Adaptive Streaming - State of the Art

Theodoros Karagioules
theodoros.karagioules@huawei.com

HUAWEI Technologies FRC
Mathematical and Algorithmic Sciences Lab
Context Aware Optimization Group
Paris, France

 Télécom ParisTech
Department of Signal and Image Processing
Multimedia Group
Paris, France

Telecom ParisTech, October 19, 2016
Today video delivery accounts for the majority (> 50%) of Internet traffic.

It is expected to rise to 75% of all Internet traffic by 2019.

The video delivery services face extreme pressure for good quality of experience.
Adaptive Streaming over HTTP

Distinction on input
Distinction on methods
Overall

Problem Formulation
HAS Flavors

MPEG-DASH & Client side adaptation

Figure in: A. Vetro and I. Sodagar, Industry and Standards The MPEG-DASH Standard for Multimedia Streaming Over the Internet, pp. 6268, 2011

**Main Problem**

**Common QoE Factors Considered**
- Re-buffering rate
- Average video bit-rate
- Video bit-rate change frequency

**In other words**

Ideally we need to make the optimal bit-rate selection, for each segment, that minimizes the frequency of re-buffering events and that satisfies the constraint set when defining the feasible domain of average video bit-rate and minimum frequency quality change.
### Road-map of the talk

- Throughout this talk we will study several ABR features being used as input for client side adaptation.
- We will also investigate the use of several methods above in the attempts of the authors to produce a principled and robust adaptation approach.
- Last we will explore the novelties and any relevant gaps.

### Algorithms

<table>
<thead>
<tr>
<th>ABR Features</th>
<th>Heuristic</th>
<th>Optimization</th>
<th>Control Theory</th>
<th>MDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer level</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>TCP Throughput</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>TCP Throughput &amp; Buffer level</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Cross-layer information</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Let’s start filing this table!
**Algorithm 1** Conventional

1. Estimate the bandwidth share $\hat{x}[n]$ by equating it to the measured TCP throughput:

   $\hat{x}[n] = \hat{x}[n - 1]$

2. Smooth out $\hat{x}[n]$ to produce filtered version $\hat{y}[n]$ by

   $\hat{y}[n] = S(\{\hat{x}[m] : m \leq n\})$

3. Quantize $\hat{y}[n]$ to discrete video bit-rate $r[n] \in \mathbb{R}$ by;

   $r[n] = Q(\hat{y}[n]; \ldots)$

4. Schedule the next download request depending on the buffer fullness:

   $\hat{T}[n] = \begin{cases} 0, & B[n - 1] < B_{\text{max}} \\ \tau, & \text{otherwise} \end{cases}$

---

**FESTIVE**

**Goals**
- Fairness: Allocate throughput fairly to multiple competing players sharing a bottleneck link
- Efficiency: A group of players must choose the highest feasible set of bit-rates
- Stability: A player should avoid needless bit-rate switches

**Steps**
- Bandwidth estimator: harmonic mean
- Stateful and delayed bit-rate update: increase the bit-rate at level \( k \) only after \( k \) chunks, but decrease it immediately if the current bit-rate is \( p = 0, 85 \times \) of the estimated bandwidth. Convergence to a fair bit-rate allocation.
- Randomized scheduler:
  \[
  t_{\text{start}}^{i+1} = \begin{cases} 
  t_{\text{end}}^i, & \text{buffer}_i < \text{randbuf}_i \\
  t_{\text{end}}^i + \text{buffer}_i - \text{randbuf}_i, & \text{otherwise}
  \end{cases}
  \]
  This ensures that the request time of a player is independent of its start time.

**Model**

**Design properties**
- Insufficiency: \( \frac{|\sum_p \tau_{p,t} - x - W|}{W} \)
- Unfairness: \( \sqrt{1 - \text{JainFair}} \)
- Instability: weighted sum of all switches over weighted sum of bit-rates in last 20 s

---

**FESTIVE**

**Results**

- **Baseline** vs. **FESTIVE**
  - Stability: FESTIVE outperforms existing solutions (Microsoft Smooth-Streaming) by $\geq 50\%$.
  - Efficiency: FESTIVE outperforms existing solutions by $\geq 10\%$.
  - Fairness: FESTIVE outperforms existing solutions by $\geq 40\%$.

**Comparison**

- **FESTIVE** outperforms existing solutions (Microsoft Smooth-Streaming) in terms of fairness by $\geq 40\%$, stability by $\geq 50\%$, and efficiency by $\geq 10\%$.

**Contribution**

- **FESTIVE** proposes a robust and fair adaptation scheme against various number of players, bandwidth variability, and different available bit-rate set.

---

Algorithm 2 PANDA

1. Estimate the bandwidth share $\hat{x}[n]$ by:

$$\frac{\hat{x}[n] - \hat{x}[n-1]}{T[n-1]} = \kappa(\omega - \max(0, \hat{x}[n-1] - \hat{x}[n-1]))$$

2. Smooth out $\hat{x}[n]$ to produce filtered version $\hat{y}[n]$ by

$$\hat{y}[n] = S(\{\hat{x}[m] : m \leq n\})$$

3. Quantize $\hat{y}[n]$ to discrete video bit-rate $r[n] \in \mathbb{R}$ by:

$$r[n] = Q(\hat{y}[n]; \ldots)$$

4. Schedule the next download request via:

$$\hat{T}[n] = \frac{r[n]}{\hat{y}[n]} + \beta(B[n-1] - B_{min})$$

where $\kappa$ is the probing convergence, $\omega$ is the additive increase rate and $\beta$ is the convergence rate.

Four step model

- Design Goals:
  - Avoid buffer under-run
  - High quality smoothness
  - High average quality
  - Fairness

- To address video oscillation:
  - Introduction of a probing mechanism: Increment $\hat{x}[n]$ by $\kappa \omega$ per unit time until $\hat{x} < \hat{x}$
  - Probing mechanism shares similarity to TCP’s congestion control

- New scheduling policy aims at driving $B[n]$ to a minimum reference level $B_{min} > 0$

PANDA, compared to the conventional approach and FESTIVE, has the best stability-responsiveness trade-off by 75% and bandwidth utilization.

**Contributions**

- Identification of Bandwidth Cliff Effect as the root cause of the bit-rate oscillation phenomenon
- General Probe and Adapt principle to address those problems
- Four-step model

Adaptive Streaming over HTTP
Distinction on input
Distinction on methods
Overall

Throughput based adaptation
Buffer-based adaptation
Cross Layer adaptation

**DBuffer**

### Control Logic
- **Buffer Derivative estimation:**
  Estimate the ideal interval for the controller to actuate:
  \[ \Delta_{safe} = \left| \frac{1}{R_t} \right| = \left| \frac{t_{act} - t_{last}}{B(t_{act}) - B(t_{last})} \right| \]
- **States**
  - Initial: Calculate \( B_{min} \geq 25s \), \( \Delta_{safe} = 10s \)
  - Actuator: Initially \( \Delta_{temp} = \Delta_{safe} \) if there is a buffer fluctuation then \( \Delta_{temp} < \Delta_{safe} \)
  - Idle: Stay while \( t_{act} - t_{last} < \Delta_{safe} \)

### State Machine

### Buffer management

---

**DBuffer**

**Results**

![Graph showing probability of DBuffer, PANDA, and Conventional against stall time]

- **HAS algorithm modeled as a state machine**
- **Decreased probability of buffer under flows:**

**Contributions**

- HAS algorithm modeled as a state machine
- Decreased probability of buffer under flows:

**Limitations**

- Heuristically chosen values

---

Buffer-based Approach

- Start-up phase
  - Usage of capacity estimation
- Steady state
  - Select video rate solely as a function of the current buffer level

Design

- Buffer-based algorithm → picks video rate as a $f(B(t))$
- Design space → $[0, B_{\text{max}}]$ on the buffer-axis and $[R_{\text{min}}, R_{\text{max}}]$ on the rate-axis
- Any curve $f(B)$ within the feasible region defines a rate map

Buffer maps

Buffer maps can visualize the relationship between video rate and buffer occupancy, showing the feasible region (where the video rate can be set) and the boundary of the safe area (indicating where the buffer is safe to operate).

Improvements

- Handling VBR (BBA-1)
- Start-up phase (BBA-2)
- Rate switches and outages (BBA-Others)

---

RESULTS

Contributions

- Strong suggestion by the authors that capacity estimation is unnecessary
- BBA-1 provides a 20-28% improvement in re-buffering rate, compared to the control algorithm and a trivial case \( R_{\text{min}} \)
- Limitation: Control algorithm is a previous version of BBA

In the original OSI networking model, strict boundaries between layers are enforced, where data are kept strictly within a given layer. Crosslayer optimization removes such strict boundaries to allow communication between layers by permitting one layer to access the data of another layer to exchange information and enable
Channel Prediction

**MDP formulation**

- Decision epochs and time horizon
  - \( T = \{1, \ldots, t, \ldots\} \) Finite of infinite horizon of fixed discrete decision time slots \( t \)
- States at time \( t \)
  - Channel: \( s_t \in S = \{1, \ldots, N\} \) Set at Layer 1
  - Buffer: \( b_t \in B = \{0, \ldots, B_{\text{max}}\} \) Set at Layer 7
- Actions: \( \alpha_t \in A = \{1, \ldots, L\} \) To download segment video segment with quality \( l \)

**Buffer evolution**

Let \( q \) be the playback rate, \( d_t \) the downloaded segments and \( y = b_t - q + d_{t+1} \)

\[
b_{t+1} = \begin{cases} 
0, & \text{if } y < 0 \\
y, & \text{if } 0 \leq y \leq B_{\text{max}} \\
B_{\text{max}}, & \text{if } y > B_{\text{max}} 
\end{cases}
\]

**Optimality**

\[
U^\pi = \lim_{T \to \infty} \frac{1}{T} E(\sum_{t=1}^{T} u_t), \text{ where } \pi \text{ is a decision policy } (S \times B \mapsto A)
\]

**Channel prediction**

### Transition Probabilities

Let $p_{ij} = P(s_{s+1} = j | s_t = i)$ and $P_d(i | s, \alpha) = P(d_t = i | s_t = s, \alpha_{t-1} = \alpha)$. Then:

$$p(s', b' | s, b, \alpha) = P(s_{t+1} = s', b_{t+1} = b' | s_t = s, b_t = b, \alpha_t = \alpha)$$

$$= p_{ss'} P(b_{t+1} = b' | s_{t+1} = s', b_t = b, \alpha_t = \alpha)$$

$$= \begin{cases} 
  p_{ss'} P(b' + q - b | s', \alpha), & \text{if } 0 < b' < B_{max} \\
  p_{ss'} \sum_{i=0}^{q-b} P_d(i | s', \alpha) & \text{if } b' = 0 \\
  p_{ss'} \sum_{i=B_{max}+q-b+1}^{D_{max}} P_d(i | s', \alpha) & \text{if } b' = B_{max}
\end{cases}$$

### Rewards

$$r(s, b, \alpha) = \mathbb{E}(u_{t+1} | s_t = s, b_t = b, \alpha_t = \alpha)$$

$$= \sum_{s'} p_{ss'} \left\{ \begin{array}{l}
  \sum_{i=q-b}^{B+q-b} P_d(i | s', \alpha) i R(\alpha) + \sum_{i=0}^{q-b-1} P_d(i | s', \alpha) [i R(\alpha) - (q - b - i) W_u] \\
  + \sum_{i=B_{max}+q-b+1}^{D_{max}} P_d(i | s', \alpha) (B_{max} + q - b) R(\alpha)
\end{array} \right\}$$

where $u$ is the utility function, reward function $R(\alpha)$ and penalty factor $W_u$ for missing one segment.

---

Algorithm 3 Channel State Information

1: Initialization $\pi(s, b) \leftarrow 1$, $u^0(s, b) \leftarrow 0$, $\forall s, b \in S \times B$, $STOP \leftarrow 0$
2: while $STOP = 0$ do
3: \hspace{1em} for each $(s, b) \in S \times B$ do
4: \hspace{2em} $u^{n+1}(s, b) \leftarrow \max_{\alpha \in A} (r(s, b, \alpha) + \sum_{(s', b') \in S \times B} p(s', b'| s, b, \alpha) u^n(s, b))$
5: \hspace{2em} end for
6: \hspace{1em} $\Delta u^{n+1}(s, b) \leftarrow u^{n+1}(s, b) - u^n(s, b)$
7: \hspace{1em} if $\max_{(s, b) \in S \times B} \Delta u^n(s, b) - \min_{(s, b) \in S \times B} \Delta u^n(s, b) < \epsilon$ then
8: \hspace{2em} $STOP \leftarrow 1$
9: \hspace{1em} end if
10: \hspace{1em} $n \leftarrow n + 1$
11: end while
12: $\pi(s, b) \leftarrow \arg \max_{\alpha \in A} r(s, b, \alpha) + \sum_{(s', b') \in S \times B} p(s', b'| s, b, \alpha) u^n(s, b)$

Contributions-Limitations

- Simplicity of formulation
- Channel state can directly be mapped to rates
- Heuristic close to optimal
- Does not support handover, load and congestion are ignored

Online Learning

Optimization Problem

\[
\max_{\alpha_t} \left\{ \sum_{t=\tau}^{\infty} \gamma^{t-\tau} [r^q(\alpha_t, D_t, q_{t-1}) - r^b(B_t, f(\alpha_t, h_t))] \right\}
\]

s.t. \( B_t - f(\alpha_t, h_t) \geq 0, \forall t \)

where: \( r^q(\alpha_t, D_t, q_{t-1}) = g(\alpha_t, D_t) - \beta |g(\alpha_t, D_t - q_{t-1})| \)
and: \( r^b(B_t, f(\alpha_t, h_t)) = \rho (\max[0, f(\alpha_t, h_t) - B_t]) + \sigma (\max[B_M - B_{t+1}], 0)^2 \).

Here \( \gamma \) is a discount factor on future rewards, \( \rho \) is a parameter that penalizes re-buffering events, \( \sigma \) is an incentive for a safe buffer level, \( \tau \) is the targeted time and \( \alpha \) the decision made.

Online Learning

Definition of state \( s_t : \{q_{t-1}, h_{t-1}, D_t, B_t\} \) where \( q_{t-1} \) is the quality of the previously downloaded segment, \( h_{t-1} \) is the estimated bandwidth experienced during the previous download, \( D_t \) is the complexity of the segment, \( B_t \) is the buffer status. The key concept is to use an intermediate state \( \tilde{s}_t \) to divide the transition from \( s_t \) to \( s_{t+1} \) in two steps (PDS), for faster learning.

Optimal State Value Function for PDS

\[
\hat{V}^*(\tilde{s}_t) = \sum_{s_{t+1}} p_{u}(s_{t+1}|\tilde{s}_t) [-r^b(s_{t+1}, \tilde{s}_t) + \gamma V^*(s_{t+1})]
\]

where: \( V^*(s_t) = \max_{\alpha_t} \{ r^q(\alpha_t, s_t) + \sum_{\tilde{s}_t} p_k(\tilde{s}_t|s_t, \alpha_t) \hat{V}^*(\tilde{s}_t) \} \)

Algorithm 4 Online Algorithm

1: Initialize $\tilde{V} = \frac{1}{\gamma}$
2: Initialize $s_t$ to starting state
3: repeat
4: for all $\alpha_j \in \mathbb{A}$ do
5:   $r^j_q \leftarrow QUALITYREWARD(s_t, \alpha_j)$
6:   $s_j \leftarrow FINDPDS(s_t, \alpha_j)$
7:   $U_t(\alpha_j) \leftarrow r^j_q + \tilde{V}_{s_j}$
8: end for
9: $\alpha_t \leftarrow SOFTMAX(U_t())$
10: $\tilde{s}_t \leftarrow FINDPDS(s_t, \alpha_j)$
11: Observe $s_{t+1}$
12: $vUPDATE(\tilde{V}, \tilde{s}_t, s_t + 1)$
13: $s_t \leftarrow s_t + 1$
14: until end of simulation

Contributions

- Learning faster than heuristic and no significant pre-training needed
- Independent of video and channel model

**Problem Formulation**

Suppose we download $N$ chunks (encoded in $M$ bit-rate levels) of duration $p$ in $k$ slots. $T_k = t_{k+1} - t_k$ is the duration of every slot and $T_{end}$ is the time that the end of the playback of the $N^{th}$ chunk. Two performance metrics are considered:

1. **Time averaging expected playback utility:**
   $$\tilde{u}_N \triangleq \frac{\mathbb{E}\{\sum_{k=1}^{N} \sum_{m=1}^{M} a_m(t_k) u_m\}}{\mathbb{E}\{T_{end}\}}$$

2. **Fraction of time spent not re-buffering:**
   $$\bar{s}_N \triangleq \frac{Np}{\mathbb{E}\{T_{end}\}} = \frac{\mathbb{E}\{\sum_{k=1}^{N} \sum_{m=1}^{M} a_m(t_k)p\}}{\mathbb{E}\{T_{end}\}}$$

**Objective:** maximize $\tilde{u}_N + \gamma \bar{s}_N$

$\alpha_m(t_k) = 1$ as an indicator of a chunk downloaded in slot $k$ with bit-rate $m$ and $\gamma$ is an input weight parameter for prioritizing playback utility with playback smoothness.

**Relaxation**

- Although DP is a good candidate for this objective it can lead to a very large state space.
- Simplify the problem in the limiting regime: $N \to \infty$
- **Lemma:** In the large $N$ regime, there exists a buffer-state-independent stationary policy that makes i.i.d control decisions in every slot and satisfies the rate stability constraint while achieving time-average utility no smaller than $u^* + \gamma s^*$ (optimal)
The technique is for stabilizing a queuing network while also minimizing the time average of a penalty function.

**DRIFT-PLUS-PENALTY**

In slot $k$ the buffer is kept stable by minimizing $\mathbb{E}\{\frac{Q(t_{k+1})^2 - Q(t_k)^2}{2} \mid Q(t_k)\}$

**Method**

- This method greedily minimizes the ratio of drift plus penalty to frame length over each slot.
- Let the buffer level be denote as: $Q(t_{k+1}) = \max[Q(t_k) - \frac{T_k}{p}, 0] + \sum_{m=1}^{M} a_m(t_k)$
- To achieve buffer stability we need to minimize $Q(t_k) \sum_{m=1}^{M} \alpha_m - \frac{T_k}{p}$

---

**BOLA continued**

**Online Control Algorithm**

Maximize

\[
\frac{\sum_{m=1}^{M} a_m(t_k)(V u_m + V \gamma p - Q(t_k))}{\sum_{m=1}^{M} a_m(t_k)S_m}
\]

subject to \(\sum_{m=1}^{M} a_m(t_k) \leq 1, a_m(t_k) \in \{0, 1\}\)

- \(S_m\) is the size of a segment in quality \(m\) and \(V\) a buffer size trade-off control parameter

**Results**

- Balance between average bit-rate and duration of re-buffering events
- Utility achieved by Lyapunov optimization within an additive factor of the optimal (First theoretical guarantee)

---

### To Sum Up

#### Algorithms

<table>
<thead>
<tr>
<th>ABR Features/Methods</th>
<th>Heuristic</th>
<th>Optimization</th>
<th>Control</th>
<th>MDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer level</td>
<td>Netflix</td>
<td>BOLA</td>
<td></td>
<td>O.Learning</td>
</tr>
<tr>
<td>TCP Throughput</td>
<td>PANDA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP Throughput &amp; Buffer level</td>
<td>FESTIVE</td>
<td>DBuffer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Metrics Considered

<table>
<thead>
<tr>
<th></th>
<th>SSIM</th>
<th>Bit-Rate</th>
<th>Rebuf.</th>
<th>Instability</th>
<th>Insufficiency</th>
<th>Unfairness</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panda</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Festive</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netflix</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dbuffer</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOLA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>
WHAT HAVE WE GOT AND WHAT IS STILL MISSING?

**Progress**

- Focus on user fairness
- Balance different QoE objectives (e.g., start-up delay vs. re-buffering)
- Formal specification of HAS
  - It used to be pure heuristics

**Gaps**

- Buffer based algorithm for multiple users
- Unified performance, quality and fairness metrics
- Lack of consensus in regard to:
  - Rate estimates vs. buffer occupancy under universal performance metric
  - Performance of specific classes of approaches under diverse operating regimes (e.g., high throughput variability, mobile environments)
- There doesn’t exist a well established input classification, nor is there a systematic comparison between them in a unified framework
- A systematic comparison on methods along with the limitations of each one
What is next?

Next steps

- Implement most relevant algorithms in same platform for comparison (Simulation + real experiments)
- Use MDPs to formally design cross-layer HAS algorithms
- Use Lyapunov optimization to model and study the stability of the designed HAS algorithms, in comparison to policies from literature
- Systematic study of cross-layer parameters

Discussion

- The formulation of optimal HAS algorithms, will allow to characterize video quality versus stability as a fundamental trade-off in HAS policy design. Numerical study of this trade-off?
- Can you suggest an effect of various cross-layer parameters on the previously-designed optimal HAS algorithms? Or similar to your research? In particular, parameters related to location and map information as well as radio network measures such as received signal strength, cell identifier and handover status.
- Do you have another stability method to propose?
- Any questions on your side?