# *Video Streaming*

## - Introducción -

Area de Ingeniería Telemática
http://www.tlm.unavarra.es

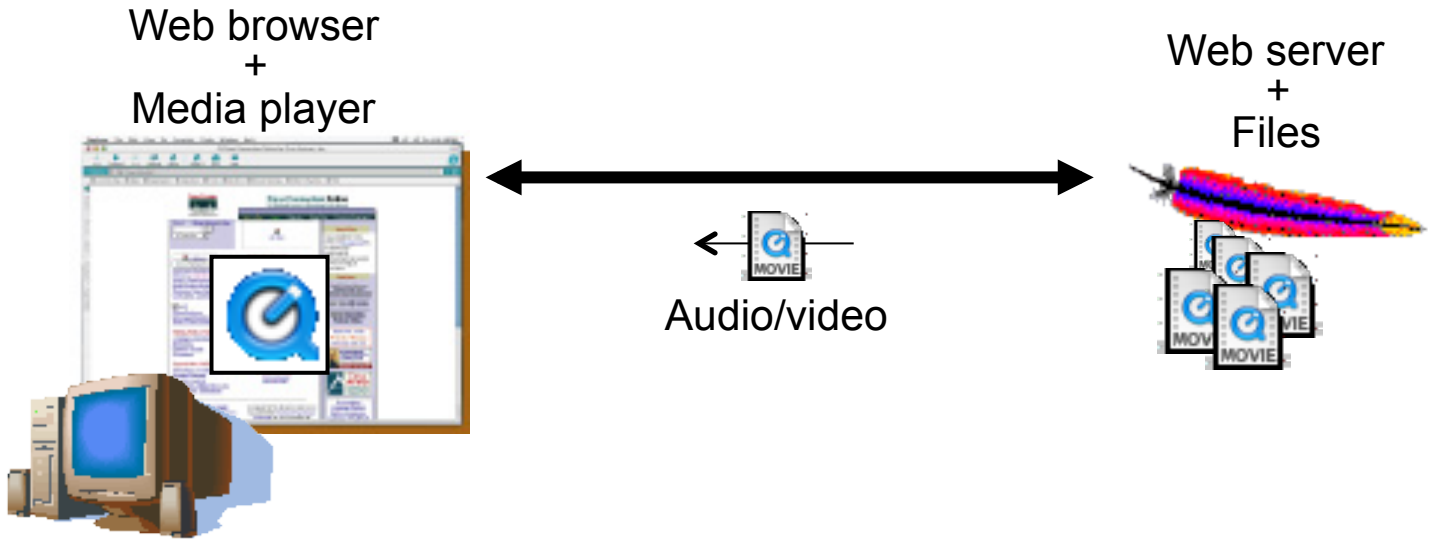
Grupo de Redes, Sistemas y Servicios Telemáticos

# Contenido

- ¿Streaming?
- Service architecture
- Video characteristics
- Network requirements

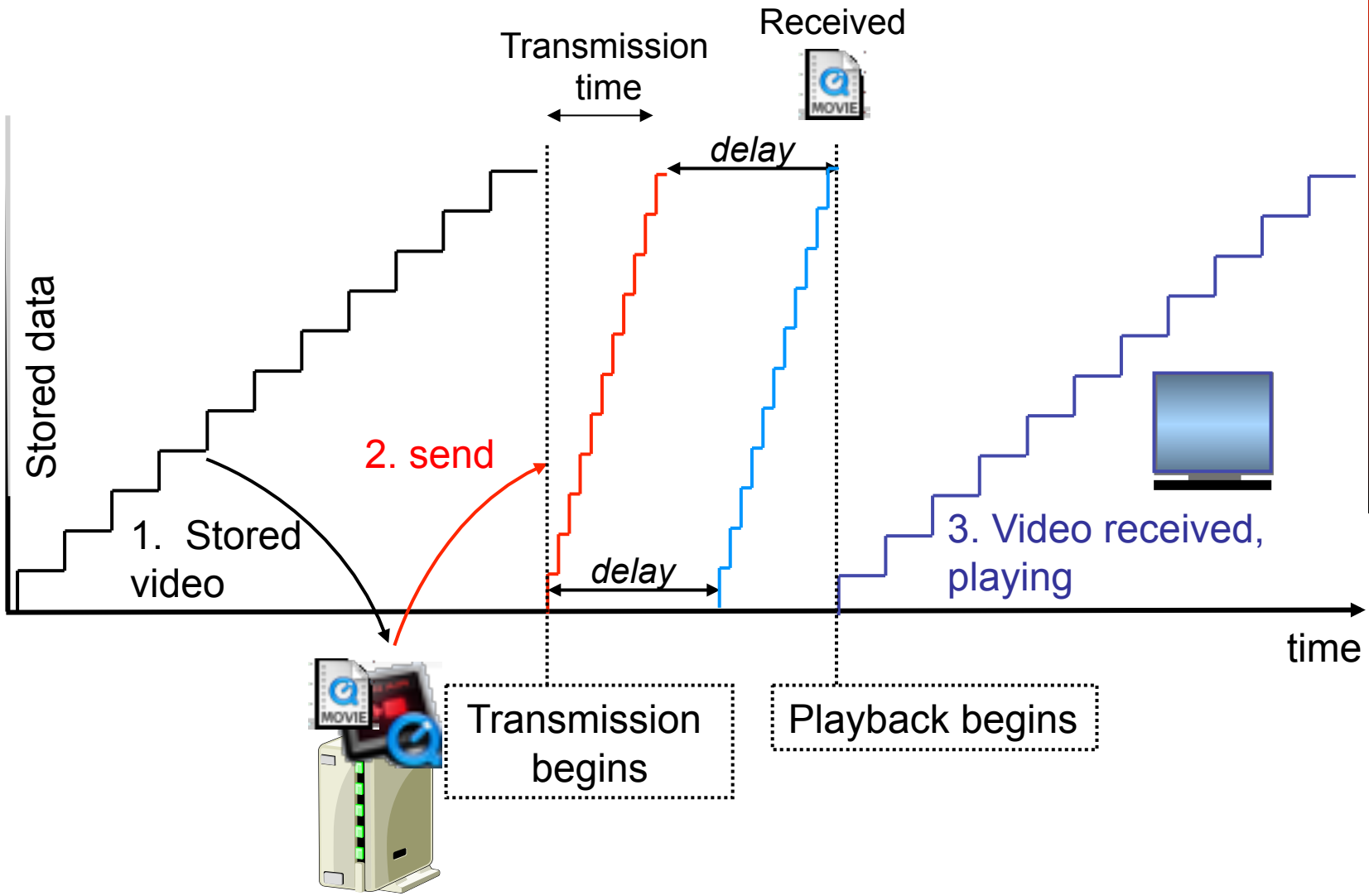
# ¿*Streaming*?

# Download+play vs “Streaming”

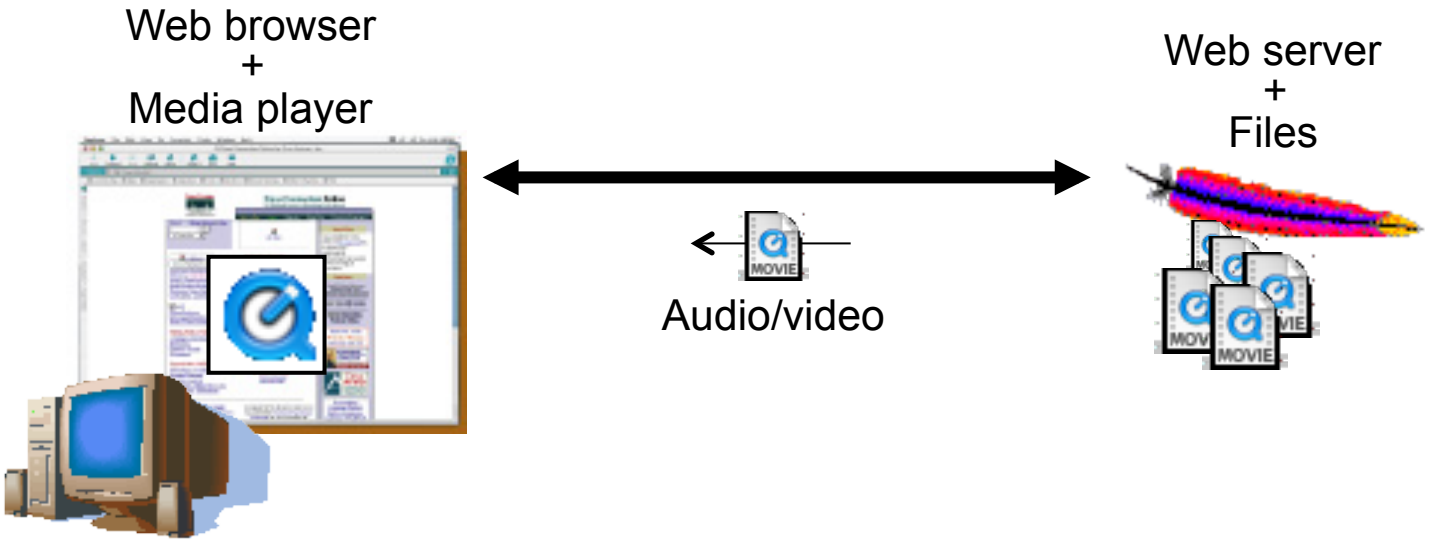



1. Download+play: file downloaded completely in advance (…)
2. Pseudo-“Streaming”
3. Streaming

# Download+play

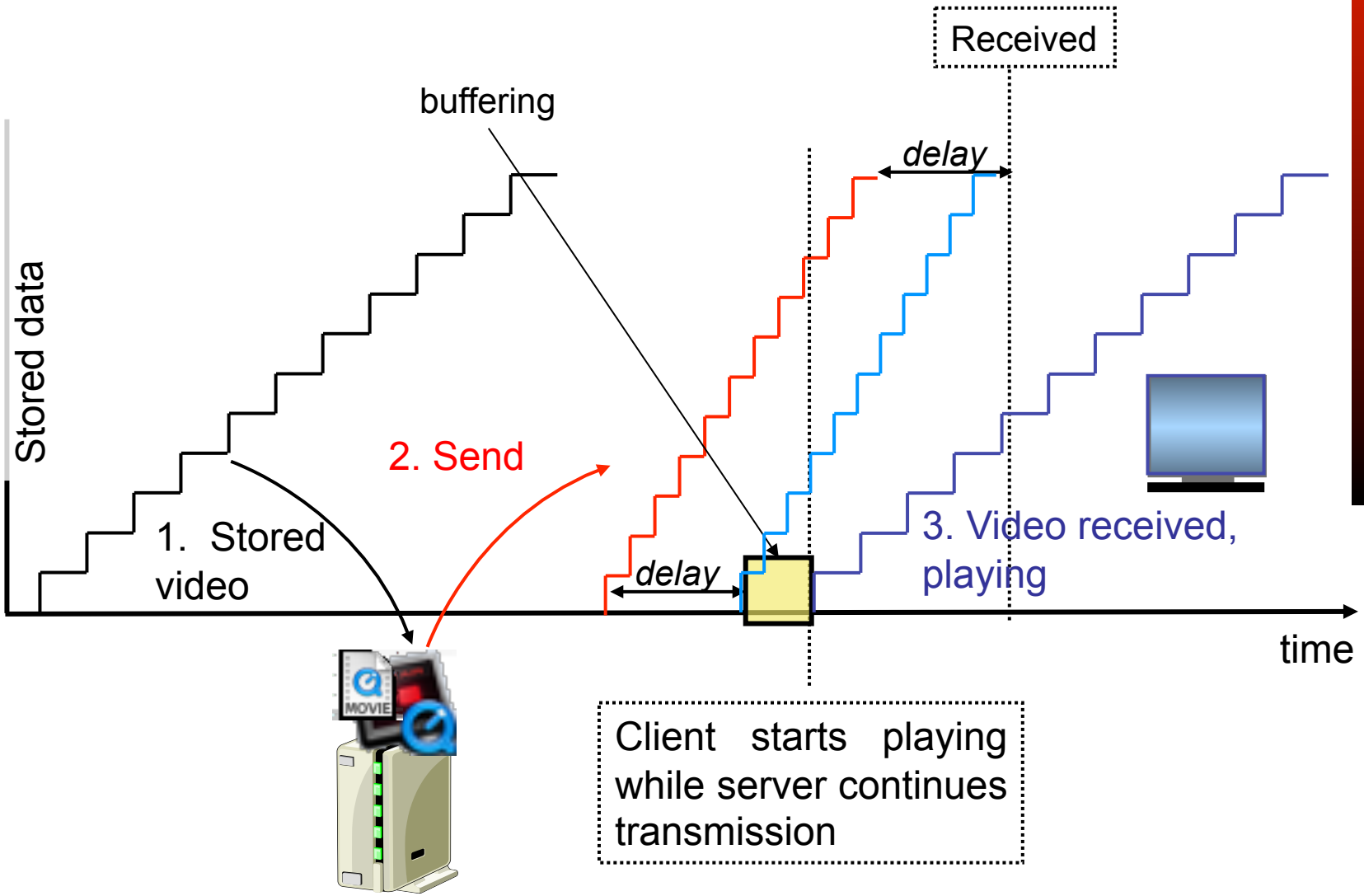


# Download+play vs “Streaming”

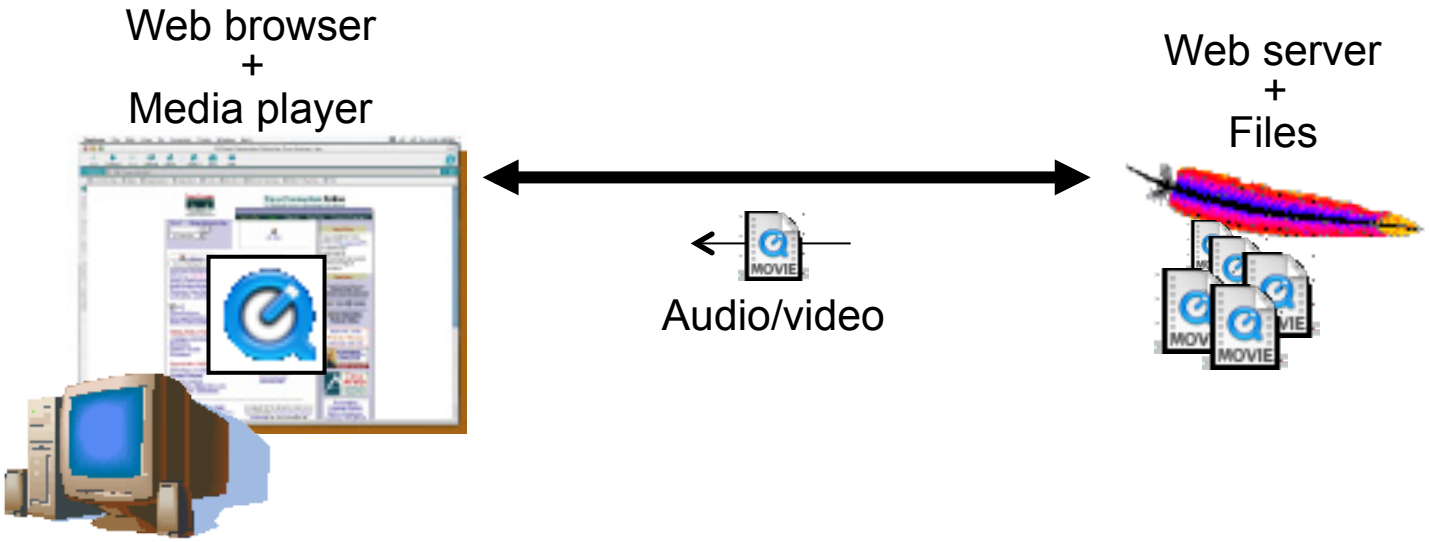



1. Download+play: file downloaded completely in advance
2. Pseudo-“Streaming”: data decoded at the player as soon as it is received (play-as-you-get) (…)
3. Streaming

# Play-as-you-get

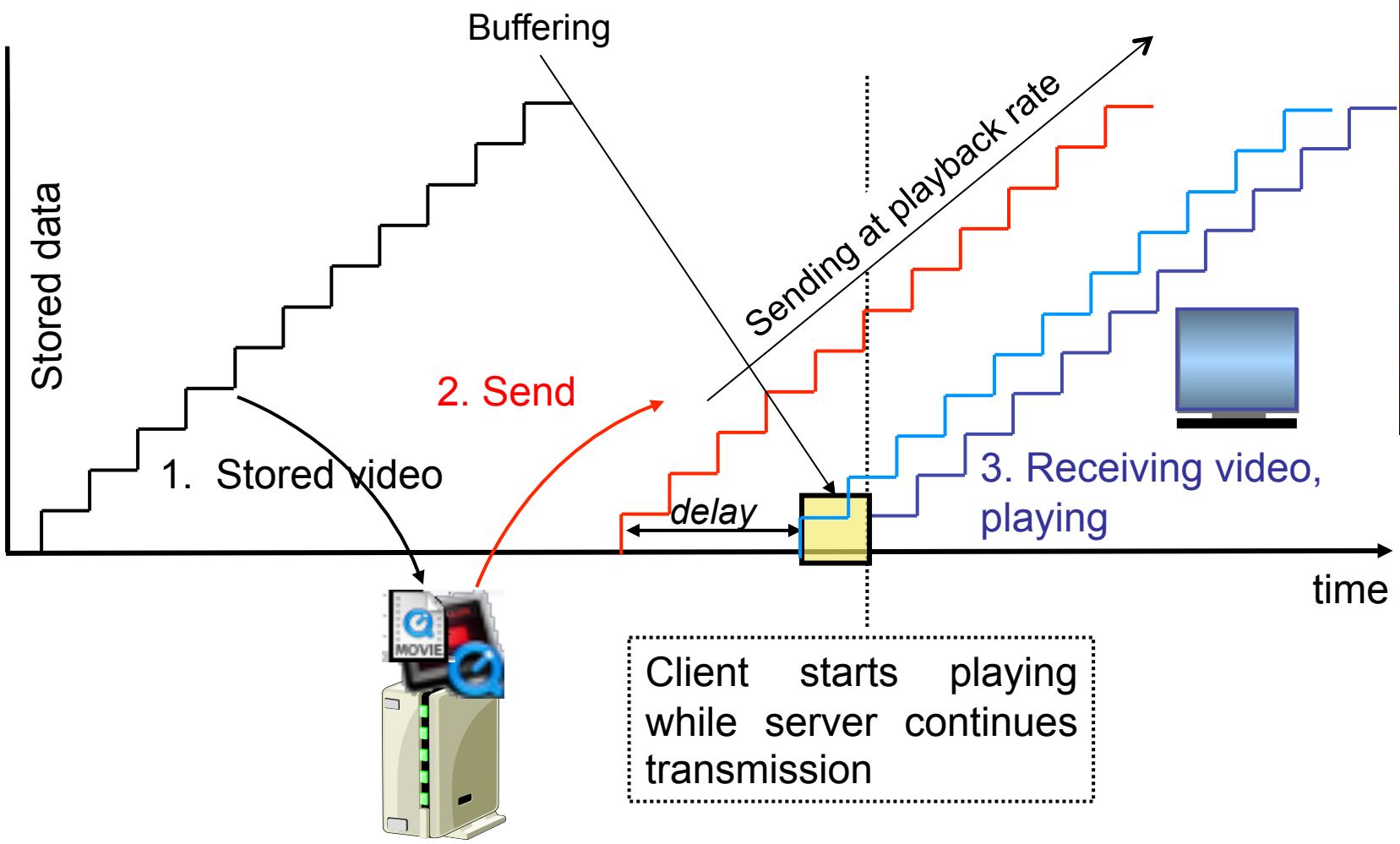


# Download+play vs “Streaming”

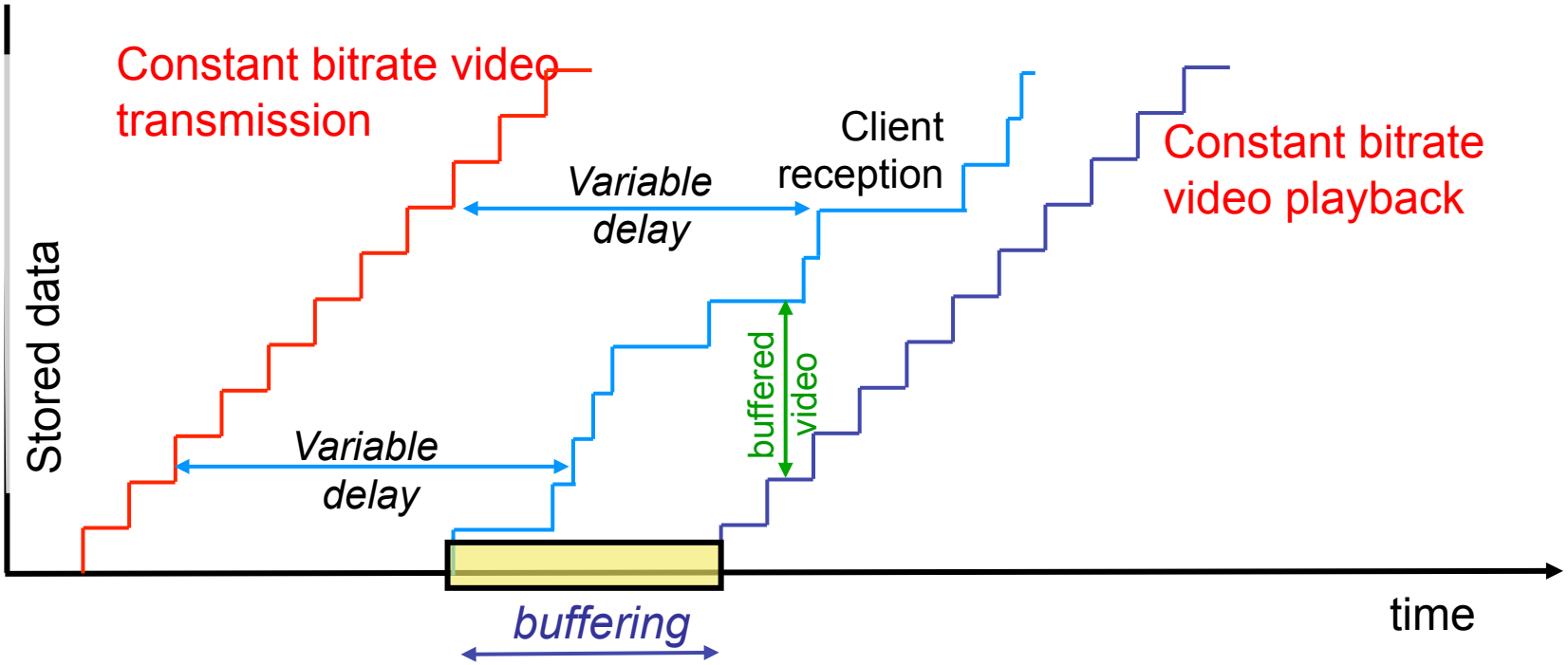



1. Download+play: file downloaded completely in advance
2. Pseudo-“Streaming”: data decoded at the player as soon as it is received (play-as-you-get)
3. Streaming: transmission rate close to playback rate (…)

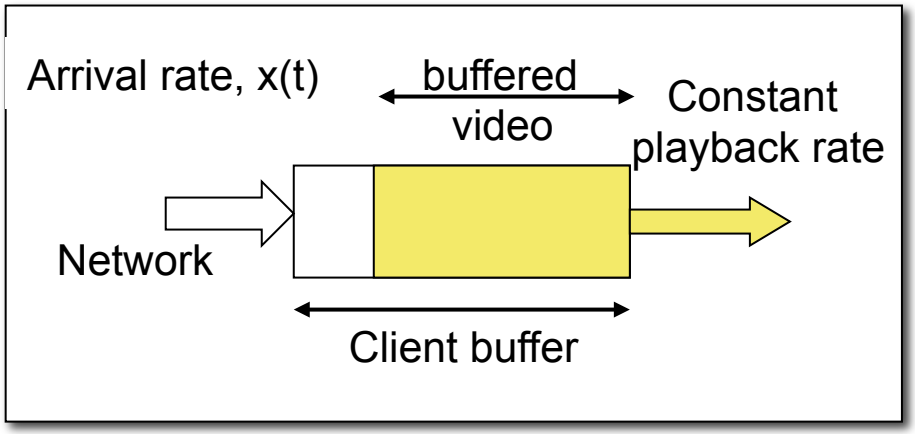
# Streaming


Buffering
Stored data
Sending at playback rate
2. Send
1. Stored video
delay
3. Receiving video, playing
time
MOVIE
Client starts playing while server continues transmission

# Streaming: Client Buffering

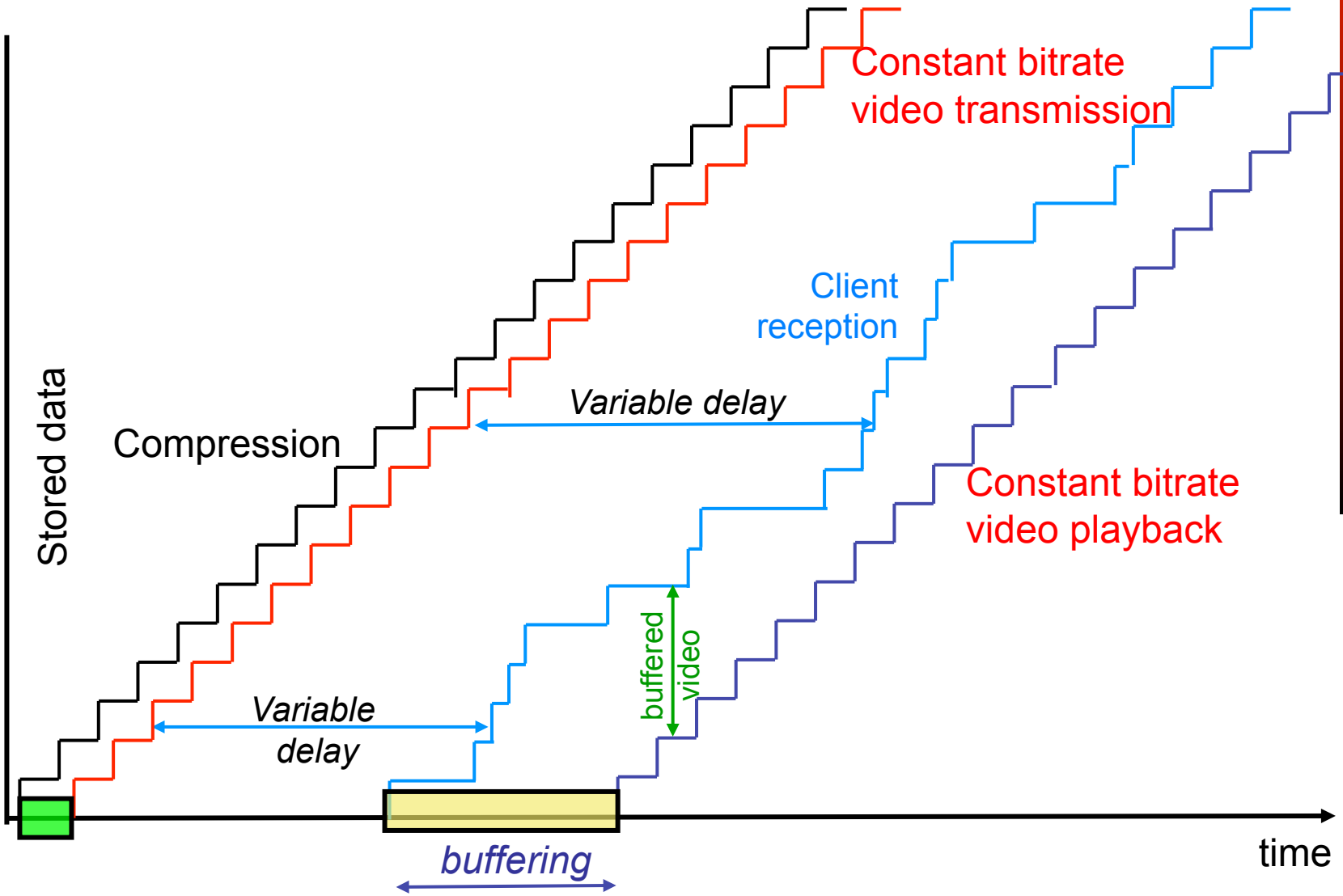



- Client buffers part of the video before starting playback
- It allows avoiding buffer exhaustion due to variable delay (*jitter*)
- If the buffer empties the player stops



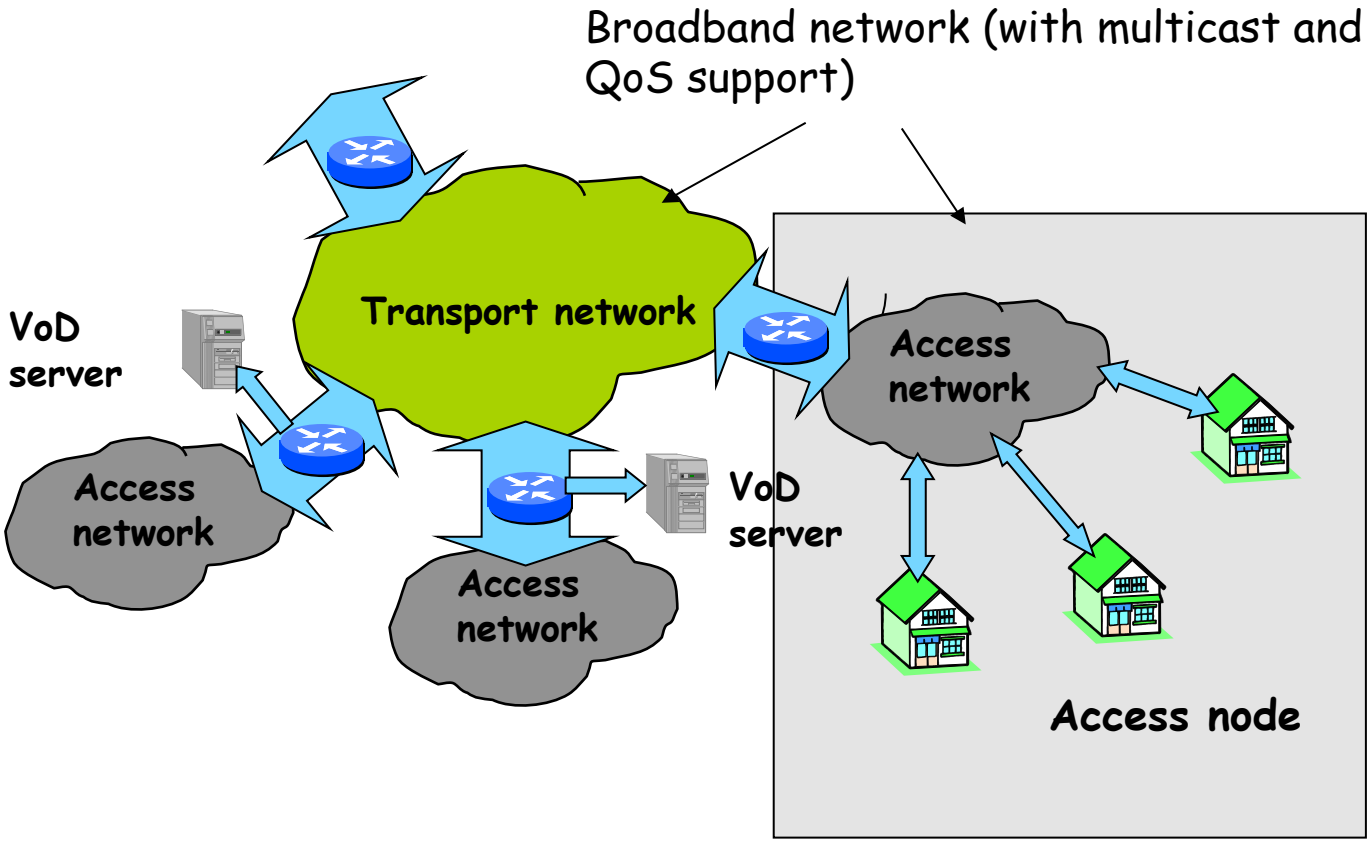
# Live Streaming

- Fast forward is impossible



# Service Architecture

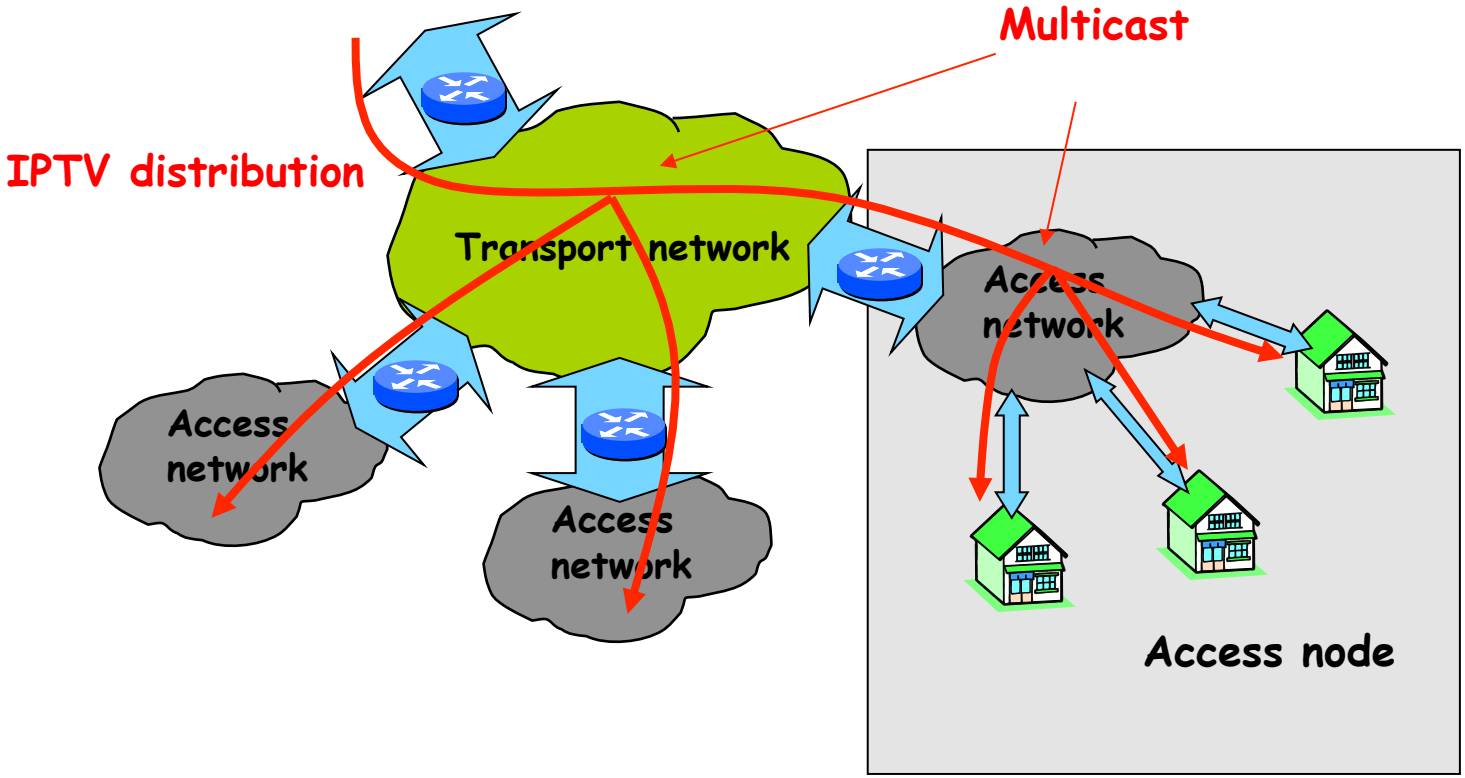
# General architecture



# TV broadcast

## Multicast for TV broadcast

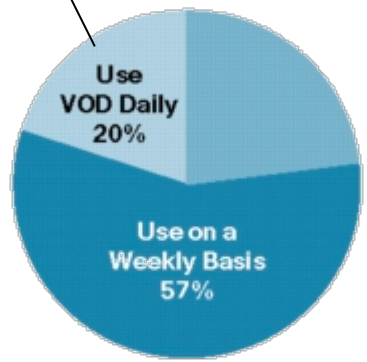
- Scalability for a large number of channels and users
- Optimum bandwidth use



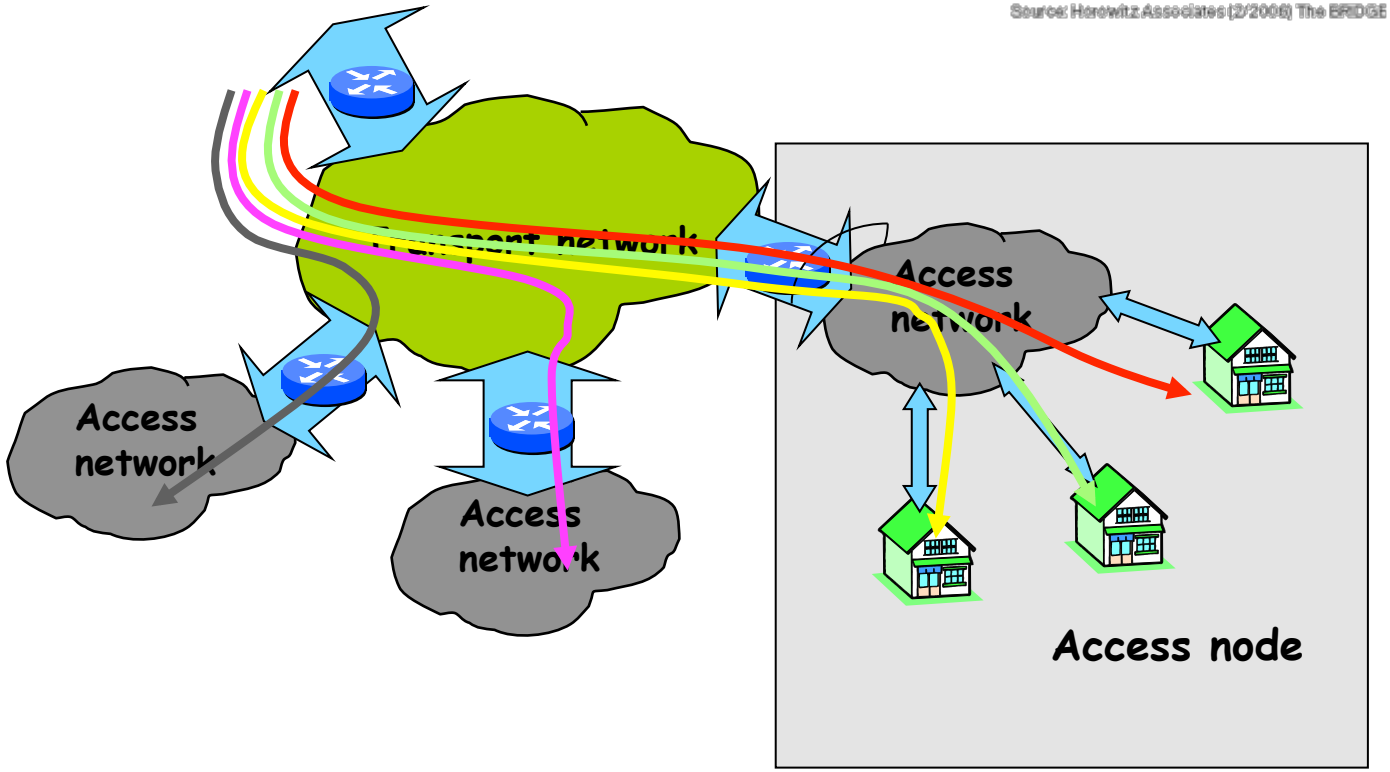
# VoD

## VoD is usually Unicast

- 1M users→ 200K/day @ 2Mb/s & 20% concurrency = 80Gb/s
- Centralized architecture offers limited scalability



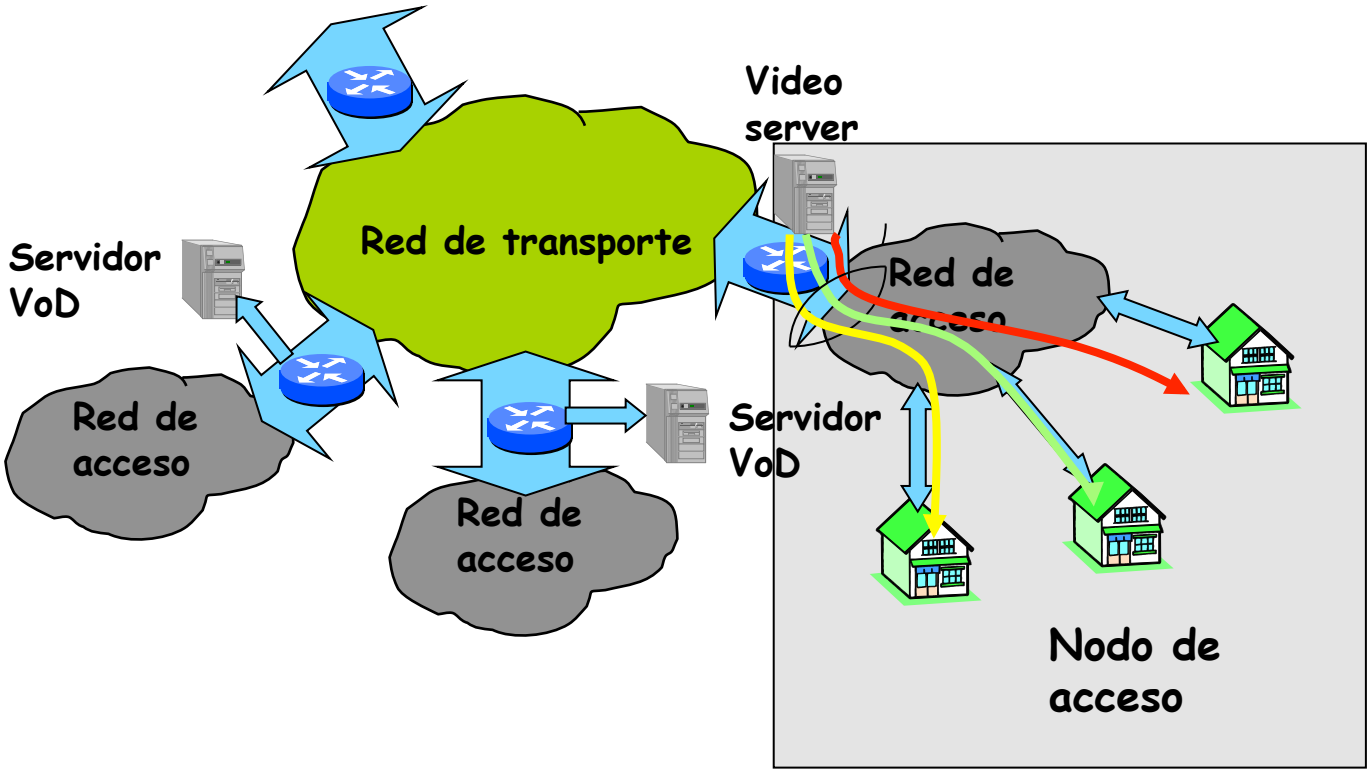

Source: Horowitz Associates (2/2006) The BRIDGE 2006



# VoD

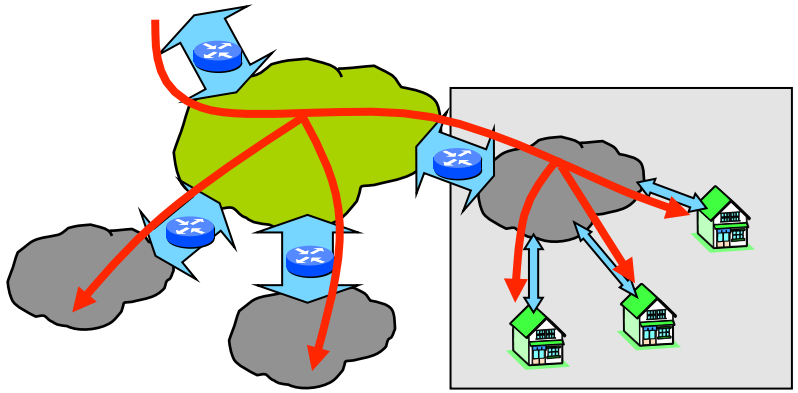
## Distributed architecture for VoD

- 20k users @ 2Mb/s & 20% concurrency = 8Gb/s
- Distributed architecture offers scalability
- A mixed architecture (centralized/distributed) is also feasible



# ¿Needs?

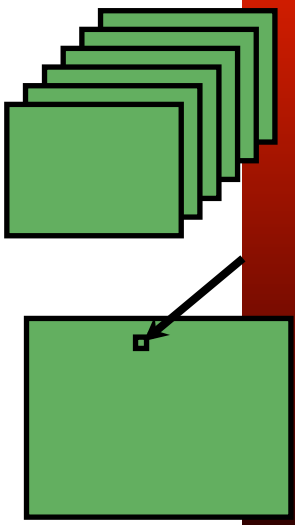
- Server:
  - Scalability
- Network:
  - Scalability ⇒ Multicast
  - Quality ⇒ QoS (technology?)
- Design:
  - Characteristics of video traffic?
  - User behaviour?



# Characteristics of video traffic

# Digital video

- Around 25 frames/sec for movement (24-30)
- Each frame is a digital image to transmit
- Quality depends on
  - Resolution (pixels)
  - Colours (number of bits to represent the colour, RGB or luma/chroma)
- Example:
  - PAL analog video
  - 625 lines (visible only 576) aspect rate 4:3, 25 fps
  - 768x576 pixels, each pixel one out of 16million colors (8:8:8 bits RGB)
  - 1300 KB per frame or **265Mbps one single TV channel**
- Compression is needed
  - Constant quality: variable bitrate
  - Constant bitrate: variable quality



# Characteristics

**Audio**

- CD: 1.411 Mbps
- MPEG-1 Part 3 Layer 3 (MP3): 96, 128, 160 kbps
- Internet telephony: 5.3-13 kbps
- For IPTV or VOD it is not a large percentaje of the traffic

**Video (+audio):**

- MPEG 1
  - Error free media (CD-ROM, VCD)
  - Video 1.2 Mbps, audio 256 Kbps
  - Best quality for a bitrate
  - Allows random access
  - Based on H.261, macroblocks and DCT in each block
- MPEG 2
  - Broadcast TV, DVD
  - 2-15 Mbps (video+audio)
  - Based on DCT but using temporal correlation for increasing compression
  - MPEG-2 part 7 advanced audio codec AAC
  - It defines the framing (transport stream TS) for network transport

# ITU-T H.26x

- ITU-T H.261 “Video codec for audiovisual services at px64 kbits”
  - CIF (352x288), QCIF (176x144) (for luma)
  - Bit stream
  - Videoconference over ISDN
- ITU-T H.263 “Video coding for low bit rate communication”
  - Based on H.261
  - sub-QCIF (128x96) , QCIF, CIF, 4CIF (704x576) and 16CIF (1408x1152)
  - Videoconference over POTS
  - Based on H.261, MPEG-1 and MPEG-2

# ITU-T H.262

- "Information technology - Generic coding moving pictures and associated audio information: Video"
- MPEG-2 Part 2
- Video over ATM and HDTV
- It offers *scalable video encoding* (layers adding refinement to the image)
- Frame types:
  - Intra Coded Pictures (I-Pictures): no reference to other frames
  - Predictive Coded Pictures (P-Pictures): they use *motion compensated prediction* based on the previous I- or P- Picture
  - Bidirectional-predictive Coded Pictures (B-Pictures): Computed related to previous and forward I- or P- Picture
  - The standard leaves flexibility for the sequence of frame types

# H.264

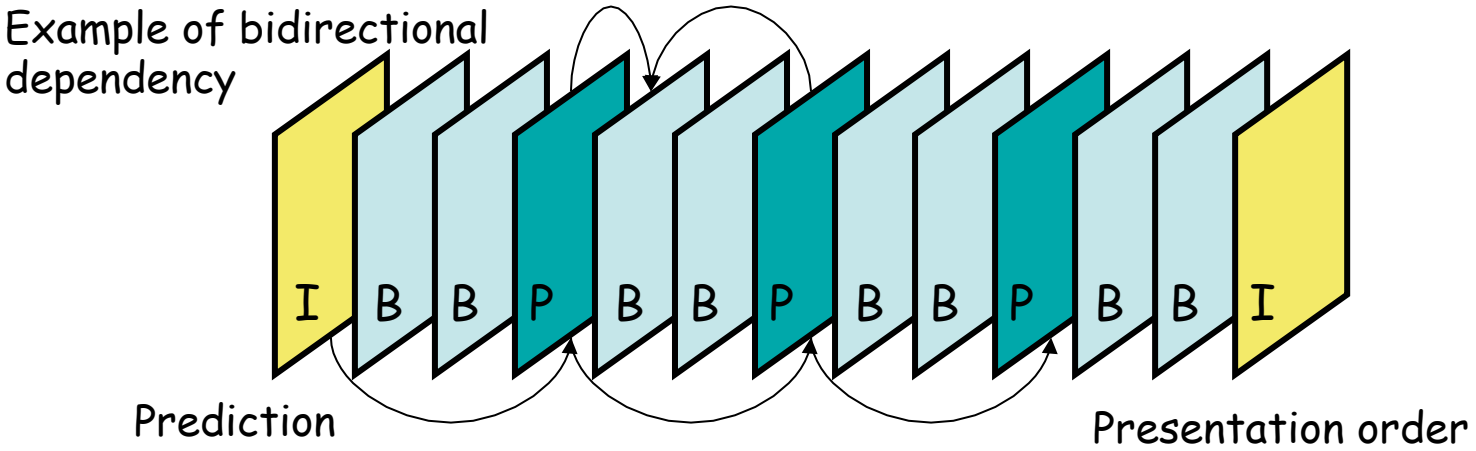
- “Advanced video coding for generic audiovisual services”
- Also MPEG-4 Part 10
- Similar quality to MPEG-2 but using half bitrate or less

| Use Scenario | Resolution & Frame Rate | Example Data Rates |
|---|---|---|
| Mobile Content | 176x144, 10-15 fps | 50-60 Kbps |
| Internet/Standard Definition | 640x480, 24 fps | 1-2 Mbps |
| High Definition | 1280x720, 24p | 5-6 Mbps |
| Full High Definition | 1920x1080, 24p | 7-8 Mbps |

# GoP structure

- *Group of Pictures*
- Usually around 1/2 sec per GoP
- Order
  - Presentation
    Ex.: IBBPBBPBBPBB ibbpbb…
  - Coding
    Ex.: I bb PBBPBBPBB i BB pbb…
- Closed or Open GoP
- Broken GoP: previous GoP is missing

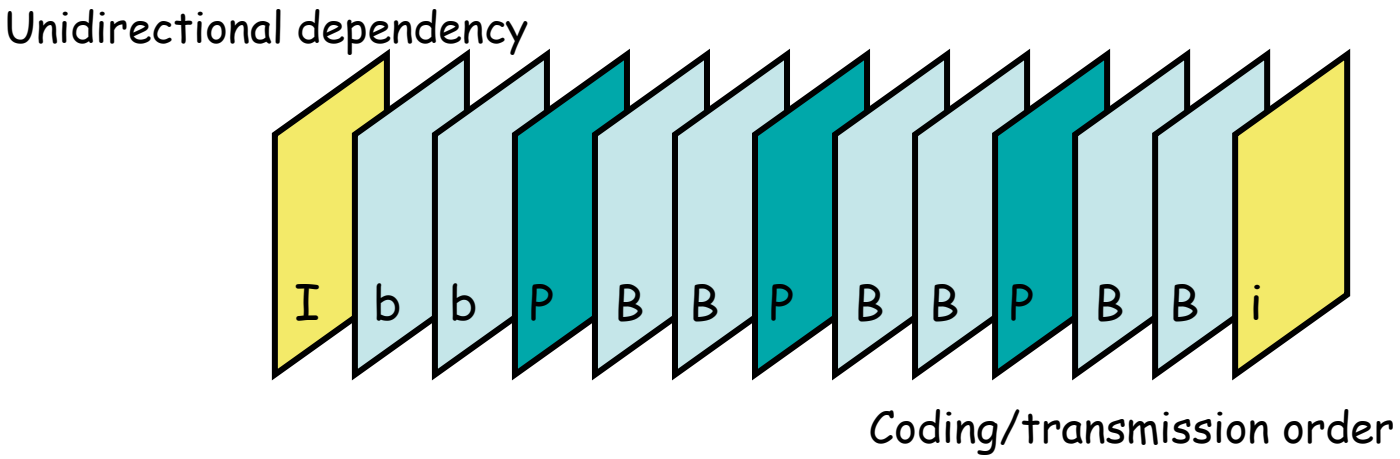
| | I frame | P frame | B frame |
|---|---|---|---|
| Compression Ratio | Low | Good | Best |
| Random Access | Best | Hard | Hardest |
| Complexity | Normal | High | Highest |



# GoP structure

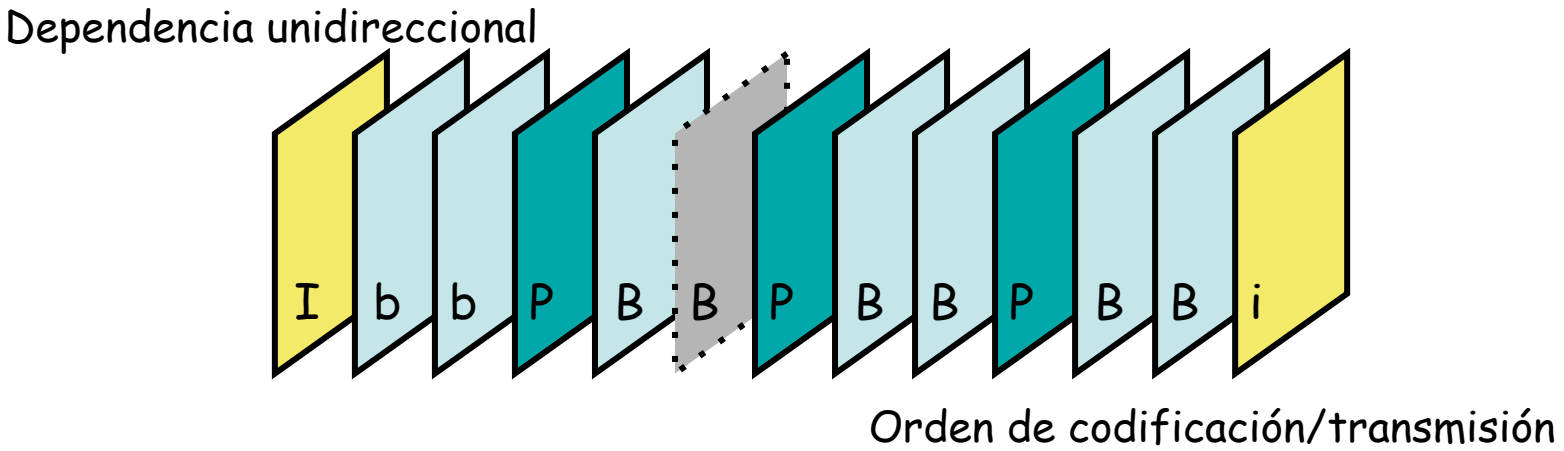
- *Group of Pictures*
- Usually around 1/2 sec per GoP
- Order
  - Presentation
    Ex.: IBBPBBPBBPBB ibbpbb…
  - Coding
    Ex.: I bb PBBPBBPBB i BB pbb…
- Closed or Open GoP
- Broken GoP: previous GoP is missing

| | I frame | P frame | B frame |
|---|---|---|---|
| Compression Ratio | Low | Good | Best |
| Random Access | Best | Hard | Hardest |
| Complexity | Normal | High | Highest |



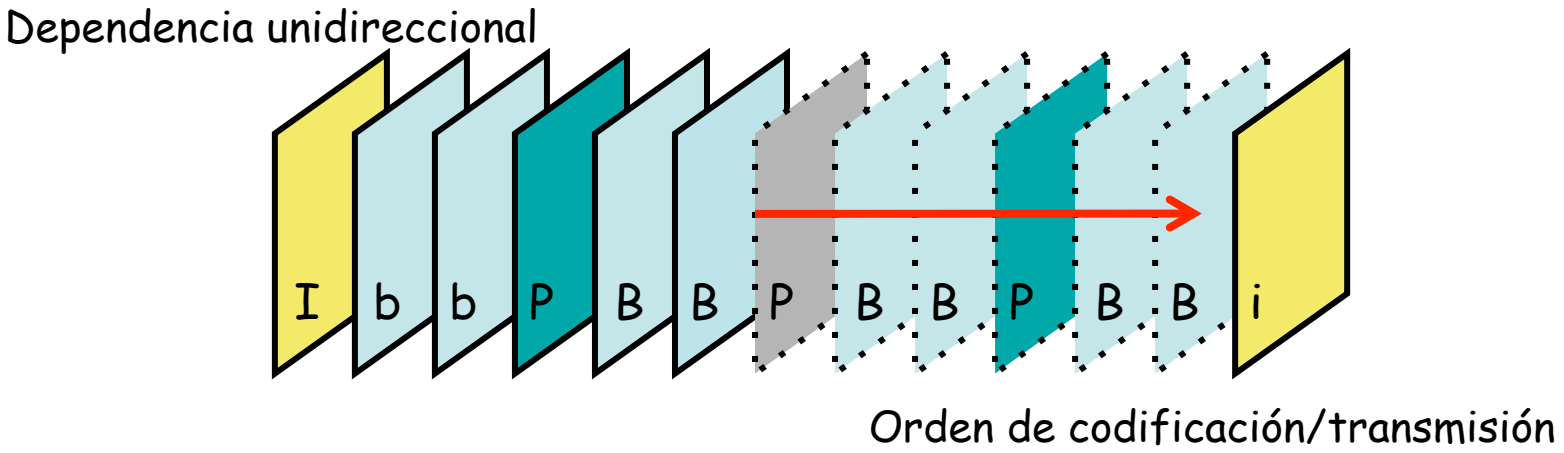
# GoP: losses

- A B-frame is lost
  - No effect on other frames



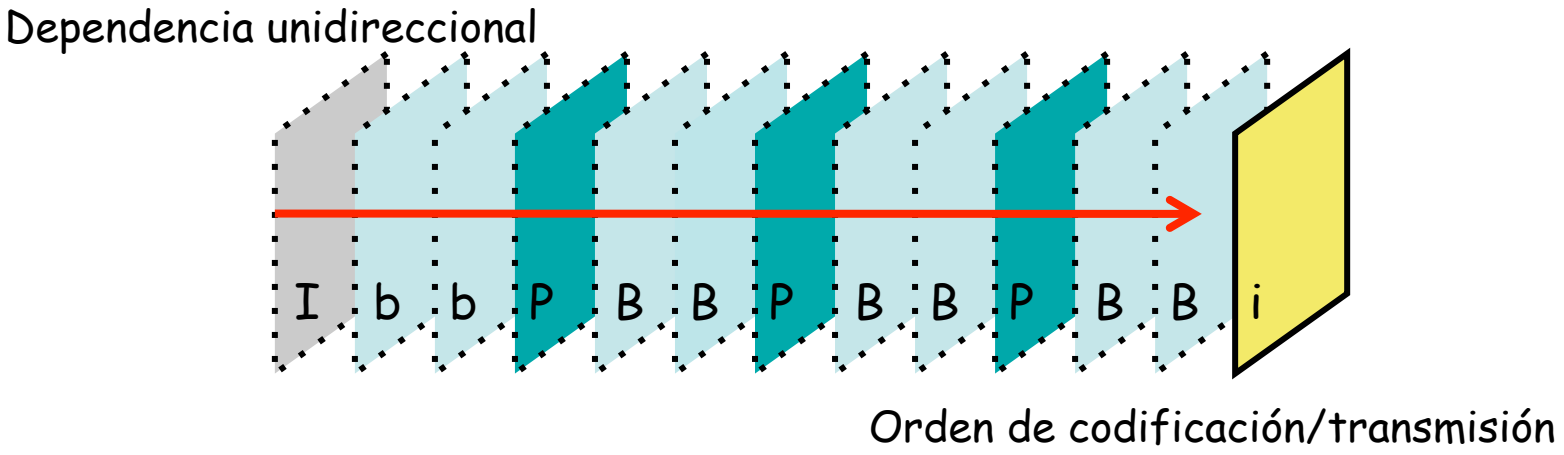
# GoP: losses

- A B-frame is lost
  - No effect on other frames
- A P-frame is lost
  - Other P- or B- frames in the same GoP cannot be decoded



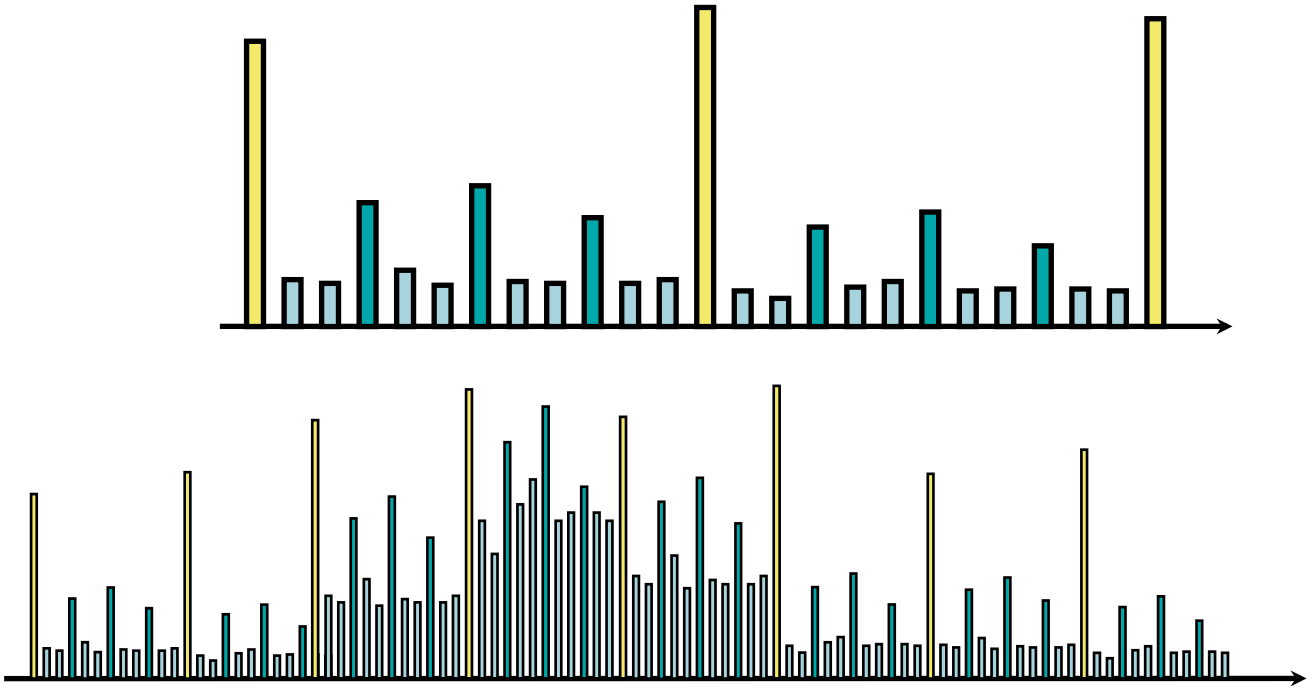
# GoP: losses

- A B-frame is lost
  - No effect on other frames
- A P-frame is lost
  - Other P- or B- frames in the same GoP cannot be decoded
- An I-frame is lost
  - As many frames as the GoP size cannot be decoded
- It could happen that only part of the frame is lost



# GoP: traffic

- Variable amount of information per frame
- More information in I-frames
- P- and B- frames are smaller (more compression)
- Variable frame size in small scales due to frame type
- Variable frame size in large scales due to amount of movement (affects the compression in constant quality coding)



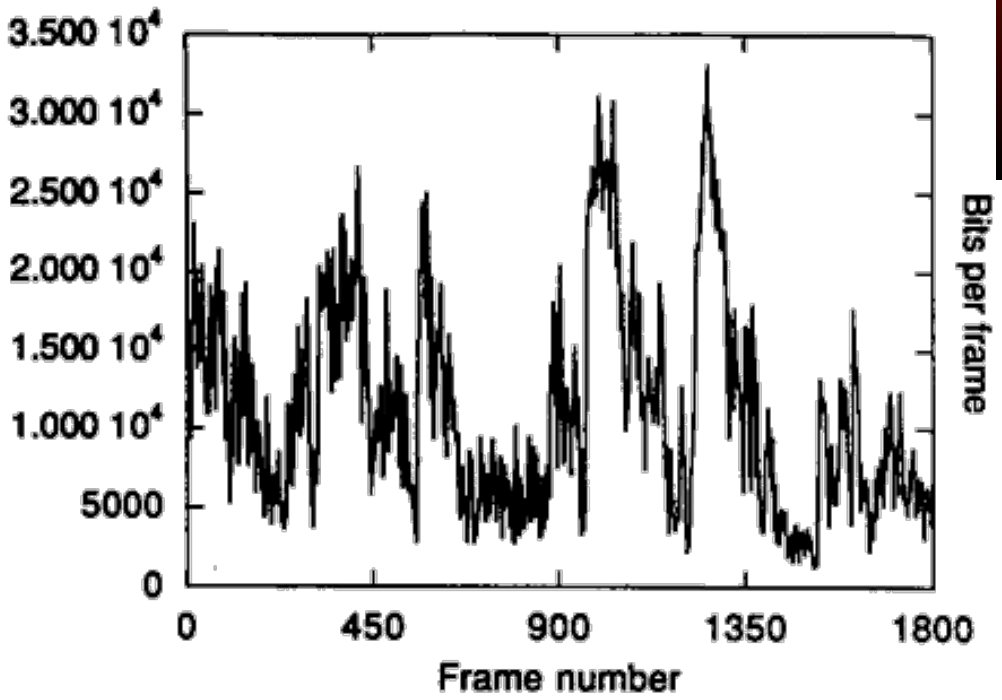
# Traffic from video sources

**For the model:**

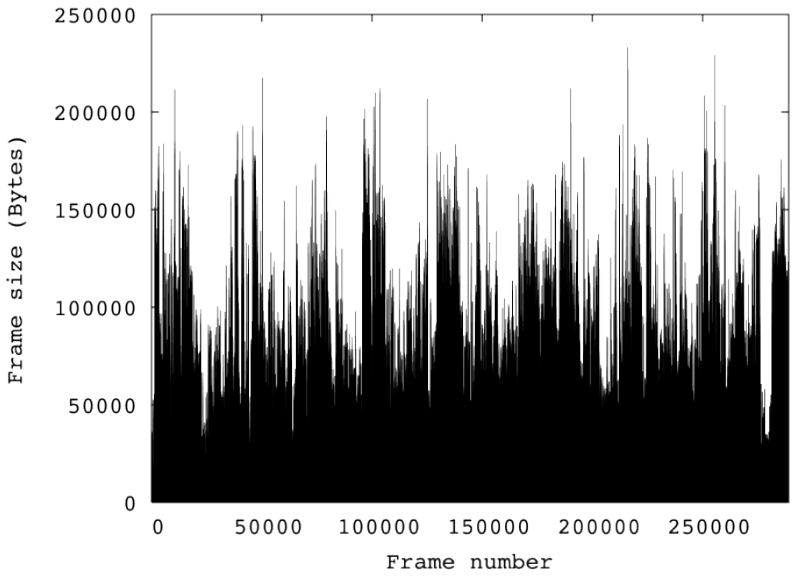
- CDF for frame sizes
- Autocorrelation (long range dependency!)

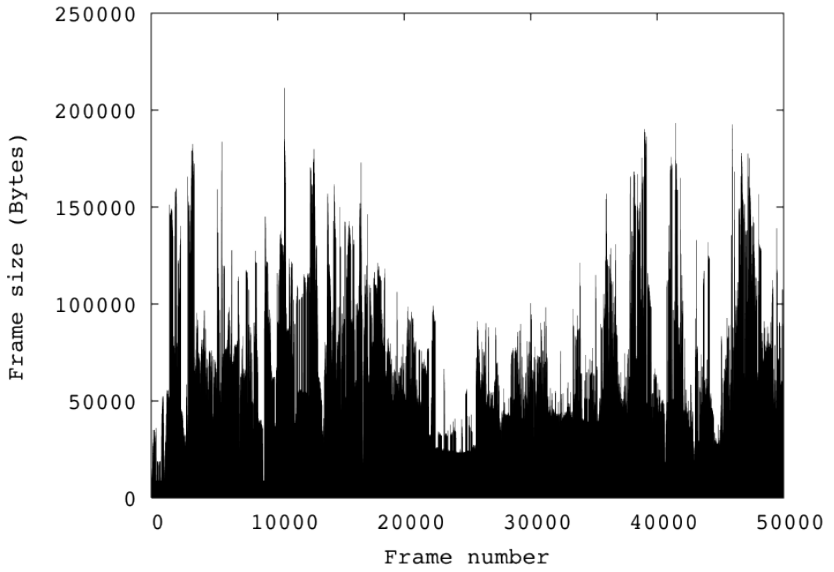
**For:**

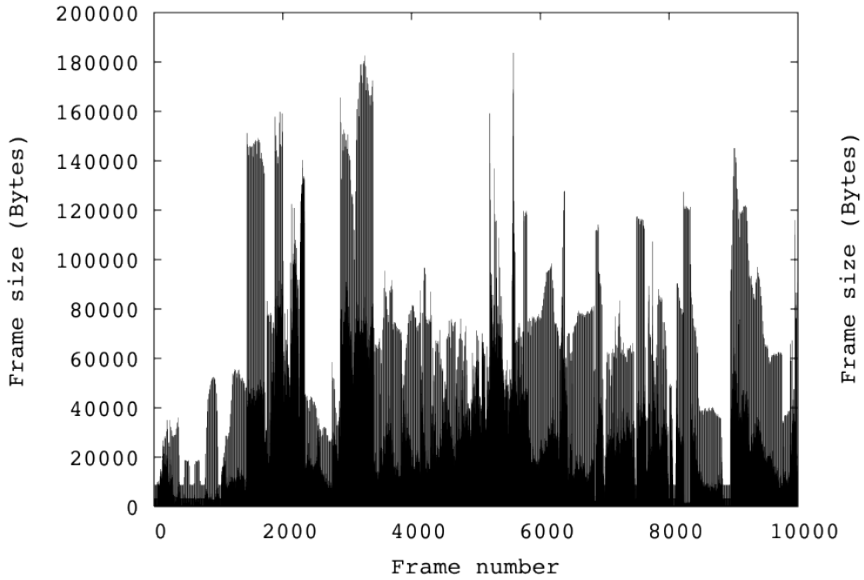
- Prediction of losses
- Decide bandwidth and buffer size needed

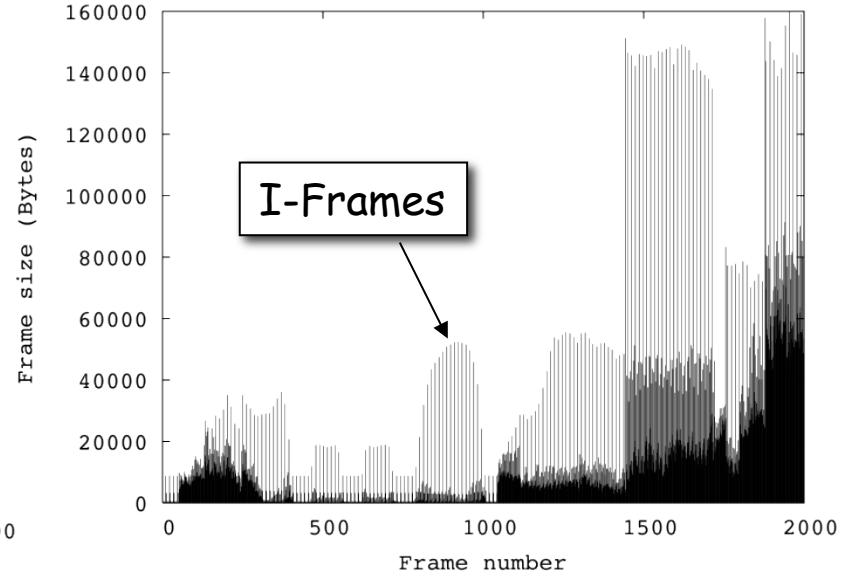


# Bursts

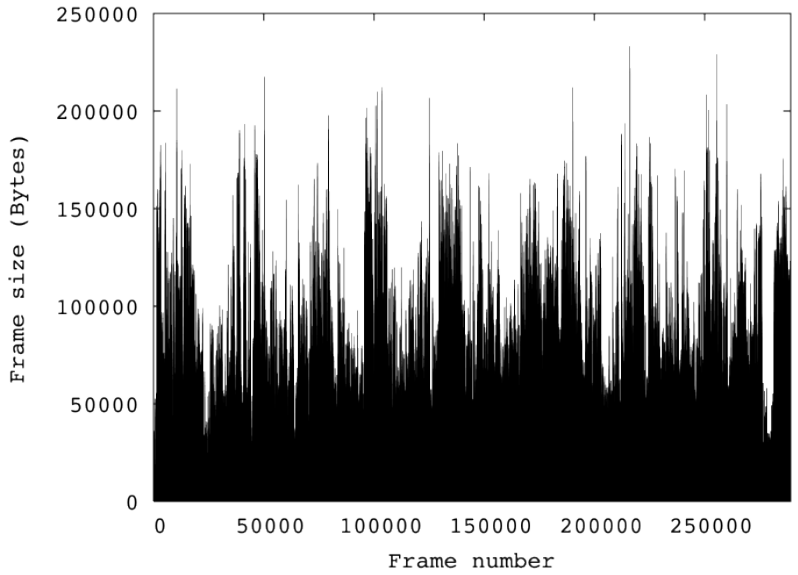

Frame size (Bytes)
250000
200000
150000
100000
50000
0
0
50000
100000
150000
200000
250000
Frame number

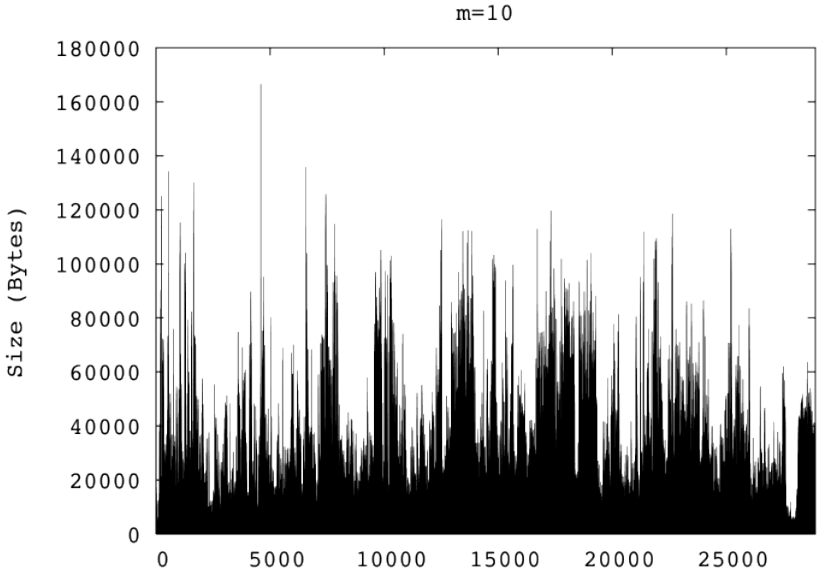

Frame size (Bytes)
250000
200000
150000
100000
50000
0
0
10000
20000
30000
40000
50000
Frame number

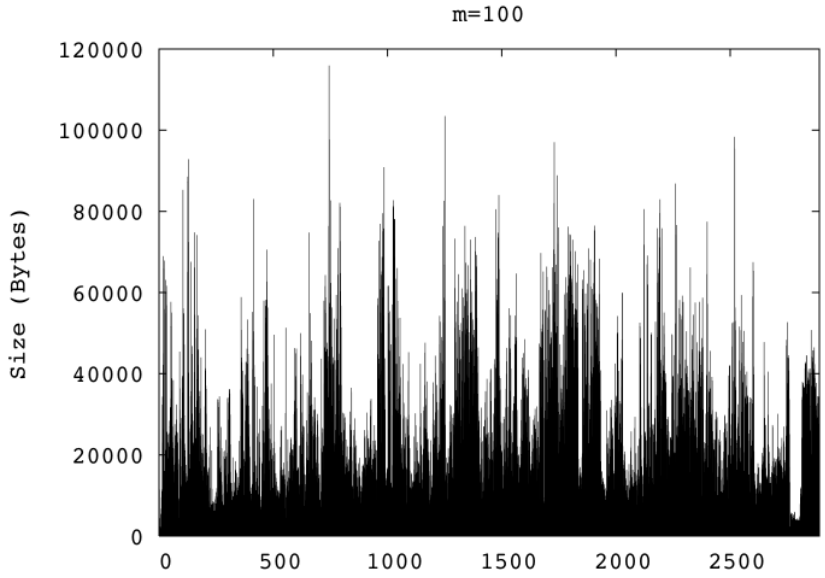

Frame size (Bytes)
200000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000
Frame number


Frame size (Bytes)
160000
140000
120000
100000
80000
60000
40000
20000
0
I-Frames
0
500
1000
1500
2000
Frame number

# In different scales

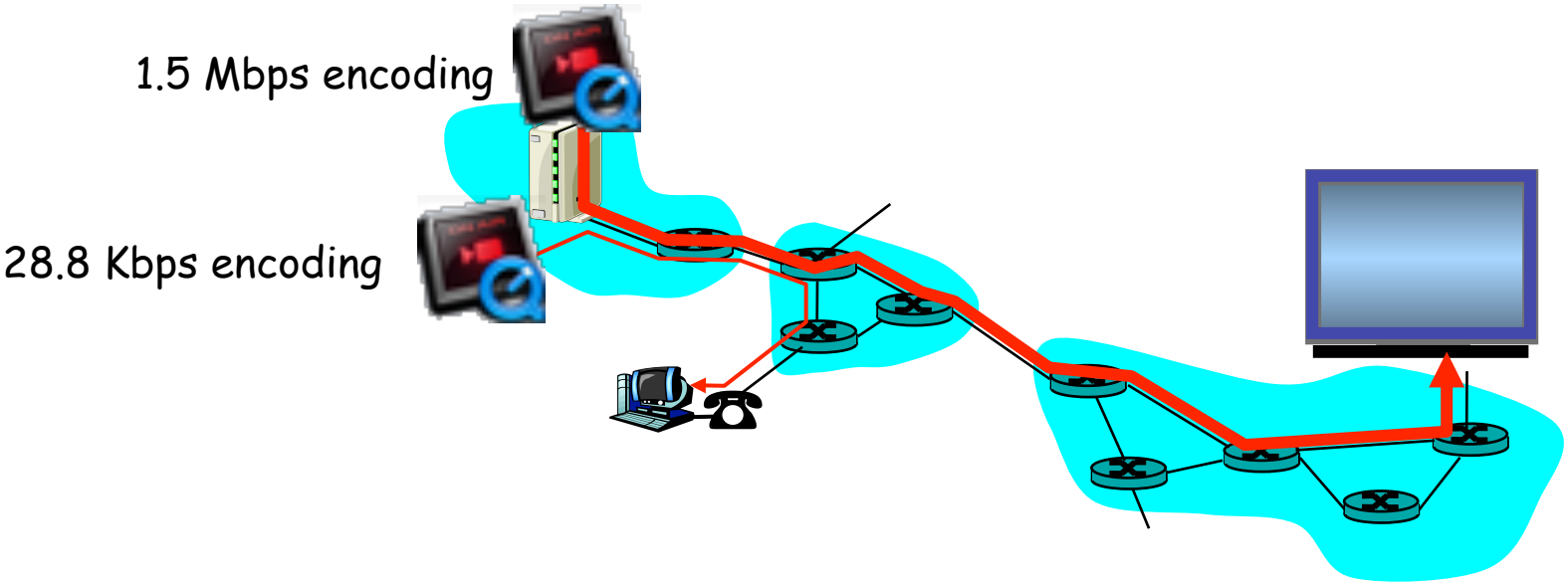


# El servidor

# Streaming: client rate(s)



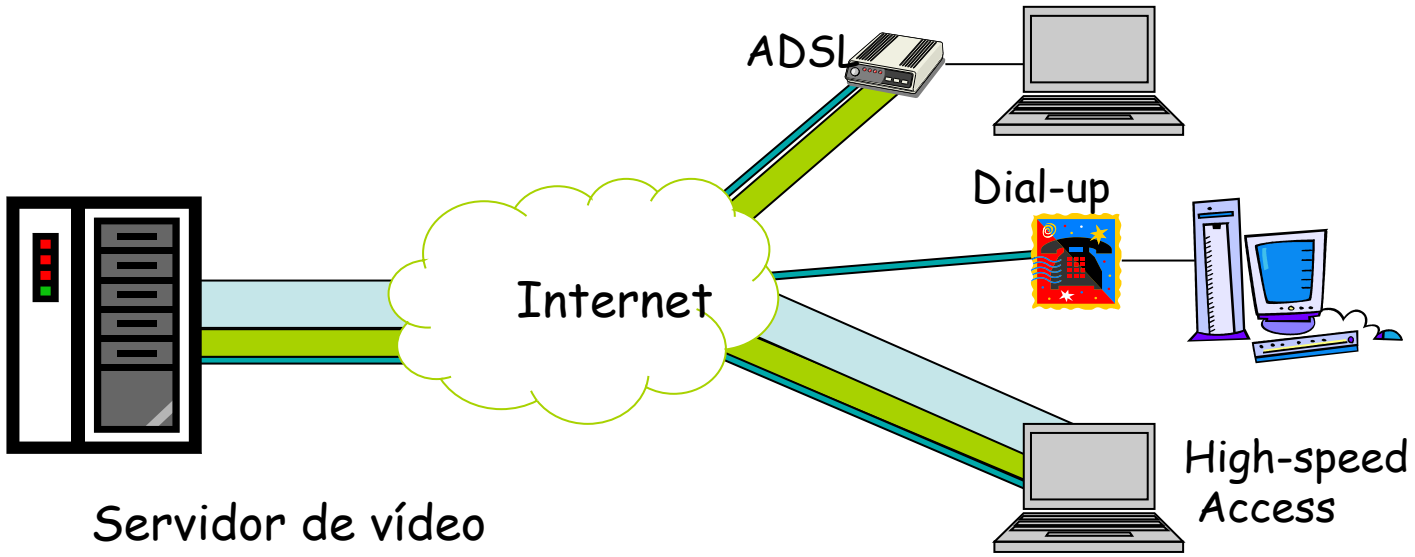

Q: ¿Cómo gestionar diferentes capacidades de recepción?

- 28.8 Kbps modem
- 100Mbps Ethernet

A: El servidor almacena y transmite copias del vídeo codificadas a diferentes velocidades

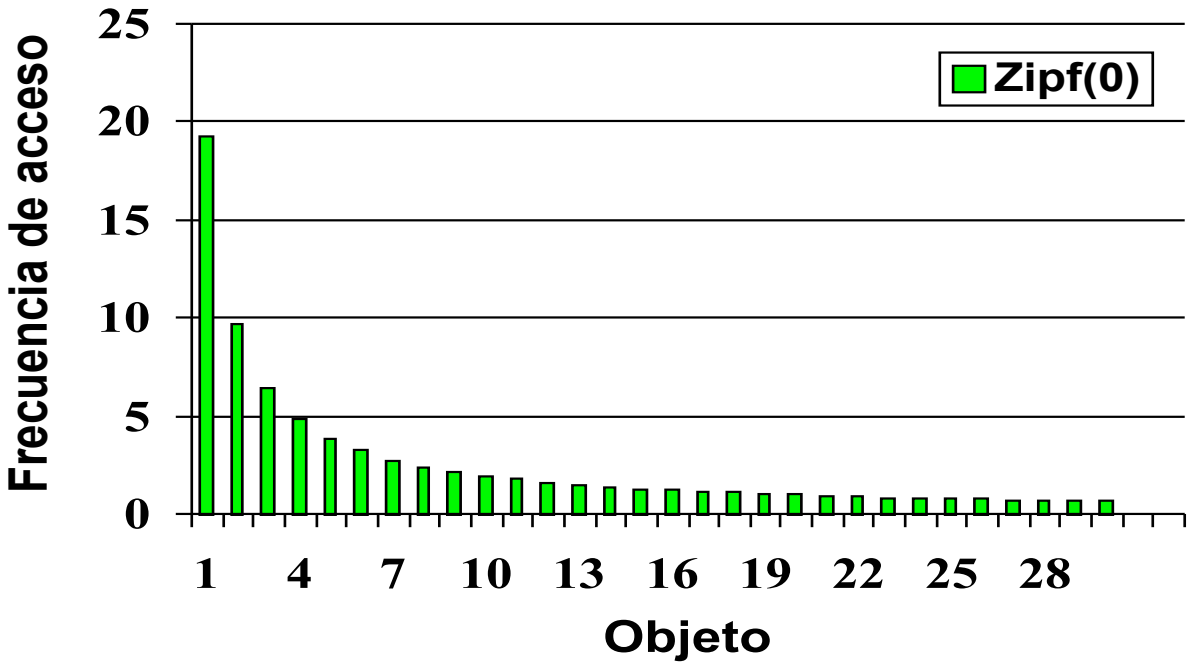
# *Streaming Live Multimedia*

- ¿Cómo hacer *streaming* a un gran número de clientes?
  - Ejemplo: Un evento deportivo popular
  - Usar multicast/broadcast
- ¿Y la heterogeneidad de clientes?
  - Los clientes pueden disponer de diferente BW
  - Vídeo en capas



# Comportamiento de los usuarios

- 100s – 1000s solicitudes de un fichero / duración del su reproducción
- Popularidad sesgada de los ficheros
  - 10% – 20% de los ficheros acaparan el 80% de las peticiones
  - Tiene sentido centrarse en los ficheros populares



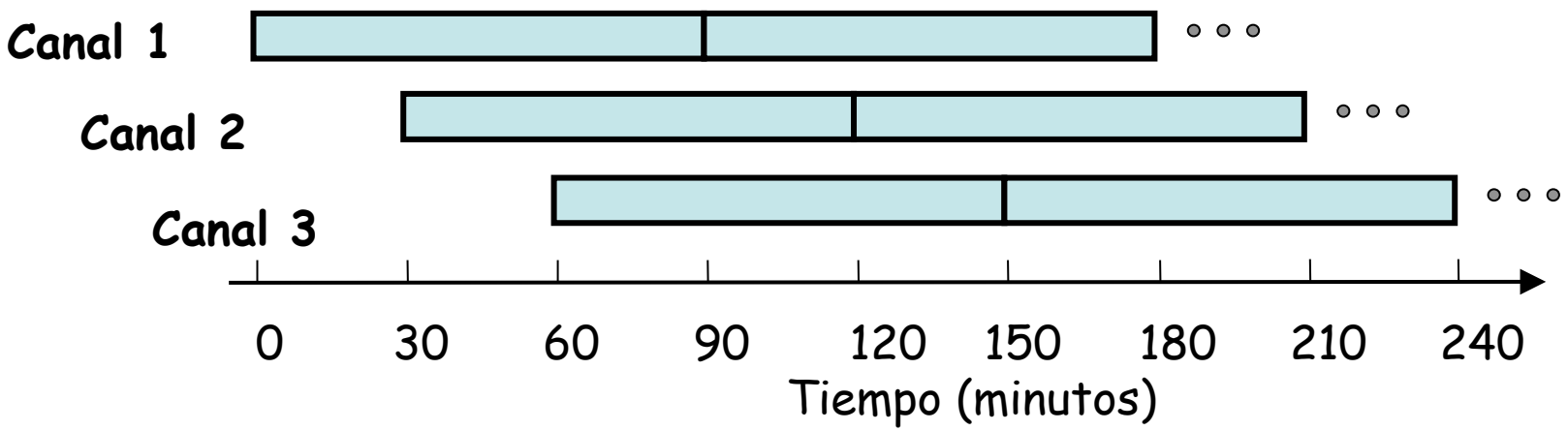
# Streaming escalable: Motivación

- Consideremos un fichero popular
  - Tasa de reproducción (*playback rate*): 1 Mbps
  - Duración: 90 minutos
  - Peticiones: una por minuto


- Comienza un nuevo stream por cada petición
  - BW requerido = 1 Mbps x 90
- ¿Cómo dar flexibilidad en el instante de comienzo sin enviar un flujo por petición?
- *Batching*
  - Acumular suficientes peticiones para que sea rentable iniciar un nuevo flujo
  - Empezar el flujo al acumular suficientes usuarios o llegar a un límite de tiempo de espera

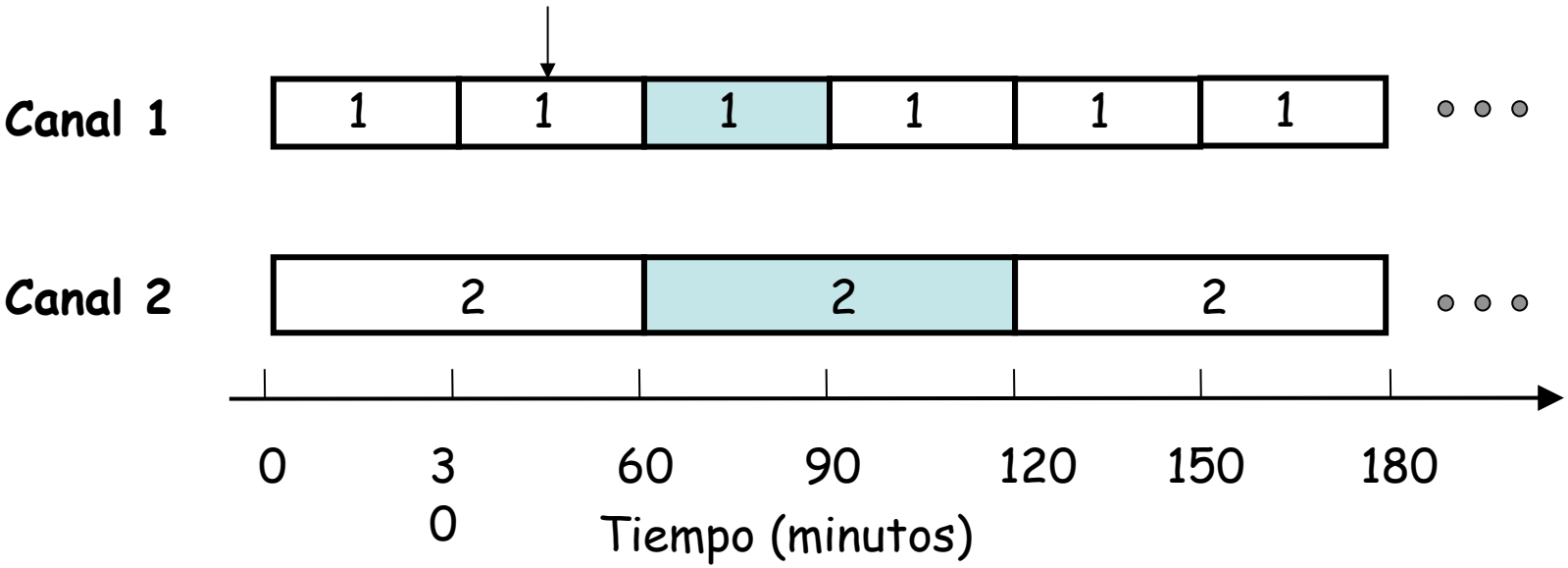
# *Batching* (Ejemplo)

- Playback rate = 1 Mbps, duración = 90 minutos
- Agrupar peticiones en intervalos no solapados de 30 minutos:
  - Máx espera comienzo = 30 minutos
  - BW necesario = 3 canales = 3 Mbps
- BW aumenta linealmente con la reducción de la espera de comienzo



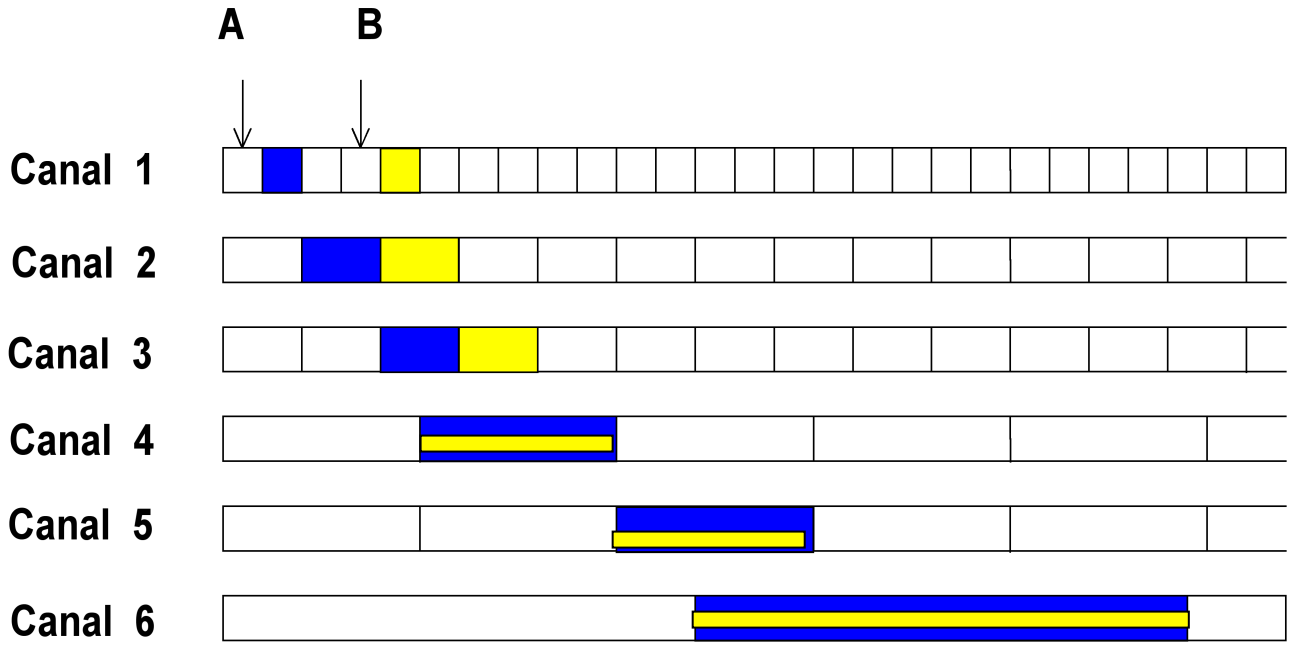
# Periodic Broadcast

- Partir el fichero en 2 segmentos con tamaños relativos {1,2}
- Para una película de 90 minutos:
  - Segmento 1 = 30 minutos, Segmento 2 = 60 minutos
- Ventaja:
  - Máximo tiempo de espera comienzo = 30 minutos
  - BW necesario = 2 canales = 2 Mbps
- Desventajas: Requiere más del cliente (recibir 2 a la vez)



# *Skyscraper Broadcasts (SB)*

- Divide el fichero en K segmentos de tamaños crecientes
  - Progresión de tamaños: 1, 2, 2, 5, 5, 12, 12, 25, …
- Multicast de cada segmento en un canal separado
- Tasa a cada cliente: 2 x playback rate



# Network requirements

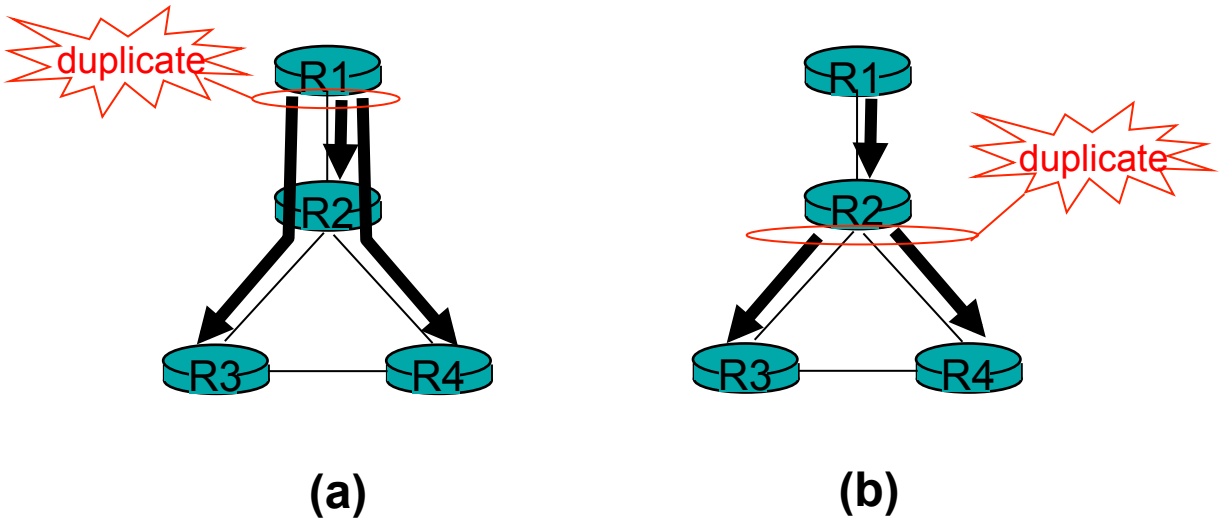
# Network requirements

- BW
- Delay (interactivity)
- Jitter (buffer size, waiting time)
- Losses (playback interruptions)

- Nowadays
  - Video over Internet: “throw BW at the problem” and wish that the other three parameters will keep within limits ➔ no guarantee
  - Video in controled environments
    - IPTV
    - VoD
    - Mixture, batching
- Let’s see what are the mechanisms to do it better

# Network requirements: Multicast

# Multicast: Introduction



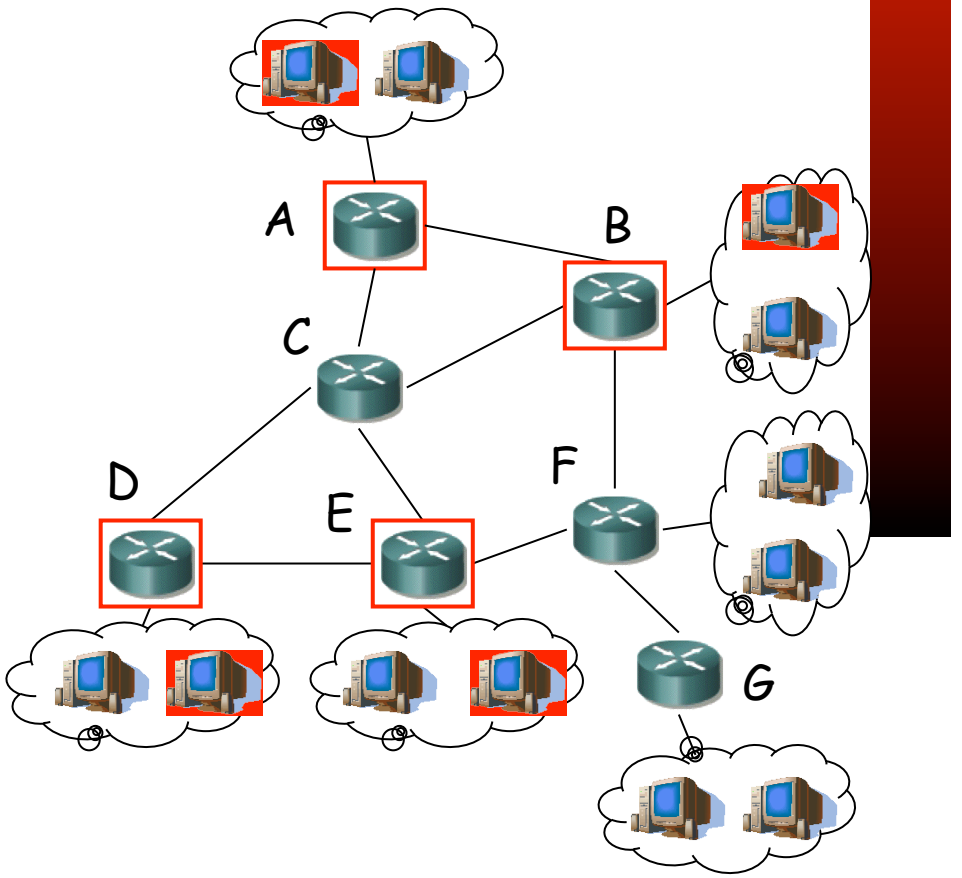

**(a)** **(b)**

Duplicate at the source or in the network
(a) Source, (b) Network

# How to reach all the multicast group members?

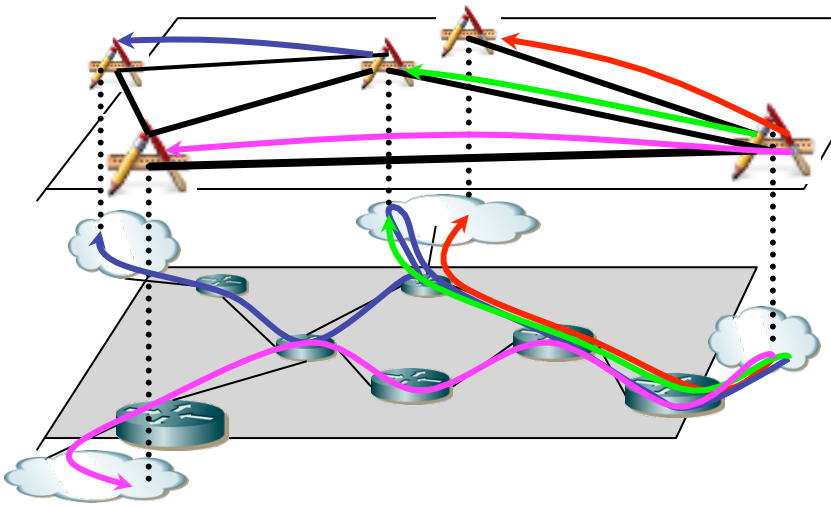
## Multicast Routing

- Find a sink-tree that interconnects all the access routers for the group members
- *Spanning trees*
- *Minimum spanning trees* for a certain metric
- Additional routers could be involved



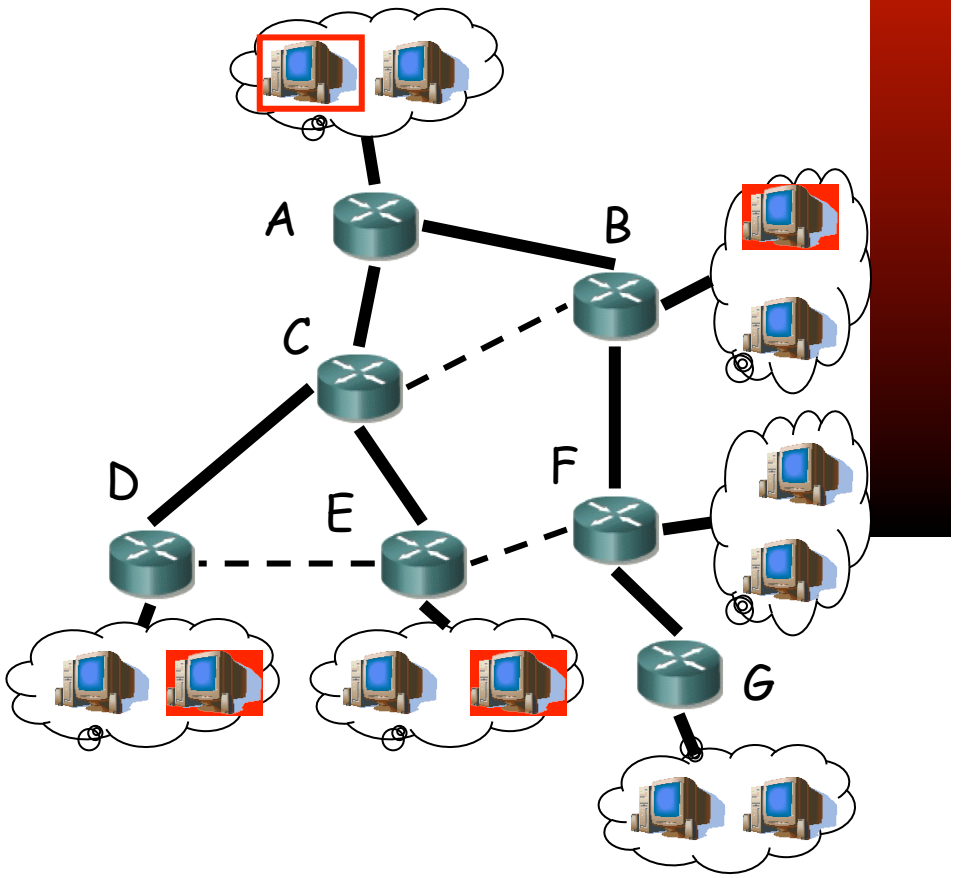
# Application layer multicast

- Based on a unicast network
- Create an overlay network between the end user applications
- Advantages
  - Use no-multicast-capable routers
  - New functionalities can be added without network changes
- Drawbacks
  - ¿How to decide a good overlay?
  - Not optimal use of links



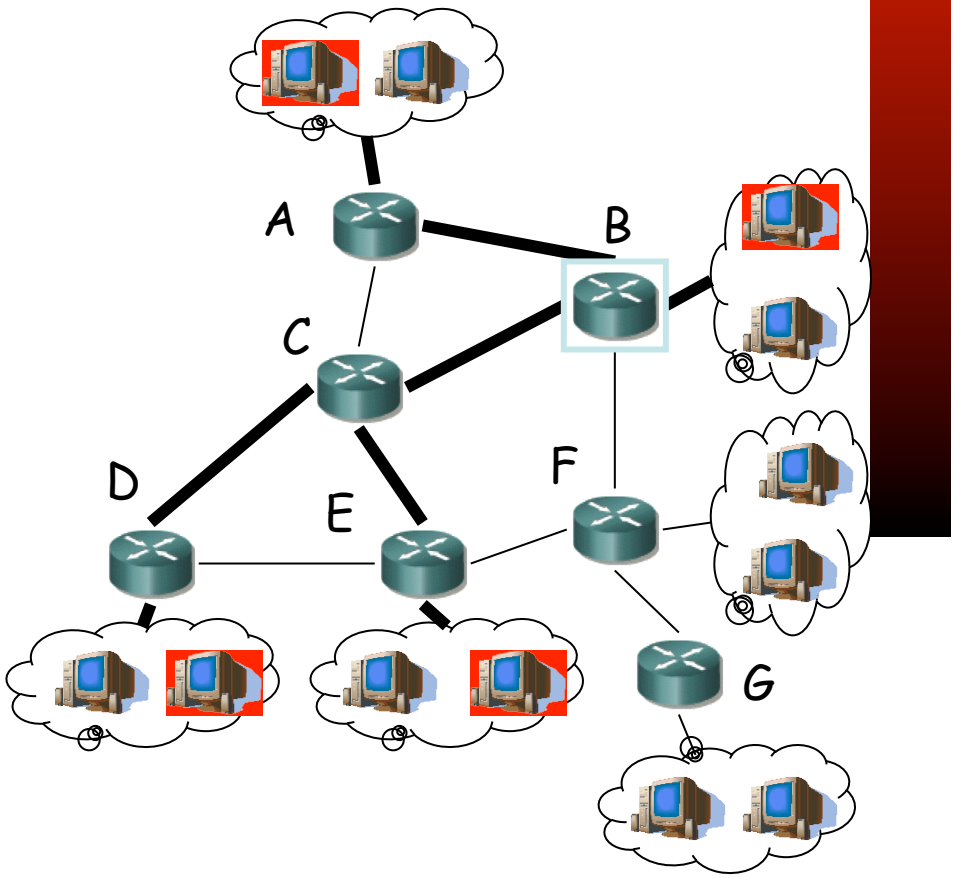
# Source-Based Trees

- One sink-tree per source in a group
- Num. Trees = Groups x Sources
- Use *Reverse Path Forwarding* (**RPF**):
  - Forward using every interface but the one it came through
  - And only if it came through the link used to reach the source
- ***Pruning***
  - A mrouter without adjacent group members or other mrouters in the tree can be *pruned*
  - Send a *prune upstream* message
  - Example: Node G



# Group-Shared Tree

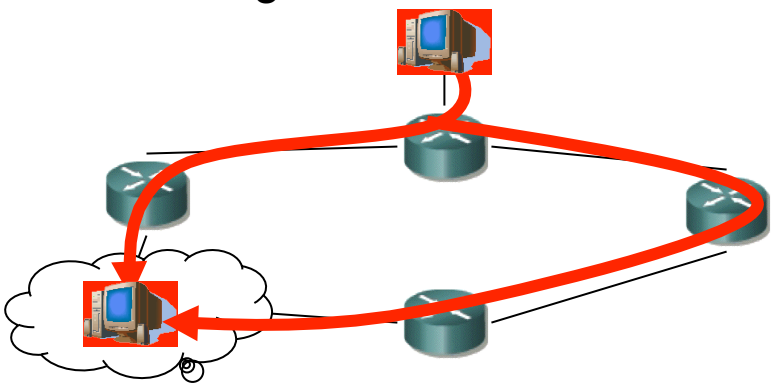
- Build a single tree per group
- Include all the routers adjacent to a group member
- It uses a ***Rendezvous-Point Tree***:
  - Choose a node to be the root tree
  - All the others send a unicast message to the root in order to join the tree
  - The message is forwarded until it reaches root or a router in the tree



# Some protocols

## DVMRP

- *Distance Vector Multicast Routing Protocol*
- It uses RPF (more than one copy can reach a network)
- *Reverse Path Broadcasting* (**RPB**)
  - Each router has a *designated parent router*
  - Only this router can forward into a network
- Use *pruning*
- Use new types of IGMP messages

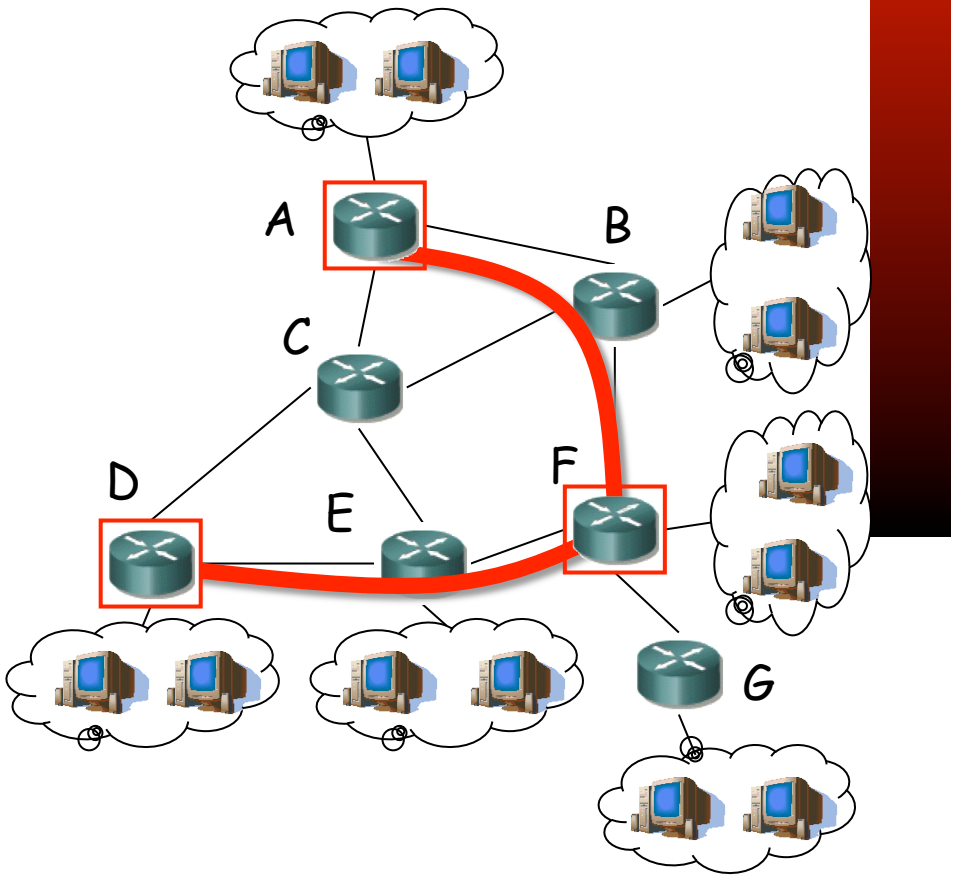


## PIM (*Protocol-Independent Multicast*)

- *Protocol-Independent* cause it may use the information obtained by any routeing protocol
- **PIM-DM (Dense Mode)**
  - Most of the routers are involved
  - Similar to DVMRP
  - *RPF Flood-and-prune*
- **PIM-SM (Sparse Mode)**
  - A small number of routers have adjacent hosts that belong to the group
  - *Group-shared*
- **Other:** MOSPF, MBGP, CBT…

# MBONE (Tunneling)

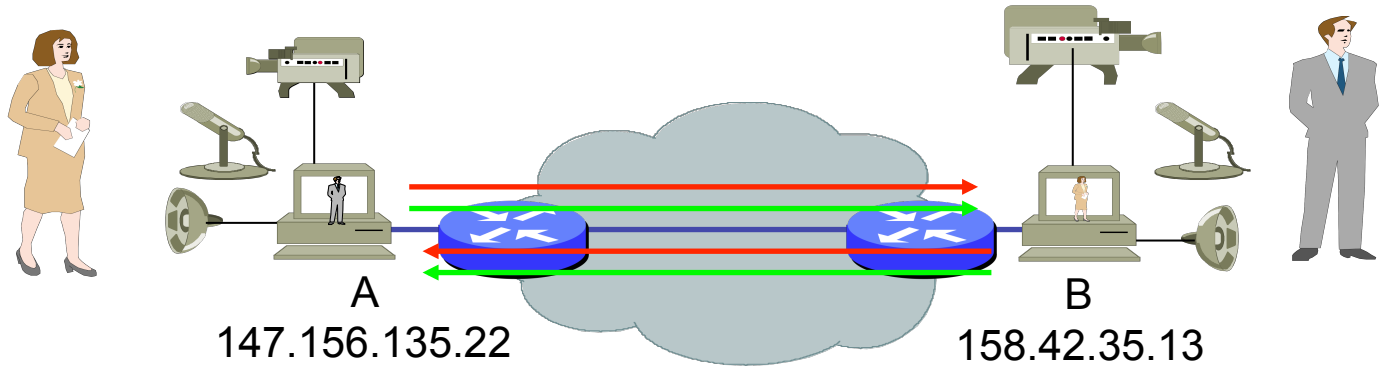
- *Multicast Backbone*
- Using a network with no-multicast enabled routers
- MRouters interconnected using **tunnels**
- Most of these mrouters were UNIX workstations
- DVMRP
- Session Description Protocol (SDP) to describe the sessions
- Session Announcement Protocol (SAP) as transport protocol for SDP



# Network requirements: QoS

# Concepto de flujo en QoS

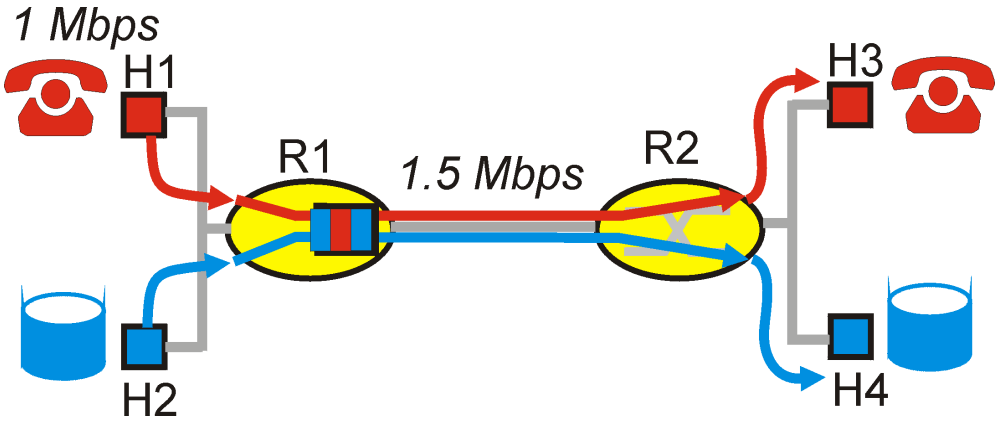
- Secuencia de datagramas que se produce como resultado de una acción del usuario y requiere la misma QoS
- Normalmente es simplex (unidireccional)
- Es la entidad más pequeña a la que los routers pueden aplicar una determinada QoS
- Ejemplo: una videoconferencia estaría formada por cuatro flujos, dos en cada sentido, uno para el audio y otro para el vídeo.
- Los flujos pueden agruparse en clases; todos los flujos dentro de una misma clase reciben la misma QoS.




Flujo vídeo A->B: 147.156.135.22:2056 -> 158.42.35.13:4065
Flujo audio A->B: 147.156.135.22:3567 -> 158.42.35.13:2843
Flujo vídeo B->A: 158.42.35.13:1734 -> 147.156.135.22:6846
Flujo vídeo B->A: 158.42.35.13:2492 -> 147.156.135.22:5387

# Principio 1: Clasificación

- Ejemplo: Teléfono IP a 1Mbps, comparte enlace de 1.5Mbps con FTP
  - Ráfagas de FTP pueden congestionar el router y causar fallos en el audio
  - Queremos dar prioridad al audio sobre el FTP

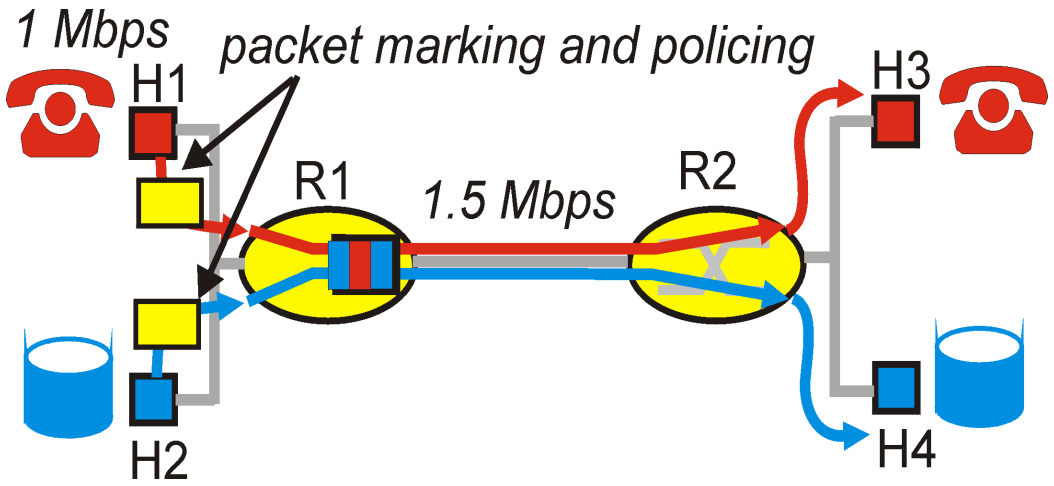



**Principio 1**

Los routers necesitan distinguir el tráfico de diferentes clases y aplicarles diferentes políticas: *packet marking* (generalmente a la entrada a la red)

# Principio 2: Aislamiento

- ¿Qué sucede si las aplicaciones no se comportan como deben?
  - Por ejemplo la aplicación de audio envía más de lo previsto
  - Necesitamos forzar que las fuentes se comporten como se ha acordado

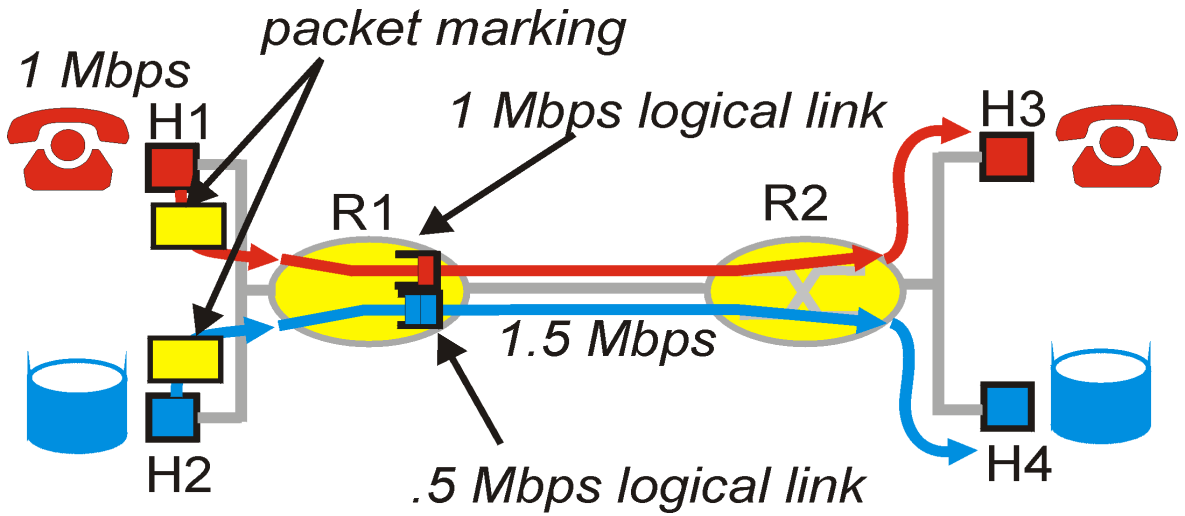



**Principio 2**

Forzar que una clase de tráfico se comporte dentro de lo contratado: *policing* (típicamente a la entrada)

# Principio 3: Eficiencia

- Reservar BW fijo (no compartido) a un flujo es ineficiente si ese flujo no lo emplea todo

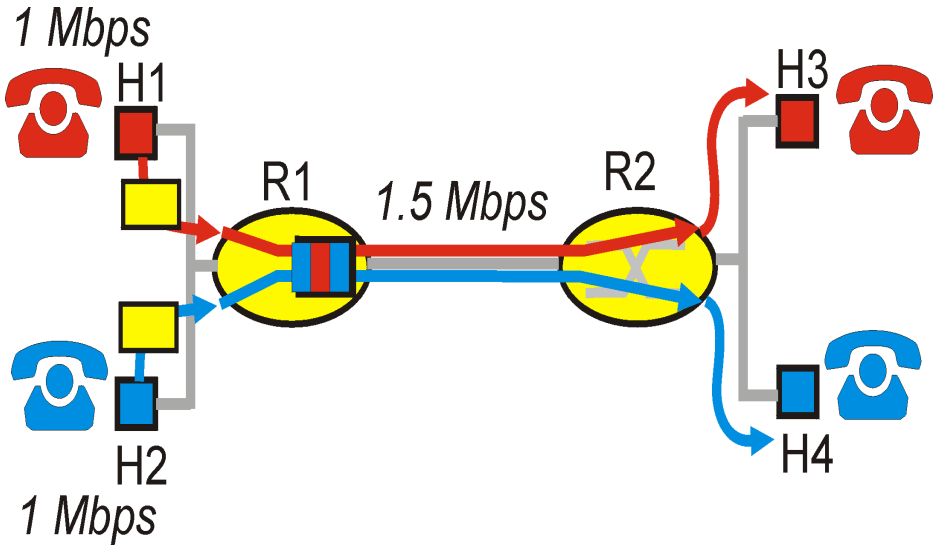



**Principio 3**

Mientras se ofrece aislamiento es deseable emplear los recursos de forma eficiente (*work conserving*): *scheduling* (en todos los routers del camino)

# Principio 4: Límite de recursos

- No se pueden satisfacer las demandas más allá de la capacidad del enlace



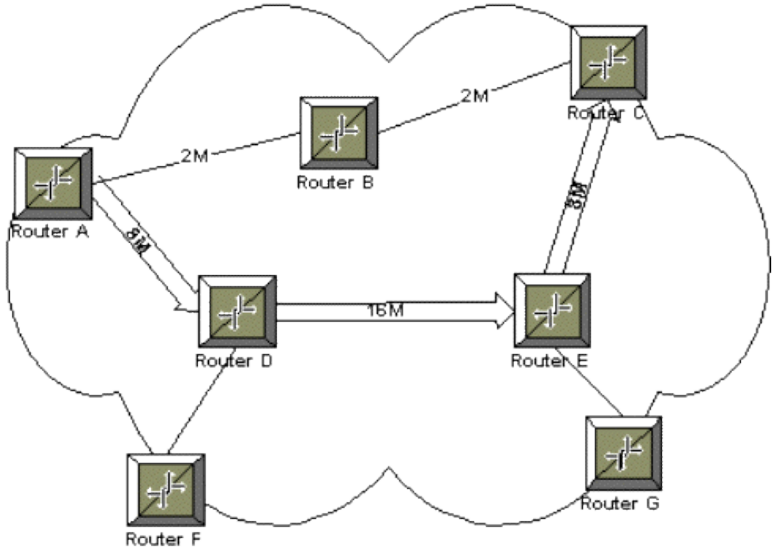

## Principio 4

El flujo declara sus necesidades pero la red puede *bloquear* al flujo si no puede satisfacerlas: *call admission*

# QoS Routing

- Encontrar caminos “buenos” para flujos con requisitos específicos de QoS
- Usar la red de forma eficiente: aumentar la probabilidad de aceptar peticiones futuras
- Es complicado:
  - Información precisa sobre el estado de la red es difícil de mantener
  - Calcular caminos que cumplan requisitos de QoS es costos (computacionalmente hablando)
- *Constraint-based Routing*
  - Calcular caminos teniendo en cuenta no solo QoS sino también políticas

A → C: 4M



# Streaming: ¿UDP o TCP?

## UDP

- El servidor envía a la velocidad apropiada para la reproducción
- Retardo de comienzo (2-5 segs) para compensar el jitter de la red
- Recuperación de pérdidas: *time permitting*

## TCP

- Puedes entregar a la velocidad de reproducción pero no sabes cuándo lo enviará TCP
- Además puede acumular bytes para formar paquetes más grandes
- Normalmente envía a la máxima velocidad
- Fluctuará debido a los mecanismos de control de flujo y control de congestión
- Retardo de comienzo más largo para poder suavizar el comportamiento de TCP
- HTTP/TCP atraviesa mejor firewalls (*HTTP Streaming*)

# Multimedia en la Internet actual

- TCP/UDP/IP: “best-effort service”
- Sin garantías de retardo o pérdidas
- Los programas emplean técnicas en el nivel de aplicación para mitigar los efectos de retardos y pérdidas
  - Buffers
  - Algoritmos de codificación resistentes ante pérdidas