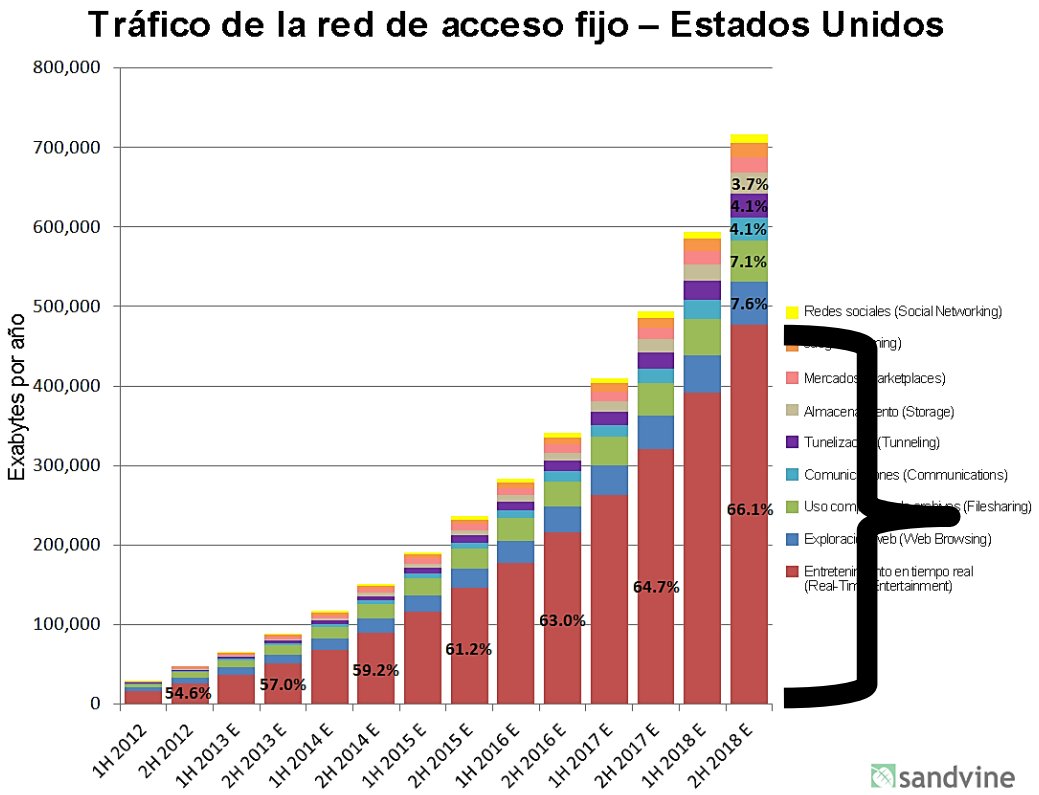
Transportando multimedia

Departamento de INGENIERIA TELEMATICA
www.it.uc3m.es

Tipos de comunicación multimedia

- **Según forma de distribución**
  - **Broadcast**
  - **Switched digital video ~ multicast**
  - **On-demand ~ unicast**
- **Según forma de comunicación: Interactiva / no-interactiva**
  - **No-interactivo: predomina *one-way vídeo***
- **No interactiva, según quién provee**
  - **IPTV: el proveedor que conecta 'la última milla'**
    - ✓ **Ej: Movistar Plus (antes Movistar TV, antes Imagenio)…**
  - **OTT (*Over The Top*): provee otro distinto del proveedor *last-mile***
    - ✓ **Ej: YouTube, RTVE, Netflix …**



Principalmente one-way vídeo, OTT

¿Qué tecnología?

Tecnología para one-way video, versión #1

Servidor de video

```
/* nos centramos en el transporte */
file = open  (fileName , O_RDONLY);
socketTCP = sock (…);
/* establece sesion TCP */

while (datosEnviado < tamanoFich)
{
    read (file, memor, TAM_BLOQ);
    write (socketTCP, memor, TAM_BLOQ);
    /* actualiza datosEnviados */
}
```

Cliente de video

```
/* nos centramos en el transporte */
socketTCP = sock (…);
/* establece sesion TCP */


while (datosRecibidos< tamanoFich)
{
    read (socketTCP, memor, TAM_BLOQ);
    /* actualiza datosRecibidos */
}

/* configura dispositivo de video */
videoDevice = open (/dev/… );

while (datosReprod < tamanoFich)
{
    write (videoDevice, memor, TAM_BLOQ);
    /* actualiza datosReprod */
}
```

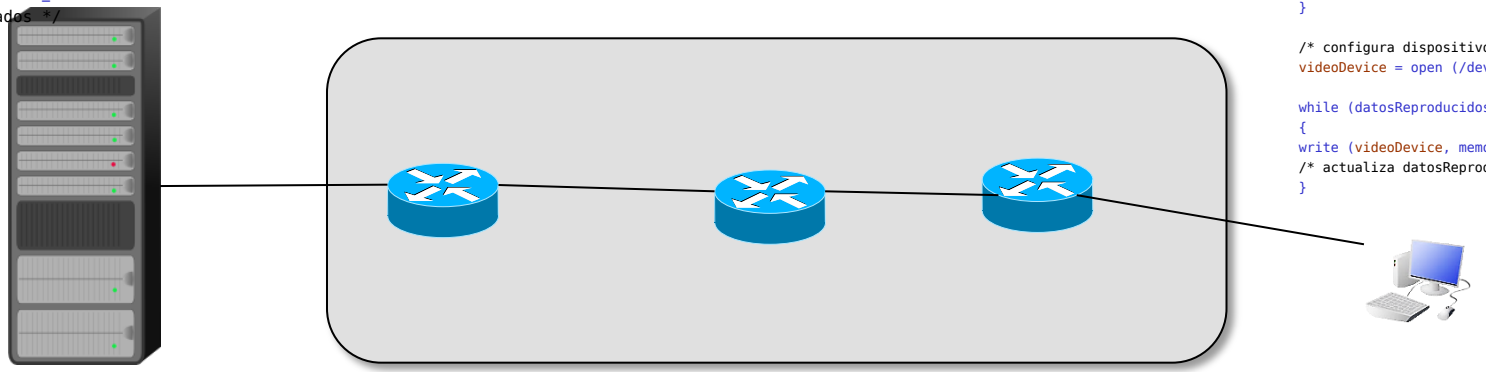
#1 = transferencia contenido completo, luego reproducción

Tecnología para one-way video, versión #1

#1 = transferencia contenido completo, luego reproducción

```
/* nos centramos en el transporte */
file = open  (fileName , O_RDONLY);
socketTCP = sock (…);
/* establece sesion TCP */

while (datosEnviado < tamanoFichero)
  {
read (file, memor, TAM_BLOQ);
write (socketTCP, memor, TAM_BLOQ);
/* actualiza datos enviados */
}
```

```
/* nos centramos en el transporte */
socketTCP = sock (…);
/* establece sesion TCP */


while (datosRecibidos< tamanoFichero)
  {
read (socketTCP, memor, TAM_BLOQ);
/* actualiza datos recibidos */
}

/* configura dispositivo de video */
videoDevice = open (/dev/… );

while (datosReproducidos < tamanoFichero)
{
write (videoDevice, memor, TAM_BLOQ);
/* actualiza datosReproducidos */
}
```



Propiedades de TCP respecto a

- Servicio de entrega de datos
- Flujo
- Congestión
- Retardos

#1: ¿cumple parámetros de calidad?

#1 = transferencia contenido completo, luego reproducción

ITU-T G.1010: 'Categorías de calidad de servicio para los usuarios de extremo de servicios multimedios', 2001

Medium	Application	Degree of symmetry	Typical data rates	Key performance parameters and target values			
				One-way delay	Delay variation	Information loss (Note 2)	Other
Audio	Conversational voice	Two-way	4-64 kbit/s	<150 ms preferred (Note 1) <400 ms limit (Note 1)	< 1 ms	< 3% packet loss ratio (PLR)	
Audio	Voice messaging	Primarily one-way	4-32 kbit/s	< 1 s for playback < 2 s for record	< 1 ms	< 3% PLR	
Audio	High quality streaming audio	Primarily one-way	16-128 kbit/s (Note 3)	< 10 s	<< 1 ms	< 1% PLR	
Video	Videophone	Two-way	16-384 kbit/s	< 150 ms preferred (Note 4) <400 ms limit	**BAJA (< 1ms)**	< 1% PLR	Lip-synch: < 80 ms
Video	One-way	One-way	16-384 kbit/s	< 10 s	**BAJA (< 1ms)**	< 1% PLR	

Jitter

NOTE 1 – Assumes adequate echo control.

NOTE 2 – Exact values depend on specific codec, but assumes use of a packet loss concealment algorithm to minimise effect of packet loss.

NOTE 3 – Quality is very dependent on codec type and bit-rate.

NOTE 4 – These values are to be considered as long-term target values which may not be met by current technology.

#1: ¿cumple parámetros de calidad?

#1 = transferencia contenido completo, luego reproducción

ITU G.1010: 'Categorías de calidad de servicio para los usuarios de extremo de servicios multimedios', 2001

Medium	Application	Degree of symmetry	Typical data rates	Key performance parameters and target values			
				One-way delay	Delay variation	Information loss (Note 2)	Other
Audio	Conversational voice	Two-way	4-64 kbit/s	<150 ms preferred (Note 1) <400 ms limit (Note 1)	< 1 ms	< 3% packet loss ratio (PLR)	
Audio	Voice messaging	Primarily one-way	4-32 kbit/s	< 1 s for playback < 2 s for record	< 1 ms	< 3% PLR	
Audio	High quality streaming audio	Primarily one-way	16-128 kbit/s (Note 3)	< 10 s	<< 1 ms	< 1% PLR	
Video	Videophone	Two-way	16-384 kbit/s	< 150 ms preferred (Note 4) <400 ms limit	**BAJA (< 1ms)**	< 1% PLR	Lip-synch: < 80 ms
Video	One-way	One-way	16-384 kbit/s	< 10 s	**BAJA (< 1ms)**	< 1% PLR	

NOTE 1 – Assumes adequate echo control.

NOTE 2 – Exact values depend on specific codec, but assumes use of a packet loss concealment algorithm to minimise effect of packet loss.

NOTE 3 – Quality is very dependent on codec type and bit-rate.

NOTE 4 – These values are to be considered as long-term target values which may not be met by current technology.

#1 = transferencia contenido completo, luego reproducción

#1: ¿cumple parámetros de calidad?

#1 = transferencia contenido completo, luego reproducción

- ¿Retardos de reproducción aceptables?
- ¿Pérdida de paquetes en reproducción?

- Problema #1.1: retardo de inicio
  - **Intrínseco al diseño inicial: primero descarga el fichero entero, luego lo reproduce**

Tecnología para one-way video, versión #2

Servidor de video

```
/* nos centramos en el transporte */
file = open  (fileName , O_RDONLY);
socketTCP = sock (…);
/* establece sesion TCP */

while (datosEnv < tamanoFich)
{
  read (file, memor, TAM_BLOQ);
  write (socketTCP, memor, TAM_BLOQ);
  /* actualiza datosEnv */
}
```

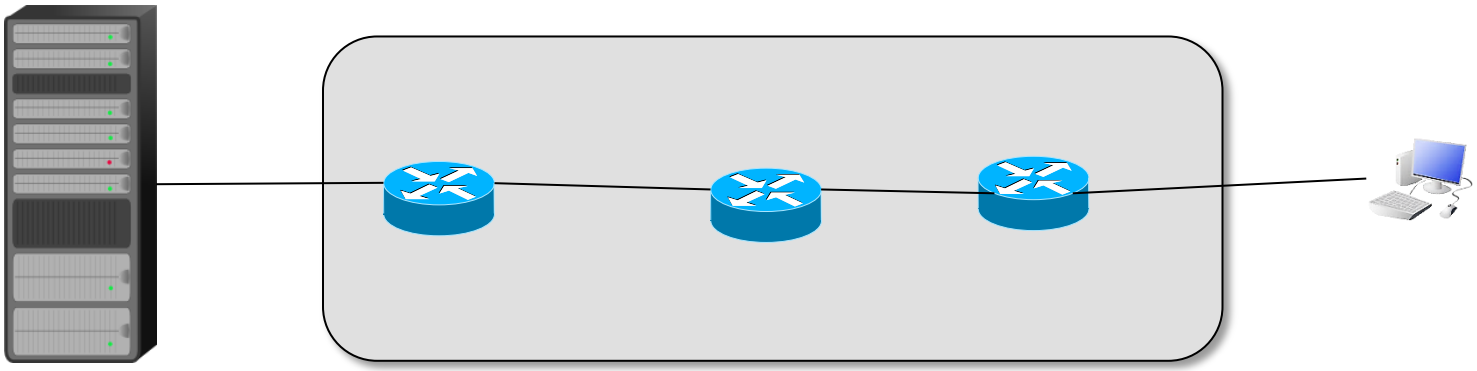
Cliente de video

```
/* nos centramos en el transporte */
socketTCP = sock (…);
/* establece sesion TCP */
/* configura dispositivo de video */
videoDisp = open (/dev/… );


while (datosRec< tamanoFich)
{
   read (socketTCP, mem, TAM_BLOQ);
   /* actualiza datosRec */
   write (videoDisp, mem, TAM_BLOQ);
}
```

#2 = reproduce bloque de datos nada más recibirlo

#2: ¿cumple parámetros de calidad?

#2 = reproduce bloque de datos nada más recibirlo



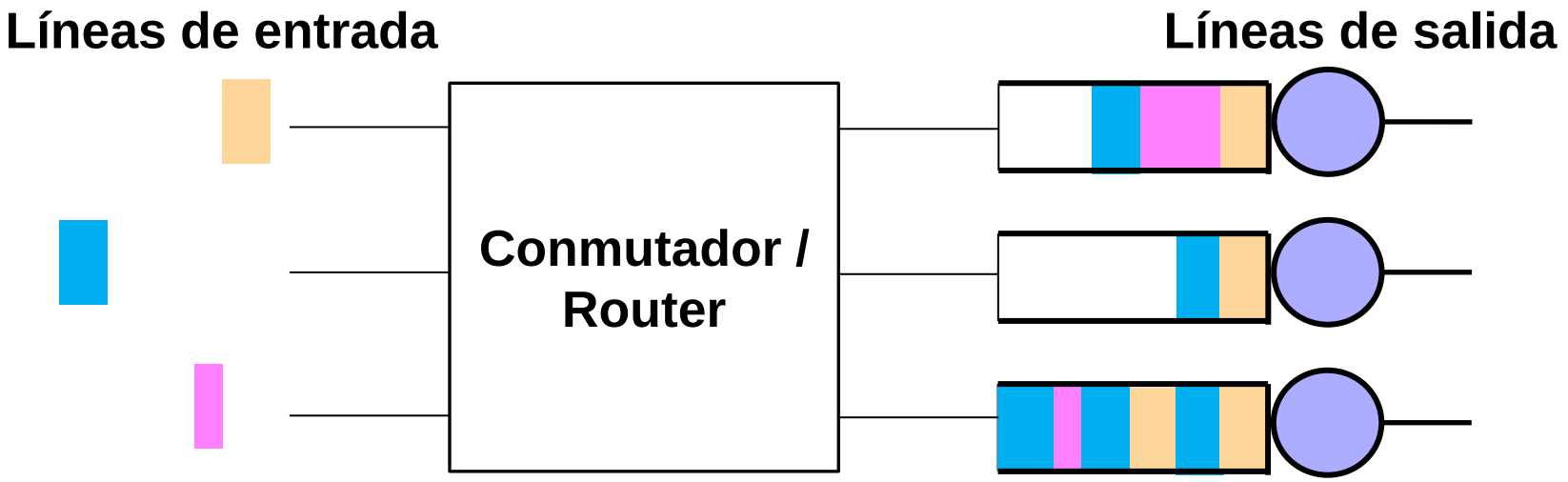
- ¿Control de flujo?

- ¿Pérdida de paquetes en reproducción?
- ¿Retardos aceptables en reproducción? ¿Se puede mantener 'delay variation'?

La RED no es ideal, y afecta

- ¿Causas de pérdida de paquetes?
- ¿Causas de retardos de paquetes?
  - **¿Puedo cuantificar el retardo máximo?**

Pérdidas y retardos en la red



- ¿Causas de pérdida de paquetes?
  - ¿Cómo son las pérdidas si utilizo TCP?
- ¿Causas de retardos de paquetes?
  - ¿Puedo cuantificar el retardo máximo 'one-way'?
  - ¿Cómo es el retardo si utilizo TCP?
- ¿Cómo varía retardo / pérdida de paquetes con la distancia?

#2: ¿cumple parámetros de calidad?

#2 = reproduce bloque de datos nada más recibirlo

- **Supongamos que de media, hay ancho de banda suficiente, pero puede haber variaciones en transitorio**
  - **Si de media no hay ancho de banda suficiente, hay que REDUCIR la calidad del vídeo**
- **Problema #2.1: ¿cumple JITTER?**

  ¿estoy seguro de que tendré datos suficientes para reproducir en cada momento?
- **Solución #2.1: que la red tenga "mejor comportamiento"**
  - **Soluciones de Calidad de Servicio**
- **Versión #3: acumular datos en el cliente antes de empezar la reproducción**
  - **BUFFERING**

Tecnología para one-way video, versión #3 (aka *progressive download*)

Servidor de video

```
/* nos centramos en el transporte */
file = open  (fileName , O_RDONLY);
socketTCP = sock (…);
/* establece sesion TCP */

while (datosEnviado < tamanoFichero)
{
   read (file, mem, TAM_BLOQ);
   write (socketTCP, mem, TAM_BLOQ);
   /* actualiza datosEnviado */
}
```

Cliente de video

```
/* nos centramos en el transporte */
socketTCP = sock (…);
/* establece sesion TCP */
/* configura dispositivo de video */
videoDisp = open (/dev/… );

memIni = mem;

while (datosRec < tamanoFichero)
{
  read (socketTCP, mem, TAM_BLOQ);
  /* actualiza datos recibidos */
  if (datosRec > BUFFER_INICIAL)
  {
    write (videoDisp, memIni, TAM_BLOQ);
  }
}
/* Cuando termina de recibir, quedan
datos que hay que terminar de
reproducir*/
```

#3 = buffering, acumula algunos datos antes de comenzar reproducción

Progressive download vs streaming

- **Progressive download:**
  - **Mandar todo el vídeo al destino**
  - **Muy simple para el servidor que envía, muy fácil de implementar con TCP**
  - ***Forward* sólo hasta parte de contenido recibida**
- **Streaming:**
  - **El cliente pide explícitamente bloques para reproducir**
    - ✓ **Facilita forward**

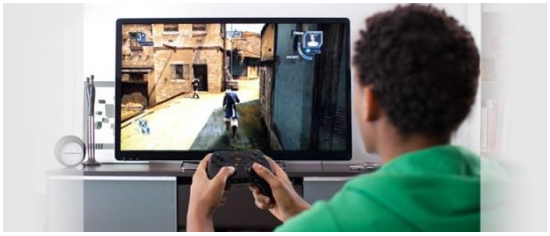
Version #3

- TCP no permite uso de multicast

~~Transporte de video one-way~~

Transporte de comunicaciones interactivas

Comunicaciones Interactivas



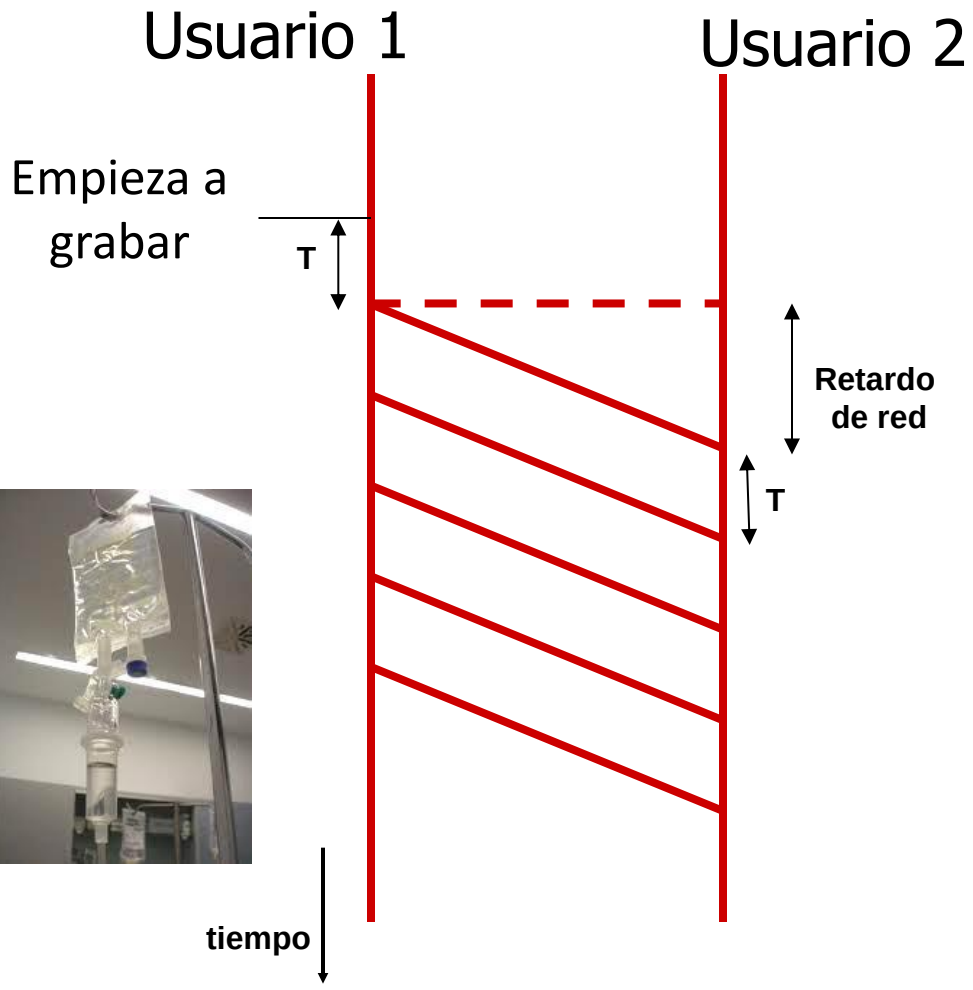
onLIVE

Cloud gaming requiere respuesta en <120 ms.

¡El contenido se genera en tiempo real

Comunicación interactiva, versión #1

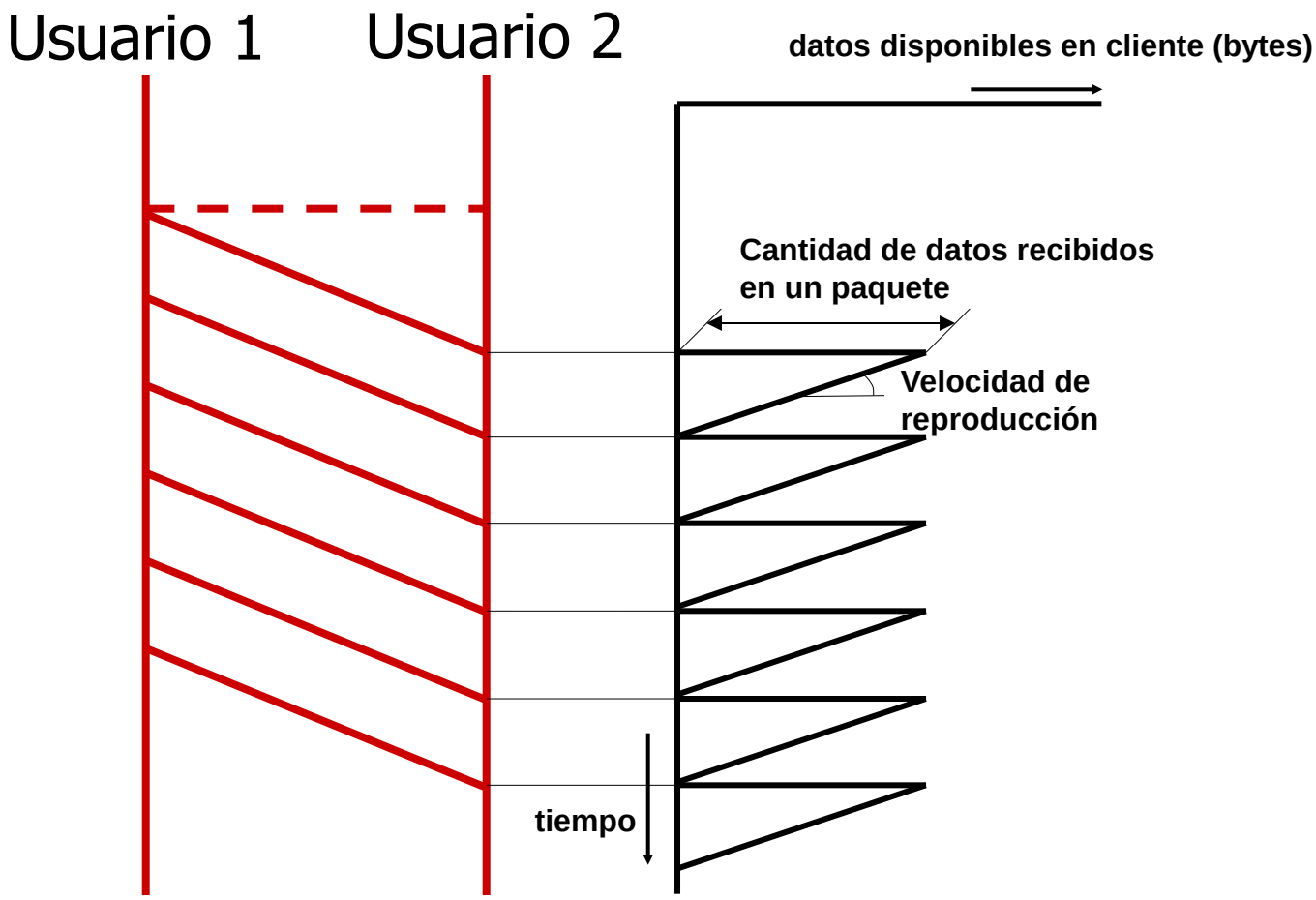
- **Ejemplo típico:**
  - Audio generado a 8000 muestras por segundo, 1 byte por muestra
  - T=20 ms (160 bytes)
  - Retardo de codificación (codec delay): T
    - ✓ Depende de tamaño de paquete
    - ✓ Siempre presente (aunque lo obviamos en el resto de la discusión)



Comunicación interactiva, versión #1

#1 = reproduce bloque de datos nada más recibirlo

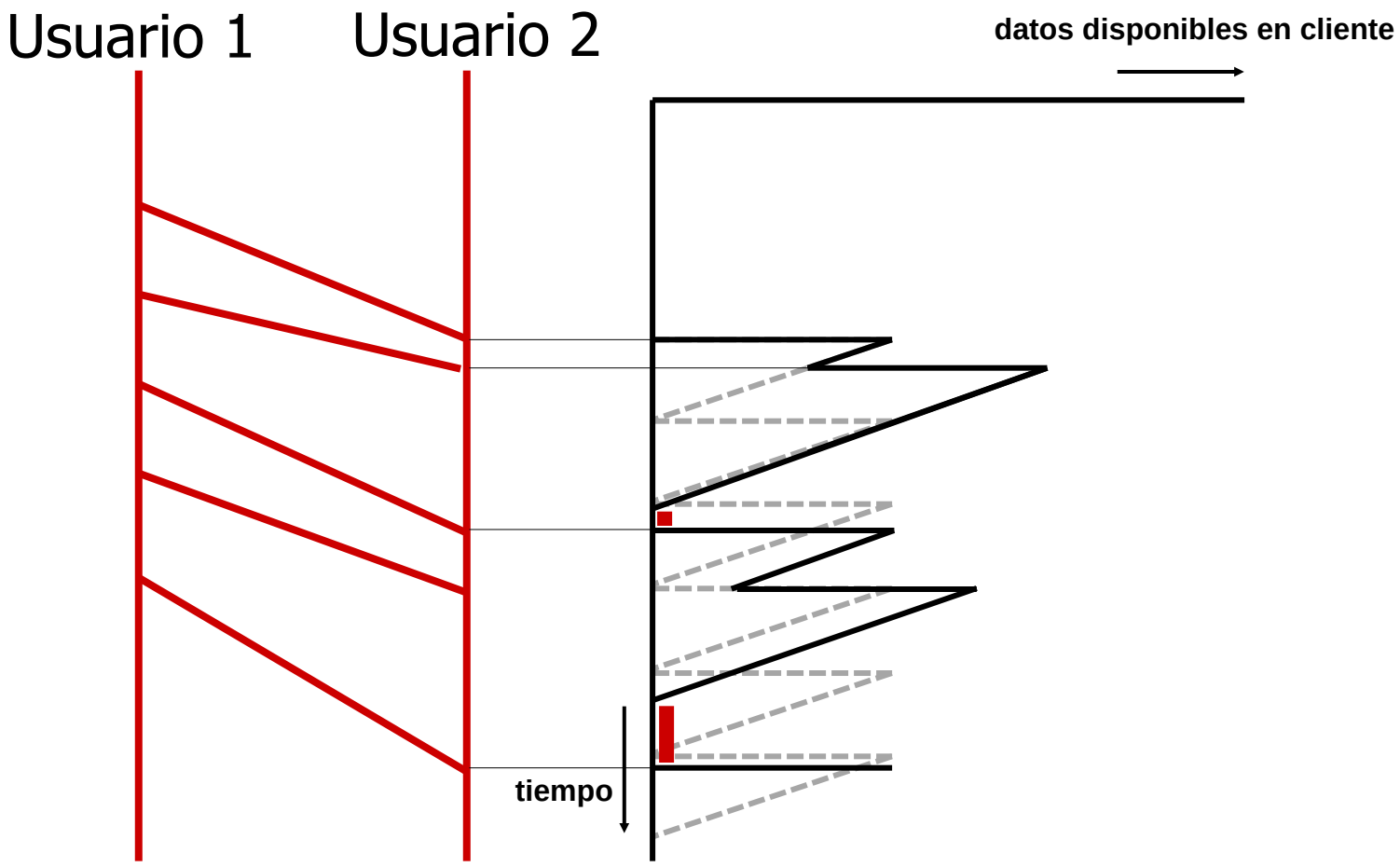
- **Análisis de datos disponibles en la memoria del cliente**



Comunicación interactiva, versión #1

#1 = reproduce bloque de datos nada más recibirlo

- **Comunicación con retardos variables en la red**



#1: ¿cumple parámetros de calidad?

#1 = reproduce bloque de datos nada más recibirlo

ITU G.1010: 'Categorías de calidad de servicio para los usuarios de extremo de servicios multimedios', 2001

Medium	Application	Degree of symmetry	Typical data rates	Key performance parameters and target values			
				One-way delay	Delay variation	Information loss (Note 2)	Other
Audio	Conversational voice	Two-way	4-64 kbit/s	<150 ms preferred (Note 1) <400 ms limit (Note 1)	< 1 ms	< 3% packet loss ratio (PLR)	
Audio	Voice messaging	Primarily one-way	4-32 kbit/s	< 1 s for playback < 2 s for record	< 1 ms	< 3% PLR	
Audio	High quality streaming audio	Primarily one-way	16-128 kbit/s (Note 3)	< 10 s	<< 1 ms	< 1% PLR	
Video	Videophone	Two-way	16-384 kbit/s	< 150 ms preferred (Note 4) <400 ms limit	BAJA (< 1ms)	< 1% PLR	Lip-synch: < 80 ms
Video	One-way	One-way	16-384 kbit/s	< 10 s	BAJA (< 1ms)	< 1% PLR	

NOTE 1 – Assumes adequate echo control.

NOTE 2 – Exact values depend on specific codec, but assumes use of a packet loss concealment algorithm to minimise effect of packet loss.

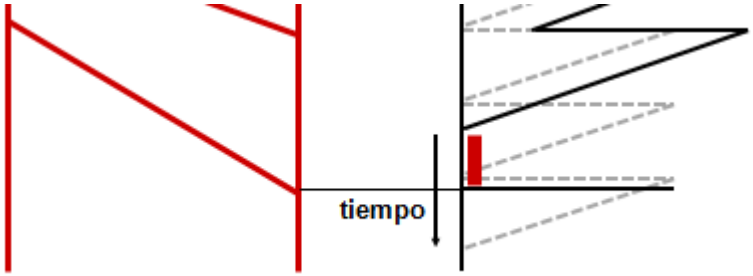
NOTE 3 – Quality is very dependent on codec type and bit-rate.

NOTE 4 – These values are to be considered as long-term target values which may not be met by current technology.

#1: ¿cumple parámetros de calidad?

#1 = reproduce bloque de datos nada más recibirlo

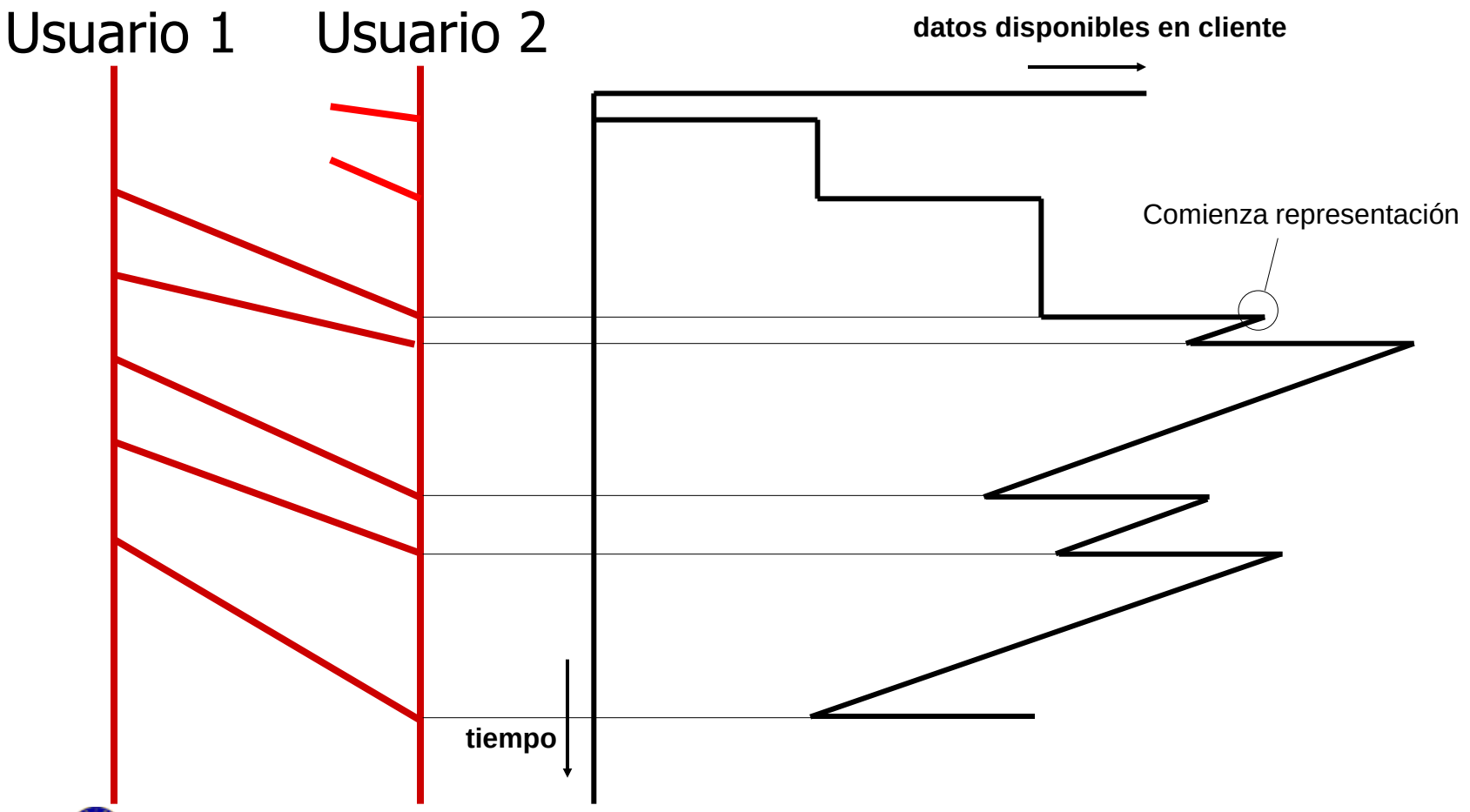
- **Delay variation <1 ms**



- **Necesitamos Versión #2: BUFFERING**

Comunicación interactiva, versión #2

#2= buffering, acumula algunos datos antes de comenzar reproducción



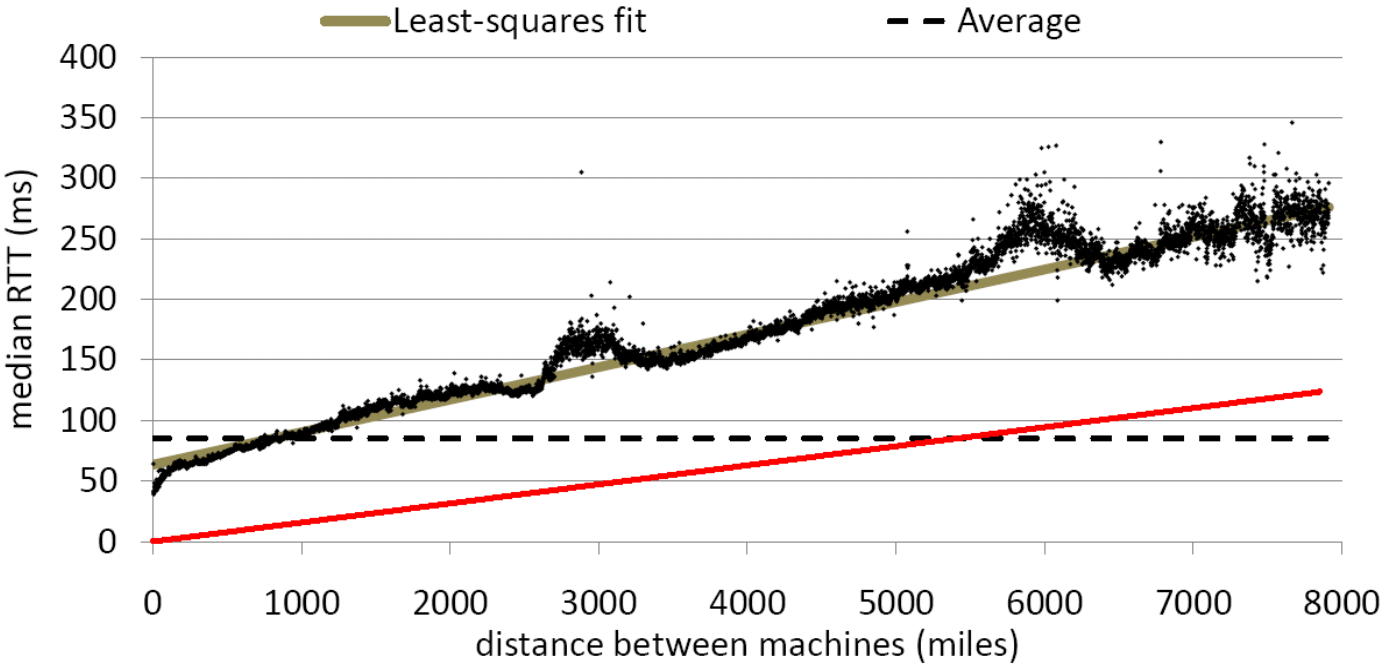
#2: ¿cumple parámetros de calidad?

#2= buffering, acumula algunos datos antes de comenzar reproducción

- **One-way delay**
  - **<150 ms preferred**
  - **<400 ms limit**

Versión #2: ¿cumple parámetros de calidad?

#2= buffering, acumula algunos datos antes de comenzar reproducción



RTT (=ida y vuelta) en función de distancia
Medido para jugadores de Halo-3.
En rojo, retardo de transmisión (RTT) mínimo
Agarwal, Lorch [Sigcomm09]

- **Retardo medido en orden de magnitud del deseado**
  - **¡Cuidado con tamaño de buffering!**

#2: ¿cumple parámetros de calidad?

#2= buffering, acumula algunos datos antes de comenzar reproducción

Suponemos que utilizábamos TCP

- **Problemas con *one-way delay***
  - **TCP puede empezar lento (control de congestión)**
  - **TCP puede incorporar retardos si se pierden paquetes**
    - Retardo por pedir retransmisión
    - Bajada de velocidad por mecanismo de control de congestión
    - ✓ **Toleramos '< 3% packet loss ratio', pero no retardos grandes**
  - **Recordar que los problemas de retardos se agravan con la distancia**
- **Solución #3: utilizar UDP**
  - **Nota: TCP puede funcionar, hay quien lo utiliza para comunicaciones interactivas**

#3: ¿cumple parámetros de calidad?

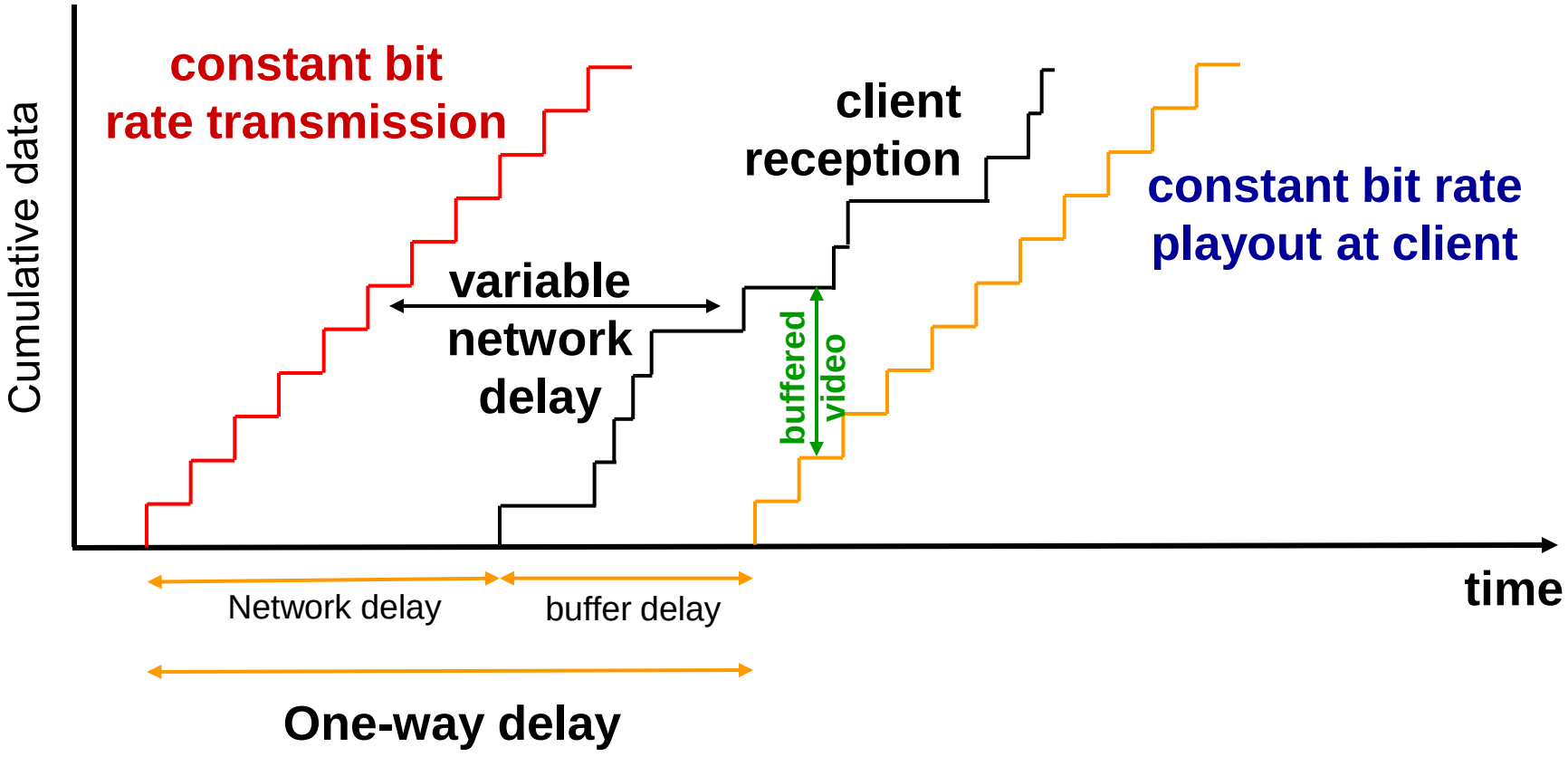
#3= UDP con buffering,

- One-way delay
  - <150 ms preferred
  - <400 ms limit
- Buffering incrementa one-way delay ☹
- ¿Cuánto buffering?

Comunicación interactiva, versión #3

#3= UDP con buffering,

- Otra forma de verlo



Computer Networking: A Top Down Approach, 6th ed,, J. Kurose, K. Ross

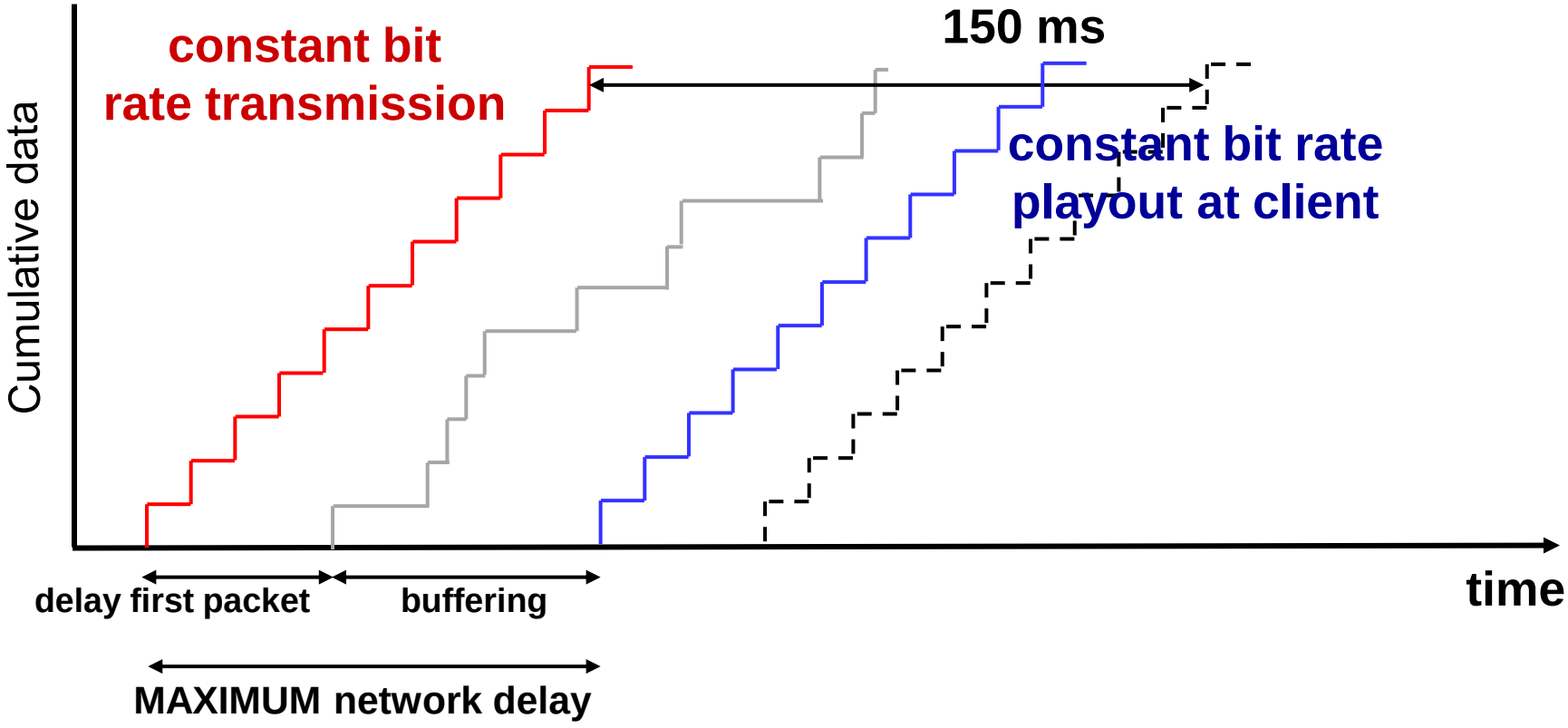
#3: ¿cumple parámetros de calidad?

#3= UDP con buffering,

- **Solución #3.1: dimensionar correctamente buffering inicial**

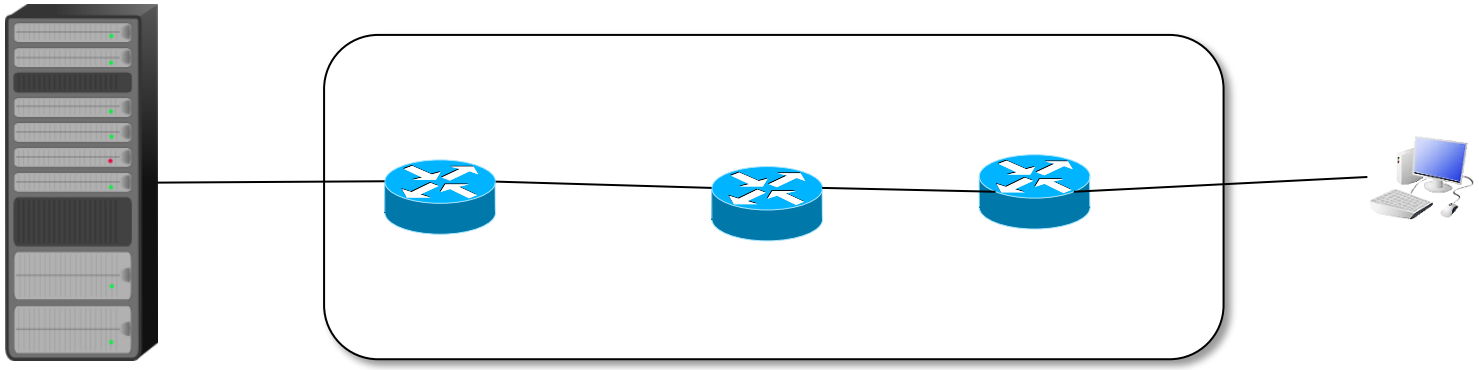
Dimensionamiento del buffer

- **Buffer delay: se establece para el primer paquete**
- **Idealmente, depende del**
  - **Retardo máximo de la red**
  - **Retardo del primer paquete**



Dimensionamiento del buffer

- ¿Puedo conocer retardo one-way máximo?
  - ¿De qué depende?
- Pregunta previa: ¿puedo MEDIR el retardo one-way?
  - Requiere relojes sincronizados en FRECUENCIA y OFFSET
  - ¿Alguna estimación del retardo one-way?



#3 y pérdidas de paquetes

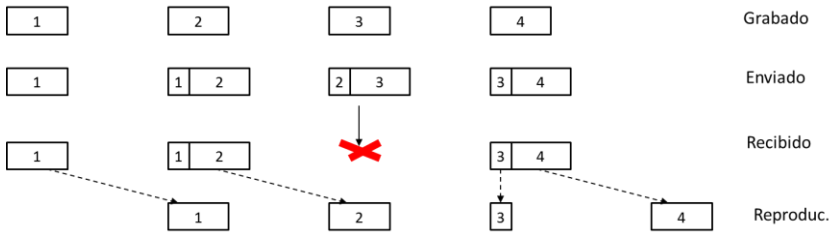
#3= UDP con buffering,

- **En solución basada en TCP, no nos teníamos que preocupar de pérdidas de paquetes**
  - **Pero UDP no gestiona esto**
- **TCP hacía control de flujo…**
  - **¿Puedo perder paquetes en #3 por 'problemas de control de flujo'?**

#3 y pérdidas de paquetes

#3= UDP con buffering,

- **Estrategia frente a pérdidas de paquetes en la red**
  - **Pérdidas puntuales**
    - ✓ **La aplicación pide retransmisión**
      - ➢ Si el retardo es muy bajo
    - ✓ **La fuente introduce redundancia**
      - ➢ Enviar más datos
    - ✓ **'Maquillar' pérdidas puntuales**
      - ➢ Interpolación de paquetes adyacentes, ruido blanco…



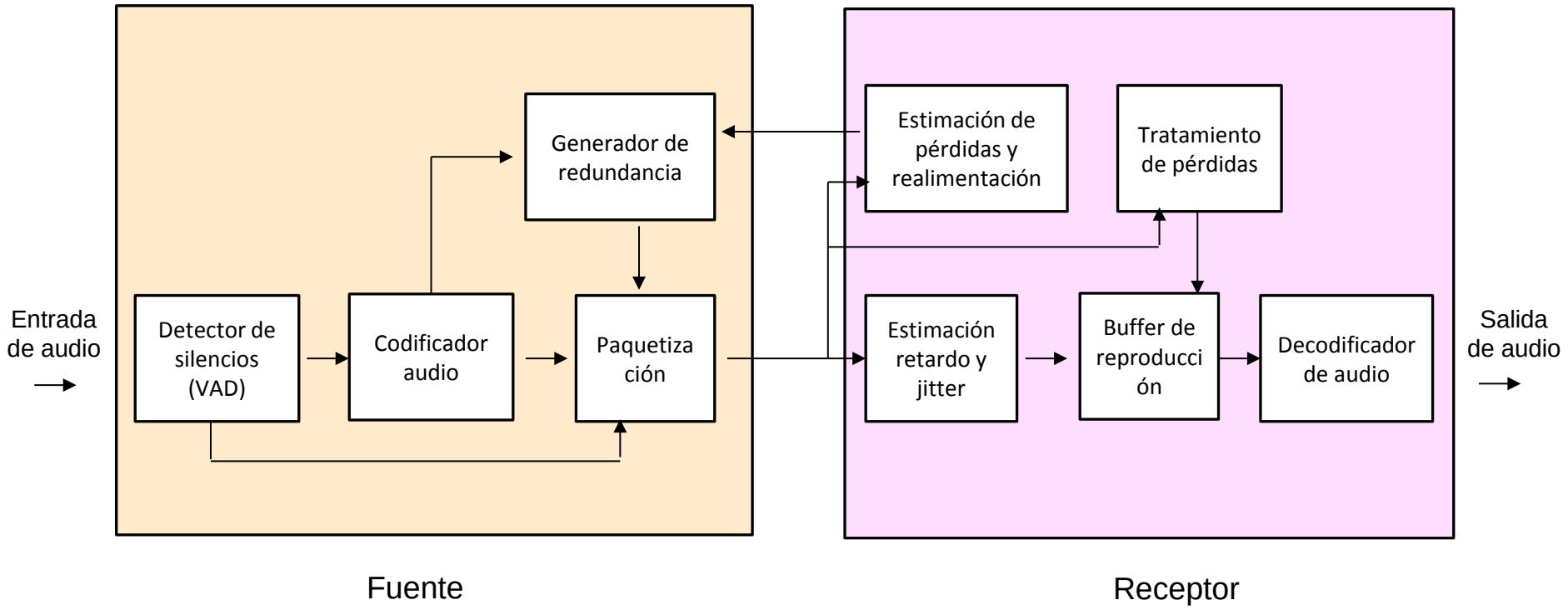
  - **Pérdidas continuadas: Reducir calidad ante pérdidas continuadas (posible congestión)**
    - ✓ **Receptor detecta pérdidas continuadas**
    - ✓ **Receptor transmite evento a fuente**
    - ✓ **Fuente cambia codec**
- **Necesitamos NUMERAR LOS PAQUETES para detectar pérdidas**
- **UDP no numera los paquetes**

#3 y medios ‘no continuos’

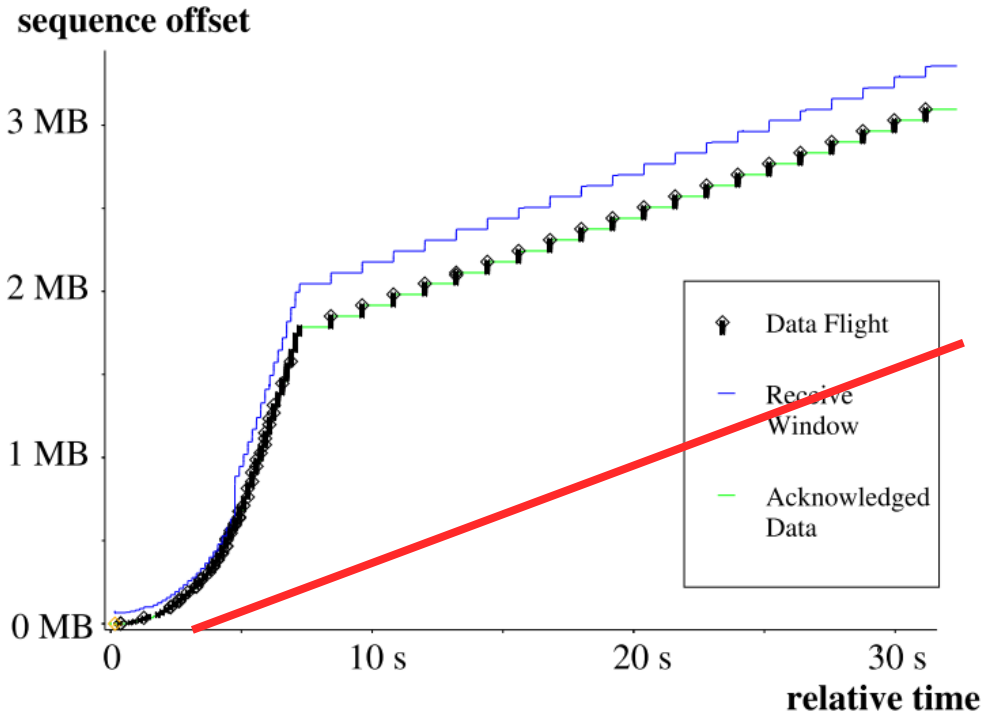
#3= UDP con buffering,

- **Medio es ‘continuo’: el tiempo de reproducción está implícito en los datos**
  - **Ejemplo, al ‘echar datos seguidos’ a la tarjeta de sonido, la tarjeta de sonido sabe qué hacer con ellos**
- **Medio ‘no continuo’: el tiempo de reproducción no está implícito en el flujo de datos**
  - **Audio con silencios: ¿cuándo termina el silencio?**
  - **Sincronización de un flujo de video y de subtítulos**
- **Para medios no continuos, hacen falta MARCAS de TIEMPO**
  - **Ni UDP ni TCP incluyen esta información**

Arquitectura de un sistema de VoIP



[NOTA al margen]



Consumo de datos en reproducción

- **También con one-way video puede tener sentido una estrategia de envío controlada**
  - **Reduce memoria en cliente**
  - **Muchos vídeos no se terminan, absurdo gastar servidor y ancho de banda en enviarlo**

RTP

Departamento de
INGENIERIA
TELEMATICA
www.it.uc3m.es

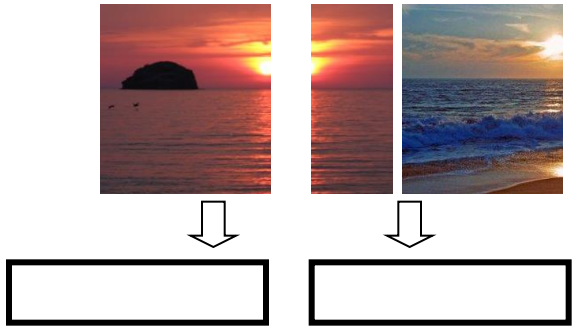
Introducción RTP

- La mayoría de las aplicaciones multimedia en red usan números de secuencia y marcas de tiempo
  - RTP ofrece este servicio de forma estandarizada
- Diseñado para poder usarse en una grandísima variedad de escenarios
  - Desde audioconferencias punto a punto a distribución de contenidos multimedia a miles de usuarios
- Permite transportar formatos comunes PCM, GSM, MP3, … para sonido y MPEG, H.263, H.264, etc., para video
  - También transporta formatos propietarios
- Es el estándar utilizado para transporte de datos en muchas arquitecturas: SIP, H.323, RTSP…
- RTP
  - No gestiona QoS (no tiene que ver con la infraestructura de red)
  - No garantiza entrega fiable ni ordenada
- Va acompañado del intercambio de información de control: RTCP

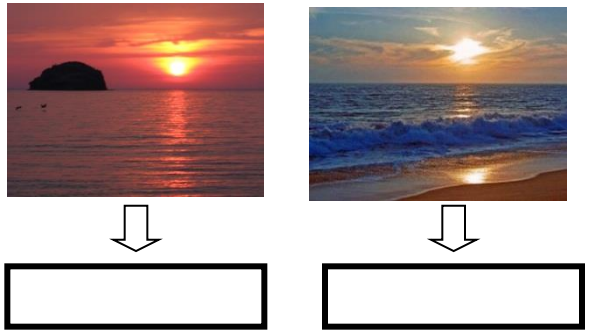
Filosofía RTP

- **Para datos multimedia, nadie mejor que la aplicación para saber qué hacer si se pierden paquetes o llegan tarde o cuáles son sus prioridades relativas**
  - **No poner funciones a nivel de transporte que no sean universalmente necesarias**
    - ✓ **El protocolo de transporte no debe implementar recuperación ante errores, control de flujo, ...**
  - ***Application level framing*: La aplicación delimita en bloques la información que se envía (bloques que puedan procesarse de forma completa e independiente)**
    - ✓ **Un mensaje (nivel de aplicación) debe empaquetarse en un paquete (nivel de red)**

Sin *application level framing*



Con *application level framing*

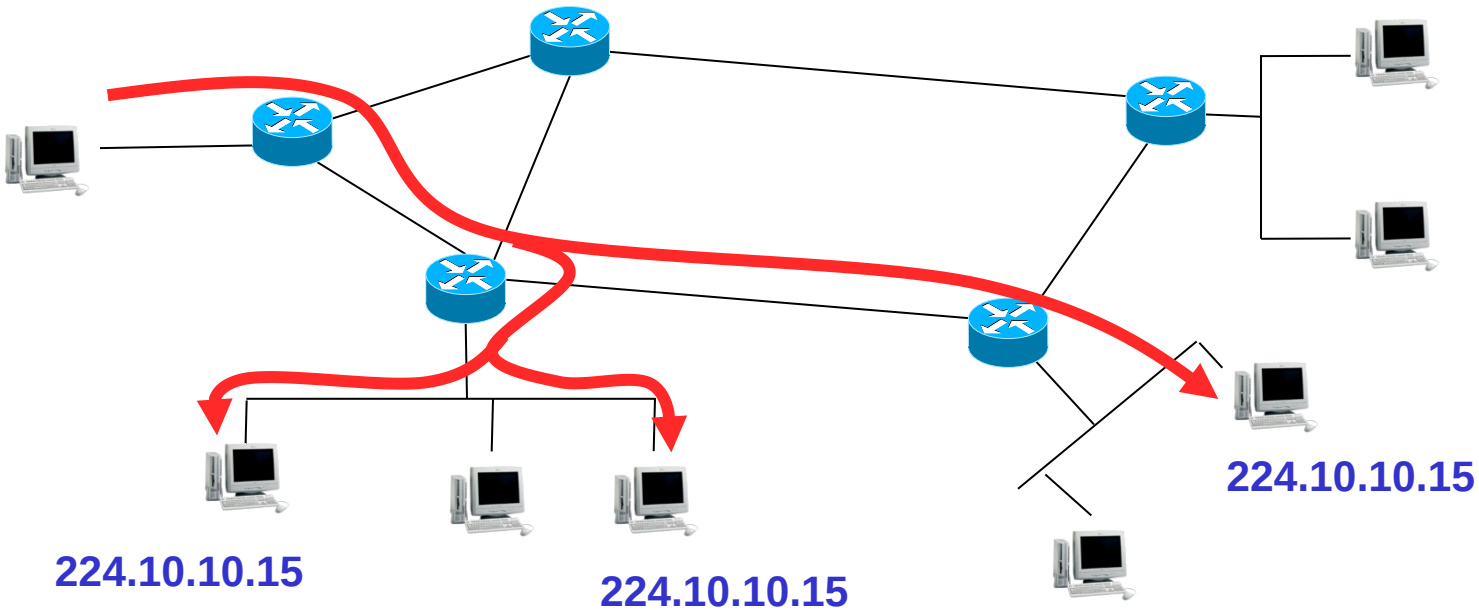


Filosofía RTP

- RTP funciona **extremo a extremo**: los nodos intermedios no interpretan RTP
- Necesita de un protocolo de transporte por debajo para multiplexación
  - RTP viaja preferentemente encima de UDP
- Conceptualmente, se puede entender como un servicio de **transporte**
  - Aunque la frontera con aplicación no está clara
- Respecto a su implementación, se suele integrar como parte del procesamiento a nivel de la aplicación que lo usa, en espacio de usuario, por encima del interfaz de sockets
  - Lo implementa el programador de la aplicación, o éste utiliza librerías que compila con su programa
  - No es un servicio del sistema operativo

Repaso: multicast en IP

- Direcciones entre 224.0.0.0 y 239.255.255.255
  - Sólo se pueden utilizar como dirección destino
- ¿Qué tiene que hacer un nodo para enviar a un grupo multicast?
- ¿Qué tiene que hacer un nodo para recibir de un grupo multicast?



Sesión RTP

- Sesiones “ligeras”: no hay control estricto ni centralizado sobre quién está en la sesión
  - Se ajusta bien al modelo multicast de IP
- *Sesión RTP* es una asociación entre un conjunto de participantes que se comunican con RTP

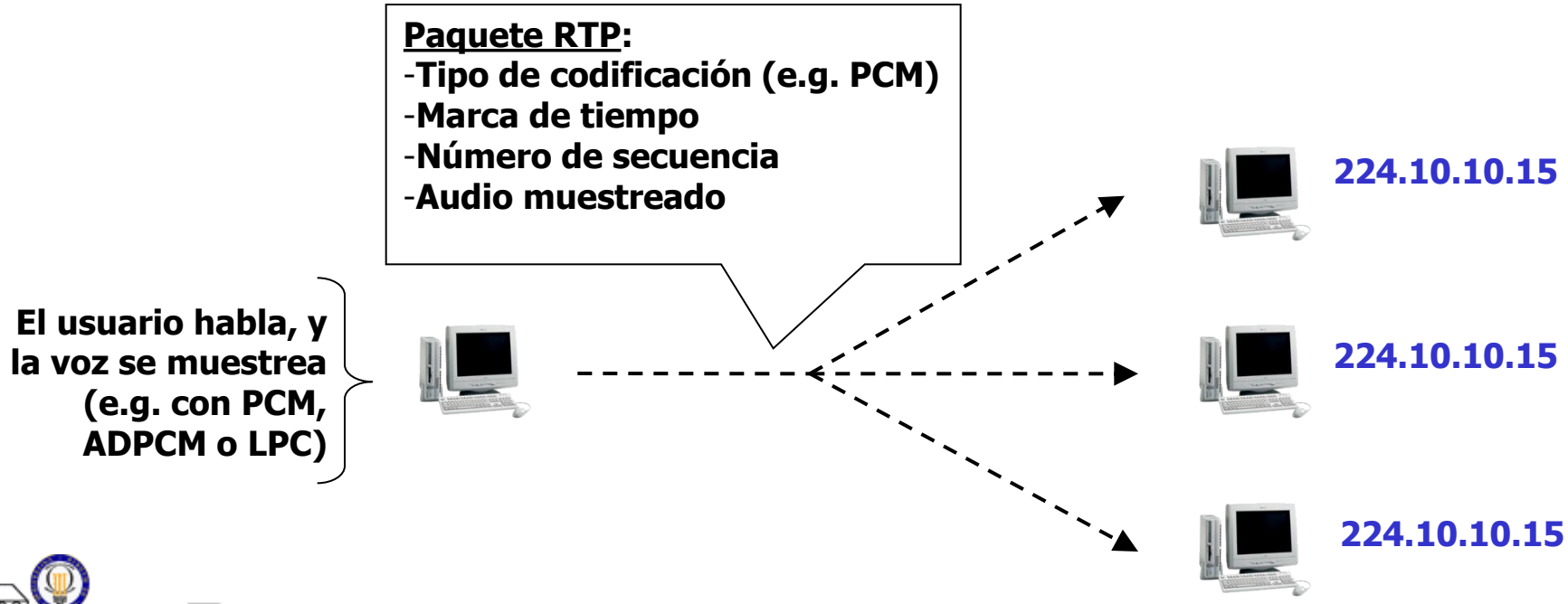
  Indica los parámetros de destino de los paquetes, y está definida por
  - Direcciones IP de los participantes
    - ✓ IP multicast
    - ✓ o lista de direcciones unicast
  - Puerto RTP: Puerto destino (UDP), en general un número par,
    - ✓ En general, lo utilizan TODOS los participantes (aunque podrían definirse puertos distintos para cada destino)
    - ✓ Puerto por defecto: 5004
  - Puerto RTCP: Puerto destino (UDP) de los mensajes de control (RTCP), en general impar
    - ✓ En general, lo utilizan TODOS los participantes (aunque podrían definirse puertos distintos para cada destino)
    - ✓ En general, puerto RTCP = puerto RTP asociado + 1
- RTP no dice cómo establecer las sesiones, determinar los parámetros que las definen
  - Esto se hace a través de otros protocolos, ej. SIP

Sesión RTP

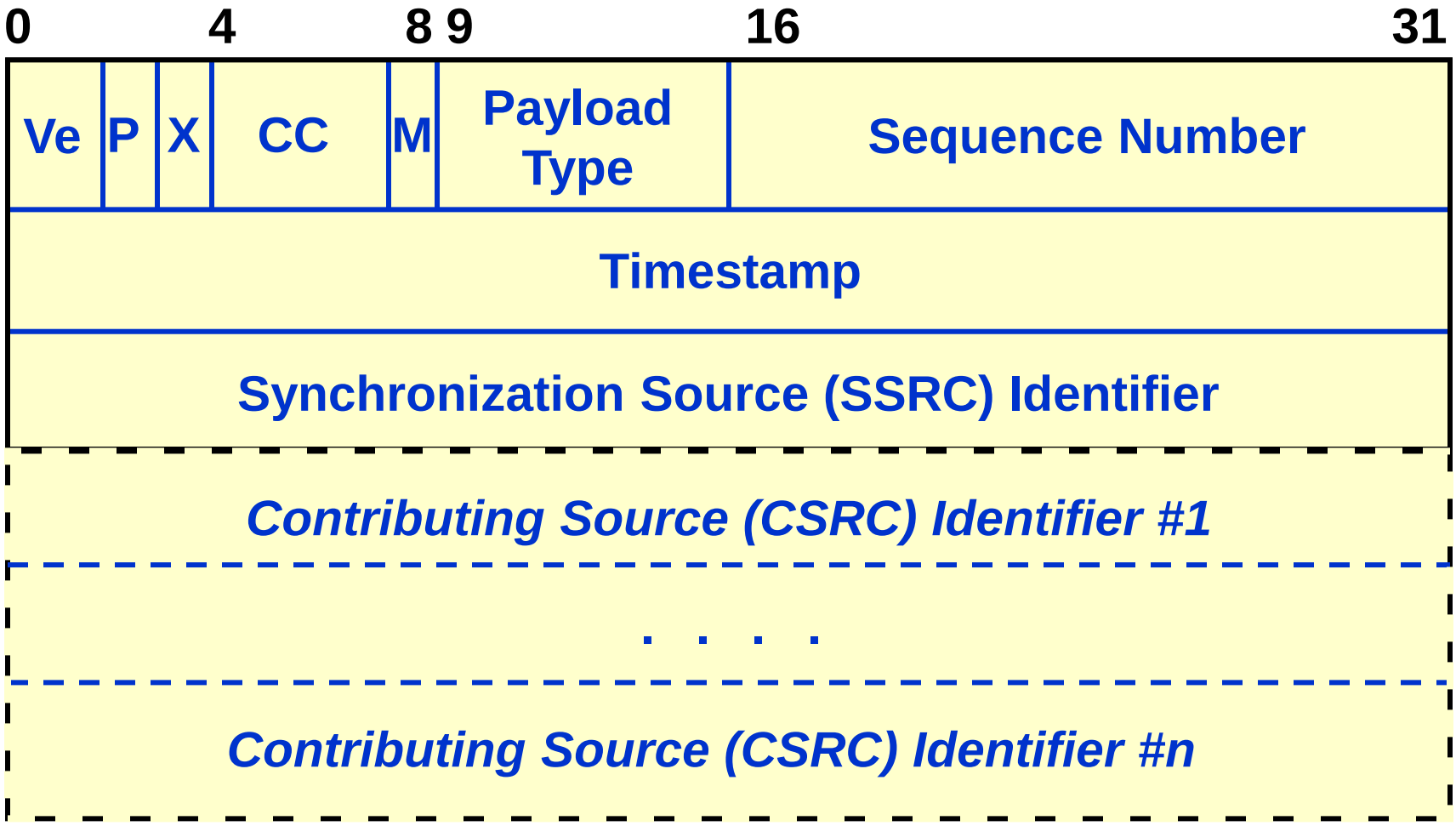
- Si un nodo participa en una sesión, es porque quiere recibir los datos (RTP) que se envían ahí
  - Además, alguno de los que participa puede mandar datos (RTP)
  - Roles posibles: 'Receptor de datos', 'Emisor+Receptor de datos'
    - ✓ Los roles pueden cambiar dinámicamente (uno puede mandar un momento, y luego sólo recibir)
- Todos los nodos mandan y reciben información de control (RTCP)
  - También los que no mandan datos

Ejemplos de uso de RTP

- **Conferencia de audio multicast:**
  - 4 participantes
  - 1 sesión RTP: dirección *multicast* 224.10.10.15, puerto RTP 5004, puerto RTCP 5005
  - El audio se muestrea en cada equipo de usuario y se envía en paquetes RTP
  - Cada paquete RTP se envía en 1 paquete UDP, con dirección destino 224.10.10.15 y puerto destino 5004



Formato de Mensaje RTP



Contenido Multimedia

Profiles y payload types

Perfil (profiles): contextos en los que se define de forma más detallada cómo usar RTP

- **Ejemplos:**
  - **RTP/AVP (Audio Video Profile): Perfil genérico para conferencias de audio/vídeo con mínimo control (RFC3551: "RTP Profile for Audio and Video Conferences with Minimal Control")**
  - **RTP/I Para *whiteboards*, y medios interactivos**
  - **Para transporte de audio y vídeo con retransmisión, …**
- **Todas las aplicaciones que pertenecen a una misma sesión RTP deben utilizar el mismo perfil**
- **Un perfil define**
  - **Números de payload, que tienen significados distintos en cada perfil**
    - ✓ **Se 'reutiliza' el mismo número en distintos perfiles, con significados diferentes**
  - **Cómo se empaqueta la información multimedia para cada tipo de payload (cómo se calcula el Timestamp, cuántos datos meter por paquete…)**
    - ✓ **También puede haber documentos específicos en los que se especifica el payload format**
  - **Incluso refinar el comportamiento de RTP/RTCP en ese perfil**
    - ✓ **Cambiar intervalo de generación de paquetes RTCP**
    - ✓ **Añadir nuevos tipos de paquete RTCP**
    - ✓ **Definir extensiones para RTP**
    - ✓ **...**
- **Nótese que NO hay un 'identificador de perfil' que viaje con los paquetes**

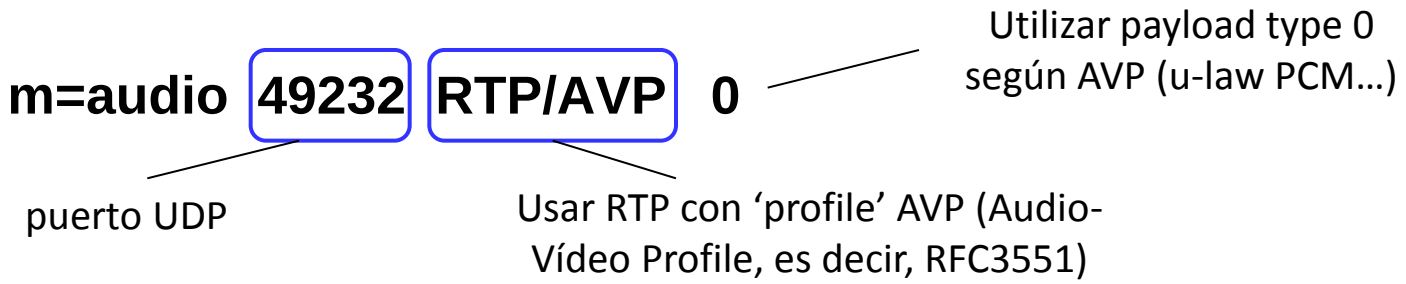
Algunos documentos de “profiles”

RTP Profile for Audio and Video Conferences with Minimal Control (RFC 3551)
The Secure Real-time Transport Protocol (SRTP) (RFC 3711)
Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF) (RFC 5124)
Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) (RFC4585)

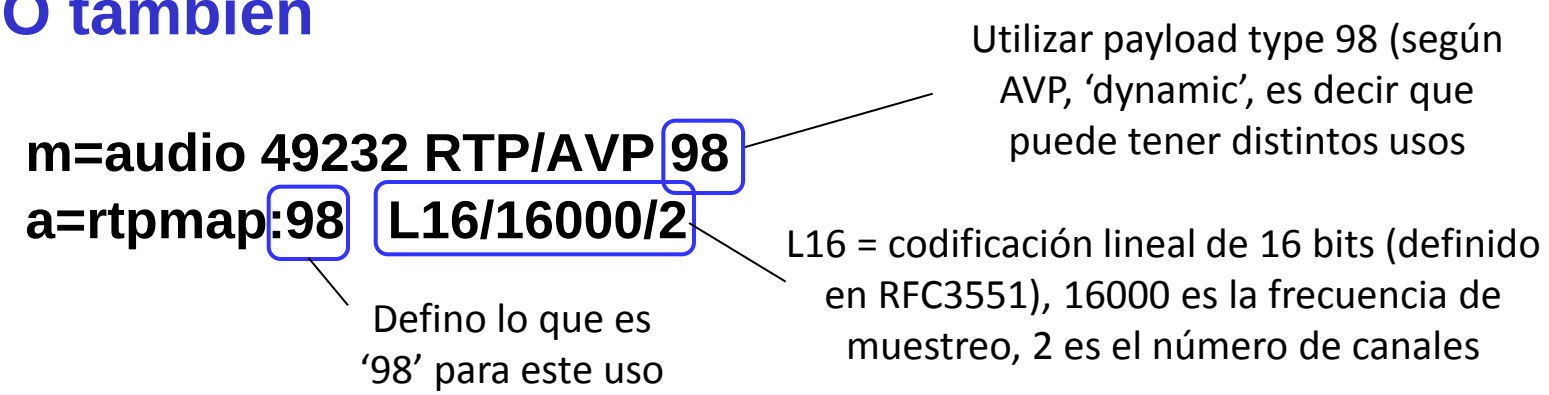
RTP Payload Format for JPEG-compressed Video (RFC 2035) obsoleted by RFC 2435
RTP Payload format for H.261 video streams (RFC 2032)
RTP Payload Format for MPEG1/MPEG2 Video (RFC 2038) obsoleted by RFC 2250
RTP Payload Format of Sun's CellB Video Encoding (RFC 2029)
RTP Payload Format for H.263 Video Streams (RFC 2190)
RTP Payload for Redundant Audio Data (RFC 2198)
RTP Payload Format for MPEG1/MPEG2 Video (RFC 2250)
RTP Payload Format for Bundled MPEG (RFC 2343)
RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+) (RFC 2429)
RTP Payload Format for BT.656 Video Encoding (RFC 2431)
RTP Payload Format for JPEG-compressed Video (RFC 2435)
An RTP Payload Format for Generic Forward Error Correction (RFC 2733)
RTP Payload for Text Conversation (RFC 2793)
RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals (RFC 2833)
RTP Payload Format for Real-Time Pointers (RFC 2862)
RTP payload format for MPEG-4 Audio/Visual streams (RFC 3016)
RTP Payload Format for ITU-T Recommendation G.722.1 (RFC 3047)
A More Loss-Tolerant RTP Payload Format for MP3 Audio (RFC 3119) (37796 bytes)
RTP Payload Format for DV Format Video (RFC 3189) (25991 bytes)
RTP Payload Format for 12-bit DAT, 20- and 24-bit Linear Sampled Audio (RFC 3190) (34977 bytes)
RTP payload format and file storage format for the Adoptive Multi-Rate (AMR) and Adaptive Multi-RTP Payload for Comfort Noise (RFC 3389) (17018 bytes)
…

Negociación de perfiles y ‘payload types’

- Se pueden utilizar distintos protocolos para negociar el perfil (nota: ‘fuera’ de RTP)
- SDP (Session Description Protocol, RFC 4556), define un formato en el que se especifican los parámetros de la sesión multimedia

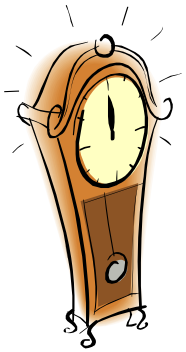


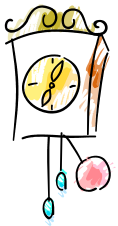
- O también



Formato de Mensaje RTP

- **Marca de tiempo (*timestamp*):**
  - **Permite sincronización y cálculo de *jitter* de paquete (útil para RTCP)**
  - **… ¿con qué formato? (con qué precisión…)**
    - ✓ **Ej: poner texto de subtítulos en instante 19.025 s, siguiente en 27.145 s**
    - ✓ **Empezar a reproducir paquete de audio en 0 s, siguiente en 1/3 de segundo, luego 2/3 de segundo…**
      - ➢ 0.33333333333333 s ?
  - **Diferentes formatos de marca de tiempo en función de perfil y payload**
  - **No debe requerir sincronización entre relojes: sólo indica valores de tiempo relativos entre paquetes**
    - ✓ **Difícil requerir sincronización de *offset* entre relojes (razonable pedir una frecuencia similar)**

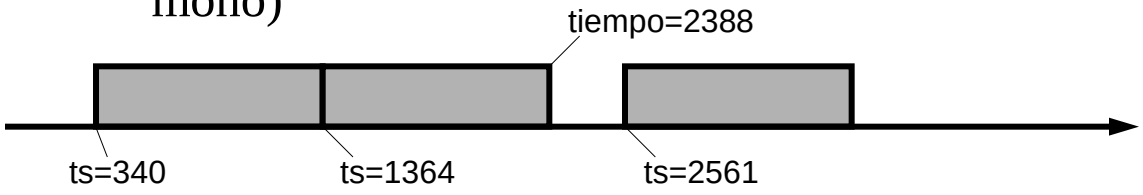




Formato de Mensaje RTP

- **Marca de tiempo (*timestamp*): 32 bits**
  - **Como es dependiente del perfil, consideremos ejemplo de 'audio' en RFC3551**
    - ✓ ***Timestamp* indica el instante de reproducción del primer octeto de datos contenido en el paquete RTP**
    - ✓ **Emplea el reloj (virtual) de muestreo del audio (NO reloj del sistema operativo, ni CPU)**
      - No tiene sentido pedir a la tarjeta de sonido un dato "en medio" de un periodo de muestreo. Tampoco iniciar una reproducción ahí (después de un silencio, por ejemplo)

      - Ejemplo: en audio sin comprimir el tiempo pasa en 1 unidad en **cada periodo de muestreo**
        - Suponer paquete de 2048 bytes, 2 bytes por muestra (16 bits mono)

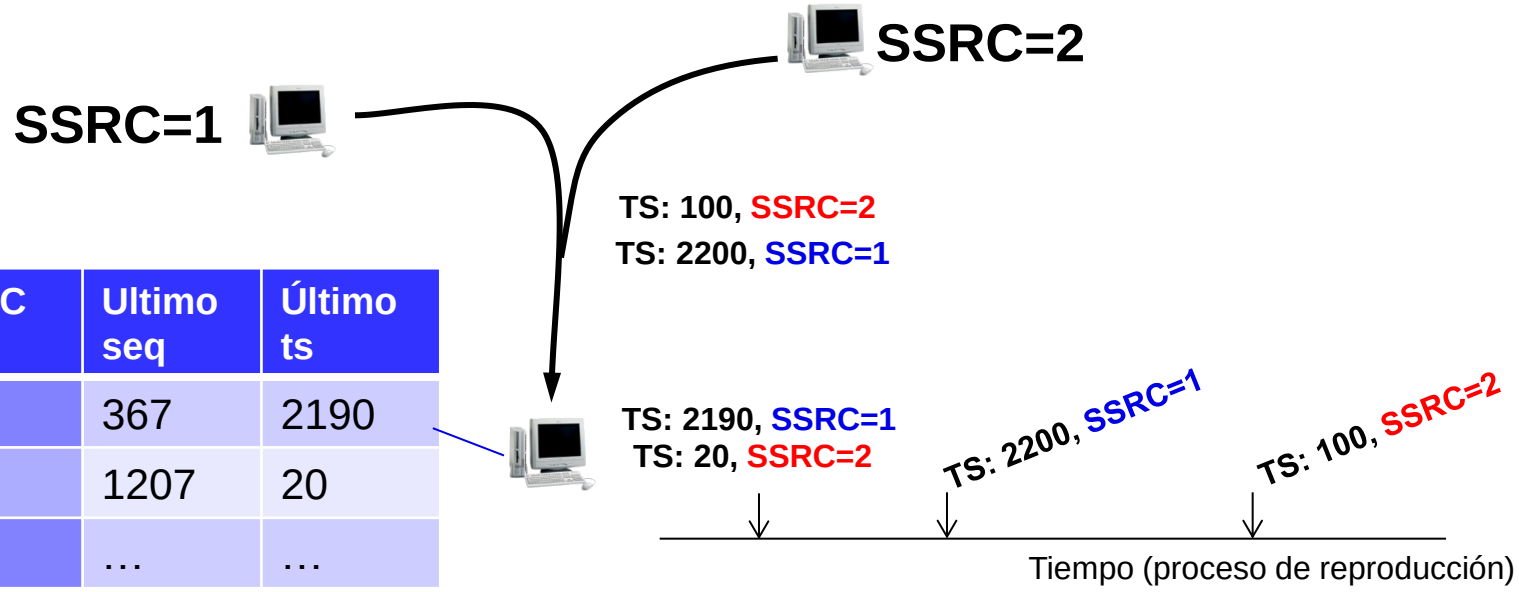


Formato de Mensaje RTP

- **Marca de tiempo (*timestamp*): 32 bits**
  - **Valor inicial es aleatorio (como el número de secuencia)**
  - **Ejemplos de otros perfiles**
    - ✓ **En video, paquetes del mismo frame (porque no quepan en un único paquete), llevan igual timestamp**
    - ✓ **MPEG marca de forma no monótona creciente (interpolados se envían después)**

Formato de Mensaje RTP

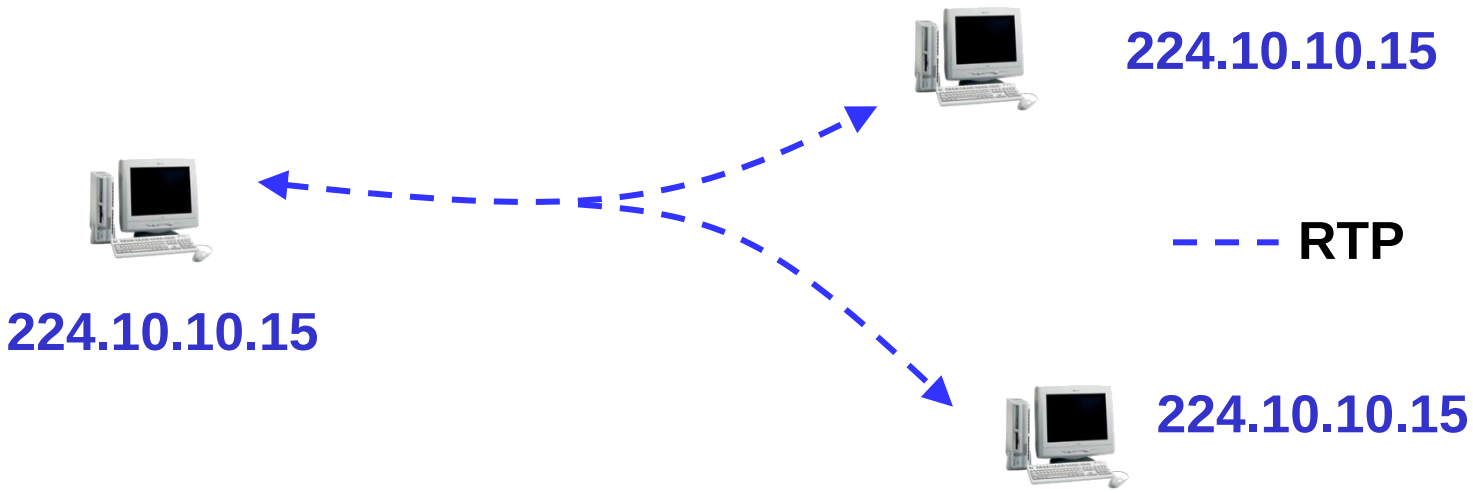
- **SSRC (*Synchronization Source Identifier*): 32 bits**
  - **Generado aleatoriamente por la fuente**
  - **Identificador único a nivel RTP de una fuente para una sesión**
  - **Permite mantener la temporización de la reproducción de cada fuente**
    - ✓ **Los números de secuencia también tienen sólo sentido para una fuente dada**



SSRC	Ultimo seq	Último ts
1	367	2190
2	1207	20
…	…	…

“Sesión ligera” en RTP

- Las sesiones RTP son “muy” ligeras, como es el control de IP multicast
  - Cualquiera puede unirse sin que nos enteremos
    - ✓ Si queremos restringir, puede hacerse cifrando y distribuyendo las claves con otros protocolos de nivel superior
  - El emisor no sabe cuántos destinatarios hay
    - ✓ ¡Puede ocurrir que no haya ninguno!
- Parece conveniente dar algún tipo de realimentación a los participantes – sobre todo a los emisores => RTCP



RTCP (Real Time Control Protocol)

Funciones

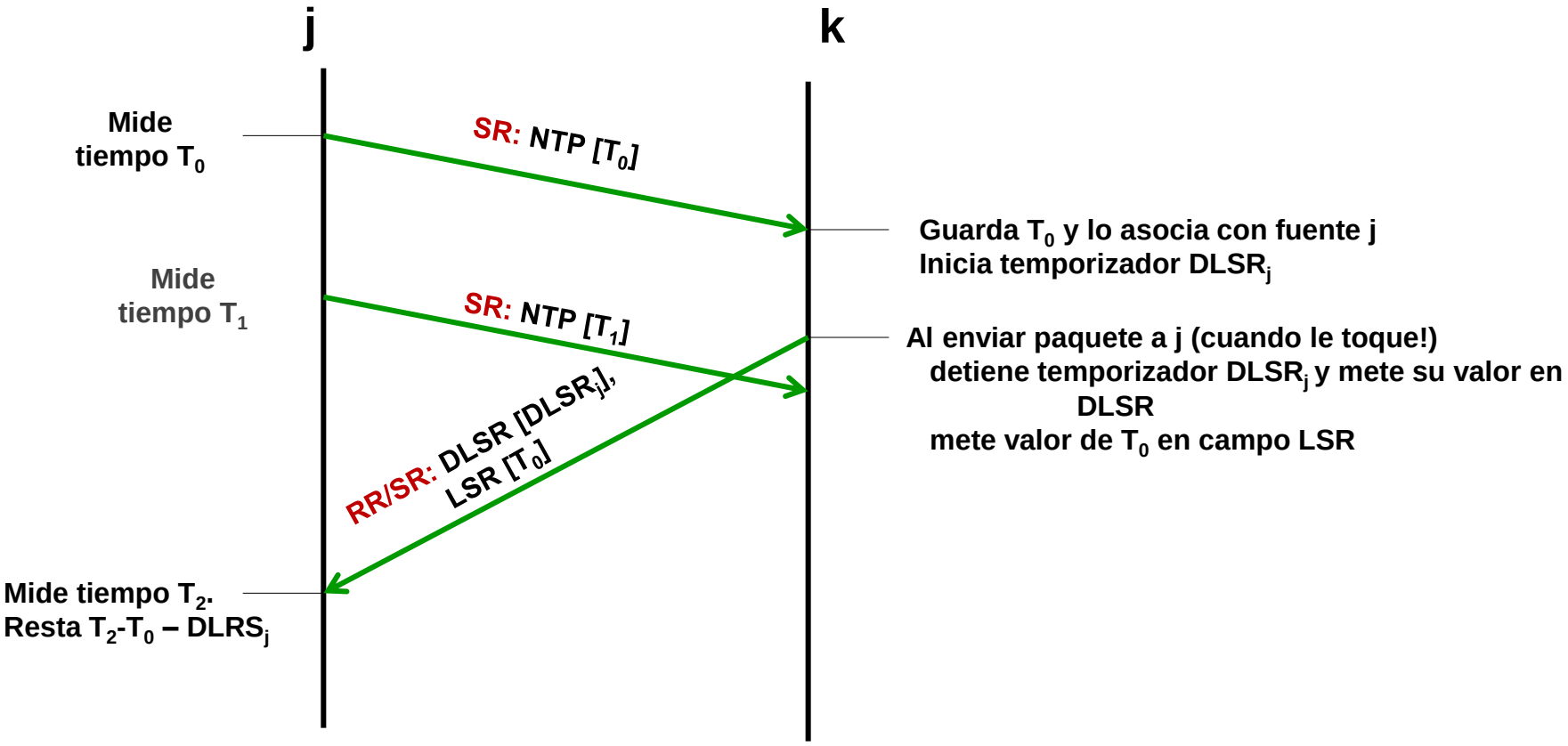
- **Monitorización de la congestión y del QoS**
  - Se envían informes sobre lo recibido de cada participante en una sesión
  - Permite medir jitter, y RTT, y calcular tiempo de buffering
- **Identificación**
  - Identificador de la fuente mediante una cadena de caracteres
  - Puede utilizarse para asociar flujos de diferentes sesiones
  - También permite enviar el nombre del usuario y otros parámetros textuales de tipo informativo
- **Control de sesión mínimo**
  - Abandono de sesión
  - Conocer el número de participantes

Paquetes RTCP

Cinco tipos

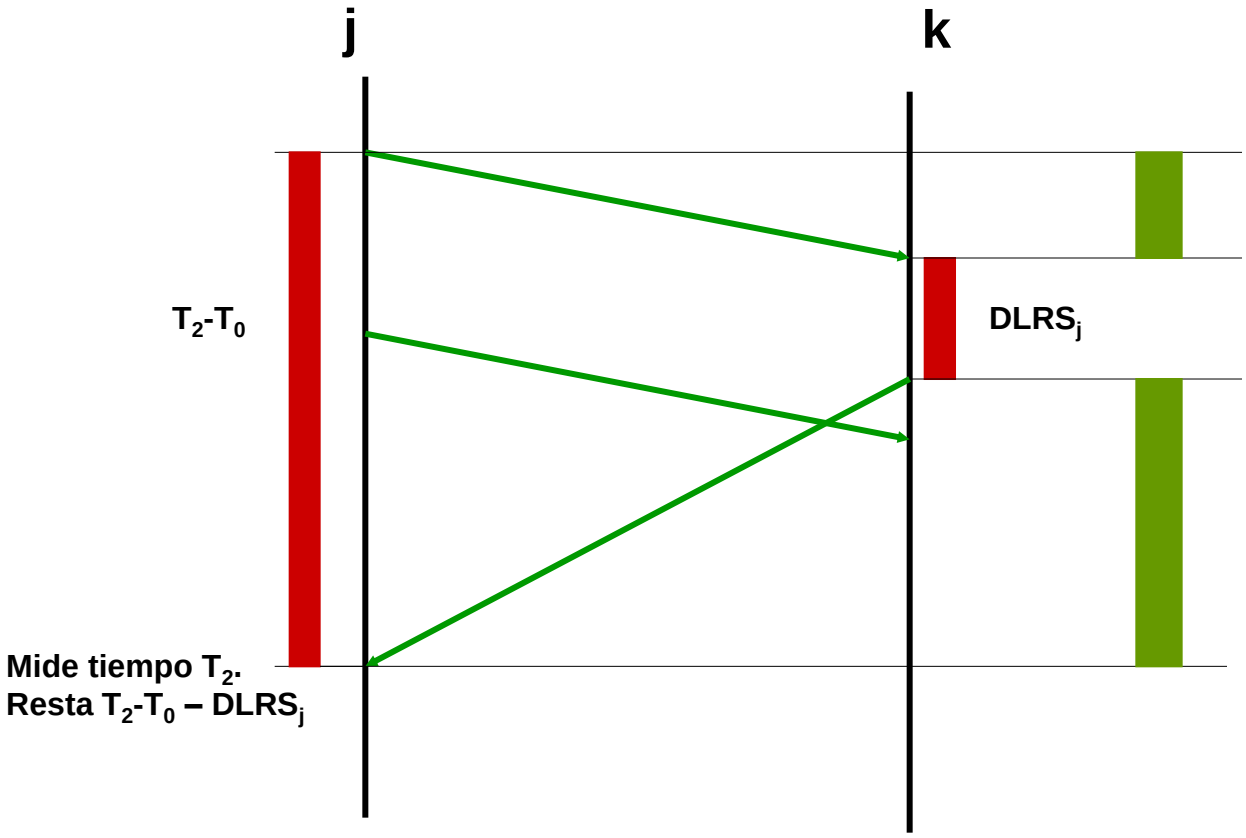
- Sender Report (SR)
  - Indica lo que ha enviado el nodo (y deberían haber recibido los demás) + lo que ha recibido el nodo
- Receiver Report (RR)
  - Informa sobre lo que ha recibido el nodo
- Source Description (SDES)
  - Descripción de la fuente: CNAME (*canonical name*), NAME, EMAIL, PHONE, LOC, TOOL, ...
- BYE
  - Opcional, indica que un equipo abandona la sesión, y la razón por la que la abandona
- Específico de aplicación
  - Opcional, permite añadir funcionalidades sin necesidad de definir nuevos paquetes en el protocolo

Cálculo del *Round Trip Time*



- ‘j’ es un ‘sender’, ‘k’ puede ser cualquier cosa
  - ’k’ manda información relevante en Rec Report Block
- Cada emisor recibe estimación de RTT con cada uno de los miembros de la sesión

Cálculo del *Round Trip Time*



- ‘j’ no tiene que almacenar estado
- ‘k’ sólo guarda por emisor RTP:
  - valor del campo NTP del último paquete recibido,
  - tiempo en el que recibió el último paquete

RTCP y Round Trip

- **Cálculo del Round Trip entre fuente j y k**
  - **j envía mensaje RTCP SR a k**
    - ✓ **instante T_0 codificado en NTP (64 bits)**
  - **k envía un tiempo después mensaje RTCP RR a j**
    - ✓ **Delay since Last SR ($DLSR_j$): tiempo desde que se recibió el mensaje de j hasta que k envía de vuelta a j**
    - ✓ **Time of Last Sender Report (LSR): tiempo de generación del primer mensaje en j según j (ahora en otro formato: 32 bits centrales del NTP)**
  - **Round trip: j recibe paquete de k en T**
    - ✓ **Round trip = T-LSR - DLSR**

Estado en los receptores

- **Cada receptor debe mantener estado PARA CADA FUENTE de la que ha recibido 'recientemente'**
  - **SSRC de la fuente,**
  - **RTP timestamp del último paquete recibido, asociado a un tiempo 'local' en el que se debe haber reproducido**
    - ✓ **Si no, no sabrá cuándo tiene que reproducir los contenidos de los paquetes de datos**
  - **Número de secuencia del último paquete recibido y del último paquete enviado antes de enviar el último mensaje SR o RR**
    - ✓ **Para poder calcular 'fraction lost', y saber si el siguiente paquete esperado se ha perdido o no…**
  - **DLSR asociado al último mensaje RTCP recibido de la fuente**
  - **Tiempo NTP recibido en el último mensaje RTCP recibido de la fuente**
  - **…**

Puertos en paquetes UDP

- RFC 3550 hace referencias sobre el puerto de destino de un paquete RTP o RTCP
  - Ej: dice que por defecto para RTP sea 5004,
  - En cualquier caso recomienda que el puerto para RTP sea par, y el de RTCP = RTP+1
  - El puerto origen podría ser cualquiera
- Sobre el puerto origen, RFC4961 sugiere que sea igual que el destino (tanto para RTP como para RTCP)
  - Ej: si capturáramos un paquete "en tránsito":
    - ✓ Dir orig: 163.117.140.180, puerto orig: 5004, dir dest: 225.10.0.3, puerto dest: 5004