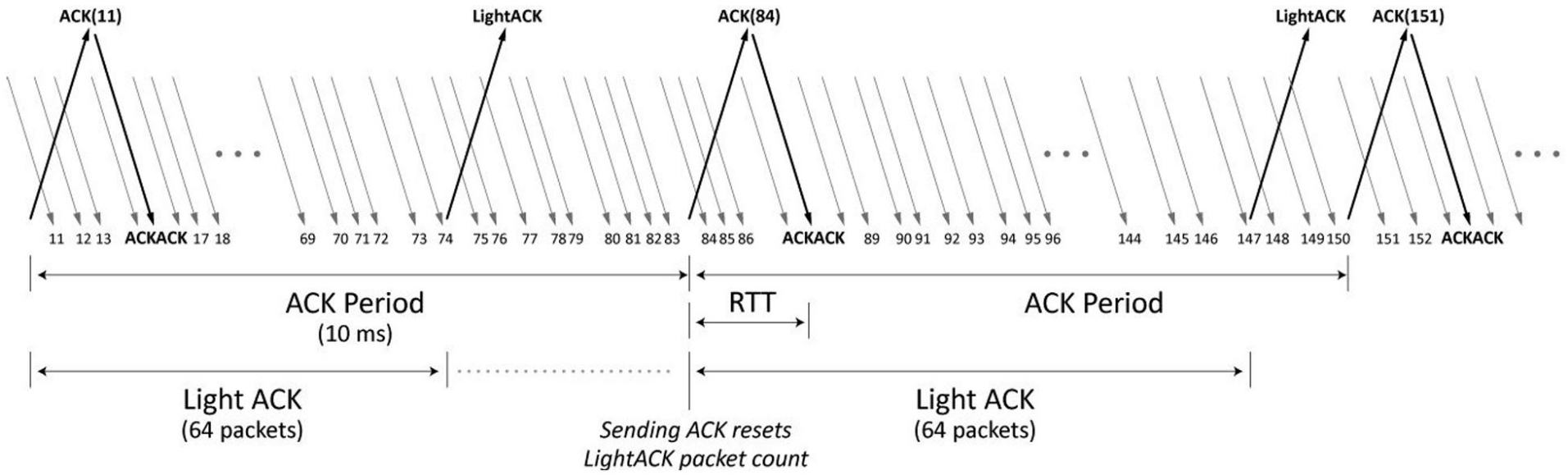
ACK se mandan regularmente para **confirmar recepción de a grupos**

Light ACK es una versión corta del ACK que se manda **cada 64 paquetes** exitosos

El ACK completo se manda cada 10 ms e incluye información adicional como RTT y capacidad del enlace

La **pérdida** de algún paquete **bloquea** el envío del siguiente ACK

Acknowledgement

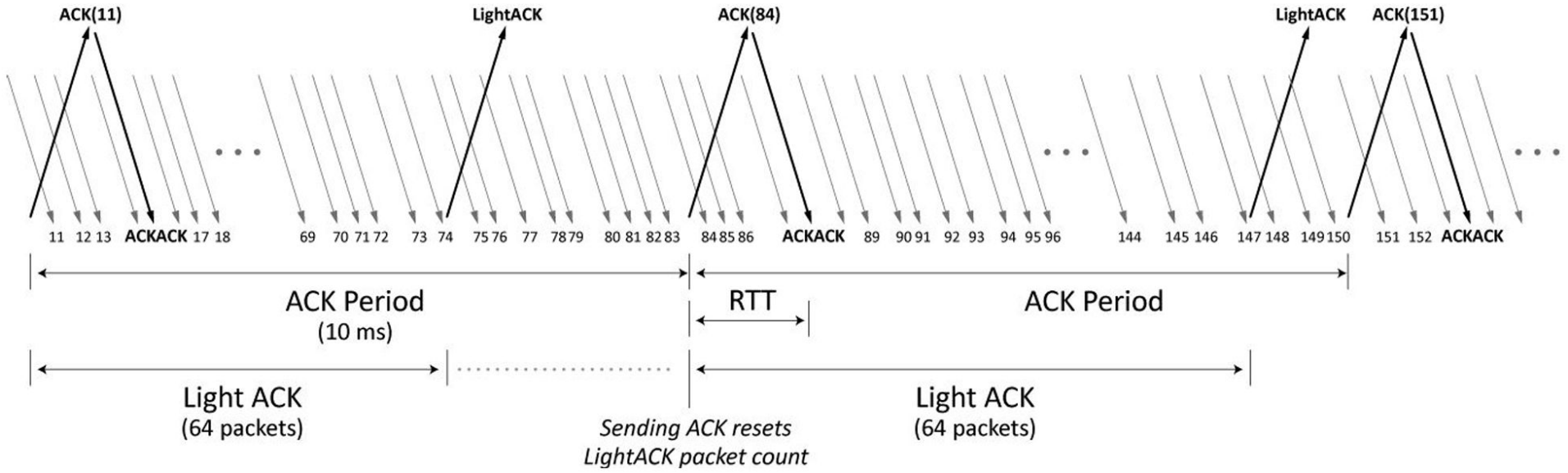


Como cumplen condiciones distintas, el ACK (10 ms) y el Light ACK (64 pk) no coinciden.

Al recibir el ACK, el Sender puede borrar los paquetes validados de su buffer.

Light ACK se usa en escenarios donde el bit rate es alto y se justifican reportes intermedios.

Estimación de RTT



Al recibir un ACK (Full), el Sender envía con bajo procesamiento una respuesta ACKACK.

Con los tiempos de ACK y de ACKACK, el receptor calcula el Round-Trip Time (RTT) en la red.

Control de congestión

Con ACK y ACKACK se intercambian parámetros para dimensionar la congestión de la red

SRT define dos algoritmos de control de congestión:

Live Congestion Control (LiveCC)	**File Transfer Congestion Control (FileCC)**
Para transmisiones en vivo	Para transferencia de archivos

LiveCC se enfoca en mantener un buffer estable para que no se llene o no sea suficiente.

De esta manera se busca una reproducción fluida en el receptor.

Packet Pacing

Complementario al control de congestión, también busca la reproducción fluida en el receptor a pesar de las condiciones de la red.

El pacing refiere al **ajuste de la velocidad de transmisión de los paquetes**, basándose en la información de delivery del receptor con la aplicación.

Reserva ancho de banda adicional para retransmisiones sin interrumpir la transmisión principal.

El **espaciado de paquetes** se calcula en función del tamaño del payload y el ancho de banda.

Se ajustan los intervalos entre paquetes de manera dinámica, según las condiciones.

Maneja el **Too-Late Packet Drop** en función de los tiempos del receptor.

¡Muchas gracias!