UNIVERSIDADE FEDERAL DE PELOTAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

TEC2/TEC4: REDES MULTIMÍDIA (RMM)

Unidade 10
Vídeo de Fluxo Contínuo

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br

Sumário

- Introdução
- UDP de Fluxo Contínuo
- HTTP de Fluxo Contínuo
- HTTP de Fluxo Contínuo Adaptativo
- Redes de Distribuição de Conteúdo

Sumário

- **Introdução**
- UDP de Fluxo Contínuo
- HTTP de Fluxo Contínuo
- HTTP de Fluxo Contínuo Adaptativo
- Redes de Distribuição de Conteúdo

Introdução

Três tipos de aplicações de fluxo contínuo

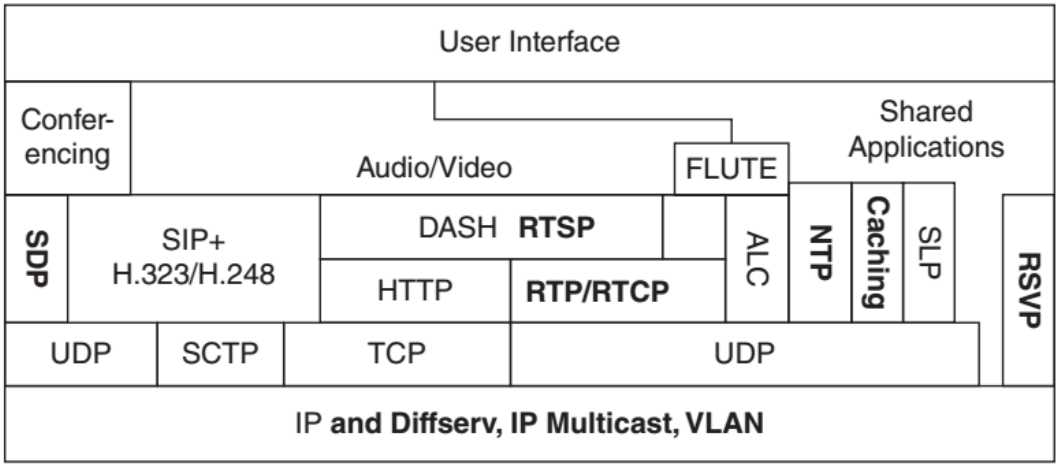
- *streaming* de áudio e vídeo *armazenado*
  - *streaming:* pode começar reprodução antes de baixar o arquivo completo
  - *armazenado (no servidor):* pode transmitir mais rápido que o áudio/vídeo é renderizado (implica em armazenar em *buffer* no cliente)
  - ex.: YouTube, Netflix, Hulu
- voz/vídeo *conversacional* sobre IP
  - natureza interativa da conversa humana limita tolerância a atrasos
  - ex.: Skype
- *streaming ao vivo* de áudio e vídeo
  - ex.: evento esportivo ao vivo (futebol)

Introdução

No caso específico de vídeo, os sistemas podem ser classificados de três formas

- UDP de fluxo contínuo
- HTTP de fluxo contínuo
- HTTP de fluxo contínuo adaptativo

} mais comuns



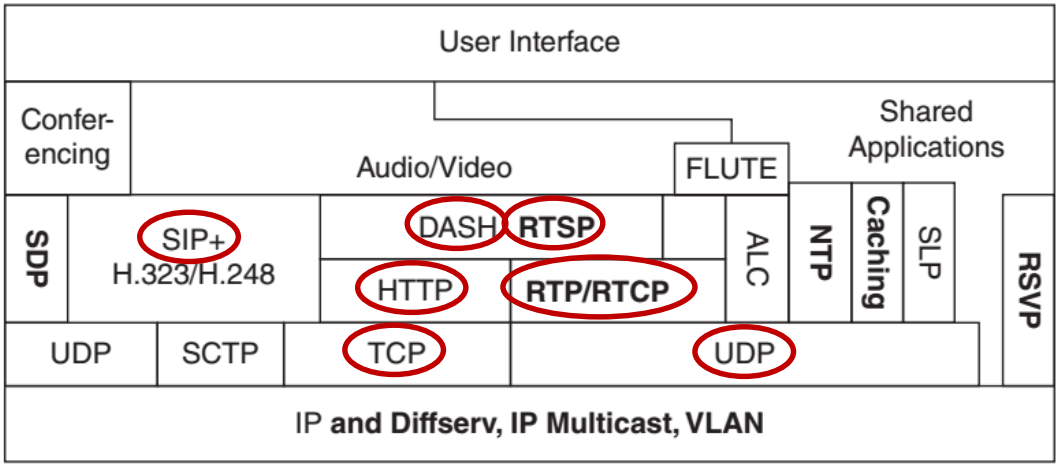
Pilha de Protocolos Multimídia

Introdução

No caso específico de vídeo, os sistemas podem ser classificados de três formas

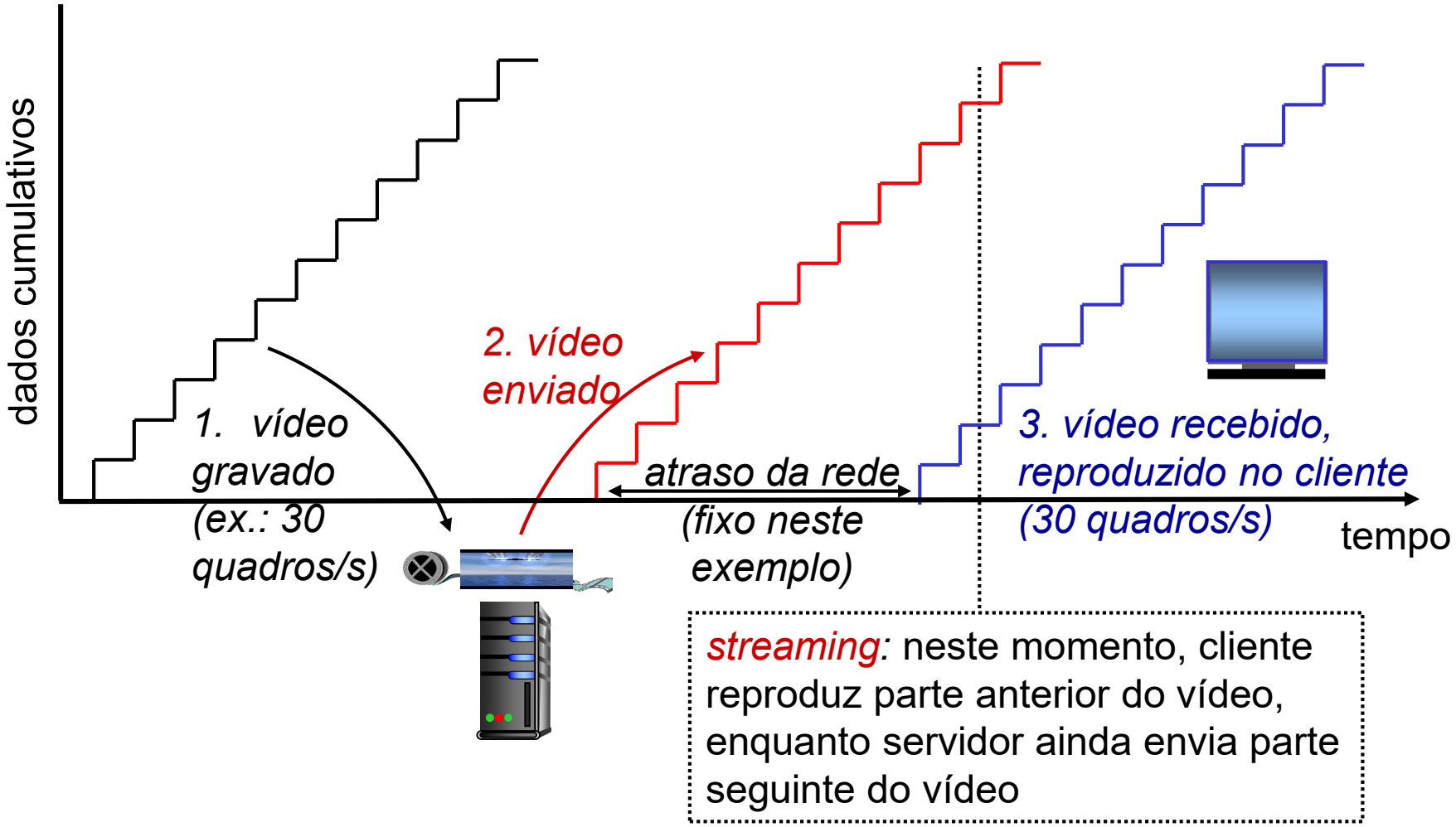
- UDP de fluxo contínuo
- HTTP de fluxo contínuo
- HTTP de fluxo contínuo adaptativo

} mais comuns



Pilha de Protocolos Multimídia

Introdução



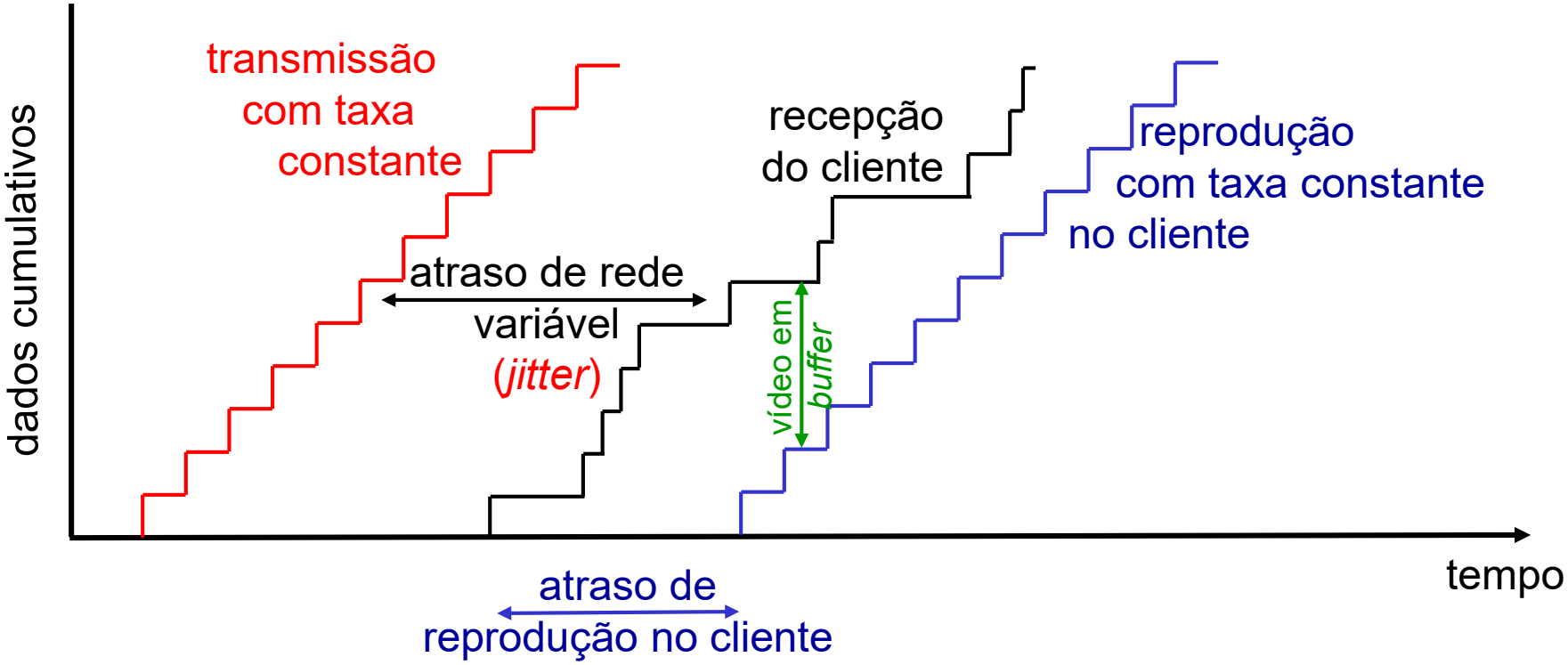
streaming: neste momento, cliente reproduz parte anterior do vídeo, enquanto servidor ainda envia parte seguinte do vídeo

Introdução

Desafios dos sistemas de *streaming* de vídeo

- *reprodução contínua*: depois que o cliente começa a reprodução, a temporização original deve ser obedecida
  - … mas *atrasos da rede são variáveis* (*jitter*), então precisamos de *buffer no cliente* para atingir requisitos de reprodução
- outros desafios:
  - interatividade do cliente: *pause*, *fast-forward*, *rewind*, saltos no vídeo
  - pacotes de vídeo podem ser perdidos e retransmitidos

Introdução



- *buffer* no cliente e atraso de reprodução: compensam o atraso na rede e *jitter*

Sumário

- Introdução
- **UDP de Fluxo Contínuo**
- HTTP de Fluxo Contínuo
- HTTP de Fluxo Contínuo Adaptativo
- Redes de Distribuição de Conteúdo

UDP de Fluxo Contínuo

- servidor transmite vídeo a uma taxa apropriada para o cliente (taxa de consumo)
  - geralmente: taxa de envio = taxa de codificação = taxa constante
  - ex.: taxa de consumo = 2 Mbits/s e pacote UDP = 8000 bits, então servidor envia um pacote a cada 4 ms
  - como UDP não faz controle de congestionamento, servidor pode enviar pacotes *sem restrições*!

UDP de Fluxo Contínuo

- cliente mantém pequeno *buffer* (2-5 segundos de vídeo) para remover *jitter*
- recuperação de erro: no nível da aplicação
- antes de entregar os dados do vídeo ao UDP, a aplicação deve encapsular os trechos (*chunks*) em pacotes RTP
- em paralelo, cliente e servidor devem manter uma sessão **RTSP** (*Real-Time Streaming Protocol*) para *controle de reprodução fora da banda*
  - comandos de pausar, retornar, reposicionar vídeo, etc.
- fluxo UDP pode não conseguir passar por *firewalls* que bloqueiam UDP para evitar congestionamento

Sumário

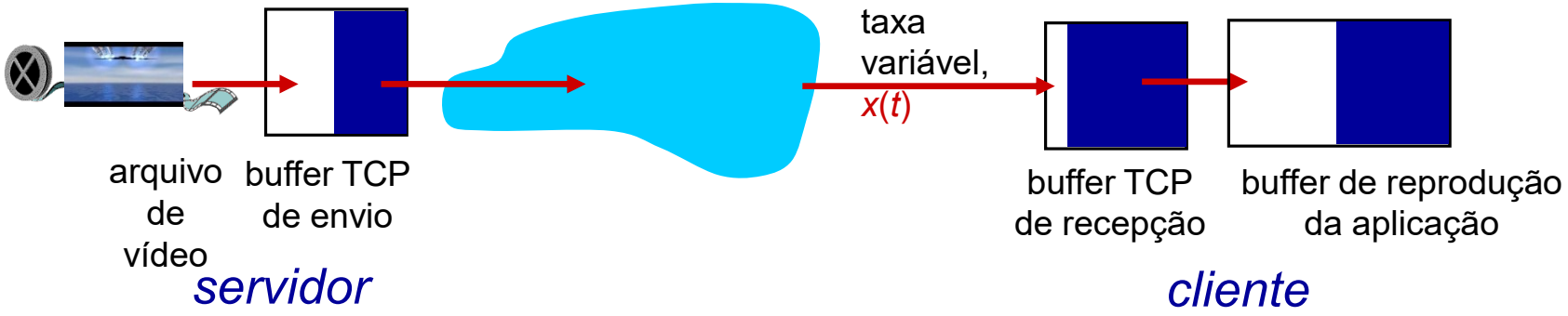
- Introdução
- UDP de Fluxo Contínuo
- **HTTP de Fluxo Contínuo**
- HTTP de Fluxo Contínuo Adaptativo
- Redes de Distribuição de Conteúdo

HTTP de Fluxo Contínuo

- vídeo completo armazenado num servidor HTTP como um arquivo comum, com uma URL específica
- cliente (ex.: *Chrome*) abre conexão TCP com servidor e envia comando HTTP GET para a URL
- servidor envia o arquivo em uma mensagem de resposta HTTP o mais rápido que consegue
  - segmentando o vídeo em segmentos TCP
  - obedecendo controle de congestionamento
  - obedecendo controle de fluxo
- cliente armazena bytes recebidos em *buffer*
- quando bytes recebidos excederem limiar, começa reprodução (decodifica e apresenta na tela)

HTTP de Fluxo Contínuo

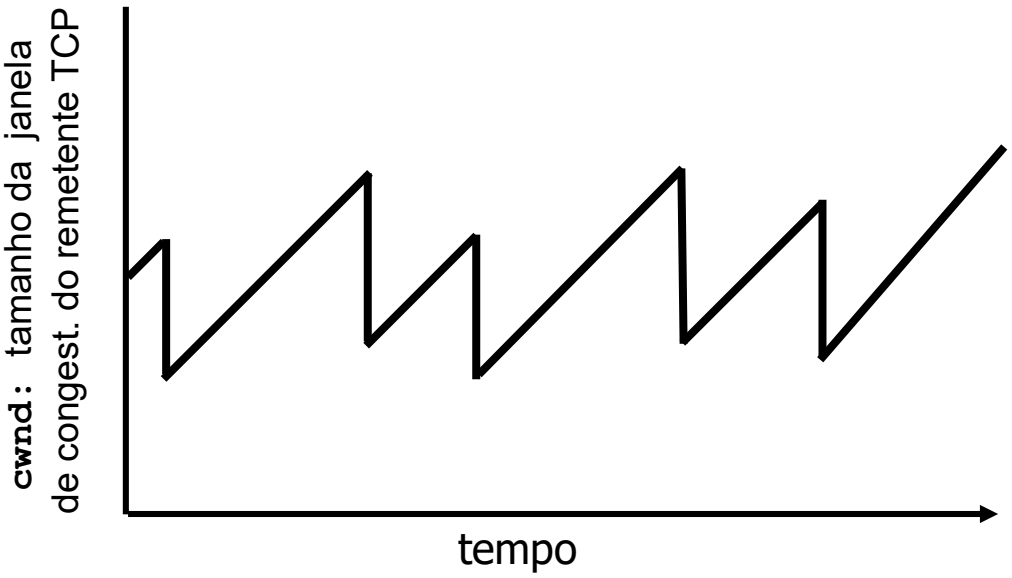
- arquivo multimídia recebido via HTTP GET
- envia na taxa máxima usando TCP



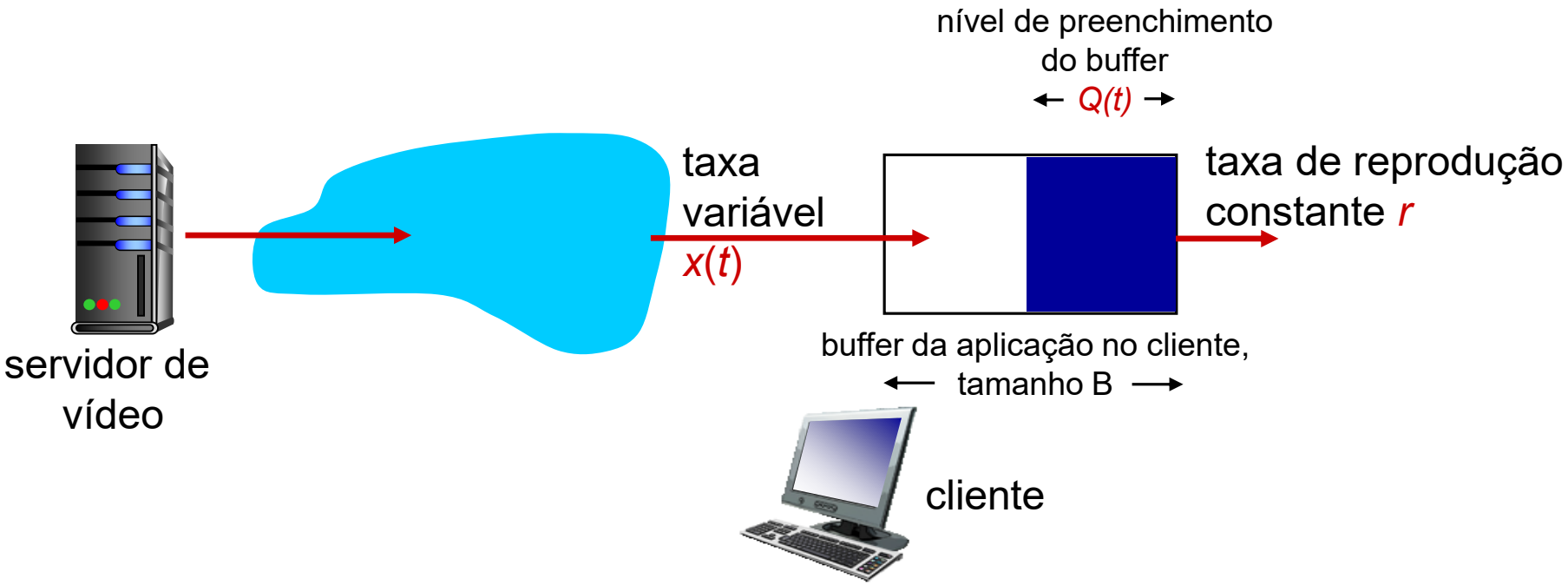
- taxa de preenchimento do *buffer* flutua devido ao controle de congestionamento do TCP e retransmissões (entrega em ordem)

HTTP de Fluxo Contínuo

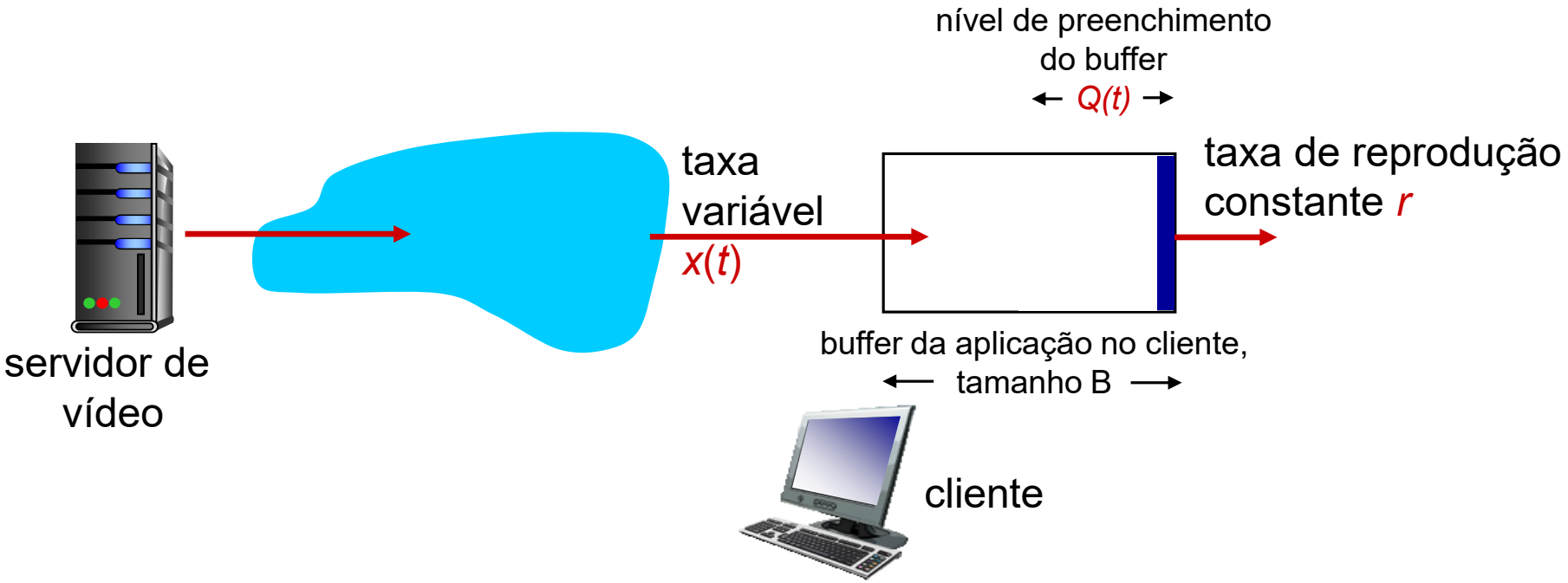
- controle de congestionamento do TCP leva a uma transmissão no formato "dente de serra"
- pacotes podem sofrer grande atraso por causa de retransmissões
- consenso geral era que TCP não servia para *streaming* de vídeo
- ... mas *buffer* do cliente e pré-busca mudaram o entendimento dos projetistas
- infra-estrutura TCP/IP disponível
- atravessa *firewalls*



Buffer no cliente e reprodução

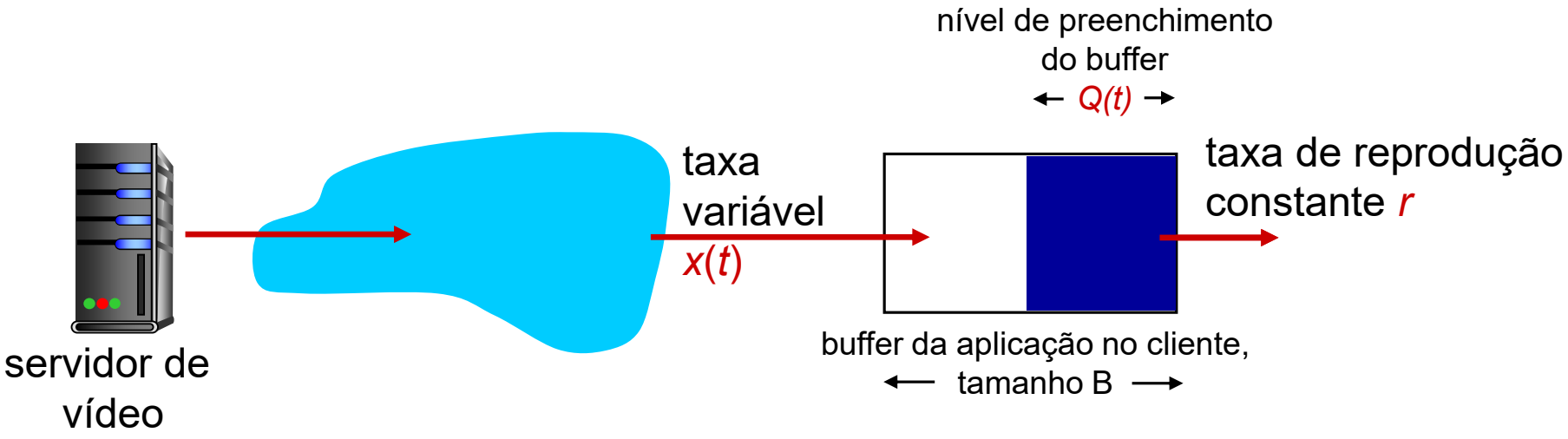


Buffer no cliente e reprodução



1. preenchimento inicial do buffer até reprodução começar em t_p
2. reprodução começa em $t_{p,}$
3. nível de preenchimento do buffer varia no tempo de acordo com variações na taxa *x(t)* e taxa de reprodução constante *r*

Buffer no cliente e reprodução

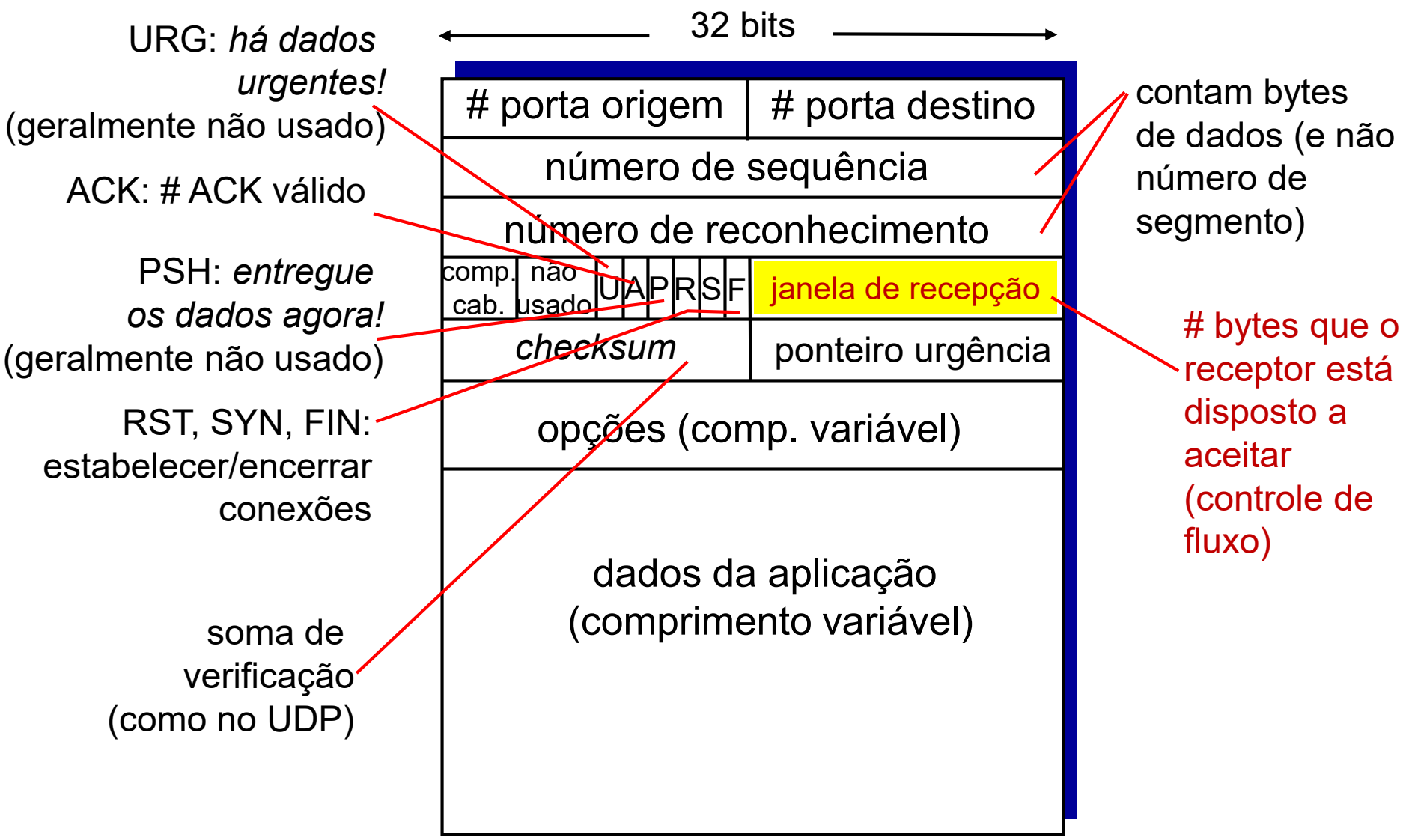


buffer de reprodução: taxa média de preenchimento (x), taxa de reprodução (r):

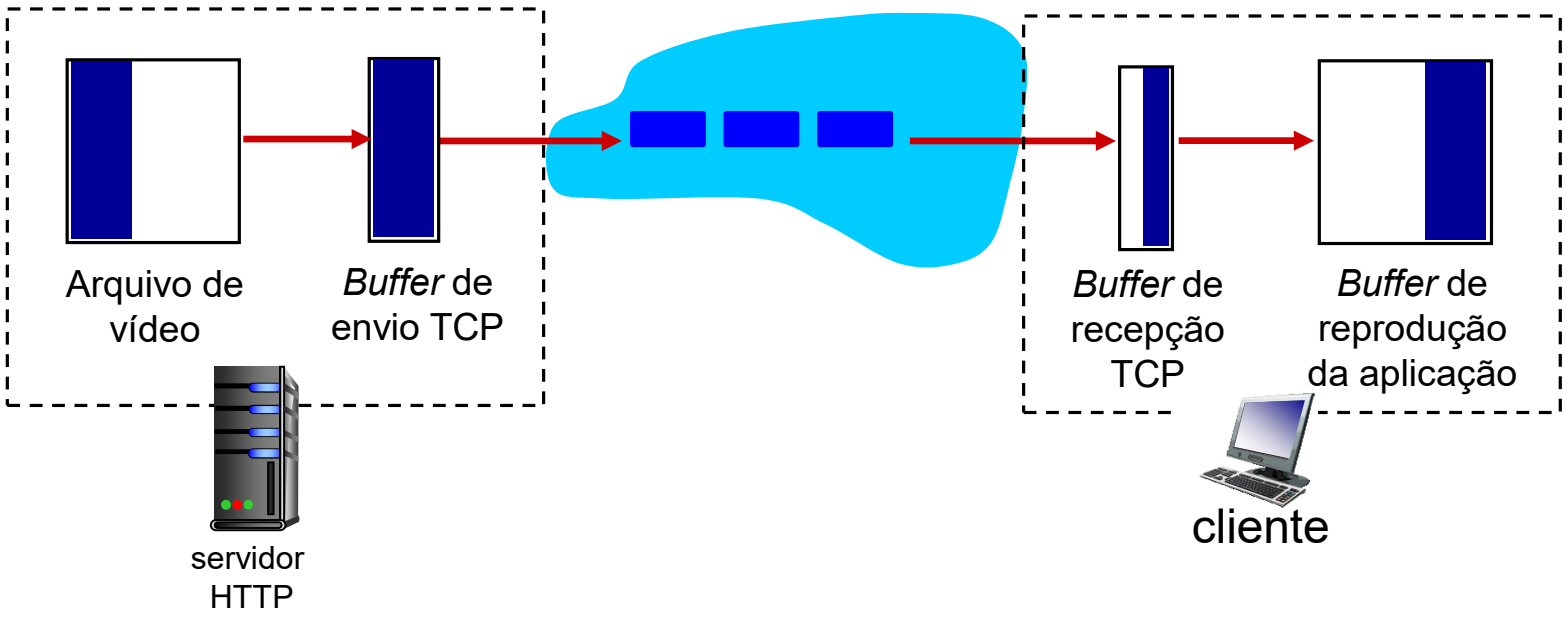
- x < r: eventualmente buffer esvazia (causa congelamento do vídeo até *buffer* preencher novamente)
- x > r: *buffer* nunca esvazia, se o atraso inicial de reprodução é grande o suficiente para absorver *jitter* em *x*(*t*)
  - *tradeoff do atraso inicial de reprodução:* esvaziamento do *buffer* menos provável com atraso grande, mas demora mais para começar a assistir vídeo

Recapitulando...

Estrutura do Segmento TCP



O que acontece quando cliente pausa o vídeo?



- bytes param de ser movidos do *buffer* de reprodução da aplicação, embora continuem a entrar no *buffer*
- *buffer* da aplicação fica cheio e bytes não podem ser mais removidos do *buffer* de recepção TCP para o *buffer* de reprodução
- *buffer* de recepção TCP fica cheio e campo "janela de recepção" = 0 no próximo segmento enviado ao servidor
- **conclusão: pausa no cliente leva a parada de envio no servidor**

Problema do HTTP de Fluxo Contínuo

- Todos os clientes recebem o mesmo vídeo, codificado da mesma forma
  - Muitas variações na largura de banda das redes
  - Diferentes perfis de clientes (móvel vs. fixo, tela grande vs. tela pequena, etc.)
  - Até no mesmo cliente há variações frequentes da rede!

- Solução: **HTTP de Fluxo Contínuo Adaptativo**
  - Vídeo codificado em muitas versões diferentes
    - Taxas de bits diferentes
    - Resoluções diferentes
    - Níveis de qualidade diferentes
  - Versões do vídeo segmentadas em trechos de poucos segundos
  - Cliente faz GET dos trechos, um de cada vez, de acordo com a característica da rede naquele momento

Sumário

- Introdução
- UDP de Fluxo Contínuo
- HTTP de Fluxo Contínuo
- **HTTP de Fluxo Contínuo Adaptativo**
- Redes de Distribuição de Conteúdo

HTTP de Fluxo Contínuo Adaptativo

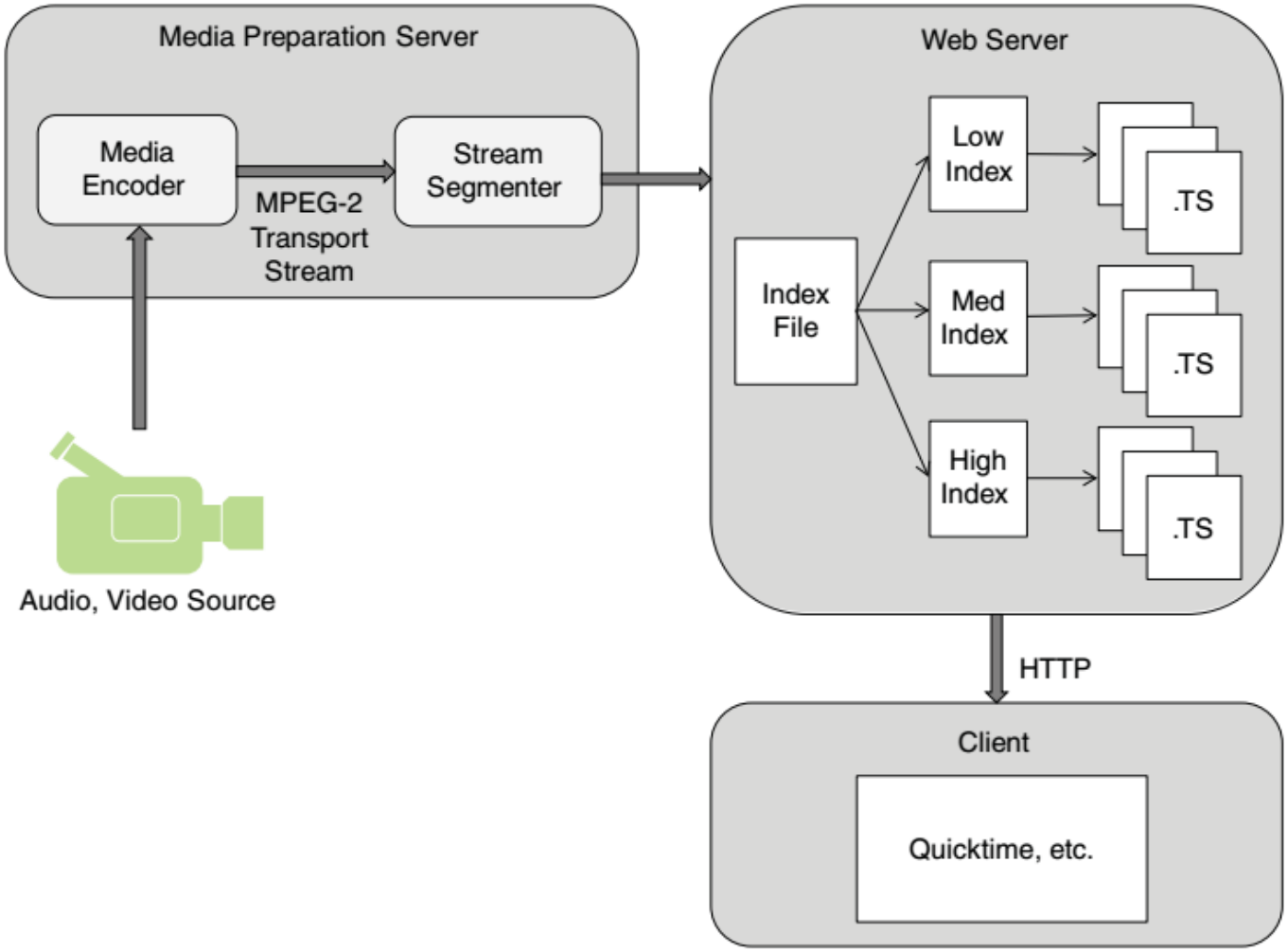
- Vídeo codificado em muitas versões diferentes:
  - Taxas de bits diferentes
  - Resoluções diferentes
  - Níveis de qualidade diferentes
- As diferentes versões do vídeo são segmentadas em trechos de poucos segundos
- Cliente faz HTTP GET dos trechos, um de cada vez, de acordo com a característica da rede naquele momento
- **Diversos padrões de *streaming* adaptativo:**
  - *HTTP Live Streaming* (**HLS**), da Apple
  - *Smooth Streaming* (**SS**), da Microsoft
  - *HTTP Dynamic Streaming* (**HDS**), da Adobe
  - *Dynamic Adaptive Streaming over HTTP* (**DASH**), do MPEG
- Não são protocolos!
  São padrões que utilizam o protocolo HTTP

HTTP *Live Streaming* (HLS) – Apple

- Padrão de *streaming* adaptativo da Apple
- Padrão proprietário
- Disponível em dispositivos iOS
- Suporte a vídeo sob demanda e ao vivo
- Único padrão de *streaming* nativamente suportado na plataforma da Apple
  - Qualquer aplicação em iPhone ou iPad que envia grandes quantidades de áudio ou vídeo deve usar HLS
- Suportado na maioria dos dispositivos de outras plataformas

HTTP *Live Streaming* (HLS) – Apple

Estrutura do HLS



HTTP *Live Streaming* (HLS) – Apple

Três componentes principais

1) Preparação de mídia

- áudio é codificado como AAC ou MP3
- vídeo é codificado como H.264
- áudio e vídeo incorporados em contâiner MPEG-TS (formato .TS)
- *stream* pode ser transcodificado para diferentes níveis de qualidade e taxa de bits
- *stream* segmentado em segmentos individuais (*chunks*) de 5-10 segundos
- cria conjunto de arquivos de índice (formato .M3U8), descrevendo todos os arquivos .TS disponíveis nos vários níveis de qualidade
- este processo todo pode acontecer em tempo real, à medida que o vídeo é gravado, codificado e transmitido (vídeo ao vivo)

HTTP *Live Streaming* (HLS) – Apple

Três componentes principais

2) Servidor web

- depois que todos os arquivos .TS (*chunks*) e .M3U8 (índices) foram criados, eles são armazenados em um servidor HTTP comum
- servidor HTTP configurado para hospedar os arquivos e entregá-los à medida que as requisições (GET) chegarem

3) Reprodutor no cliente

- reprodução totalmente controlada pela aplicação do cliente
- cliente requisita os arquivos de índice e arquivos .TS na qualidade desejada
- cliente altera a qualidade requisitada à medida que notar mudanças na qualidade do serviço de entrega (largura de banda, congestionamento, etc.)
- controle transparente para usuário

HTTP *Live Streaming* (HLS) – Apple

Exemplo de arquivo índice HLS (.M3U8)

HLS Index fil for 3 segments of 10 seconds each

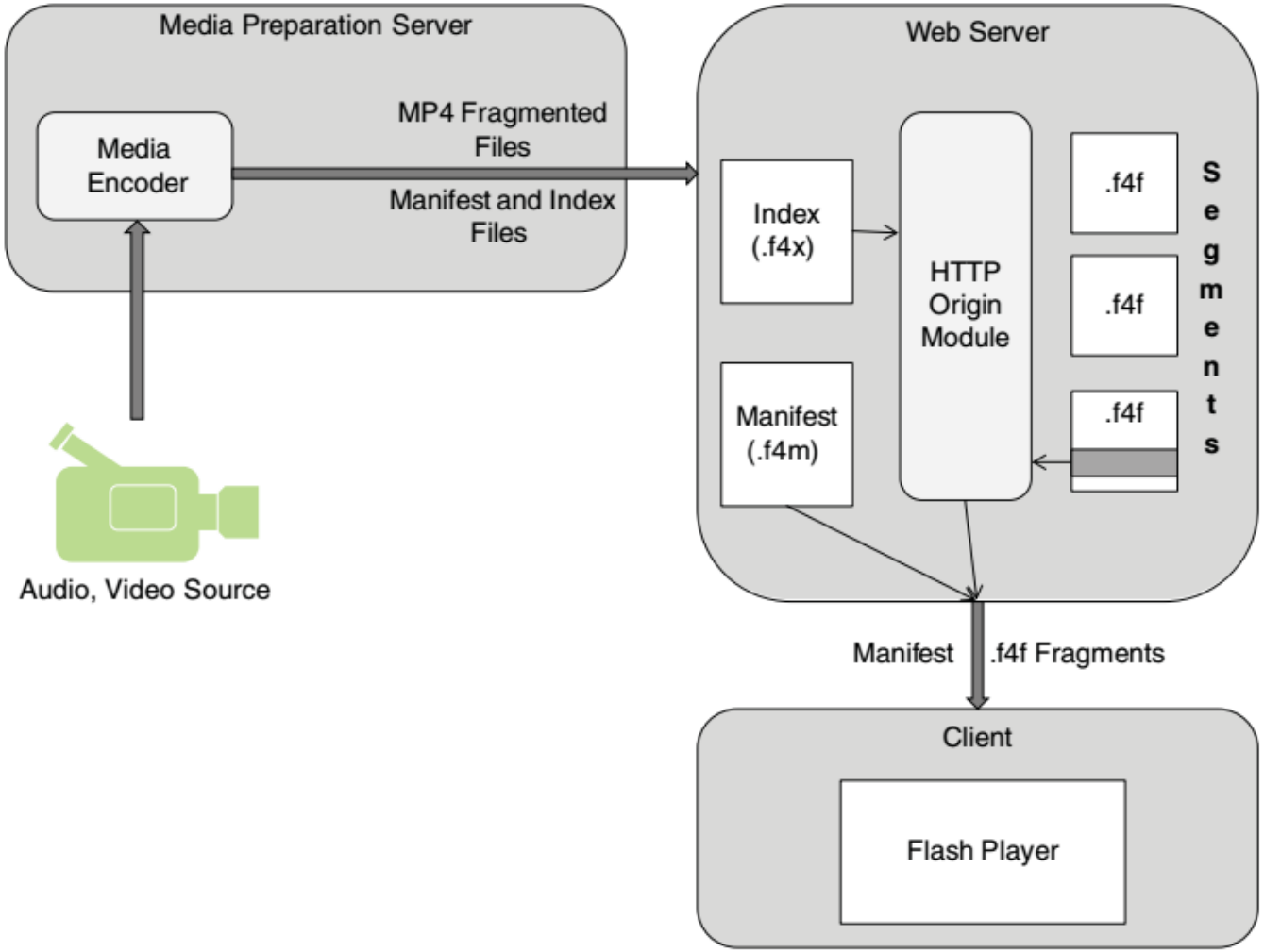
```
#EXT-X-VERSION:3
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:1

#EXTINF:10.0,
http://media-server.example.com/segment0.ts
#EXTINF:10.0,
http://media-server.example.com/segment1.ts
#EXTINF:9.1,
http://media-server.example.com/segment2.ts
#EXT-X-ENDLIST
```

HTTP *Dynamic Streaming* (HDS) – Adobe

- HDS é o padrão de *streaming* da Adobe
- Vídeo codificado em H.264
- Conteúdo armazenado em arquivos .f4f
  - um segmento contém um vídeo inteiro
  - cada segmento codificado para uma determinada taxa de bits
- Arquivo manifesto (.f4m)
  - descreve conteúdo para o cliente
- Arquivo índice (.f4x)
  - descreve como o servidor deve extrair fragmentos dos arquivos .f4f
- Servidor contém plugin *HTTP Origin Module*
  - transfere e interpreta os arquivos manifesto e índice

HTTP *Dynamic Streaming* (HDS) – Adobe

Estrutura do HDS



HTTP *Dynamic Streaming* (HDS) – Adobe

Eventos para reproduzir um vídeo com HDS

- cliente HDS envia requisição HTTP ao servidor
  - ex.: GET http://www.server.com/media/sample-video.f4m
- servidor encaminha requisição ao *HTTP Origin Module*
  - *HTTP Origin Module* envia arquivo manifesto (.f4m) ao cliente
- cliente recebe manifesto (.f4m)
  - usa dados no conteúdo para traduzir temporização do vídeo para número de segmento, número de fragmento e formato de representação que vai solicitar
- cliente envia requisição HTTP pedindo fragmento específico
  - ex.: GET http://www.server.com/media/sample-video-seg1-frag1
- servidor envia requisição ao *HTTP Origin Module*
  - *HTTP Origin Module* usa arquivo índice (.f4x) para definir o deslocamento (em bytes) do arquivo de mídia (.f4f) e envia o fragmento solicitado ao cliente

Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

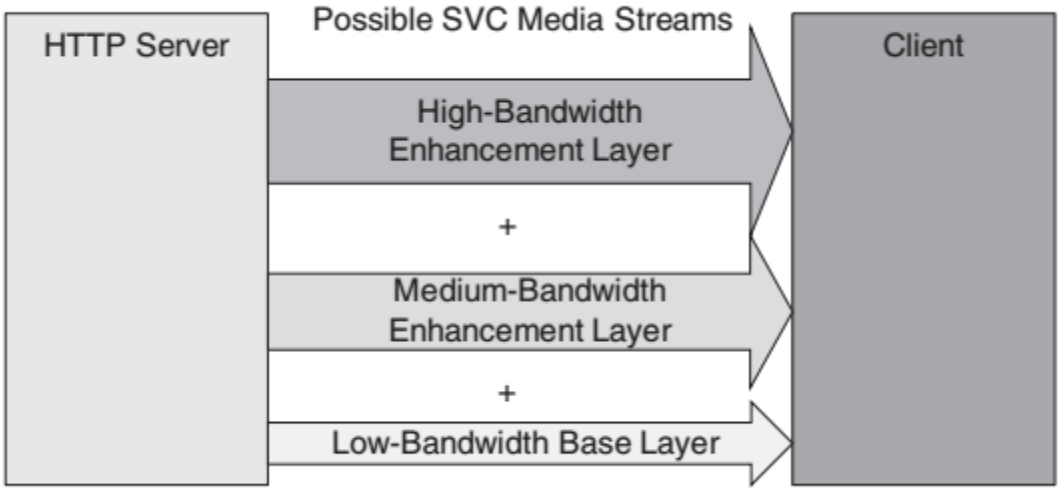
- Padrão do MPEG para *streaming* de vídeo (**MPEG-DASH**), lançado em 2012 (atualizado em 2014)
- DASH também serve como termo genérico para descrever qualquer serviço de *streaming* por HTTP
- Similar aos padrões anteriores nos seguintes aspectos:
  - HTTP usado para o transporte
  - embora não seja estritamente especificado, a codificação mais comum é com H.264 (atualmente em migraçaõ para HEVC e VP9)
  - cliente escolhe e muda qualidade / taxa de bits do *stream*
  - segmentos armazenados como arquivos individuais (como no HLS)
  - utiliza um índice e arquivo de descrição do *stream*

Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

- *servidor:*
  - divide arquivo do vídeo em múltiplos segmentos
  - cada segmento é armazenado após codificado em diferentes taxas de bit
  - *arquivo manifesto (MPD):* provê as URLs para diferentes segmentos
- *cliente:*
  - periodicamente mede a largura de banda do servidor para o cliente (cada *player* faz isso de forma diferente)
  - consulta manifesto e requisita um segmento por vez
    - escolhe a taxa de bits máxima possível dada a largura de banda atual
    - escolhe taxas de bit diferentes em diferentes momentos
    - **Repetindo:** cada *player* faz isso de forma diferente!

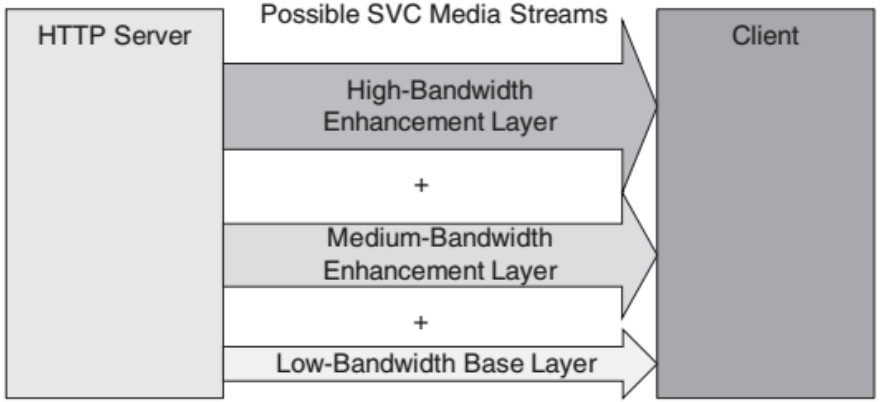
Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

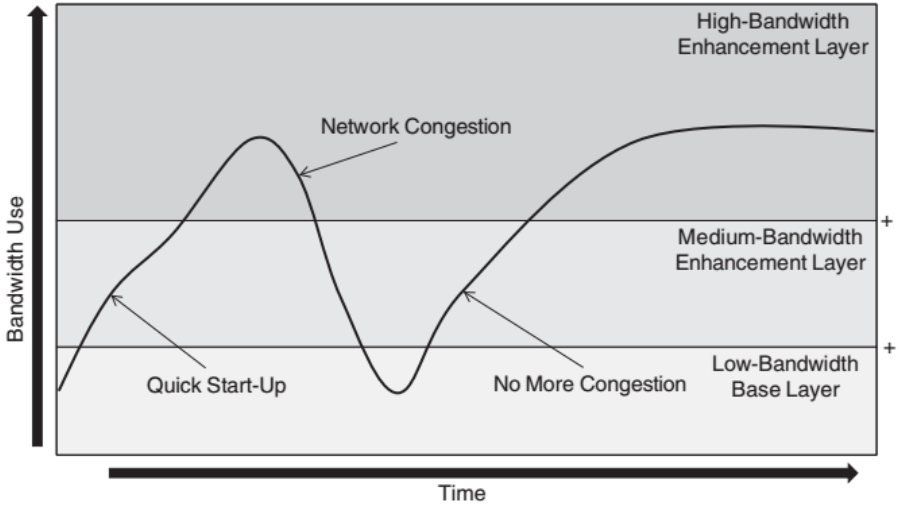
- A diferença do DASH em relação aos demais padrões é a **escalabilidade**
- ***Scalable Video Coding* (SVC)**
  - camadas aditivas do vídeo
  - cliente **sempre recebe** o a camada-base (*base layer*)
  - para melhorar a qualidade, camadas de enriquecimento (*enhancement layers*) são adicionadas



Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

- Para iniciar rapidamente transmissão, apenas a camada-base é transmitida no início
- **Com condições apropriadas**, camadas de enriquecimento são adicionadas
- Camadas de enriquecimento são removidas com há congestionamento na rede
- E assim por diante...
- Cada *player* implementa diferentes métodos de controle



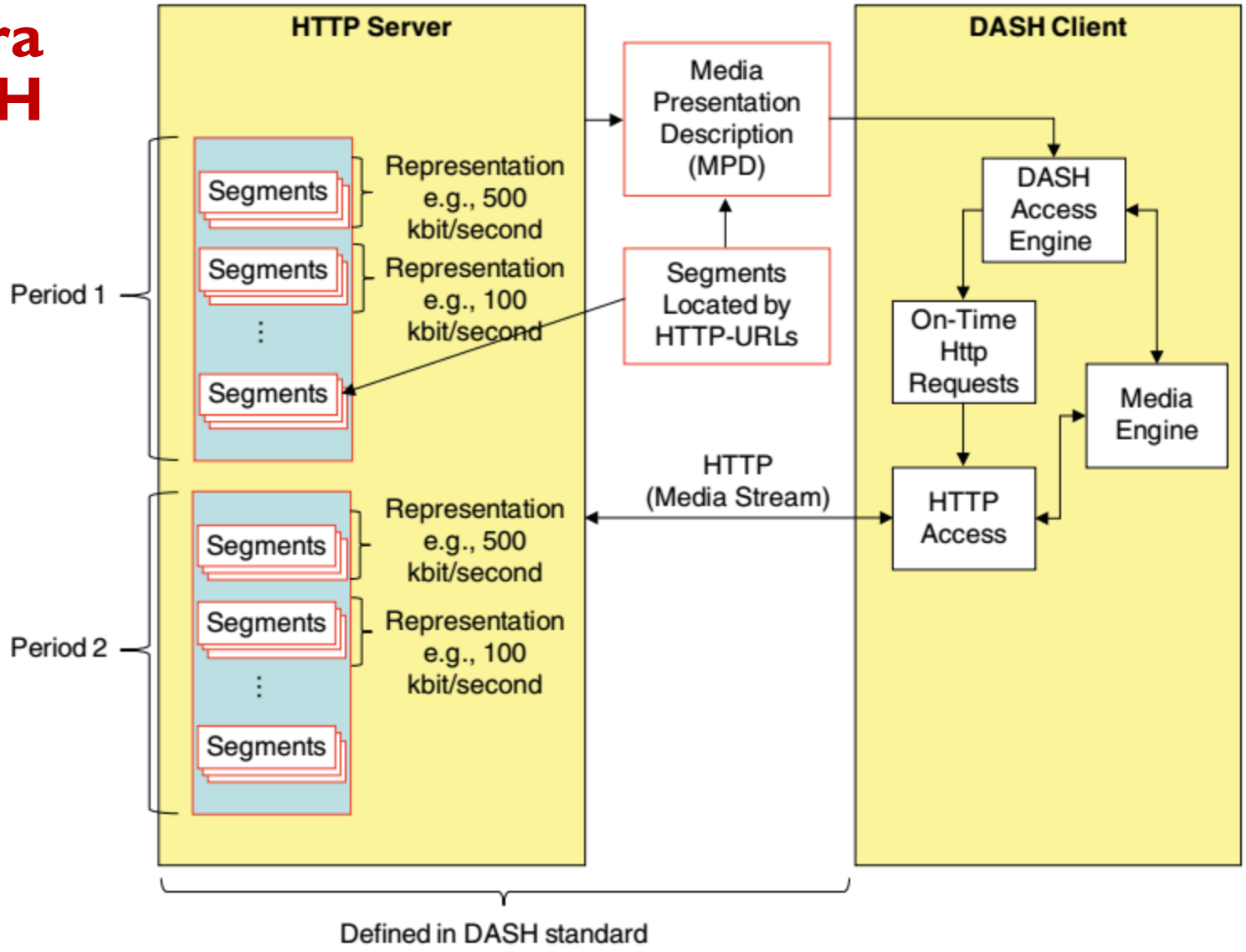


Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

- arquivo manifesto: *Media Presentation Description (MPD)*, em formato XML
- MPD define, para áudio e vídeo:
  - **Períodos** (*Period*): maior unidade de trecho de um *stream* (geralmente na ordem de minutos). Todos períodos concatenados formam a mídia completa.
  - **Conjuntos de Adaptação** (*Adaptation Set*): contém um *stream* de mídia ou um conjunto de *streams* de mídia. Geralmente, um Período contém um Conjunto de Adaptação de áudio e um Conjunto de Adaptação de vídeo. Geralmente são escolhidos pelo usuário ou player usando preferências do usuário.
  - **Representações** (*Representation*): cada período tem múltiplas representações possíveis (codificador, resolução, quadros por segundo, taxa de bits, etc.). Também são chamadas de camadas.
  - **Segmentos** (*Segment*): cada representação consiste em uma coleção de segmentos, cada um identificado por uma URL única.

Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

Estrutura do DASH



Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

Estrutura do MPD

MPD Sample

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S" minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760" codecs="avc1.640028"
height="1080" id="1" mimeType="video/mp4" width="1920">
        <BaseURL>sample1-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
          <Initialization range="0-673" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460" codecs="avc1.4d401e"
height="480" id="3" mimeType="video/mp4" width="854">
        <BaseURL>sample1-87.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="264835" codecs="avc1.4d4015"
height="240" id="5" mimeType="video/mp4" width="426">
        <BaseURL>sample1-85.mp4</BaseURL>
        <SegmentBase indexRange="672-1147">
          <Initialization range="0-671" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet>
      <ContentComponent contentType="audio" id="2" />
      <Representation bandwidth="127236" codecs="mp4a.40.2"
id="6" mimeType="audio/mp4" numChannels="2" sampleRate="44100">
        <BaseURL>sample1-8c.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="31749" codecs="mp4a.40.5"
 id="8" mimeType="audio/mp4" numChannels="1" sampleRate="22050">
        <BaseURL>sample1-8b.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

MPD Sample

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S" minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
```

Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

MPD Sample

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S" minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760" codecs="avc1.640028"
height="1080" id="1" mimeType="video/mp4" width="1920">
        <BaseURL>sample1-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
          <Initialization range="0-673" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460" codecs="avc1.4d401e"
height="480" id="3" mimeType="video/mp4" width="854">
        <BaseURL>sample1-87.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="264835" codecs="avc1.4d4015"
height="240" id="5" mimeType="video/mp4" width="426">
        <BaseURL>sample1-85.mp4</BaseURL>
        <SegmentBase indexRange="672-1147">
          <Initialization range="0-671" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet>
      <ContentComponent contentType="audio" id="2" />
      <Representation bandwidth="127236" codecs="mp4a.40.2"
id="6" mimeType="audio/mp4" numChannels="2" sampleRate="44100">
        <BaseURL>sample1-8c.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="31749" codecs="mp4a.40.5"
 id="8" mimeType="audio/mp4" numChannels="1" sampleRate="22050">
        <BaseURL>sample1-8b.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Estrutura do MPD

```
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760" codecs="avc1.640028"
height="1080" id="1" mimeType="video/mp4" width="1920">
        <BaseURL>sample1-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
          <Initialization range="0-673" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460" codecs="avc1.4d401e"
height="480" id="3" mimeType="video/mp4" width="854">
        <BaseURL>sample1-87.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="264835" codecs="avc1.4d4015"
height="240" id="5" mimeType="video/mp4" width="426">
        <BaseURL>sample1-85.mp4</BaseURL>
        <SegmentBase indexRange="672-1147">
          <Initialization range="0-671" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
```

Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

Estrutura do MPD

MPD Sample

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S" minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760" codecs="avc1.640028"
height="1080" id="1" mimeType="video/mp4" width="1920">
        <BaseURL>sample1-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
          <Initialization range="0-673" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460" codecs="avc1.4d401e"
height="480" id="3" mimeType="video/mp4" width="854">
        <BaseURL>sample1-87.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="264835" codecs="avc1.4d4015"
height="240" id="5" mimeType="video/mp4" width="426">
        <BaseURL>sample1-85.mp4</BaseURL>
        <SegmentBase indexRange="672-1147">
          <Initialization range="0-671" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet>
      <ContentComponent contentType="audio" id="2" />
      <Representation bandwidth="127236" codecs="mp4a.40.2"
id="6" mimeType="audio/mp4" numChannels="2" sampleRate="44100">
        <BaseURL>sample1-8c.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="31749" codecs="mp4a.40.5"
 id="8" mimeType="audio/mp4" numChannels="1" sampleRate="22050">
        <BaseURL>sample1-8b.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

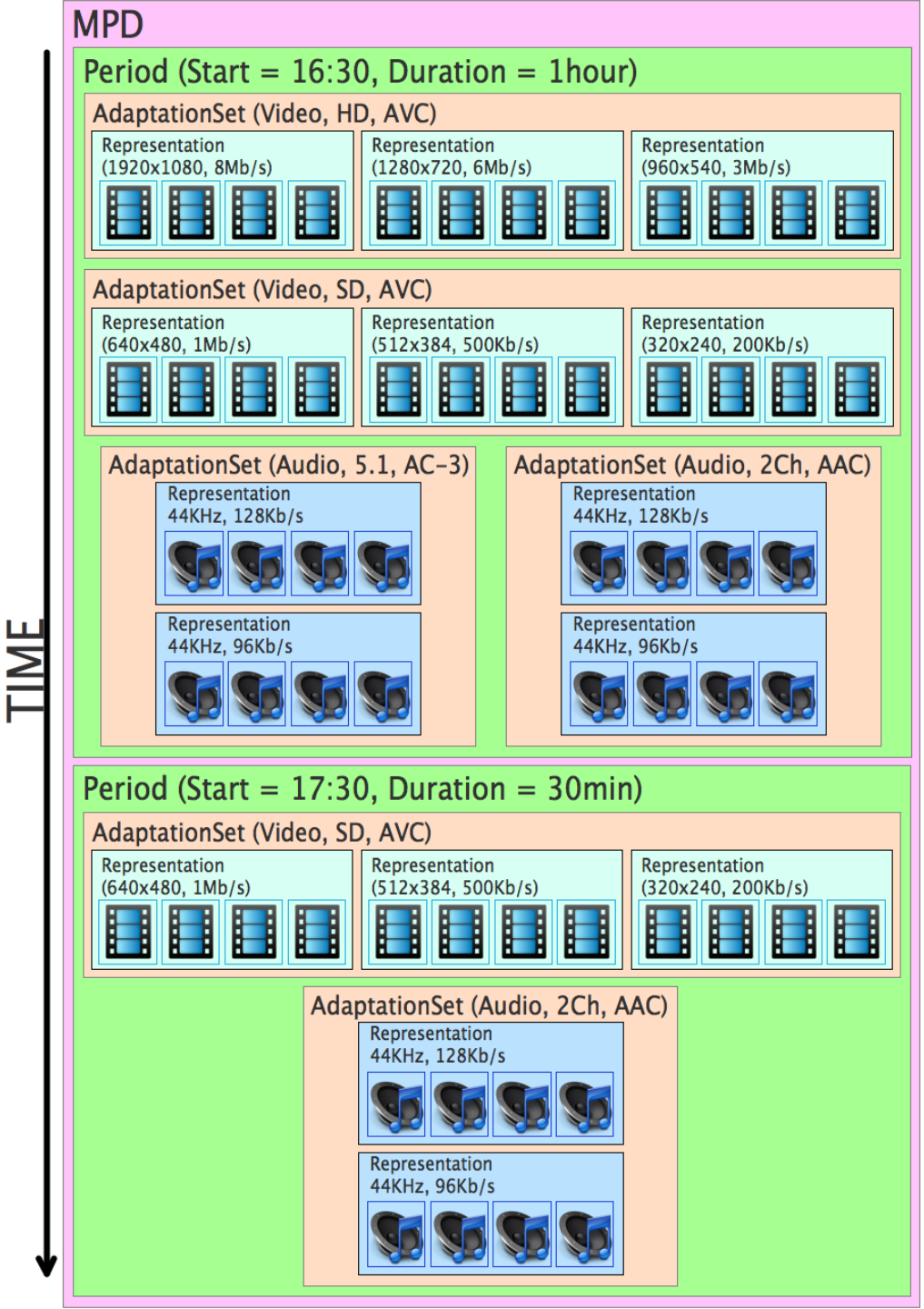
```
    <AdaptationSet>
      <ContentComponent contentType="audio" id="2" />
      <Representation bandwidth="127236" codecs="mp4a.40.2"
id="6" mimeType="audio/mp4" numChannels="2" sampleRate="44100">
        <BaseURL>sample1-8c.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="31749" codecs="mp4a.40.5"
 id="8" mimeType="audio/mp4" numChannels="1" sampleRate="22050">
        <BaseURL>sample1-8b.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
```

Dynamic Adaptive Streaming over HTTP (DASH) – MPEG

Estrutura do MPD

MPD Sample

```
<MPD xmlns="urn:mpeg:DASH:schema:MPD:2011"
mediaPresentationDuration="PT0H3M1.63S" minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H3M1.63S" start="PT0S">
    <AdaptationSet>
      <ContentComponent contentType="video" id="1" />
      <Representation bandwidth="4190760" codecs="avc1.640028"
height="1080" id="1" mimeType="video/mp4" width="1920">
        <BaseURL>sample1-89.mp4</BaseURL>
        <SegmentBase indexRange="674-1149">
          <Initialization range="0-673" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="869460" codecs="avc1.4d401e"
height="480" id="3" mimeType="video/mp4" width="854">
        <BaseURL>sample1-87.mp4</BaseURL>
        <SegmentBase indexRange="708-1183">
          <Initialization range="0-707" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="264835" codecs="avc1.4d4015"
height="240" id="5" mimeType="video/mp4" width="426">
        <BaseURL>sample1-85.mp4</BaseURL>
        <SegmentBase indexRange="672-1147">
          <Initialization range="0-671" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
    <AdaptationSet>
      <ContentComponent contentType="audio" id="2" />
      <Representation bandwidth="127236" codecs="mp4a.40.2"
id="6" mimeType="audio/mp4" numChannels="2" sampleRate="44100">
        <BaseURL>sample1-8c.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
      <Representation bandwidth="31749" codecs="mp4a.40.5"
 id="8" mimeType="audio/mp4" numChannels="1" sampleRate="22050">
        <BaseURL>sample1-8b.mp4</BaseURL>
        <SegmentBase indexRange="592-851">
          <Initialization range="0-591" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```


MPD
Period (Start = 16:30, Duration = 1hour)
AdaptationSet (Video, HD, AVC)
Representation (1920x1080, 8Mb/s)
Representation (1280x720, 6Mb/s)
Representation (960x540, 3Mb/s)
AdaptationSet (Video, SD, AVC)
Representation (640x480, 1Mb/s)
Representation (512x384, 500Kb/s)
Representation (320x240, 200Kb/s)
AdaptationSet (Audio, 5.1, AC-3)
Representation 44KHz, 128Kb/s
Representation 44KHz, 96Kb/s
AdaptationSet (Audio, 2Ch, AAC)
Representation 44KHz, 128Kb/s
Representation 44KHz, 96Kb/s
TIME
Period (Start = 17:30, Duration = 30min)
AdaptationSet (Video, SD, AVC)
Representation (640x480, 1Mb/s)
Representation (512x384, 500Kb/s)
Representation (320x240, 200Kb/s)
AdaptationSet (Audio, 2Ch, AAC)
Representation 44KHz, 128Kb/s
Representation 44KHz, 96Kb/s

Sumário

- Introdução
- UDP de Fluxo Contínuo
- HTTP de Fluxo Contínuo
- HTTP de Fluxo Contínuo Adaptativo
- **Redes de Distribuição de Conteúdo**

Redes de Distribuição de Conteúdo

- *desafio:* como enviar conteúdo (selecionado entre milhões de vídeos) para centenas de milhares de usuários simultâneos?

- *opção 1:* grande e único "mega-servidor"
  - longos caminhos até clientes distantes
  - ponto de congestionamento da rede (enlace de gargalo)
  - várias cópias do mesmo vídeo enviadas no enlace de saída
  - desperdício de largura de banda (e de dinheiro)
  - único ponto de falha

… simplesmente: esta solução *não é escalável!*

Redes de Distribuição de Conteúdo

- *desafio:* como enviar conteúdo (selecionado entre milhões de vídeos) para centenas de milhares de usuários simultâneos?

- *opção 2:* armazenar/oferecer múltiplas cópias dos vídeos em múltiplos locais geograficamente distribuídos *(CDN)*
  - *enter deep:* coloca servidores CDN "profundamente" dentro das redes de acesso dos *Internet Service Providers* (ISPs)
    - perto dos usuários
    - estratégia da *Akamai*, *clusters* de servidores em 1700 locais
  - *bring home:* menor número (dezenas) de *clusters* enormes próximos (mas não dentro) das redes de acesso
    - estratégia da *Limelight*

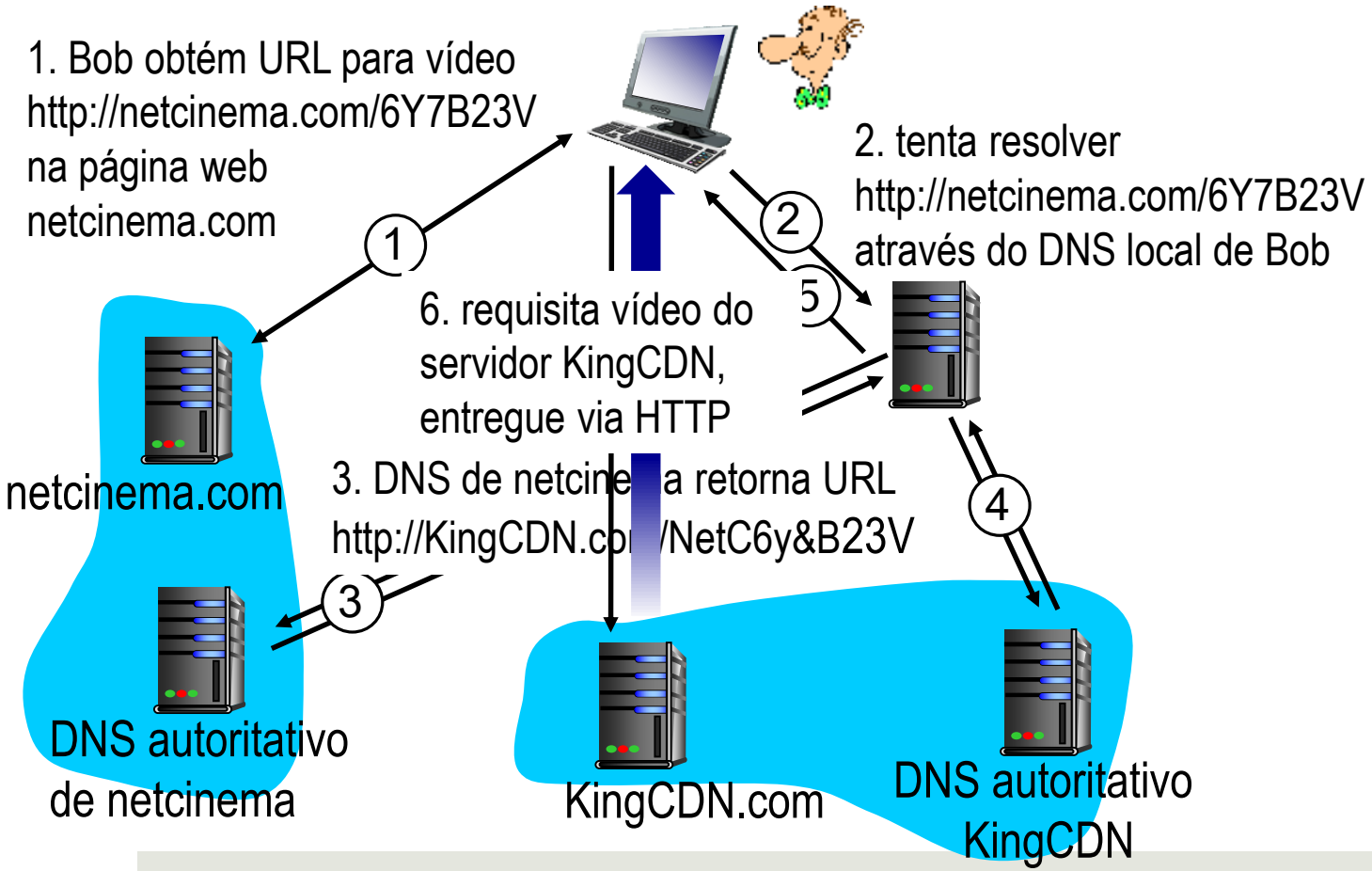
Redes de Distribuição de Conteúdo

- uma vez que os *clusters* estejam operando
  - CDN replica conteúdos mais vistos através dos *clusters*
  - funcionamento lembra o de uma cache web
- se cliente requisita vídeo de um *cluster* e este não possui o vídeo
  - *cluster* busca o vídeo no repositório central (ou em outro *cluster*)
  - armazena cópia do vídeo localmente
  - ao mesmo tempo, envia o fluxo contínuo para o cliente
- quando a memória do *cluster* fica cheia, remove vídeos menos frequentemente requisitados

Operação da CDN

Bob (cliente) requisita vídeo http://netcinema.com/6Y7B23V

- vídeo armazenado na CDN em http://KingCDN.com/NetC6y&B23V



Estratégias de Seleção de *cluster*

- *desafio:* como direcionar cliente para um nó CDN "bom" dentro da CDN ?
  - escolhe nó CDN geograficamente mais perto do cliente
  - escolhe nó CDN com atraso menor (ou número mínimo de saltos) para o cliente (nós CDN periodicamente fazem um acesso *ping* aos ISPs, reportando resultados aos DNS do CDN)
  - IP *anycast*

- *alternativa:* deixar o *cliente* decidir – entrega ao cliente uma lista de diversos servidores CDN
  - cliente faz *ping* nos servidores, escolhe "melhor"
  - estratégia da Netflix

Estratégias de Seleção de *cluster*

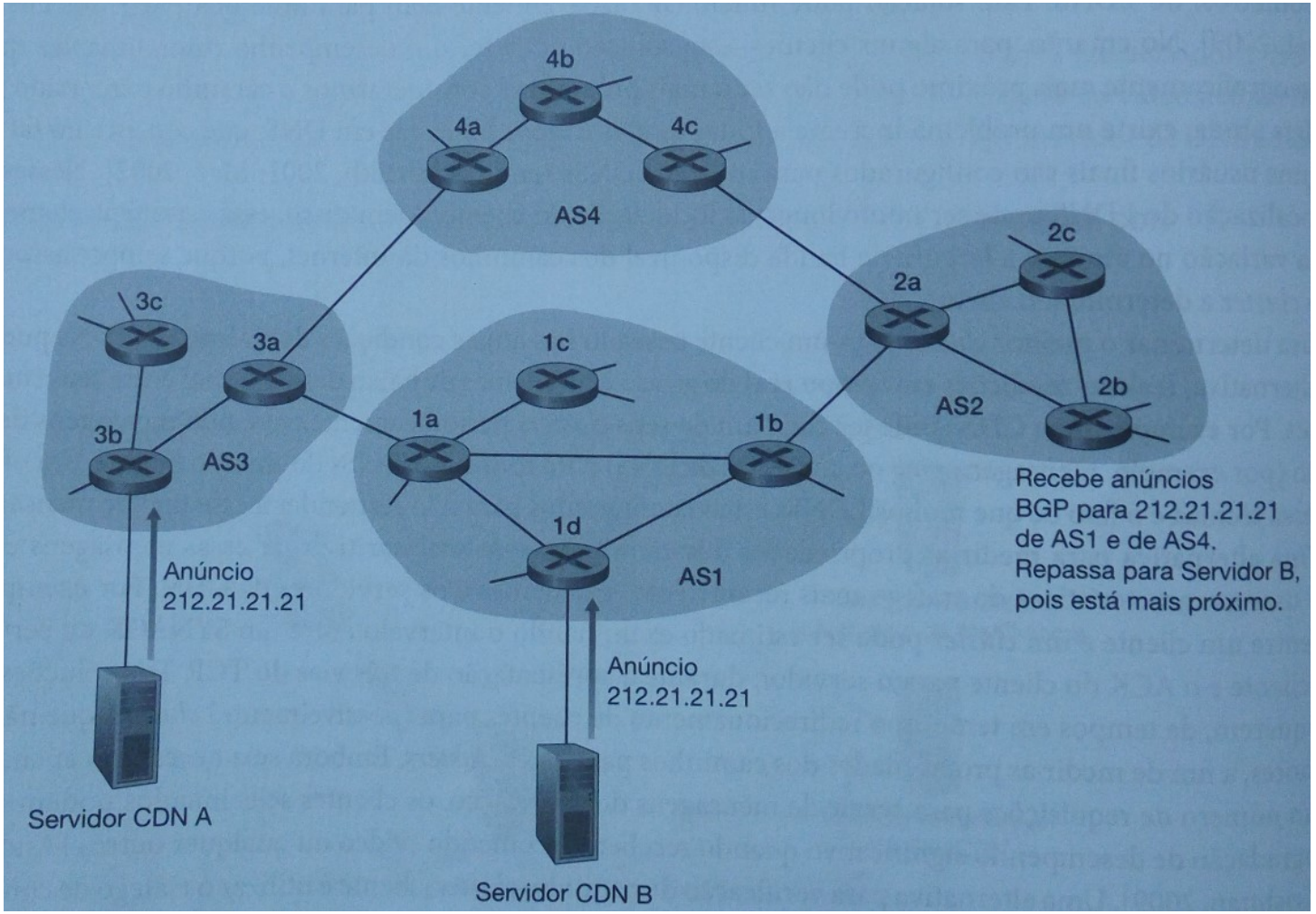
- escolher nó CDN geograficamente mais perto do cliente
  - mapeamento IP – localização (distância em km)
  - localização pode não representar caminho da rede
  - usuários que configuram DNS locais remotas

- escolhe nó CDN dinamicamente com menor atraso (ou menor número de saltos) para o cliente
  - requer que os *clusters* em uma CDN enviem periodicamente mensagens de verificação aos servidores DNS locais do mundo inteiro
  - muitos DNS locais estão configurados para não responder este tipo de mensagem

Estratégias de Seleção de *cluster*

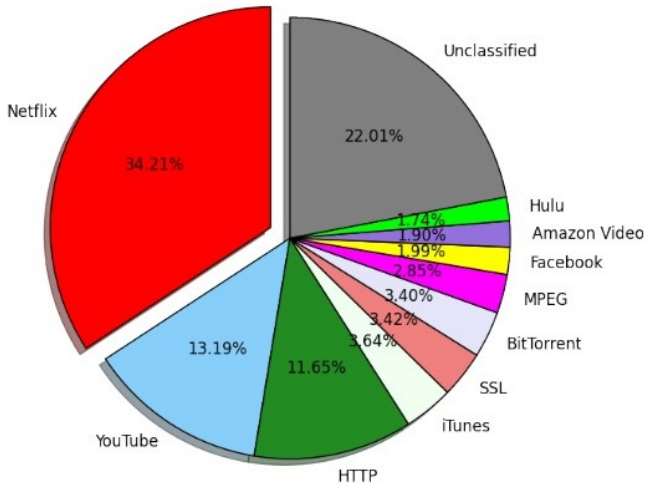
- IP *anycast*: utilizar mesmo IP para qualquer membro do grupo de servidores CDN
  - utiliza propriedade do protocolo BGP de roteamento
  - todos nós CDN configurados com mesmo endereço IP
  - todos nós CDN anunciam a sua existência aos roteadores
  - quando um roteador BGP recebe múltiplos anúncios de rota com mesmo endereço IP, trata-os como diferentes caminhos para o mesmo local físico
  - BGP seleciona melhor rota de acordo com o número de saltos de AS (sistemas autônomos)

Estratégias de Seleção de *cluster*



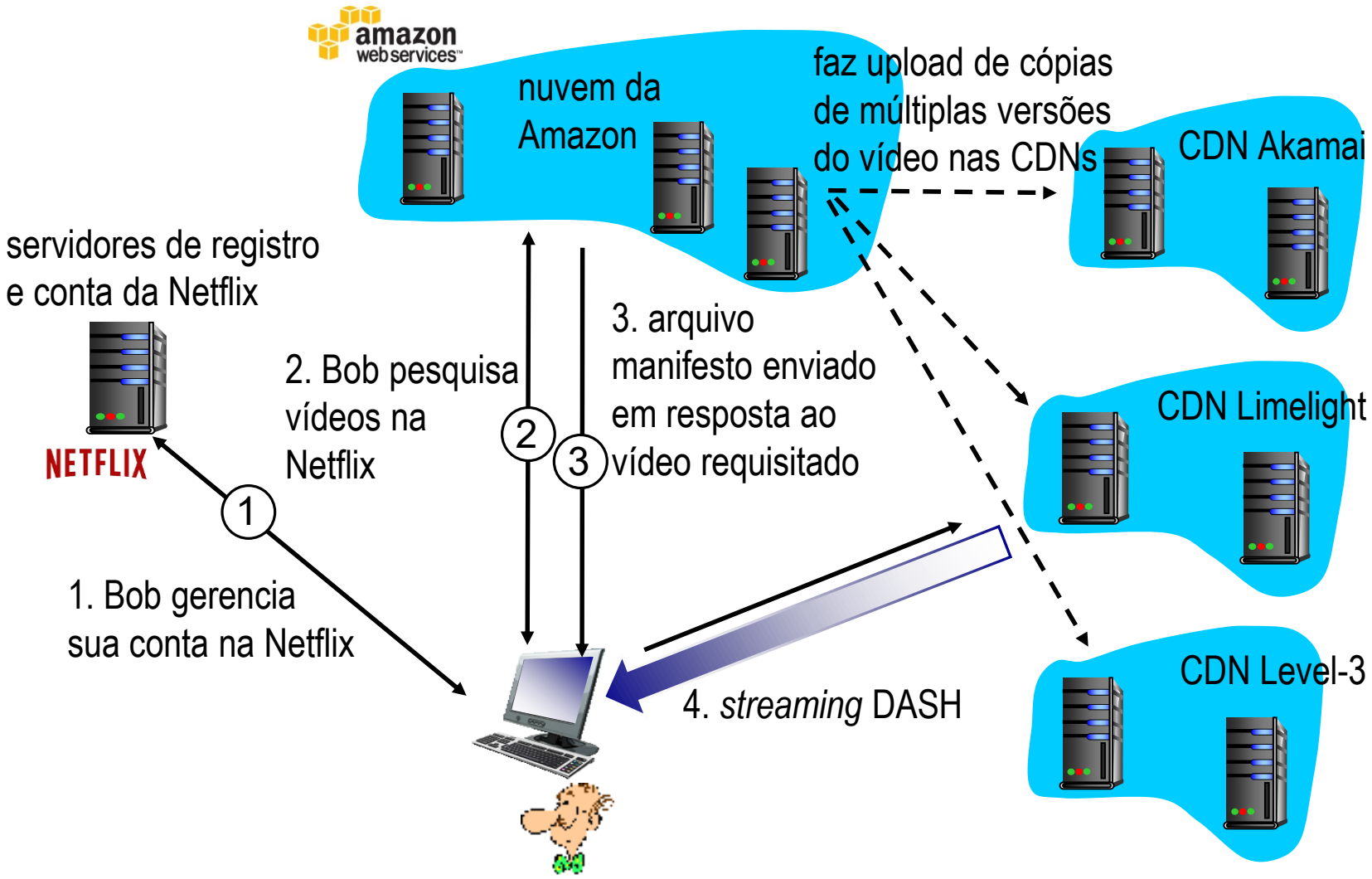
Estudo de Caso: *Netflix*

- 34,2% do tráfego de rede nos EUA em 2014
- possui infraestrutura própria muito pequena e usa serviços de terceiros:
  - servidores de registro e pagamento próprios
  - serviços de nuvem da Amazon (terceirização):
    - Netflix faz upload do *studio master* para a nuvem
    - cria múltiplas versões de filme (codificações diferentes) na nuvem
    - faz upload das versões da nuvem para as CDNs
    - nuvem armazena páginas web da Netflix para navegação do usuário
  - em 2011, *três* CDNs para *stream* do conteúdo da Netflix: *Akamai*, *Limelight*, *Level-3*



2014 Sandvine North America Traffic Report

Estudo de Caso: *Netflix*



Estudo de Caso: *Netflix*

- a partir de 2011, programa *Netflix Open Connect*
  - utiliza propriedade do protocolo BGP de roteamento
- parcerias entre Netflix e provedores de Internet
  - centenas de provedores aderiram ao redor do mundo
- ISPs/IXPs solicitam ao Netflix a instalação de OCAs (*Open Connect Appliances*)
  - hardware que armazena e disponibiliza localmente no ISP/IXP o conteúdo de vídeo
  - Netflix instala gratuitamente o equipamento nos ISPs/IXPs
  - Netflix oferece suporte para configurar sessões BGP nos OCAs para direcionar o tráfego
  - ISPs/IXPs fornecem espaço físico, energia e conectividade

Estudo de Caso: *Netflix*

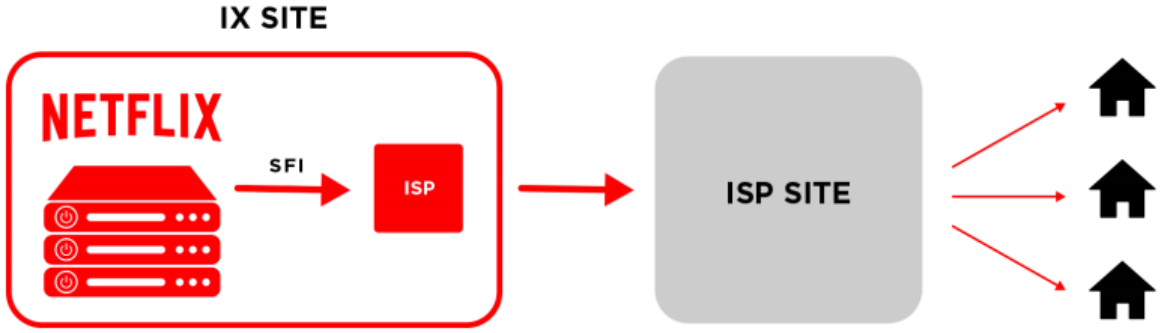
- OCAs (*Open Connect Appliances*)
  - vários SSDs de 1 TB
  - OCAs de 14 TB a 280 TB
  - vazão de 9 Gbps a 36 Gbps
  - CPU de 8 cores

"*Esses dispositivos servem um propósito simples: impedir a Netflix de entupir a internet. A capacidade total do backbone da internet entre um país e outro é de 35 TB por segundo. Nosso pico de tráfego é mais do que isso… Nossa escala é realmente maior do que a capacidade internacional da internet.*"

Ken Florance
vice-presidente de entrega de conteúdo na Netflix

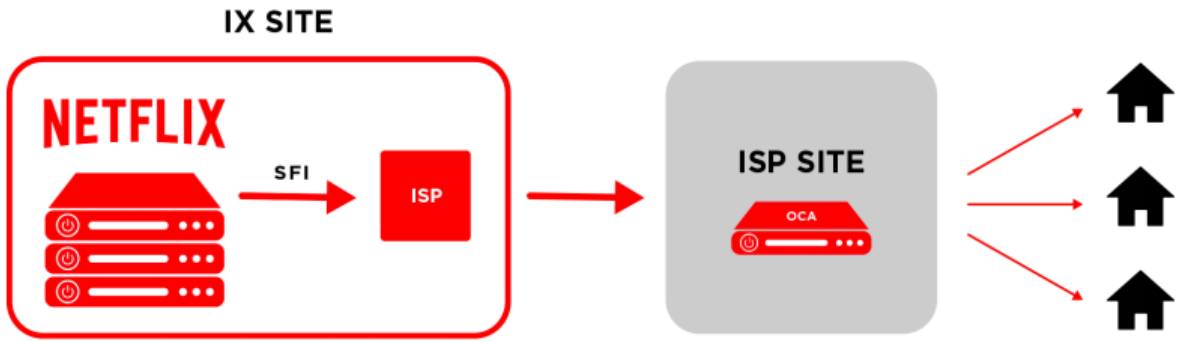
Estudo de Caso: *Netflix*

- Opção 1: OCAs instalados em *Internet Exchange Points* (IXPs)
  - IXPs são estruturas físicas nas quais vários ISPs e CDNs compartilham conteúdo de suas redes (AS)
  - *Settlement-Free Internconnect* (SFI): geralmente os ISPs envolvidas trocam conteúdo gratuitamente entre si (já que todos são beneficiados)

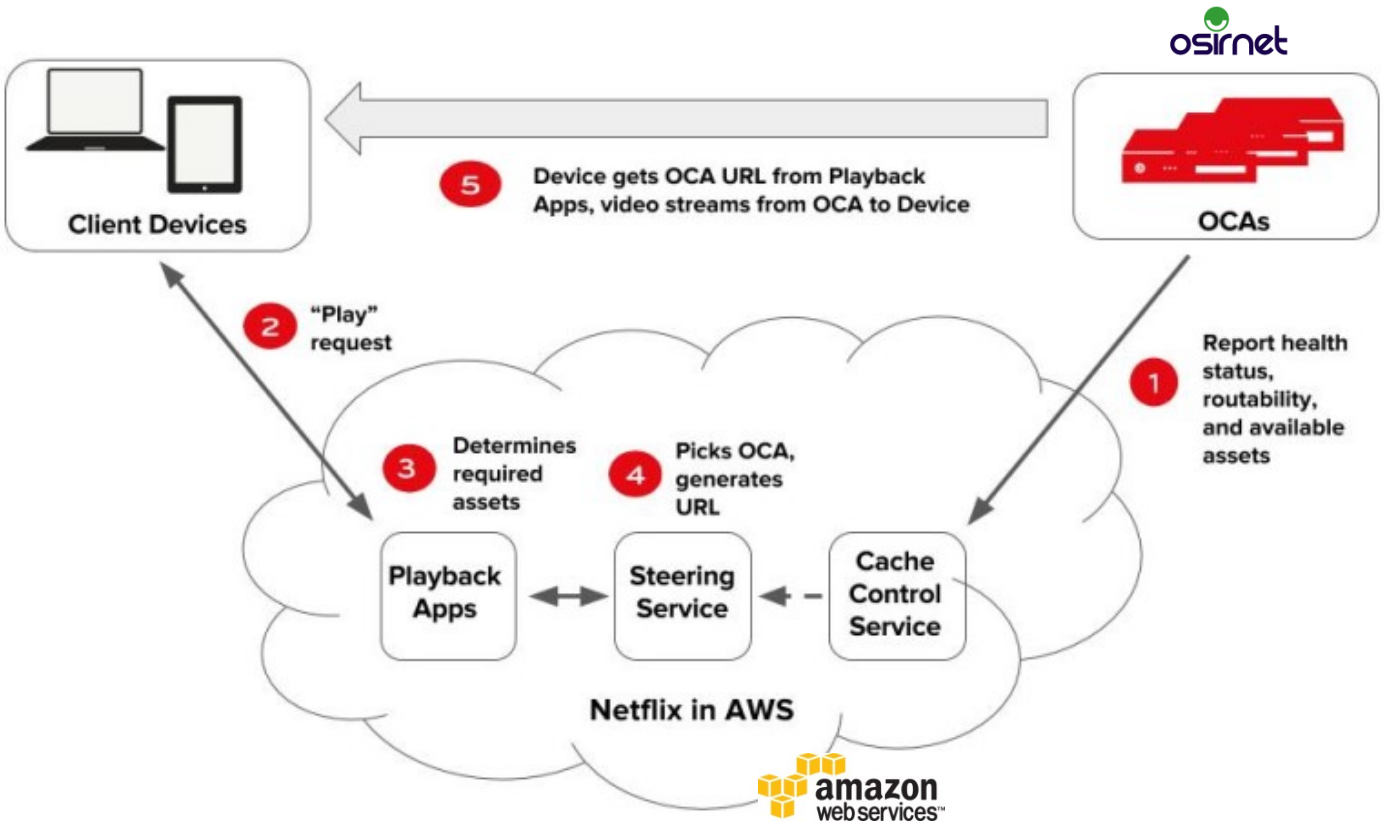


Estudo de Caso: *Netflix*

- Opção 2: OCAs instalados em *Internet Service Providers* (ISPs)
  - mais próximo do cliente que IXPs
  - OCAs com mesma capacidade daqueles instalados em IXPs
  - ISP provê energia, espaço físico e conectividade
  - ISP controla quais dos seus clientes são roteados para seus OCAs
  - Netflix monitora desempenho e uso dos OCAs e aumenta a capacidade à medida que a demanda cresce



Estudo de Caso: *Netflix*



UNIVERSIDADE FEDERAL DE PELOTAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

TEC2/TEC4: REDES MULTIMÍDIA (RMM)

Unidade 10
Vídeo de Fluxo Contínuo

Prof. Guilherme Corrêa
gcorrea@inf.ufpel.edu.br