Optimized Strategies for Real-Time Multimedia Communications from Mobile Devices

Enrico Masala

Dept. of Control and Computer Engineering, Politecnico di Torino, Torino, Italy

(Part of this work has been done jointly with Aalto University, Finland)

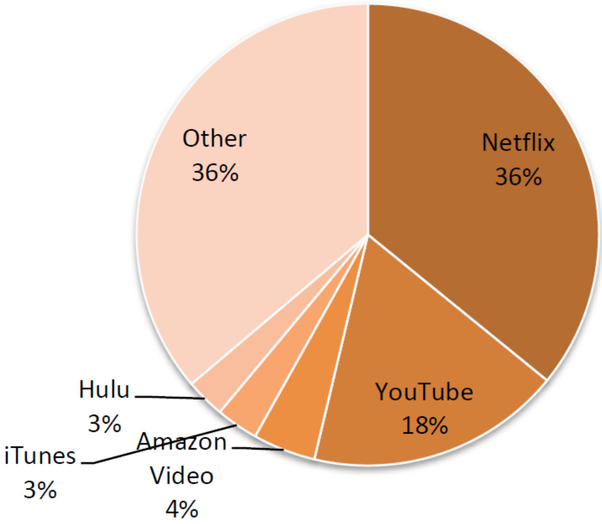
Dip. di Informatica, University of Verona, May 23, 2017

Outline

- Introduction
- Background technologies: SVC, DASH
- Live mobile streaming optimization
  - Problem formulation
  - Proposed solution and results
- Outlook on the mobile live video trend
  - Example: Periscope
- Conclusions

Multimedia Communications

- Dramatically increased in recent years
  - Netflix video accounts for more than 1/3 of traffic in North America at peak hours [1] *(Downstream peak period applications, North America, Fixed Access, Jun 2016)*



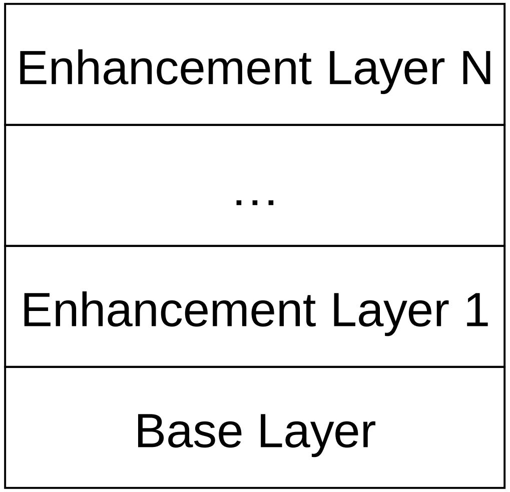
- Anybody can produce content
  - Using, e.g., a mobile device
  - Upload it on streaming platforms (Youtube, etc.)
  - Can even be done live!

Background

- Multimedia encoding and transmission: can be done in many different ways
- We focus on:
  - Scalable coding (different resolution, quality, frame rate)
  - Streaming using HTTP (through the DASH standard)

Scalable Coding

An embedded way to represent a compressed bitstream so that players can extract different versions (layers) of the content using only some portions of the bitstream



Coding example with spatial scalability:

Original image

Base layer

Upsampled base layer

Layered Structure and Advantages

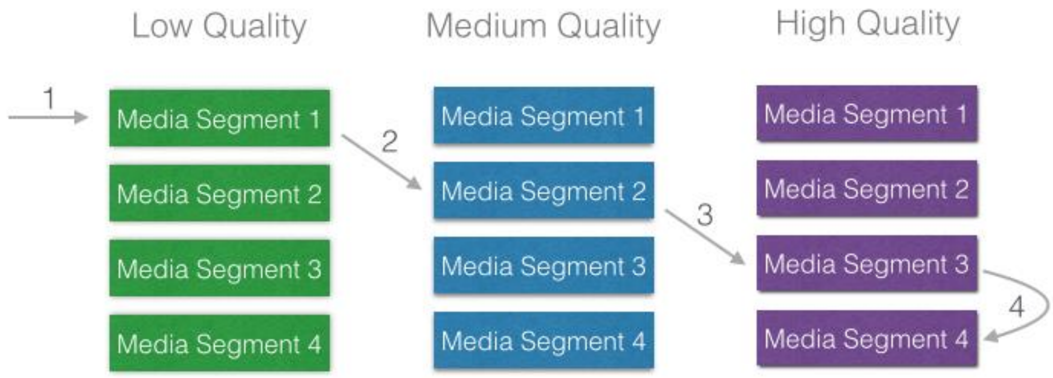
- For efficiency reasons, every layer only adds “refining” information (to improve quality) to the information already present in another layer
  - Other layers are needed to fully decode one layer
  - Only the base layer (the lower one) can be independently decoded
- Advantages:
  - No need to keep more versions of the same content encoded at different qualities: **space savings**
  - No need to process data which are useless to extract a reduced quality version: **complexity savings**

Dynamic Adaptive Streaming over HTTP

- If the network is good, resources can be downloaded in any way, e.g., using HTTP
- How to handle bandwidth variation? → Adaptation
- How to adapt on HTTP?
  - TCP cannot be explicitly controlled

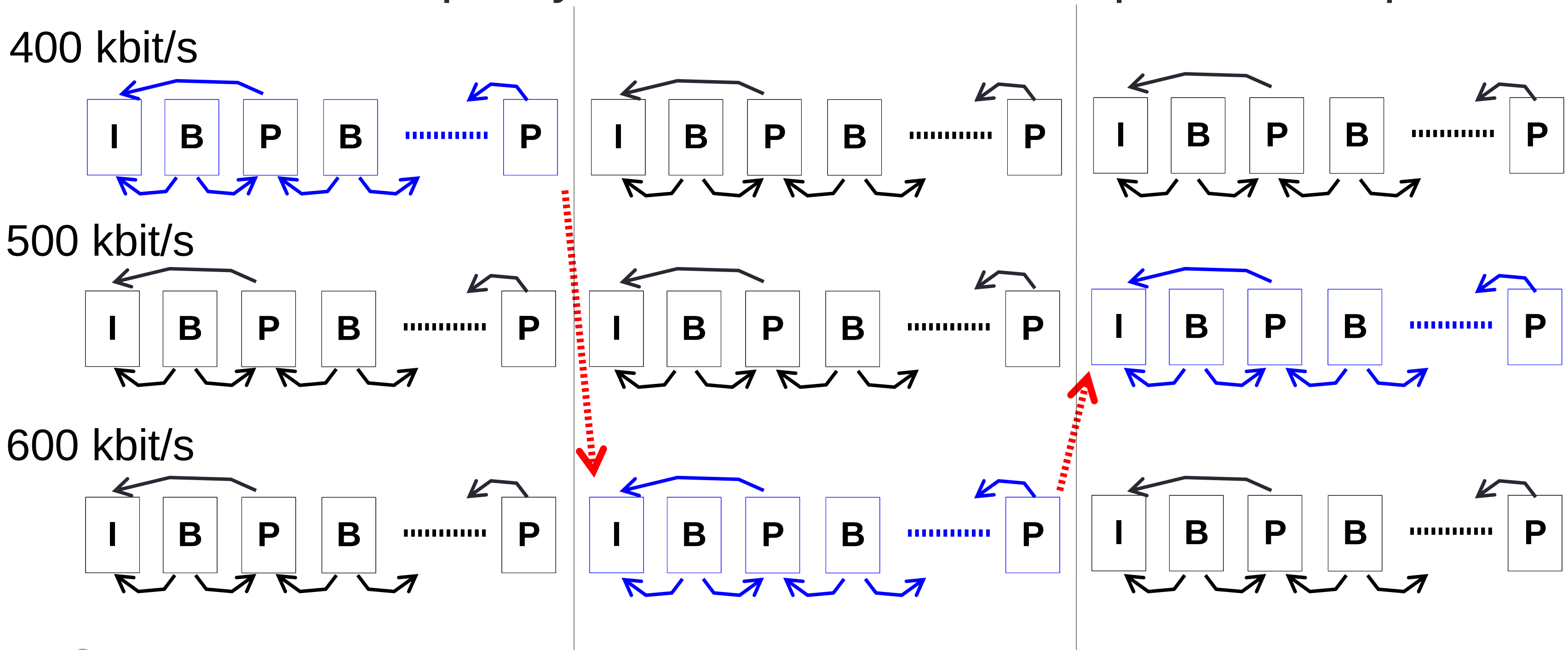
Dynamic Adaptive Streaming over HTTP

- Content is split into “chunk”, temporally aligned, with different characteristics (e.g., bitrate)
- The client requests chunks as **independent** HTTP resources
- The client request different resources over time in order to adapt to the time-varying network conditions
  - The **client drives** the adaptation process



Dynamic Adaptive Streaming over HTTP

- Can switch quality/rate/resolution etc. at predefined points



- Optimized strategies difficult to design (they are not included in the standard)
- Scalable video is supported (more or less layers requested)

Multimedia Communication Optimization

- General problem statement

$$\min_{\{\Pi_i\}} E[D(\Pi_i)] \quad \text{subject to} \quad R(\Pi_i) < R_{max}$$

To be solved for each media segment (e.g., interval between two I frames)

• $E[D(\Pi_i)]$ = **expected distortion** for a given coding and transmission policy i

• **Policies**: set $\{\Pi_i\}$ (for the units – e.g., frames – in the media segment)

• Policy i = (e.g.) an assignment to a certain **coding parameters and channel transmission policies** for each unit (e.g., a frame) in the segment

• $R(\Pi_i)$ = **rate** caused by using the coding and protection level corresponding to the policy Π_i.

(notation from [2])

Difficulties and Possible solutions

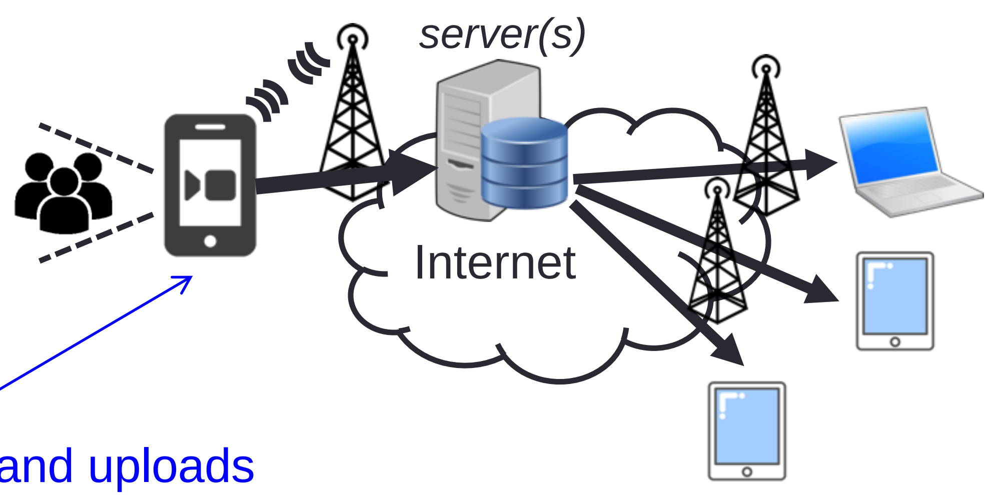
- Estimating the distortion is difficult due to
  - Dependency between coding units (linear additive approximations)
  - Uncertainty in estimating the channel conditions
- The problem grows exponentially in complexity
  - Lagrangian-based solutions (if it is possible to express the terms as sums)
  - Heuristic algorithms

Specific Cases

- The problem needs to be tailored to the specific cases
- Good understanding of the context is essential to adapt and simplify the analytical formulations
- For this presentation, we focus on:
  - upload from mobile devices
  - using stateless HTTP servers
  - that serve multiple clients

Live Mobile Streaming to Many Users

- Constraints
  - Stateless HTTP server (for simplicity and low cost)
  - Support dynamic adaptive streaming, optimized for many users
  - Save, in any case, the maximum quality video and eventually send everything to the server



Captures, encodes, and uploads live video

Results from [3]

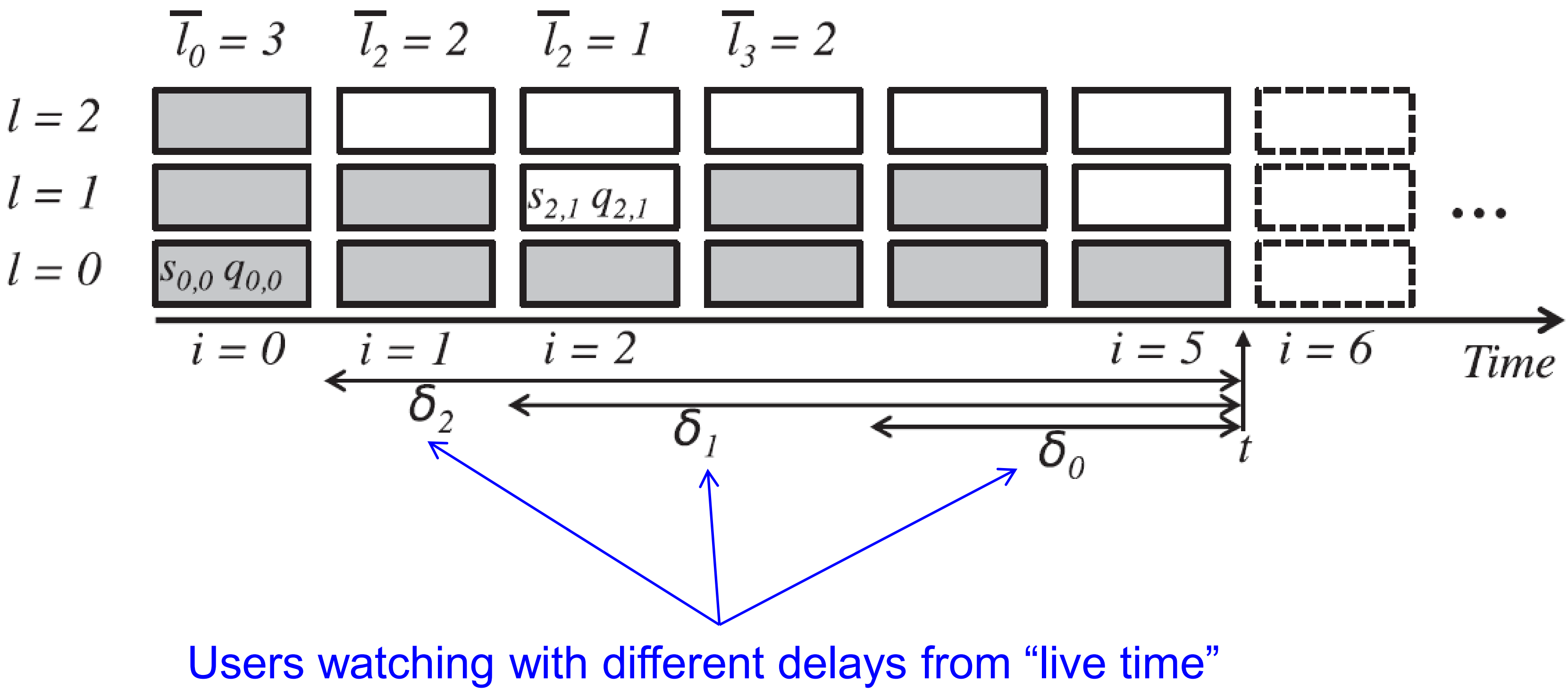
Proposed Solution

- Use scalable video encoding
- Upload scheduling problem: **optimize the order** of chunk uploading, depending on available mobile upload bandwidth, to satisfy the largest number of users watching the video according to their “wishes”
- Use DASH
  - low-cost stateless HTTP server
  - each **user** drives the adaptation, it can choose a **different** delay/quality **tradeoff**
    - The longer the delay from the live point, the better the quality

(from [Siekkinen,Masala17])

Example of Situation

- Chunks in grey have already been uploaded to the server



Analytical Analysis

- Total number of combinations: unfeasible unless number of chunks is very low
- Simple formulation: how can chunks be put into segments?
- Example for constant size chunks: allocate V elements in t bins (multinomial coefficient)

$$\left(\!\!\binom{V}{t}\!\!\right) = \binom{V+t-1}{t} = \frac{(V+t-1)!}{t!(V-1)!}$$

- Optimize for the quality of all clients, while considering the bandwidth constraints

Problem Formulation

- Quality / Distortion of a client, watching the video with a given delay (δ_n) and upload policy (π_k):

$$Q_n^{\pi_k} = \sum_{(i,l)} \left(x_{i,l}^{\pi_k}(i - 1 + \delta_n) q_{i,l} \right)$$

- Optimization

$$\underset{\pi_k \in \Pi}{\text{maximize}} \; f(Q_0^{\pi_k}, \ldots, Q_{N-1}^{\pi_k})$$

$$\text{subject to: } S(\pi_k) \leq \sum_{i=1}^{d+\max\delta_1,\ldots,\delta_N} B_i$$

- Possible combination of client qualities, e.g., average

$$f(Q_0^{\pi_k}, \ldots, Q_{N-1}^{\pi_k}) = \frac{1}{N} \sum_{n=0}^{N-1} Q_n^{\pi_k}$$

General Intuition of the Problem

- It is better a more recent chunk of a lower layer that benefits all clients

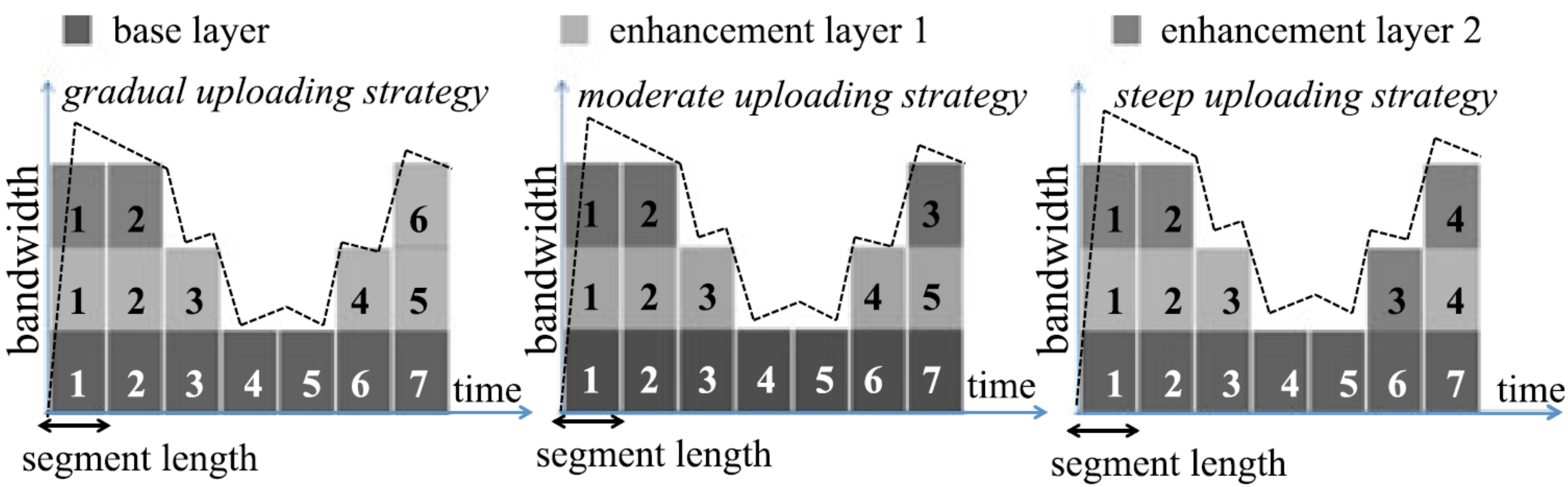
or

- It is better a less recent chunk of a higher layer that improves the quality only for some clients?

- If time allows, 2nd would be better, but channel is uncertain, there might be the risk that important layers are not transmitted for clients with low delay

Possible Strategies

- Example of “naïve” fixed strategies: gradual, moderate, steep
  - Note the different chunks uploaded when bandwidth is available



Other Proposed Strategies

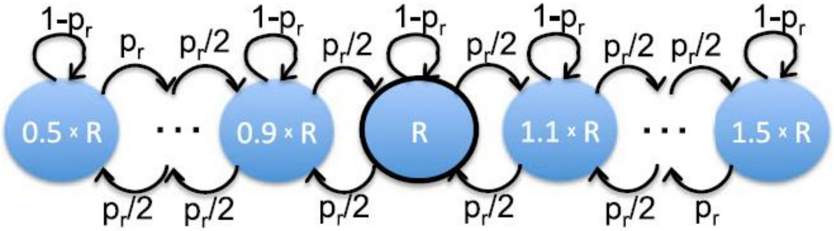
- Greedy approach: send the chunk that has the best quality(increase)/size ratio

$$h_i \leftarrow \frac{q_{i,\bar{l}_i+1} - q_{i,\bar{l}_i}}{s_{i,\bar{l}_i+1}} \qquad \mathcal{O}((t+V)\log t)$$

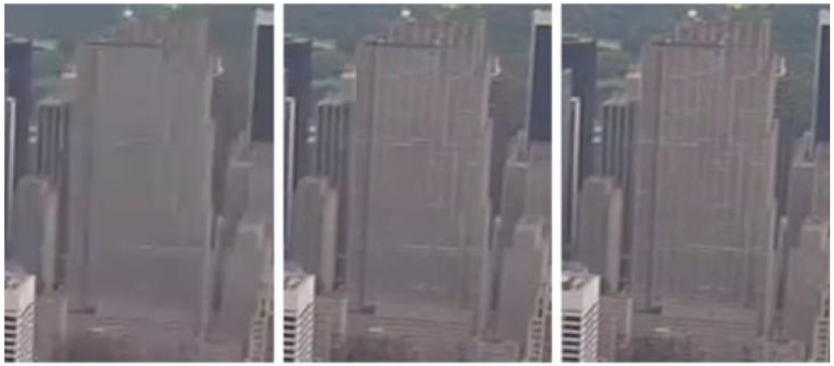
- Dynamic programming for 0/1 knapsack problem
  - Chunks that can be fitted into the available bandwidth
  - One chunk can be used only once (0/1 knapsack)
  - Local knapsack or global knapsack (upper bound, if the channel were known)

Simulation Setup

- Channel: Markov chain of different rates



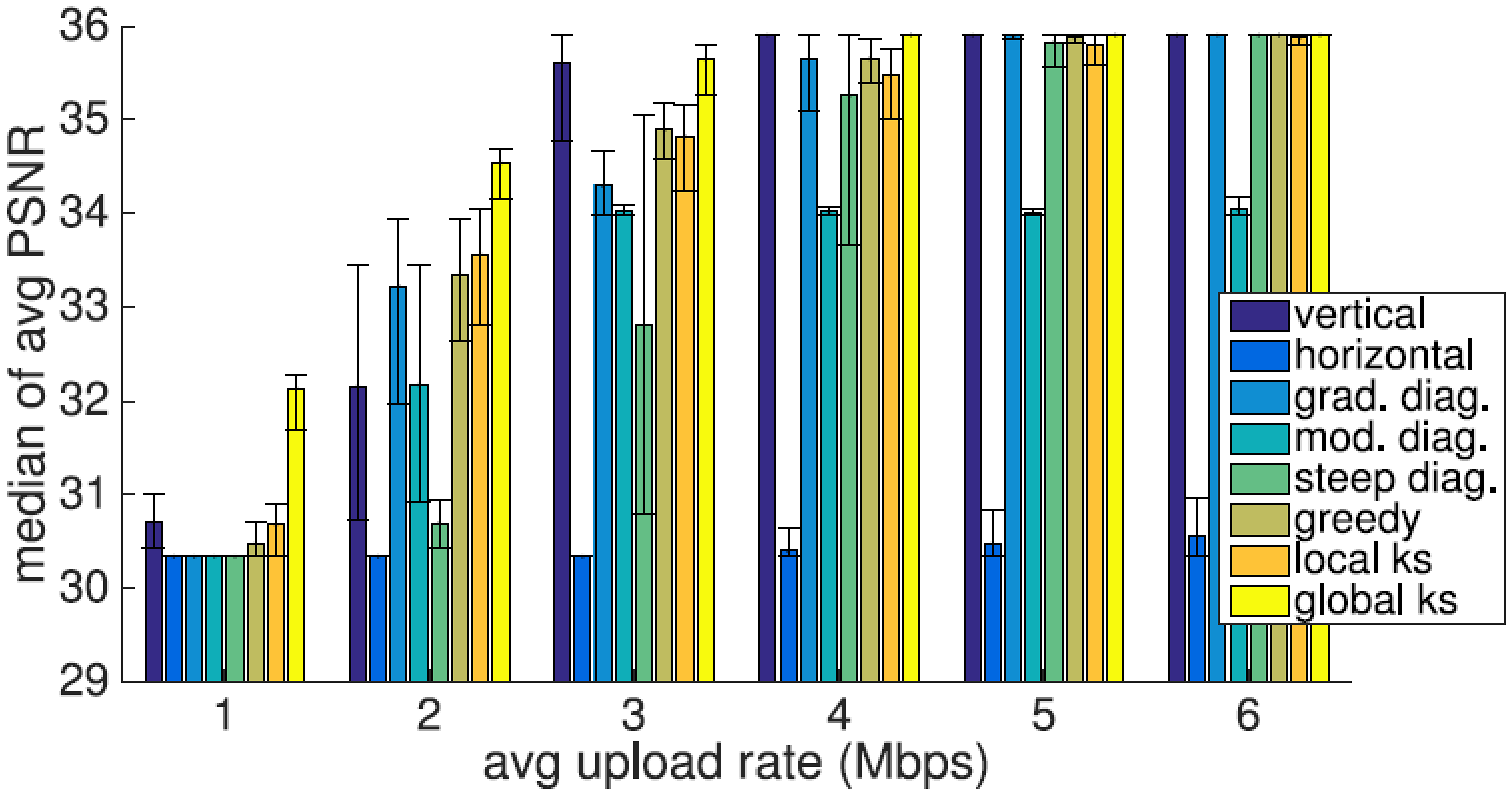
- Spatial and SNR scalability, standard test sequences, from QCIF (176x144) to 4CIF (704x576) resolution



(a) Base layer (L0) (b) Enh. layer #1 (c) Enh. layer #2

Results

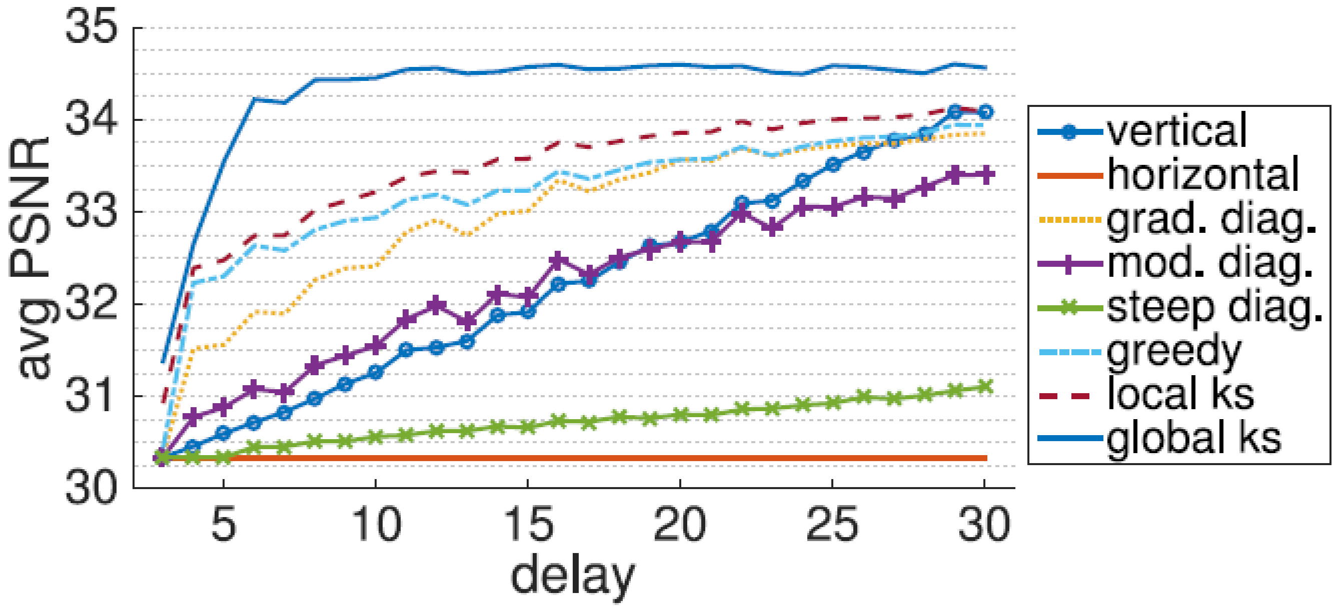
- Quality measured through PSNR (Peak Signal-to-Noise Ratio) w.r.t. the original video sequence at full resolution



Results

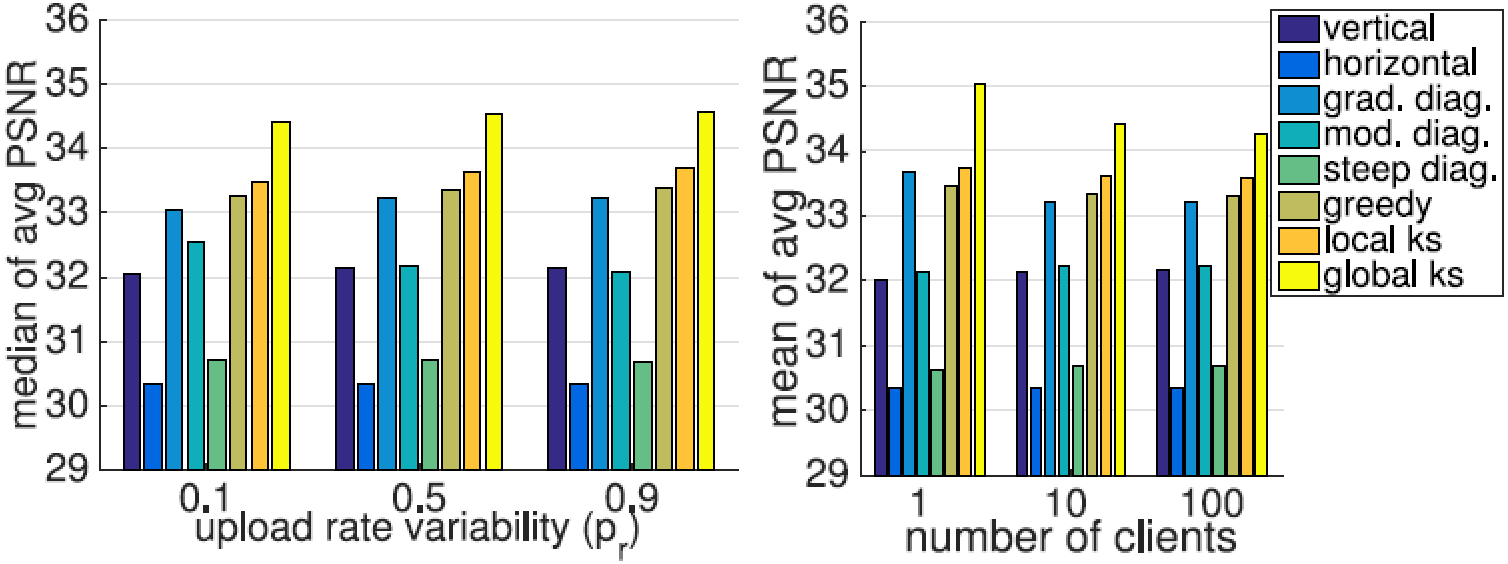
- Tradeoff between quality and delay, for clients

  (2 Mbps upload rate)



Results

- Stability vs simulation parameters



First Conclusions

- Optimized adaptation strategies for live multimedia communications from mobile devices have been designed
- Simple greedy and local optimal algorithms have been provided
- They are shown to perform not far from the global optimum which has channel knowledge in advance
- The algorithms are simple and can be easily implemented in mobile devices

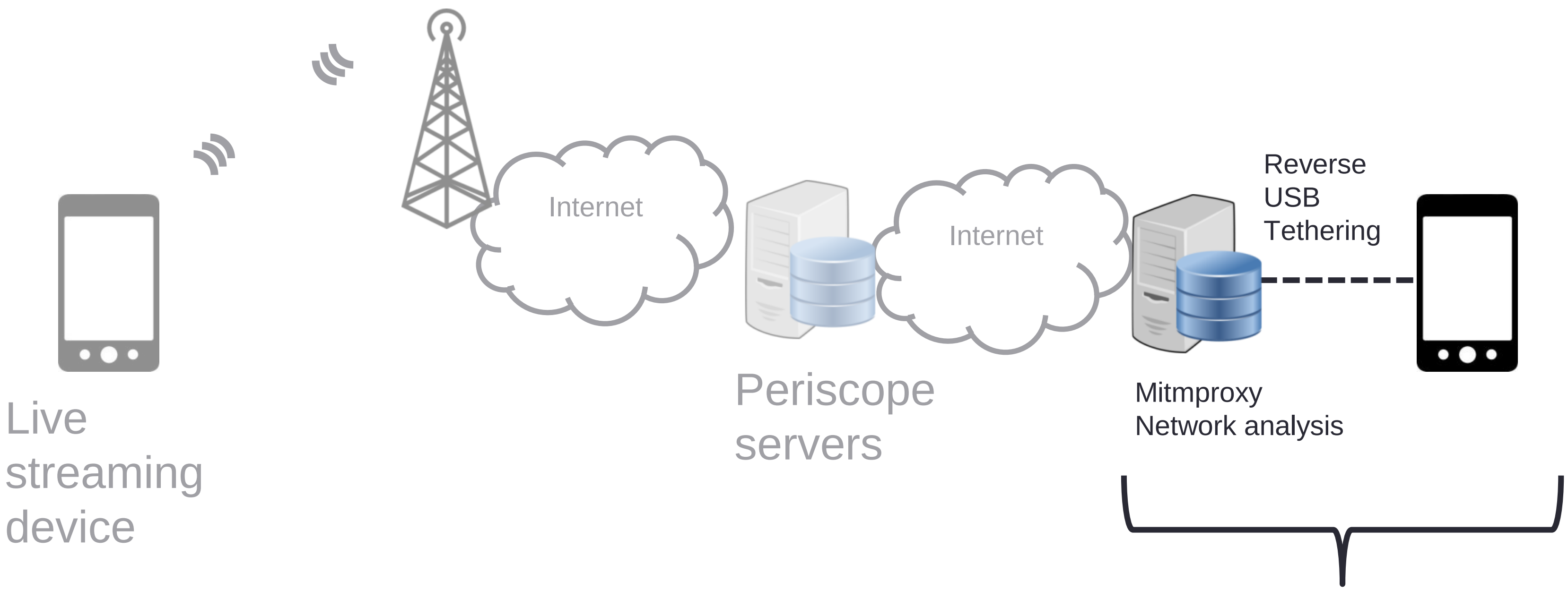
Outlook on the Mobile Live Video Trend

- Mobile live video broadcasting is becoming increasing popular. For instance:
  - App for live streaming from mobile devices:
    - Periscope, Facebook Live, Meerkat, etc.
  - Very popular applications: tens of thousands of users, growing
- Number of receivers per single event can vary significantly
  - Few or 100s / 1,000s

We focused on Periscope

- App for live streaming from mobile devices
  - Similar to Facebook Live, Meerkat, etc
  - Very popular application: tens of thousands of users, growing
- Possibility of selecting a (public) random broadcast through the app “Teleport” button
  - Used for our analysis
- Live streaming with different protocols: RTMP and HLS
  - RTMP: Real Time Multimedia Protocol (Adobe)
  - HLS: HTTP Live Streaming (Apple)

Periscope Analysis Scenario



Our analysis

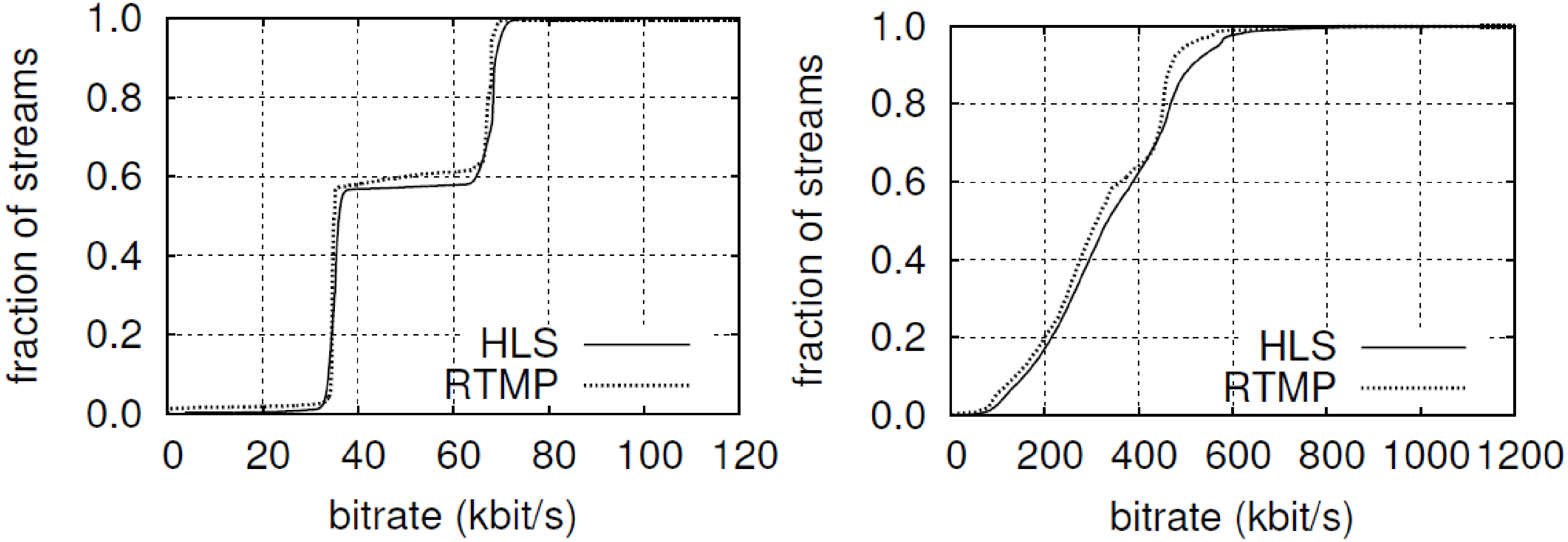
Results from [4] (Sep 2016)

Periscope Challenges

- Mobile upload: unreliability of wireless channel
- Avoid freeze events (rebuffering events) at the receiver side
- Tradeoff: latency vs freeze probability in playback

Media Characteristics

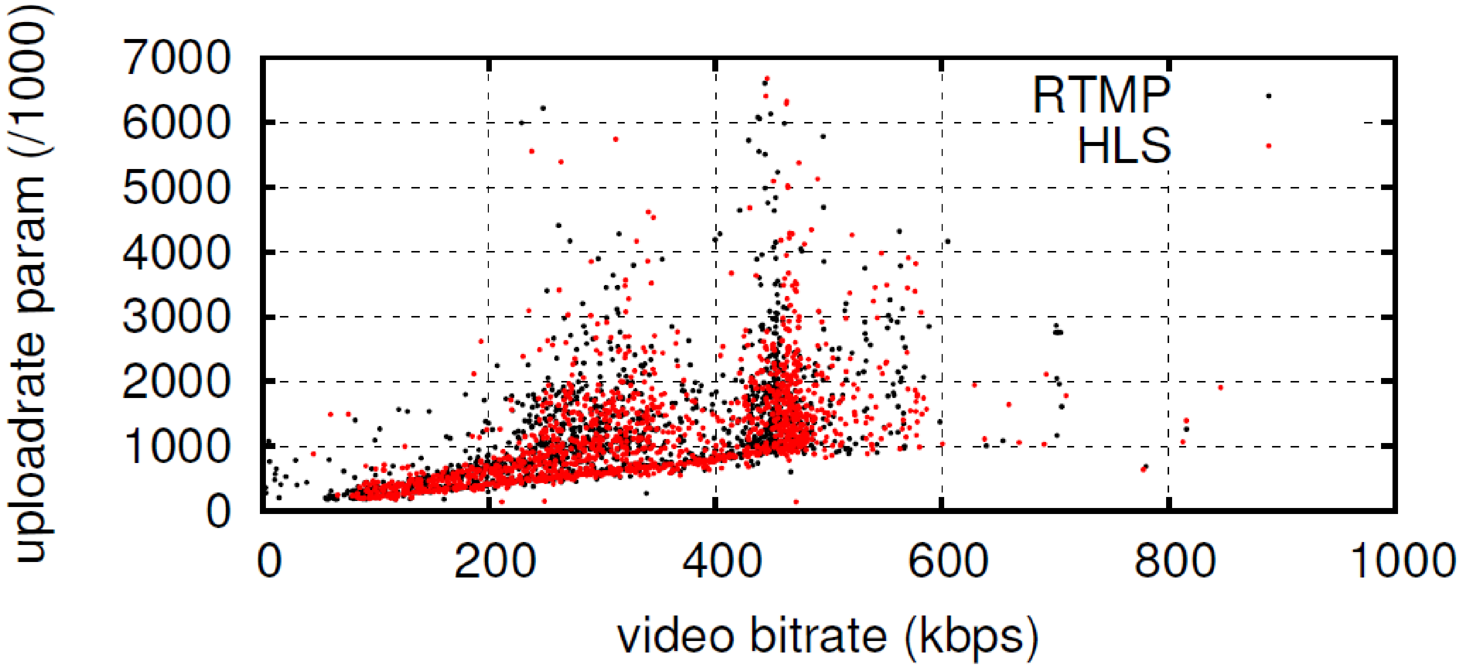
- Audio: 32 and 64 kbit/s
- Video: mostly from 100 to 600 kbit/s, resolution: 320 x 568
- Independent of the protocol: RTMP or HLS



(a) audio (b) video

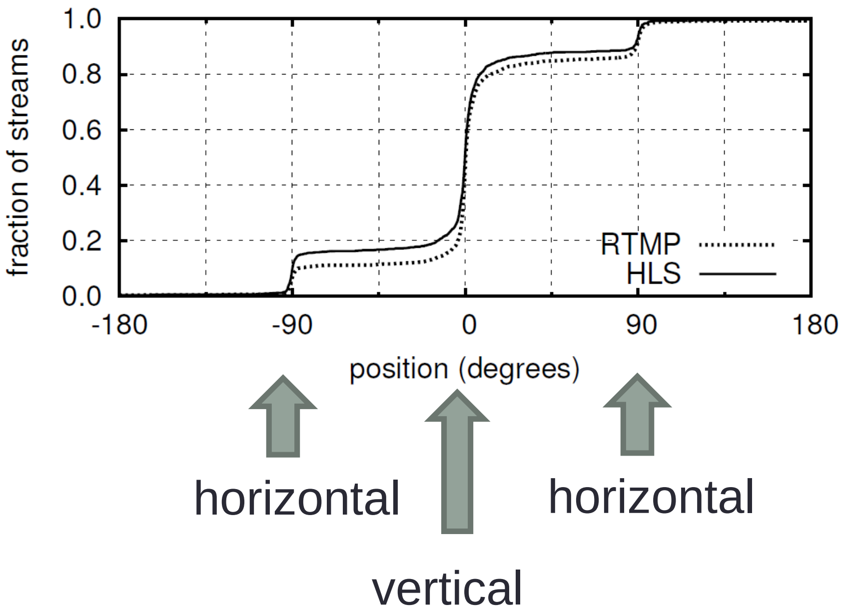
Insights from Embedded Information

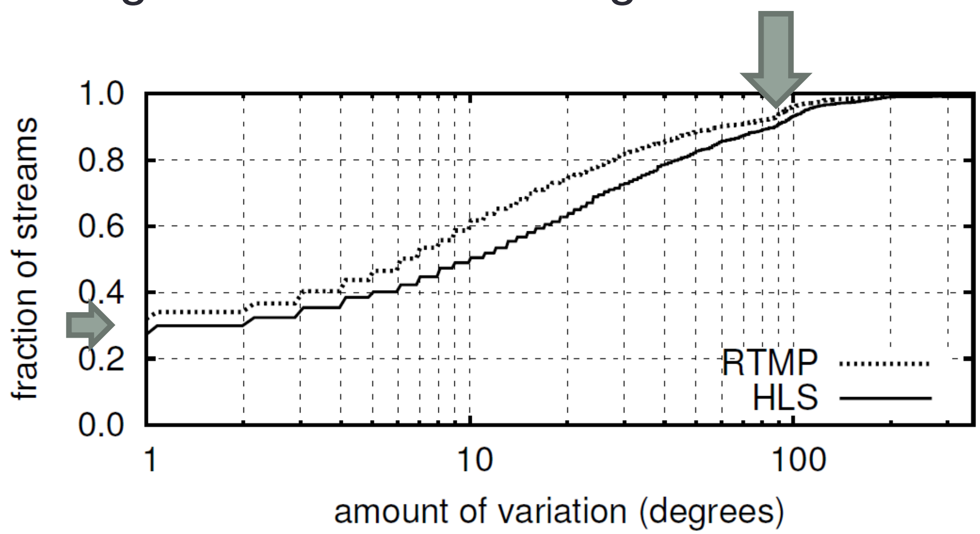
- The stream contains embedded information from Periscope
- Most interesting is *uploadrate* (probably the estimated available upload bandwidth)
- Video rate is capped at about 450 kbps. HLS similar to RTMP



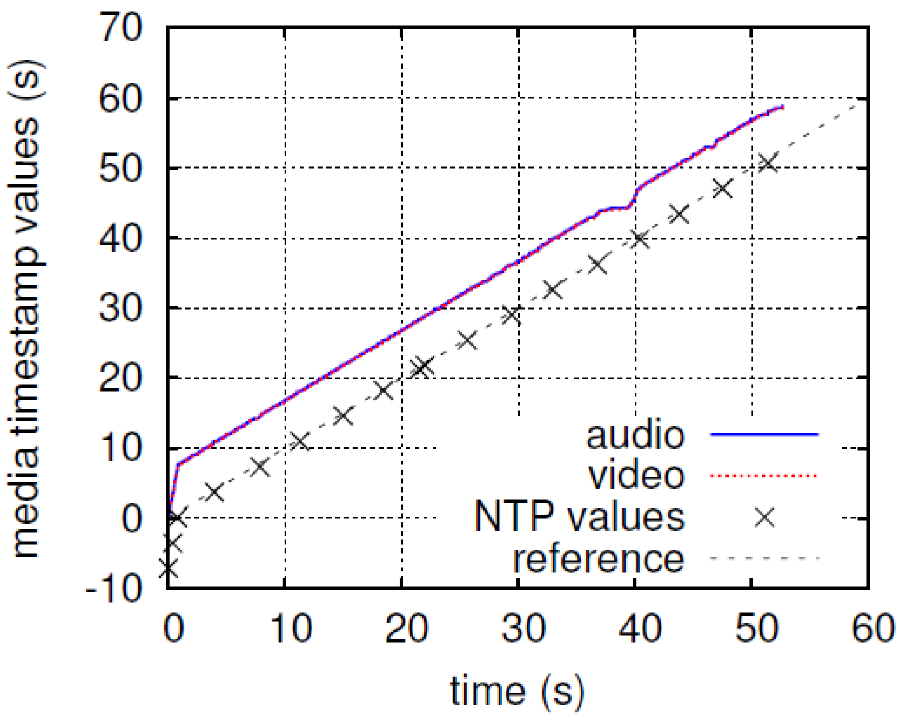
How the Mobile Device is Handled

- From information embedded in the stream:
  - Average position: about 60% vertical
  - 30% of the cases: almost no movements
  - 10% of the cases: rotation > 90 degrees while streaming

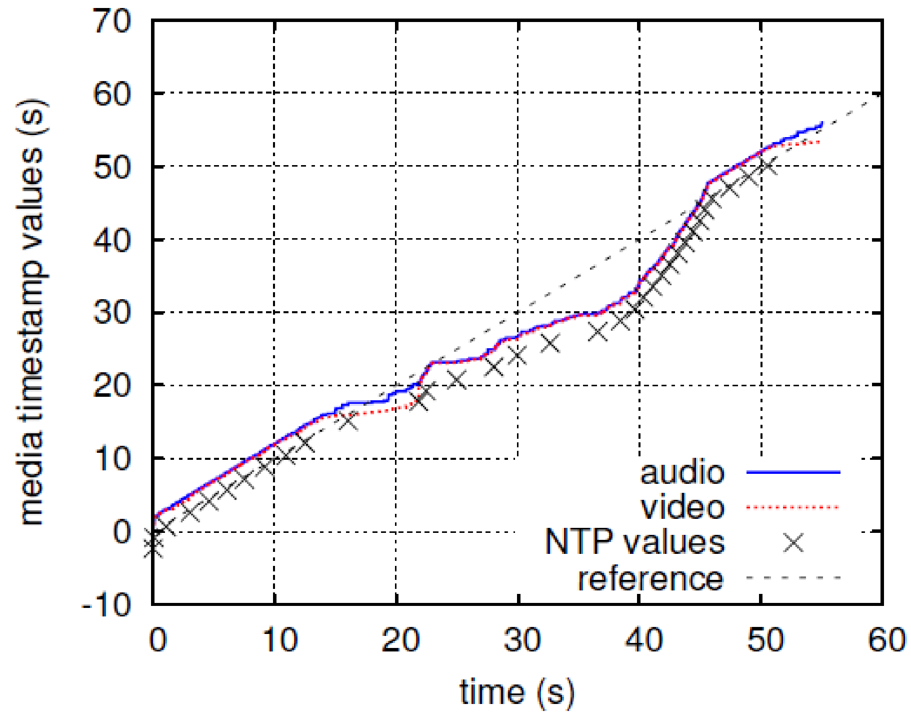




Behavior Over Time: RTMP

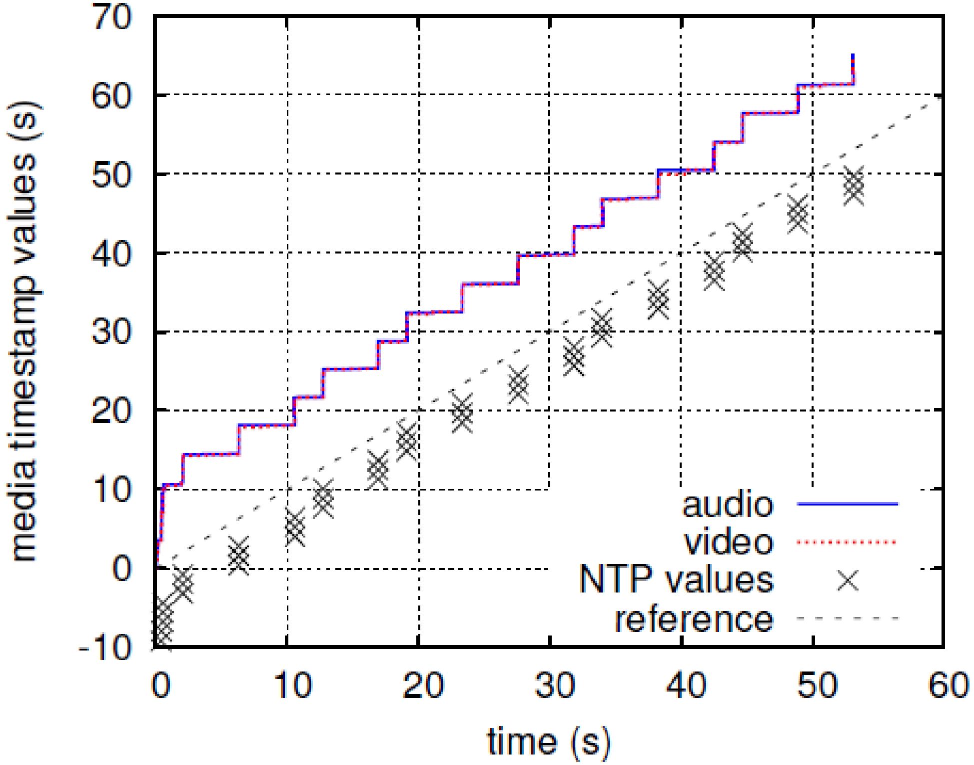


(a) RTMP sample #1



(b) RTMP sample #2

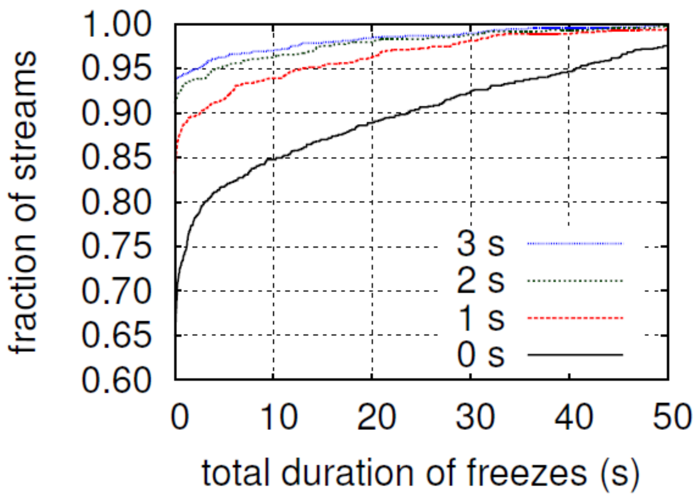
Behavior Over Time: HLS



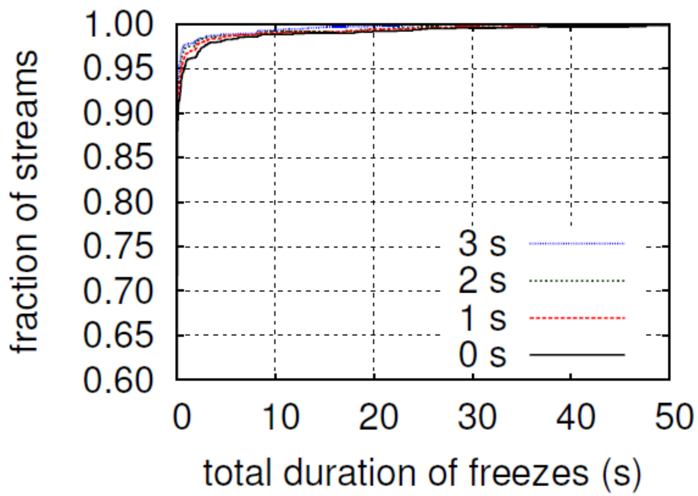
(c) HLS sample #1

Playback Impairments

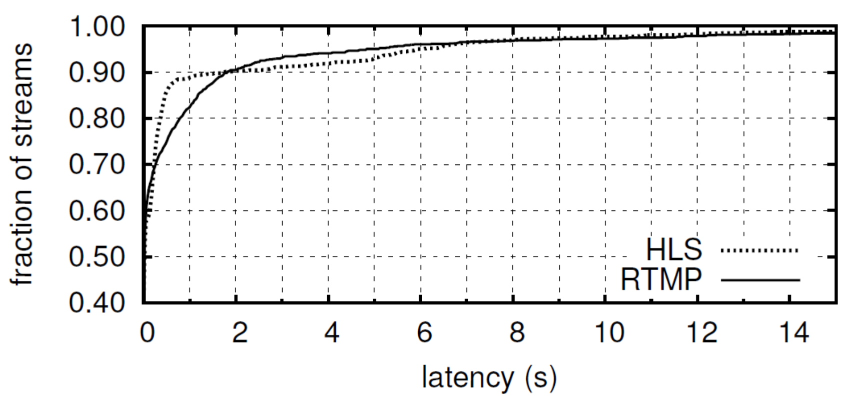
As a function of a simulated initial playout delay (no access to the app…)



(a) RTMP



(b) HLS



1 or 2 s enough for 90% of users, depending on the protocol

Conclusions

- We provided an overview of the status of real-time multimedia communications from mobile devices
- A general framework for multimedia communication optimization has been discussed, with particular reference to optimization strategies for mobile live streaming
- An outlook about current mobile streaming services has been delineated, focusing on the specific characteristics of “Periscope”
- Future work will be devoted to further experiment with adaptation strategies, both in the case of upload and in the case of existing applications

References

[1] Sandvine Global Internet Phenomena Report – Latin America & North America, June 2016, Downstream peak period applications, North America, Fixed Access.

[2] P.A. Chou, Z. Miao, “Rate-Distortion Optimized Streaming of Packetized Media”, IEEE Transactions on Multimedia, vol. 8, n. 2, Apr 2006, pp. 390-404.

[3] M. Siekkinen, E. Masala, J. K. Nurminen, “Optimized Upload Strategies for Live Scalable Video Transmission from Mobile Devices”, IEEE Transactions on Mobile Computing, DOI: 10.1109/TMC.2016.2585138 (ISSN: 1536-1233), Apr 2017.

[4] L. Favario, M. Siekkinen, E. Masala, “Mobile Live Streaming: Insights from the Periscope Service”, IEEE Workshop on Multimedia Signal Processing (MMSP), Montreal, Canada, Sep 2016.