

Vídeo y Audio

Vídeo en HTML

Vídeo en HTML5

- ⦿ Antecedentes y necesidad
- ⦿ Formatos
- ⦿ Elemento de vídeo en HTML5
- ⦿ Codificación
- ⦿ Descarga progresiva, streaming y difusión HTTP adaptativa
- ⦿ Posibilidades en la manipulación del elemento vídeo
 - ⦿ CSS, CSS3, Javascript, SVG con vídeo, Web Workers

Antecedentes

- ⦿ 2005 > primera mención a la posibilidad de la existencia de <video>
- ⦿ 2007 > primera implementación de prueba
- ⦿ 2010 > se une IE (Microsoft)
- ⦿ Algunos “fracasos”:
 - ⦿ HTML+Time (1998 por MS)
 - ⦿ SMIL (nunca desarrollado en navegadores)

Antecedentes

- ⦿ En 5 años se consiguió llegar a un acuerdo entre todos para <audio> y <video>
- ⦿ ¿quienes son todos? > QuickTime, MS Windows Media, Real Networks, Real Media, Xiph Ogg, ISO/MPEG, Adobe Media y Silverlight

Antecedentes

- ⦿ Antes de <video> y <audio> : <object> y <embed>
- ⦿ Requieren complementos de terceros
- ⦿ Macromedia: introduce compatibilidad con vídeo en su complemento, que dependía del códec Sorenson Spark (también por QT en aquel entonces)

Antecedentes

Real Media
QuickTime
Windows Media

Flash Player 9 (2007)

compatibilidad con familia de
códigos MPEG (H.264)

se pasó del FLV a MP4

Flash
(aparece con
soporte pobre)

Flash 8 (2005)
código avanzado de vídeo VP6 de On2
transparencia alfa en vídeo
cue points en FLV
componente de reproducción avanzado

coge mucha ventaja

supone solución de vídeo online sin tener que
codificarlo en varios formatos, ya que plugin
en casi todos los navegadores

ej: Google Video en 2005 publicaba con
Flash, al igual que YouTube

Antecedentes

- ⦿ Mientras tanto...
 - ⦿ debate sobre si <video> si o <video> no
- ⦿ Por lo tanto...
 - ⦿ vídeo en la web dominado por Flash hasta la aparición de <video> en HTML5

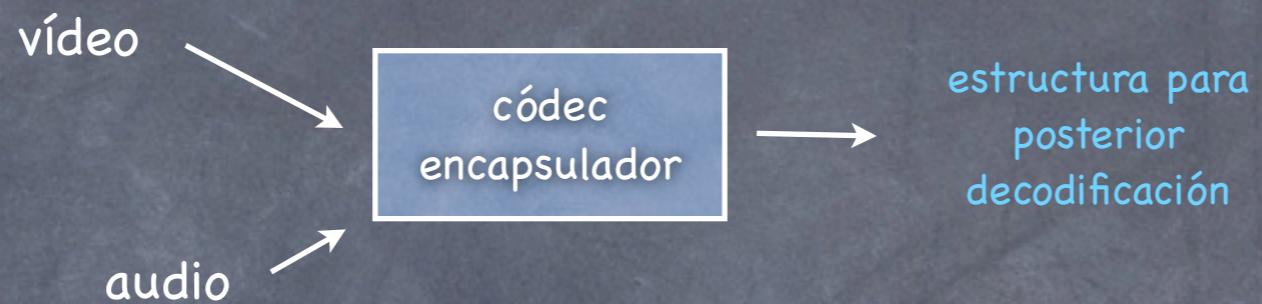
Antecedentes

- ⦿ ¿por qué <video>?, ¿por qué no usar <embed>, SMIL, HTML +Time...?
 - ⦿ el elemento debería ser tan fácil de integrar como una
 - ⦿ integrado en todas las capas de aplicaciones web, incluyendo DOM, CSS y Javascript para poder progresar
 - ⦿ de lo contrario: pérdida frente a los complementos
 - ⦿ y todos los navegadores deberían aceptarlo
- ⦿ nace la necesidad de una estandarización del elemento <video>

Antecedentes

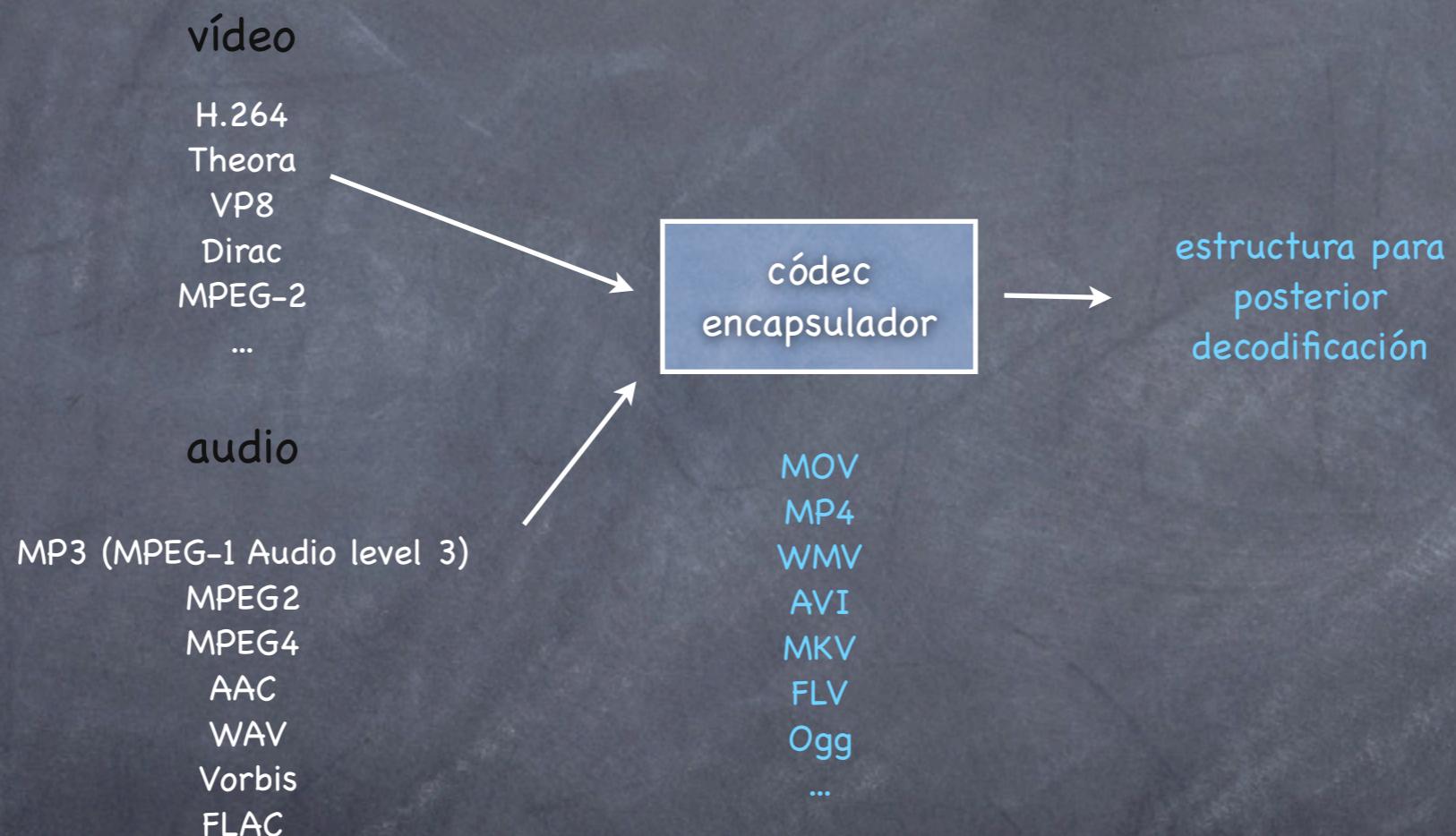
- ⦿ Necesidad de un formato común base de codificación
- ⦿ Que esté implementado y compatible con todos los navegadores
- ⦿ códec: audio y vídeo comprimidos en un formato contenedor o encapsulador

Antecedentes



- ⦿ Cada uno de estos puede (en teoría) ser compatible con la encapsulación de cualquier secuencia de datos del códec
- ⦿ en la práctica: tan sólo se encuentran algunos códecs en contenedores específicos
 - ⦿ MP4: MP3, AAC y H.264
 - ⦿ Ogg: Theora, Vorbis, Speex, FLAC

Antecedentes



Ante esto... para HTML5 es FUNDAMENTAL disponer un códec base

Antecedentes

- ⦿ Wium Lie (CTO Opera): necesidad de hacer esto bien desde el principio y no como en el caso del PNG, p.j.
- ⦿ Requisitos que debería tener:
 - ⦿ compatible con amplio rango de dispositivos
 - ⦿ libre de derechos de autor (RF ppo. de W3C)
- ⦿ Propone Ogg Theora como candidato

Antecedentes

- ⦿ Ogg formato contenedor
 - ⦿ códec vídeo Theora
 - ⦿ códec audio Vorbis (superior a MP3 y similar a AAC)
- ⦿ Desarrollado por Xiph.org como código abierto. Primer formato libre de gravámenes
- ⦿ Theora es una derivación de VP3 de On2
- ⦿ Google se hizo con On2 en 2010

Antecedentes

- ⦿ Posible competidor de Ogg Theora > Dirac
- ⦿ Desarrollado por BBC (<http://diracvideo.org/>)
- ⦿ Calidad de compresión buena
- ⦿ No muy eficiente para web

<http://keyj.emphy.de/video-encoder-comparison/>

Antecedentes

- ⦿ **Conclusión:** a comienzos de 2007, Ogg Theora y Vorbis fueron incluidos en la especificación HTML5 como los códexs base para vídeo y audio
- ⦿ Sin embargo...

Antecedentes

- ⦿ ...estaba claro que no todos lo implementarían
- ⦿ Safari 3.1 (sólo H.264), por calidad, etc
- ⦿ Nokia + Microsoft confirmaron lo mismo
- ⦿ Con lo cual, H.264 fue aprobado por la ITU e ISO/IEC como estándar, a pesar del desacuerdo por el pago de derechos, etc

Antecedentes

- ⦿ ¿Por qué H.264 en vez de Theora?
 - ⦿ Mayor calidad de codificación
 - ⦿ Estándar para publicación en web establecido por Youtube (Adobe Flash con compatibilidad MP4 H.264/AAC). Esto implica que escoger este códec facilitaría las migraciones
 - ⦿ Existen implementaciones de hardware móviles compatibles con H.264
- ⦿ Pero Ogg crece profesionalmente (implementaciones en ARM de Google), es menos complejo...

Antecedentes

- ⦿ En resumen... en mayo 2010....
- ⦿ ¿H.264 o Ogg ?

Antecedentes

- ⦿ Google lanza el proyecto WebM
 - ⦿ oportunidad para satisfacer a Apple, Nokia y Microsoft sobre Theora
 - ⦿ basado en código abierto y libre de derechos
- ⦿ WebM
 - ⦿ códec de vídeo VP8 (adquirido de On2)
 - ⦿ códec de audio Vorbis
 - ⦿ contenedor derivado de Matroska

Antecedentes

- ⦿ Intención de Google
 - ⦿ resolver los problemas del códec base de vídeo
 - ⦿ licencia de código abierto
 - ⦿ concede licencia mundial de patente libre de derechos, no exclusiva y gratuita
- ⦿ Reacciones: positivas de MS y Apple...?

Antecedentes

- ⦿ Con lo cual...
- ⦿ Safari sólo funciona con MP4 H.264/AAC
- ⦿ Mozilla y Opera no compatibles con MP4
- ⦿ Chrome: compatible con los tres
- ⦿ IE: MP4 H.264/ACC

Antecedentes

- ⦿ Conclusión final...
 - ⦿ un sitio web de contenido de vídeo que quiera ser compatible con cualquier navegador, tendrá que publicarlas al menos en MP4 y OggTheora.
 - ⦿ y si partimos de cero... MP4 y WebM
- ⦿ O los tres!
- ⦿ Esto implica que no está resuelto todavía la necesidad de un códec base común para <video> en HTML5

Vídeo en HTML5

- etiquetado simple y básico

```
<video src="video.mp4"></video>
```

- **contenido de retroceso:** cualquier elemento dentro de la etiqueta es interpretada por un navegador que no la soporte. En los que la soportan, se ignoran y se muestra el elemento

Vídeo en HTML5

```
<video src="video.mp4">  
    tu navegador no soporta el elemento de vídeo  
</video>
```

- se puede usar cualquier etiqueta dentro de <video>
- <object> y <embed> para alternativas al uso con Flash
- Muchas bibliotecas interesantes basadas en Javascript

Vídeo en HTML5

- ⦿ Si usamos un único formato...
- ⦿ Debemos capturar errores de apertura y mostrar alternativas
- ⦿ Sino... mostrar todos los recursos disponibles

Vídeo en HTML5

```
<video>  
    <source src="video.mp4">  
    <source src="video.webm">  
    <source src="video.ogv">  
</video>
```

- no es necesario detectar compatibilidad
- el navegador las recorre por orden

Vídeo en HTML5

- Atributo `type`: tipo de recurso referenciado
- El navegador lo analiza y se emplea como “pista” para ver si es capaz de reproducir el archivo o no
- En función del valor de `type`, el navegador puede responder:
 - `no`: no compatible con el recurso
 - `puede ser`: existe alguna posibilidad de ser compatible
 - `probablemente`: seguro que va a funcionar (sólo cuando se envían los codecs)
- Compatibilidades: en lista en el navegador o capas multimedia intermedias (DirectShow, QuickTime, GStreamer...)

Vídeo en HTML5

```
<video>
  <source src="video.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
  <source src="video.webm" type='video/webm; codecs="vp8, vorbis"'>
  <source src="video.ogv" type='video/ogg; codecs="theora, vorbis"'>
</video>

<video>
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  <source src="video.ogv" type="video/ogg">
</video>
```

Vídeo en HTML5

```
<script type="text/javascript">
    var types = new Array();
    types[0] = "video/ogg";
    types[1] = 'video/ogg; codecs="theora, vorbis"';
    types[2] = "video/webm";
    types[3] = 'video/webm; codecs="vp8, vorbis"';
    types[4] = "video/mp4";
    types[5] = 'video/mp4; codecs="avc1.42E01E, mp4a.40.2"';
    // create a video element
    var video = document.createElement('video');
    // test types
    for (i=0; i<types.length; i++) {
        var support = video.canPlayType(types[i]);
        if (support == "") support="no";
        document.write( "<p><b>" +types[i]+ "</b> : "+support+"</p>" );
    }
</script>
```

Vídeo en HTML5

- ⦿ Más atributos:
 - ⦿ autoplay (inicio automático)
 - ⦿ loop (repetición)
 - ⦿ controls (cada navegador el suyo)
 - ⦿ poster (fotograma de representación)
 - ⦿ media (adaptación a dispositivos)
 - ⦿ width, height
 - ⦿ preload

Codificación de recursos

- ⦿ Vídeo MPEG-4 H.264
 - ⦿ uso de la biblioteca `x264` (GNU GPL)
 - ⦿ software: `ffmpeg`
 - ⦿ otros: `handbrake`, `VLC`, `Miro Video Converter`...
- ⦿ Vídeo Ogg Theora
 - ⦿ uso de la biblioteca `libtheora` (BSD Xiph.org)
 - ⦿ software: `ffmpeg2theora` y `ffmpeg` (no es lo mismo)
- ⦿ Vídeo WebM
 - ⦿ uso de la biblioteca `libvpx` (BSD Google)
 - ⦿ `VLC`/ `Miro Video Converter`
- ⦿ Audio MP3 y Ogg Vorbis
 - ⦿ uso de la biblioteca `lame` y `libmp3lame`

Transmisión de información de vídeo

- ⦿ Descarga progresiva
- ⦿ Streaming
- ⦿ Difusión HTTP adaptativa

Descarga progresiva

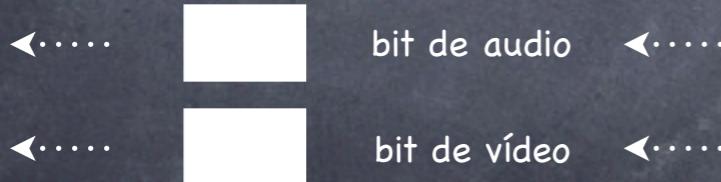


1

obtiene bytes iniciales
que indican de que tipo es
y facilitan la info para
comenzar la tubería de
codificación/decodificación

2

se reciben los datos multiplexados
en rangos de tiempo



3

Mientras se reproduce el vídeo se descargan de forma
progresiva el resto de bits

la reproducción y descarga se realiza en paralelo, y esto
implica una restricción: *¿y si busco una posición no descargada?*

4

El servidor detiene la descarga y comienza a recibir los datos
futuros

esto es posible gracias a Byte Range de HTTP 1.1

Difusión y streaming

- ⦿ RTP (Real Time Protocol)
 - ⦿ No se almacenan los rangos de bytes, son volátiles! (transmisiones de eventos en vivo)
 - ⦿ No se pueden realizar búsquedas en estos casos
 - ⦿ También útil para transmisión de vídeo bajo demanda

Difusión y streaming

- ⦿ RTSP (Real Time Streaming Protocol)
 - ⦿ compatible con encabezados con rango para poder hacer búsquedas dentro de un vídeo
 - ⦿ más difícil de copiar el contenido
 - ⦿ el navegador debe ser compatible con RTSP
 - ⦿ difusión en directo + transmisión cifrada: se puede obtener con descarga progresiva bajo HTTPS
 - ⦿ requiere servidor especial de difusión, no sirve solo con web. Con puertos específicos (554/8554) -> problema con firewalls
 - ⦿ no bueno para transmisión de vídeo bajo demanda, ya que HTTP con rangos de bytes ofrece mismos resultados
 - ⦿ RTSP con servidor específico y clientes específicos han fracasado. Domina HTTP

Difusión y streaming

- ⦿ Youtube trasmite completamente mediante descarga progresiva > desarrolladores han creado extensiones propias para mejorar este método
- ⦿ Objetivo:
 - ⦿ Seguir usando servidores web normales
 - ⦿ no instalar extensiones de terceros
 - ⦿ no incrementar complejidad en el cliente
- ⦿ Evitamos:
 - ⦿ creación de nuevos protocolos
 - ⦿ problemas con firewalls
 - ⦿ instalación de software nuevo en servidores y clientes

Difusión HTTP adaptativa

- Ajusta la calidad de un vídeo transmitido a través de HTTP, basándose en el cambio del software del usuario y red, para ofrecer la mejor experiencia visual posible.
- HTTP Live Streaming de Apple (m3u8)
- Smooth Streaming de Microsoft (SMIL)
- HTTP Dynamic Streaming de Adobe (F4M)
- Grupo MPEG trabaja en una técnica estándar de la difusión adaptativa > DASH (Dynamic Adaptive Streaming for HTTP)

Difusión HTTP adaptativa

Servidor HTTP



Archivo de manifiesto



- 1 se almacenan diferentes versiones del recurso a diferentes tasas de bits y resoluciones

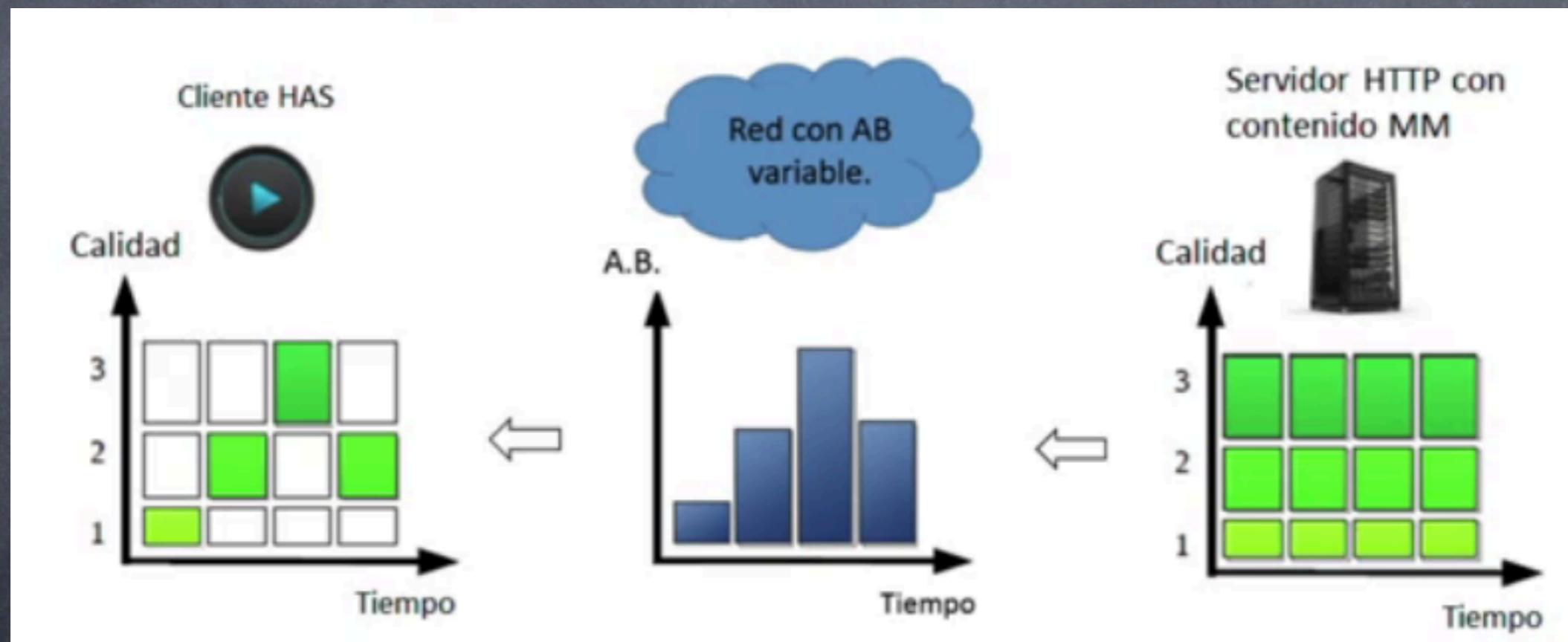
2 Analiza de forma continua los recursos y sus características con el archivo de manifiesto

3 Revisa la situación en el reproductor multimedia desde un punto de vista del ancho de banda y CPU

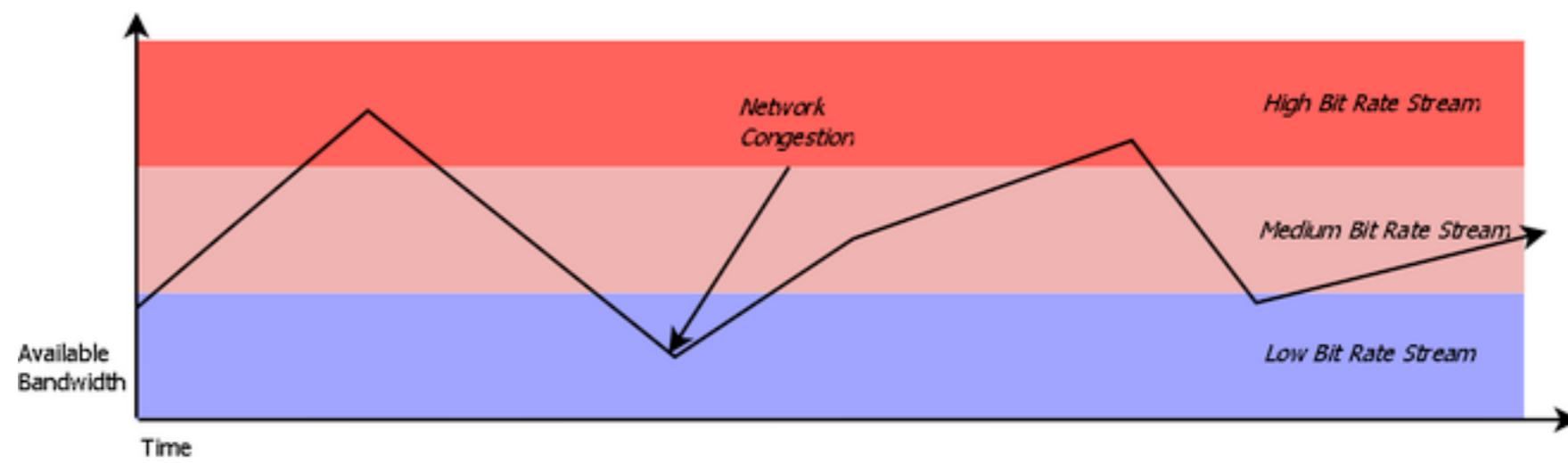
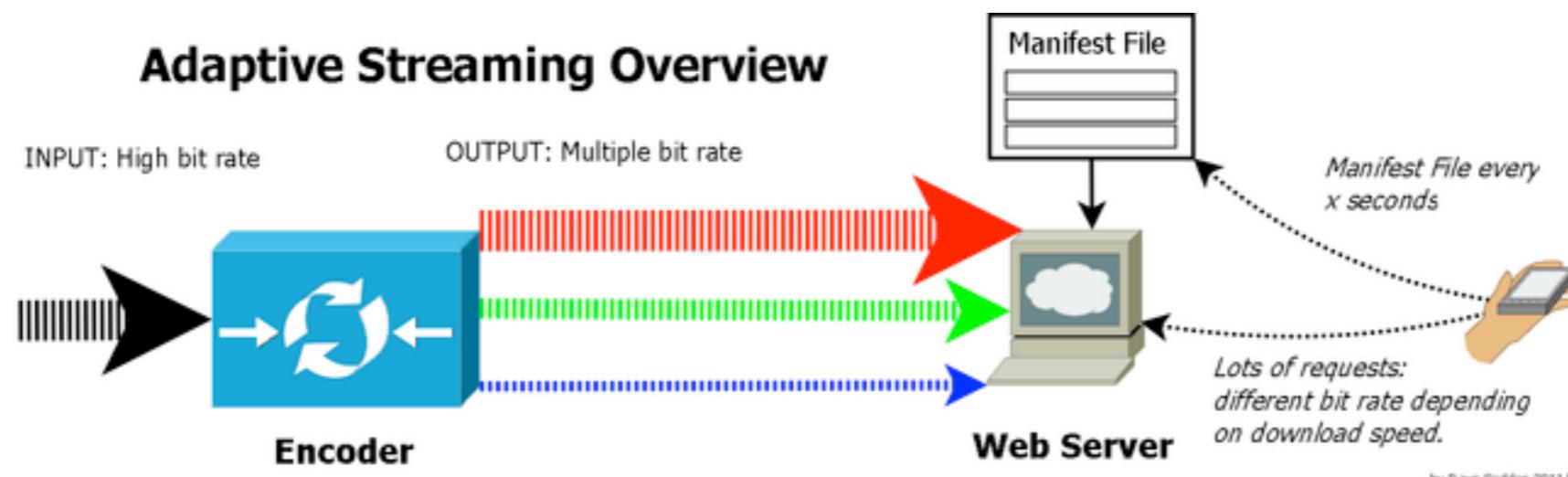
4 Selecciona del archivo de manifiesto cuál será el siguiente bit del recurso que va a solicitar durante la descarga progresiva

Safari es el único navegador que puede hacerlo de forma nativa

Difusión HTTP adaptativa



Difusión HTTP adaptativa



Difusión HTTP adaptativa



Posibilidades

- ⦿ CSS/CSS3
- ⦿ SVG (en línea, enmascaramiento, video en SVG)
- ⦿ Canvas
- ⦿ Web Workers
- ⦿ Javascript