MPEG-DASH Standard with SVC Video Streaming on Android Mobile Devices

CHIEN CHANG CHEN

Outline

Introduction

Background

- SVC
- MPEG-DASH

System Architecture

Implementations

- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

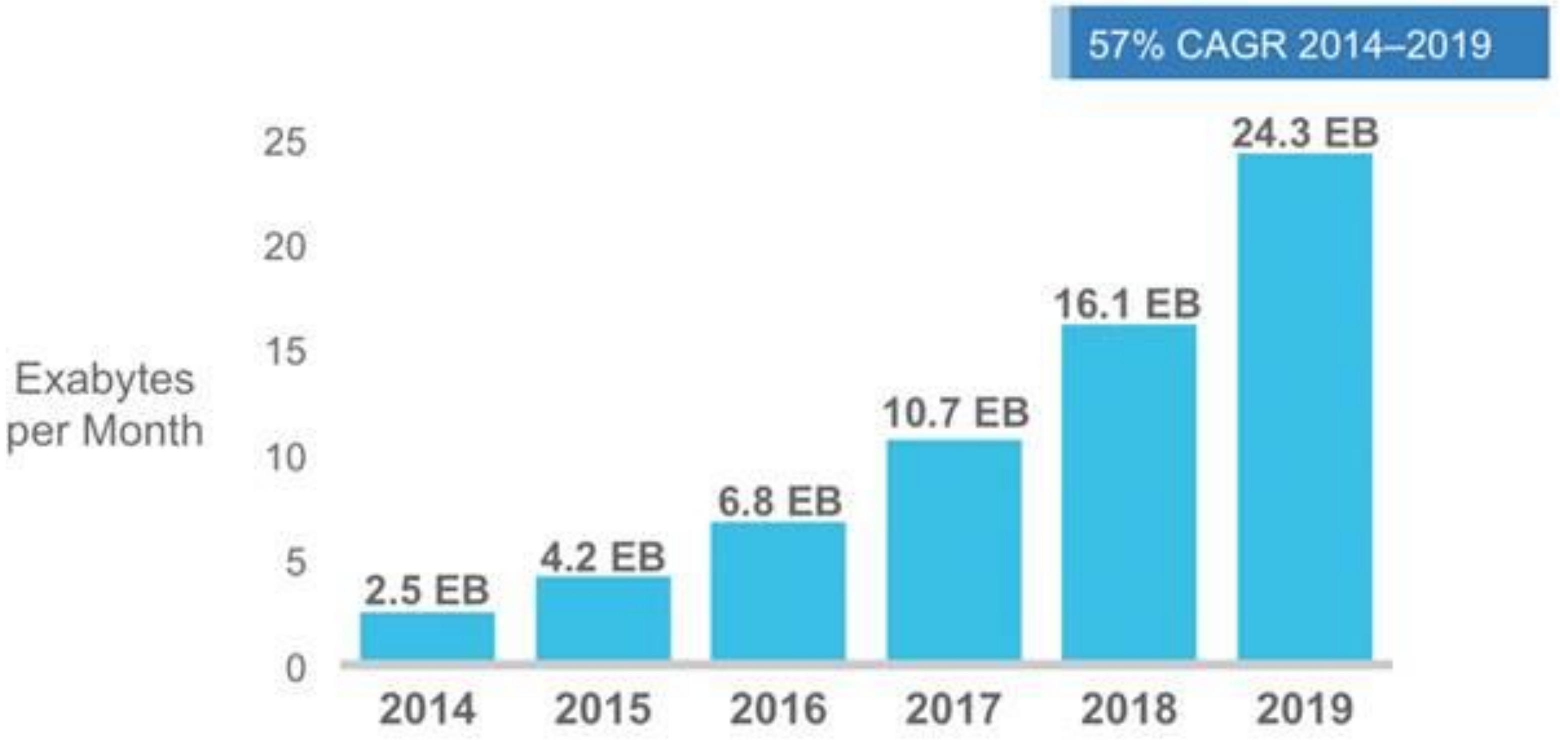
Experiments

- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

Mobile Data Traffic

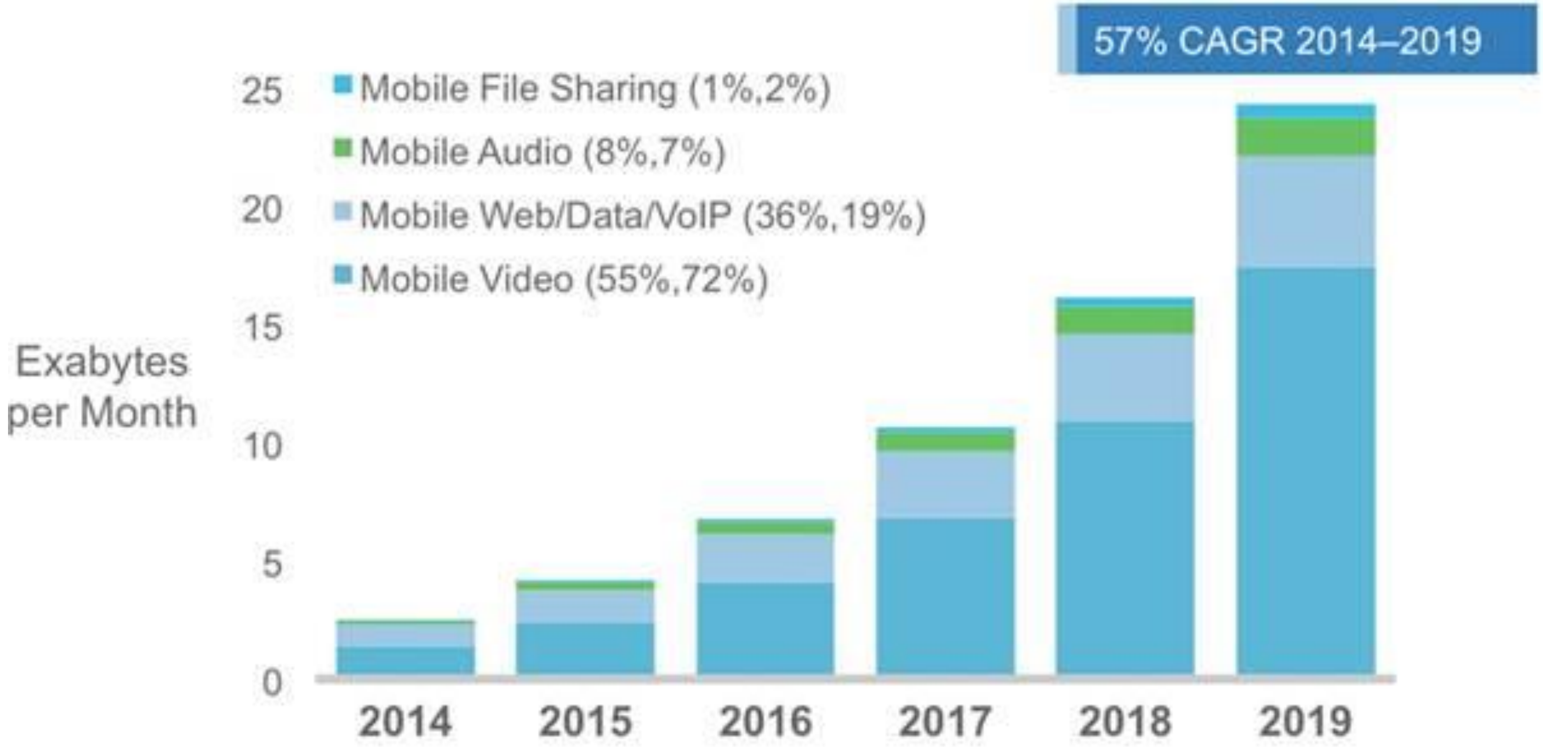
Cisco expected that[1] the mobile data traffic in 2019 will be **10** times the traffic in 2014



[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

Traffic Data for Mobile Video Streaming

Over **70%** of mobile data traffic is video streaming

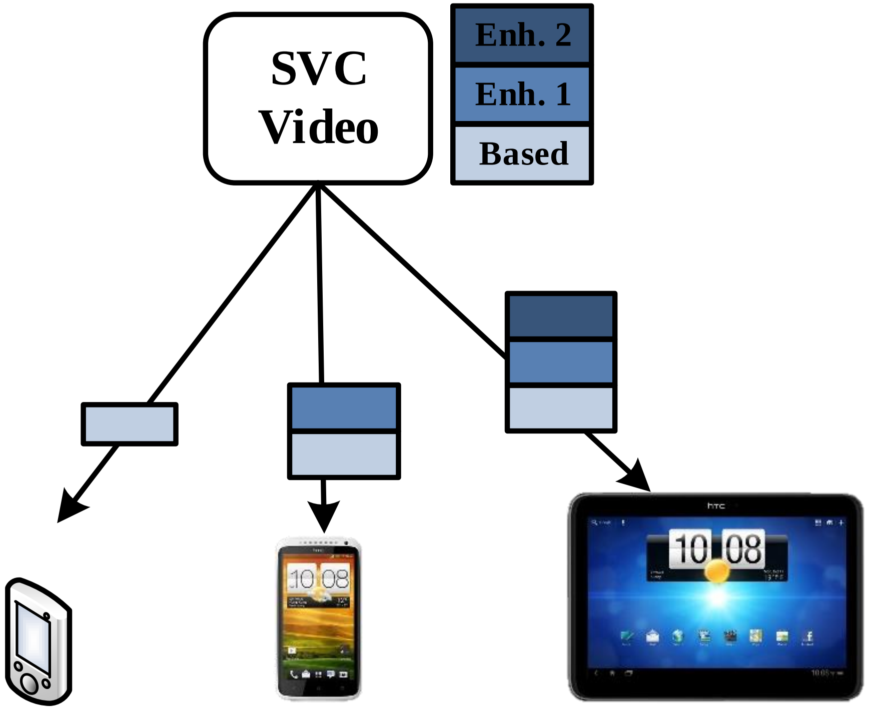


It is getting more and more important to **provide better mobile video streaming service**

SVC is More Suitable for Heterogeneous Mobile Streaming

Non-scalable video codecs are not suitable for mobile devices

- Devices are heterogeneous
  - Different power computation and screen size
- Wireless network conditions are dynamic
  - Can’t receive entire media data when bandwidth is insufficient



Scalable Video Codec (SVC)

- Divided videos into dependent layers
- Request proper layers

There is **no existing SVC chip**

- **We implemented the pioneering SVC decoder on Android devices**

Efficient Approach for Transferring Video

Dynamic Adaptive Streaming over HTTP (MPEG-DASH)

- Video is divided into segments with the same length
- Reuse existing technologies:
  → proxies, caches, codecs, container
- Switch events are decided by client side:
  → more accurate decisions
- Use common apache web server
  → easy to extend
- Without NAT traversal problem

mpeg-DASH

We port MPEG-DASH client from PC to android and integrate it with SVC decoder

- High frame-rate and high throughput
- Better user experience

Contributions

We implement the pioneering SVC decoder on android phones

- Leverage multi-threading to enhance decoding performance

We integrate MPEG-DASH client into our SVC decoder

- Our DASH client achieves high throughput

Publicize our source codes on Github

Outline

Introduction

Background
- **SVC**
- MPEG-DASH

System Architecture

Implementations
- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

Experiments
- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

Scalable Video Codec - SVC

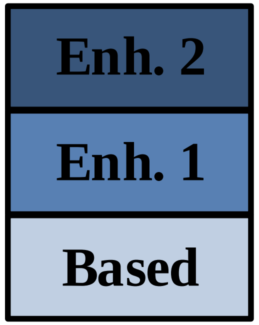
SVC is extended by H.264/AVC

Supported Scalability

- Temporal (Frame-rate)
- Spatial (Resolution)
- Quality (QP)

Features

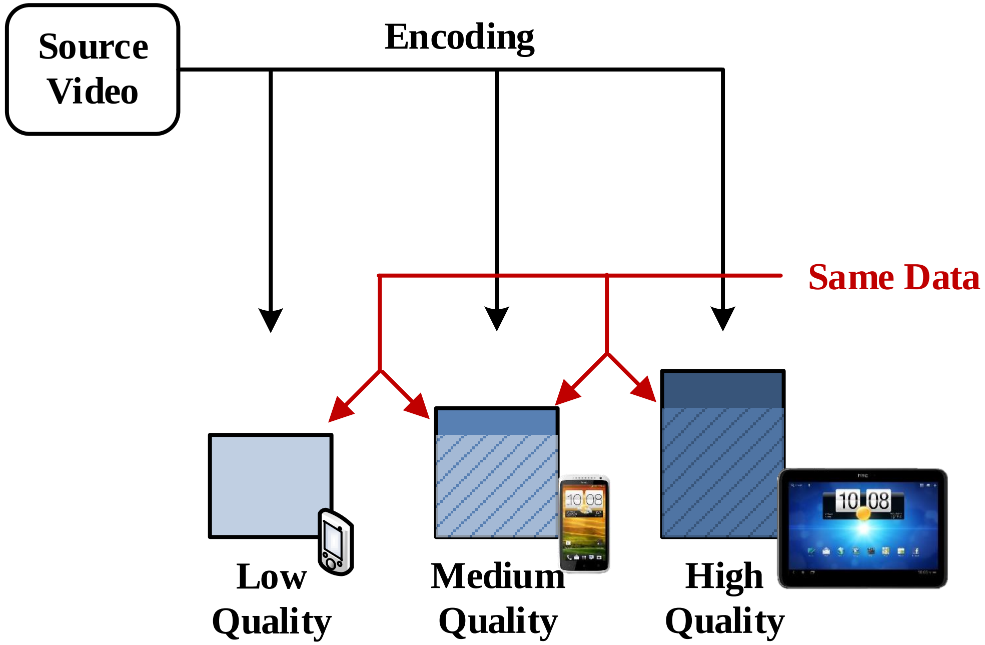
- Videos are encoded into dependent layers
- Based layer is necessary
- More enhancement layers → better video quality



Non-Scalable Video Content

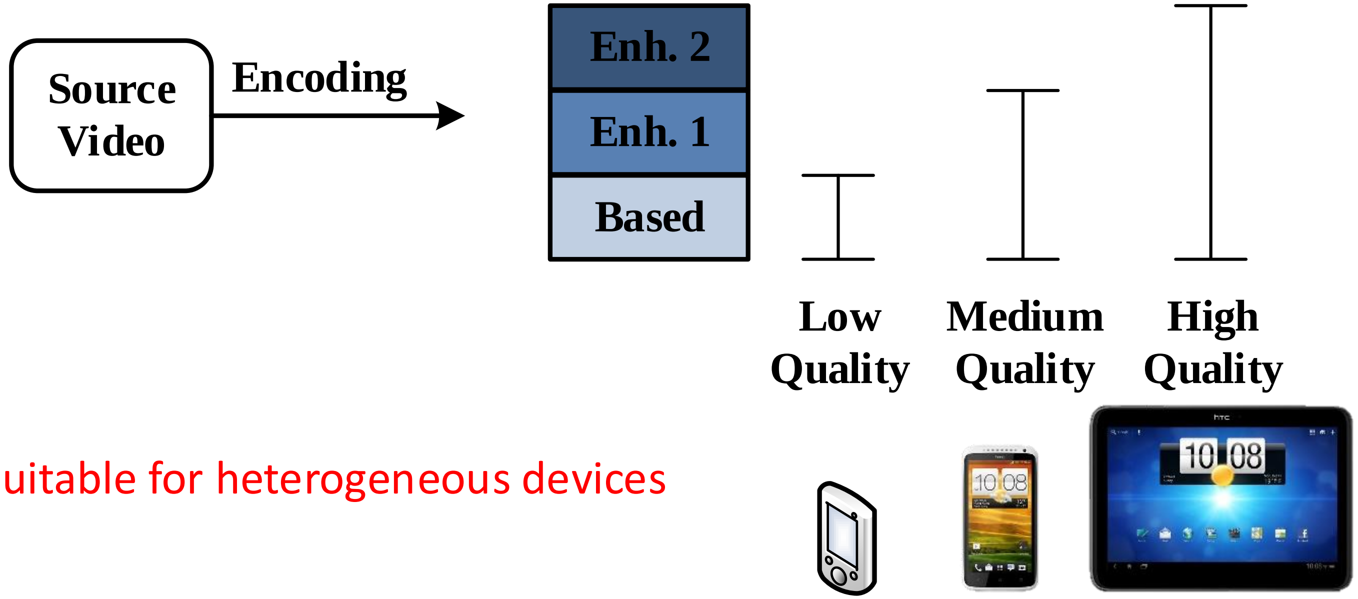
One copy for each combination of video encoding parameters

Duplicate data among multiple copies



SVC Video Content

Without duplicate data among multiple layers



SVC is more suitable for heterogeneous devices

Outline

Introduction

Background

- SVC
- **MPEG-DASH**

System Architecture

Implementations

- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

Experiments

- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

MPEG-DASH

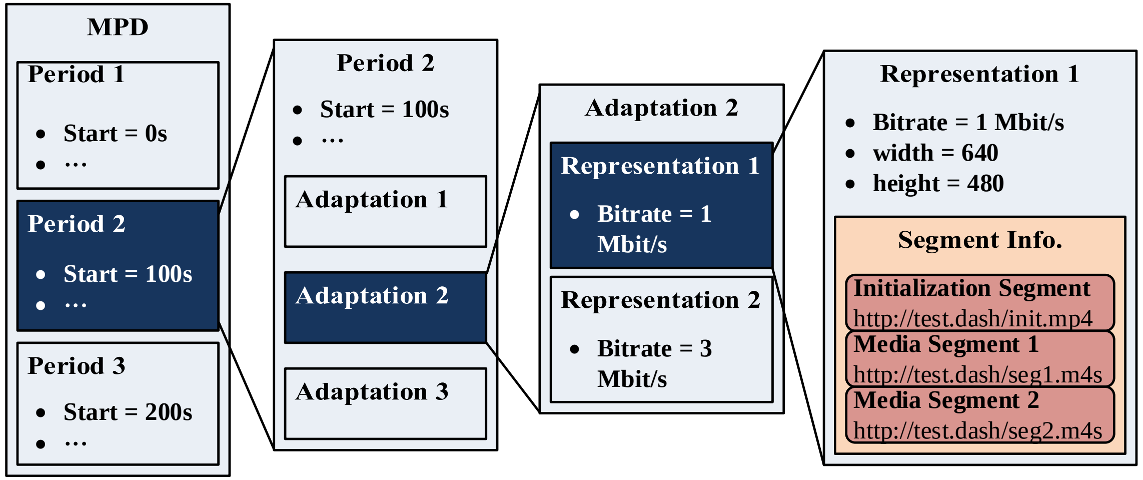
Dynamic Adaptive Streaming over HTTP

Features

- Videos are divided into segment with the same length
- Requesting unit is segment
- MPD (Media Presentation Description) is manifest file

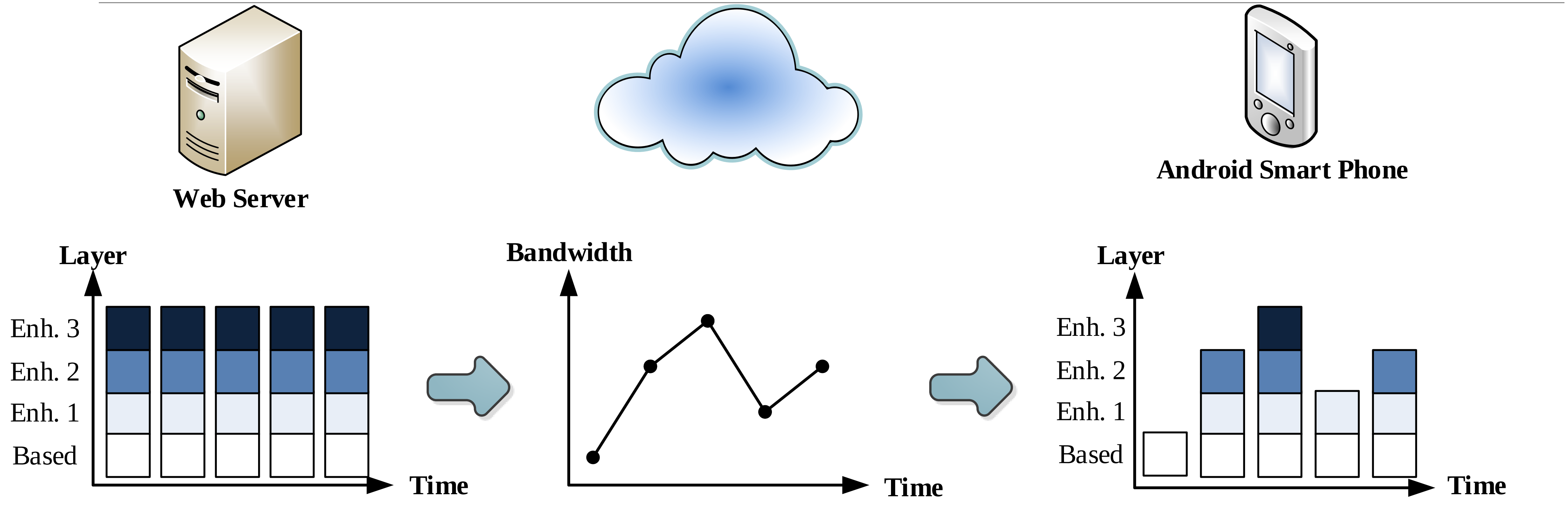
Use existing technologies

- Container
- Codec
- Proxy
- Etc.



Enable seamless switching to multiple bit-rate streams

MPEG-DASH with SVC Streaming



Video with multiple layers on Web Server

Measured available bandwidth

Received segments on Client

MPEG-DASH with SVC streaming provides more flexible video streaming service

Outline

Introduction

Background
- SVC
- MPEG-DASH

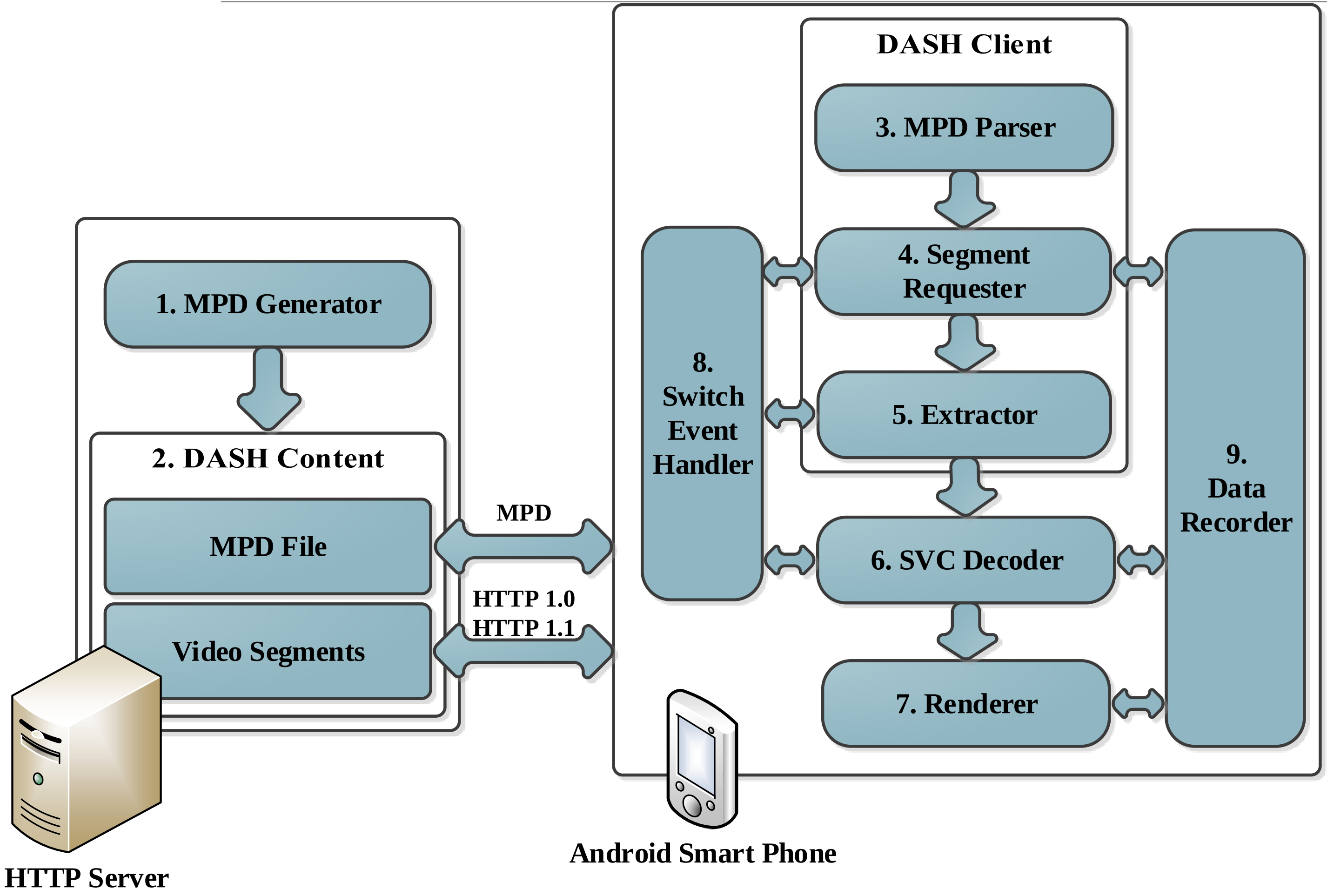
System Architecture

Implementations
- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

Experiments
- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

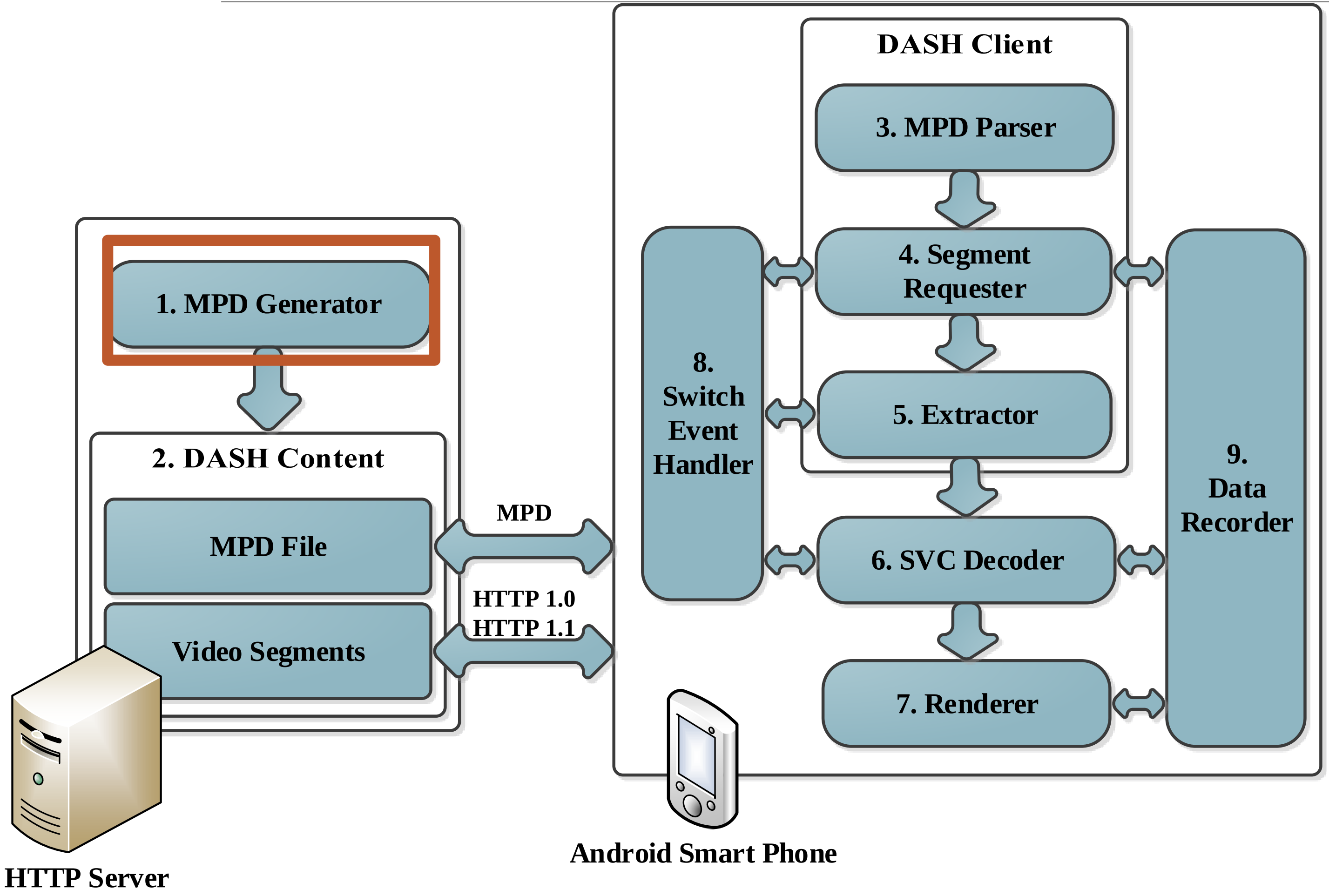
System Architecture



The components in our system

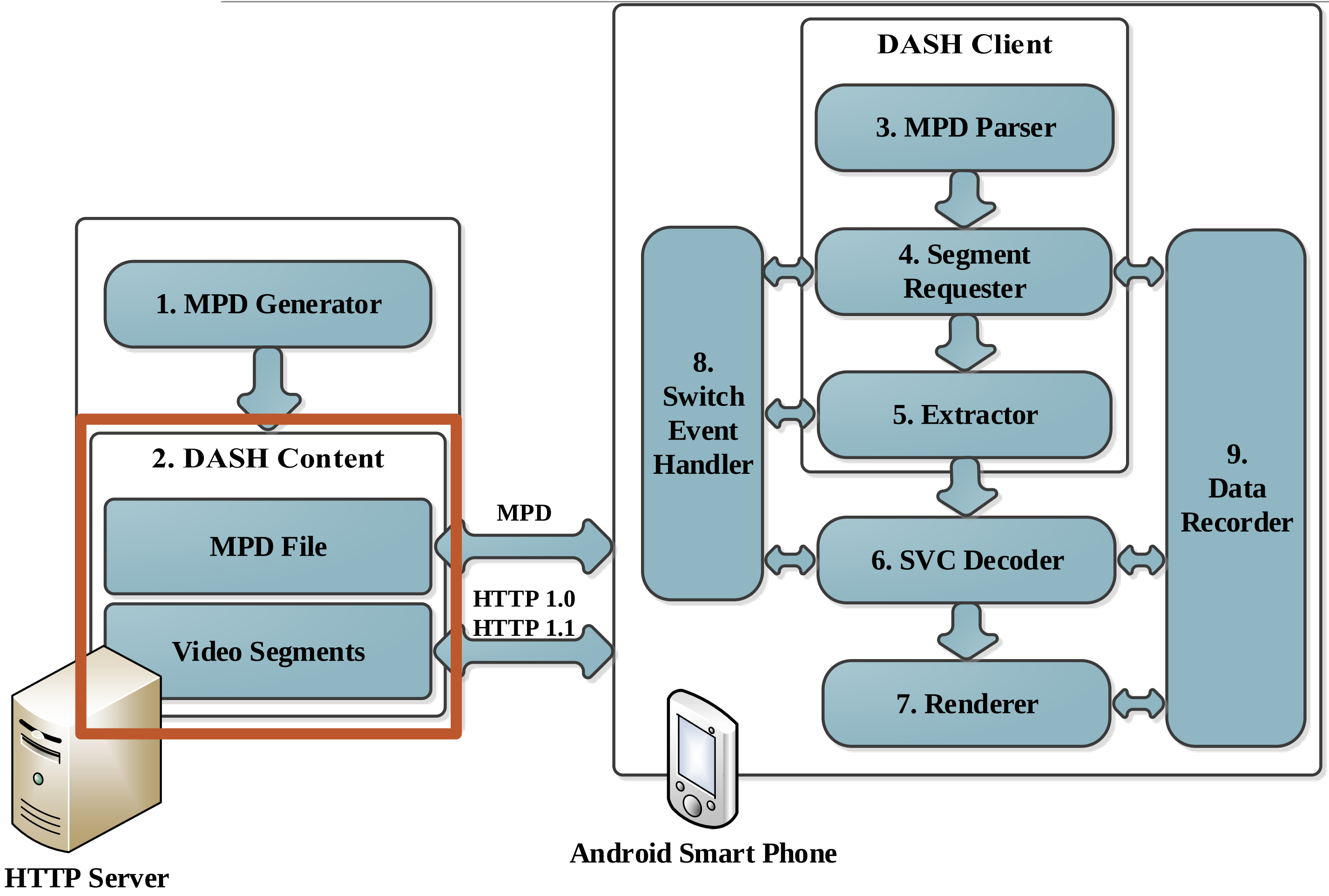
1. MPD Generator
2. DASH Content
3. MPD Parser
4. Segment Requester
5. Extractor
6. SVC Decoder
7. Renderer
8. Switch Event Handler
9. Data Recorder

MPD Generator



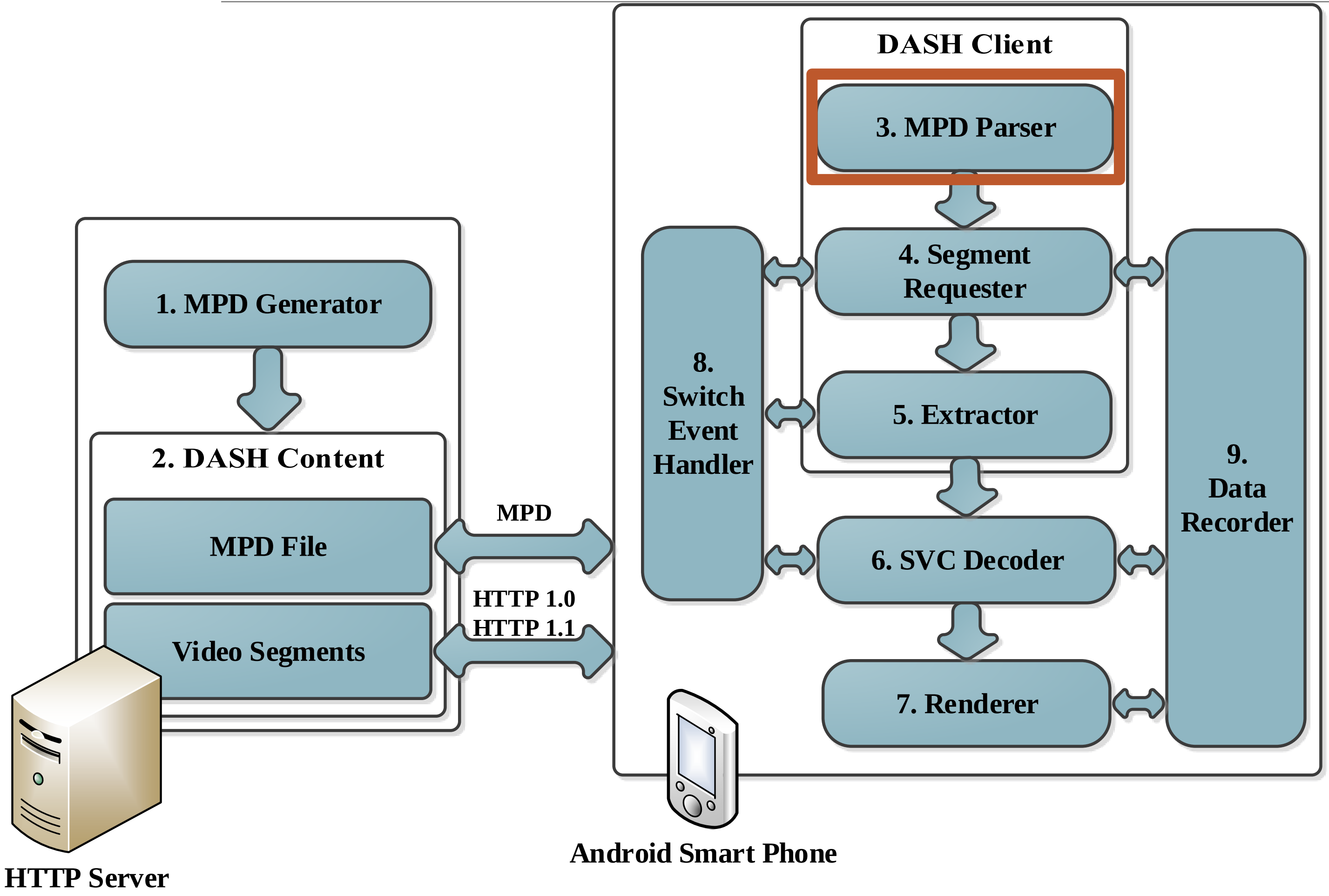
1. SVC Videos Encoding
   1) Encode videos into raw format
   2) Encode raw video into SVC video
2. For MPEG-DASH Streaming
   1) Chop SVC videos into segments and generate corresponding MPD

DASH Content



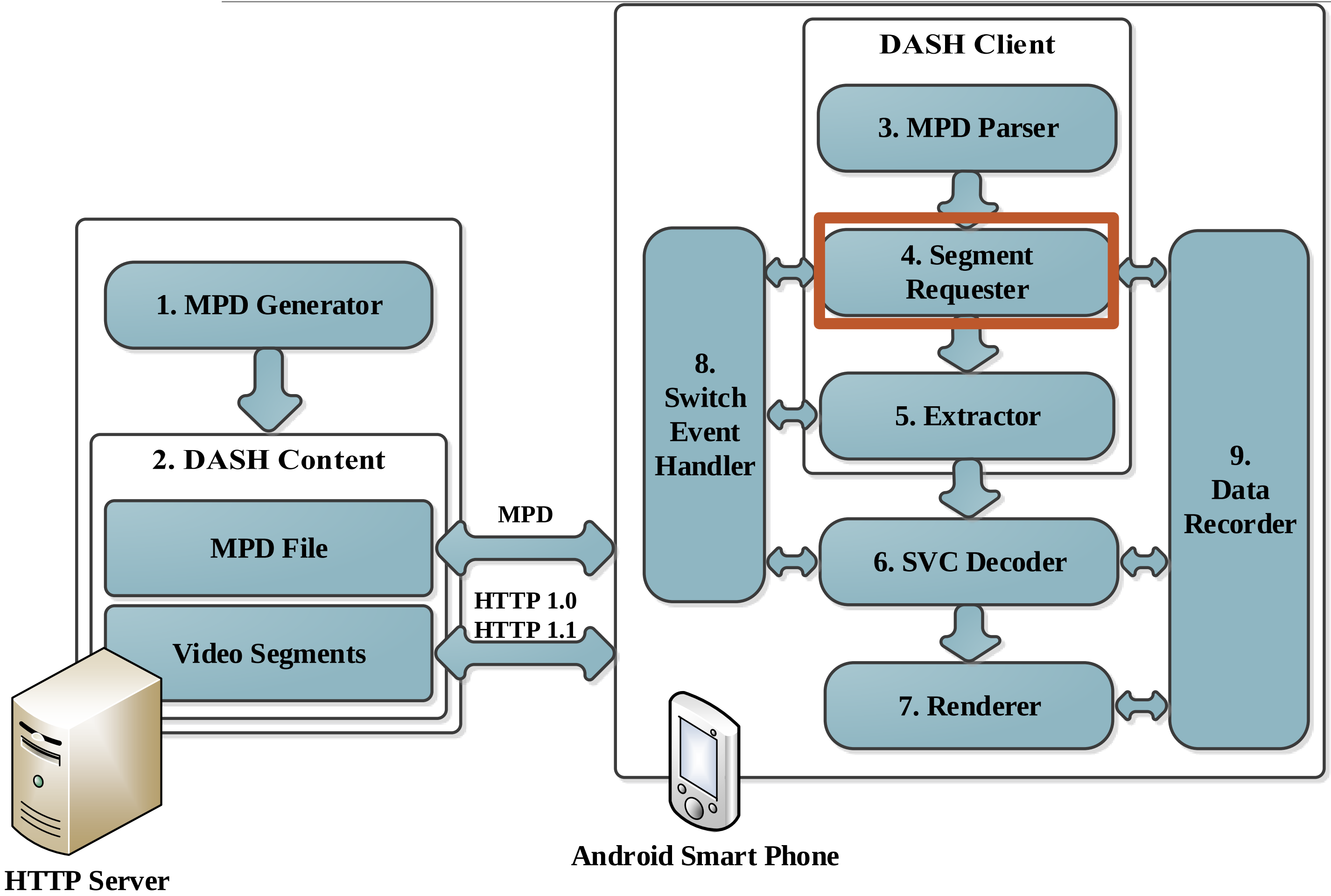
1. Place MPDs and segments
2. Requested by accessible URL over HTTP

MPD Parser



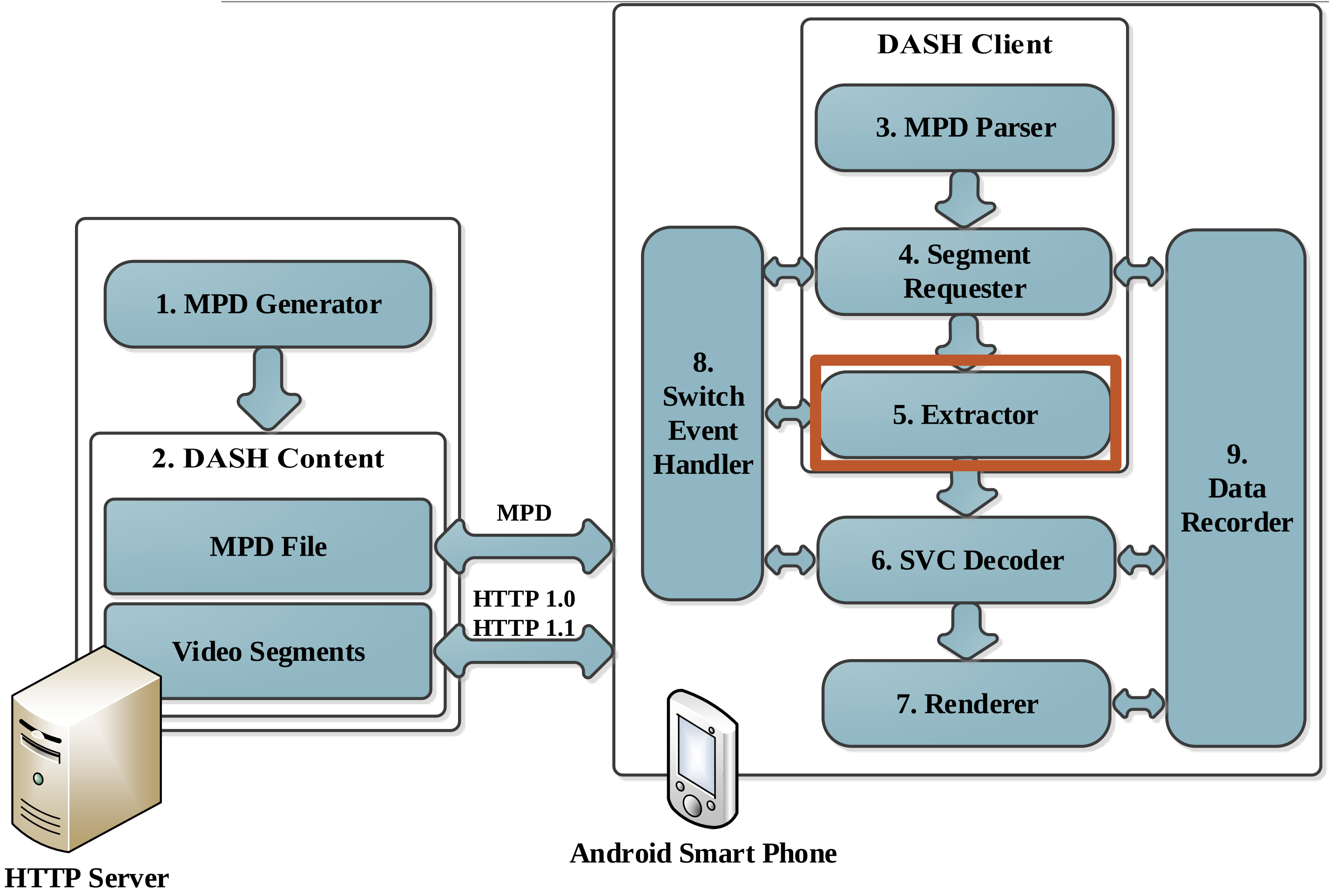
1. Parse MPD file
   - URL of segments
   - Codec
   - Resolution
   - Dependency id
   - Bit-rate
2. Information used for **Segment Requester**

Segment Requester



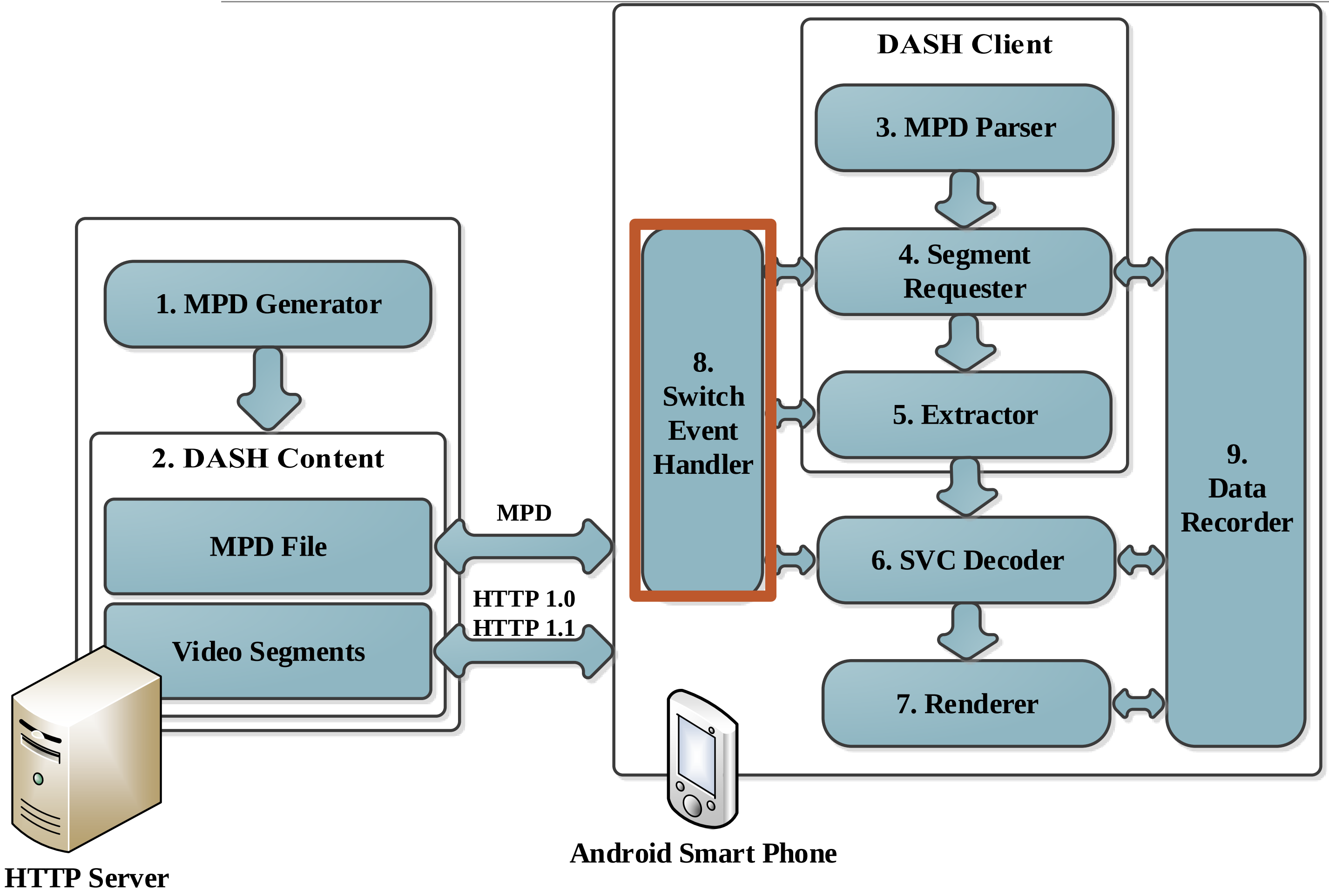
1. Request segment according to
   1) Information in MPD
   2) Selected layer number
2. The ISO base media file format (ISOBMFF)
   1) A number of different boxes
   2) Decoder does not know this format

Extractor



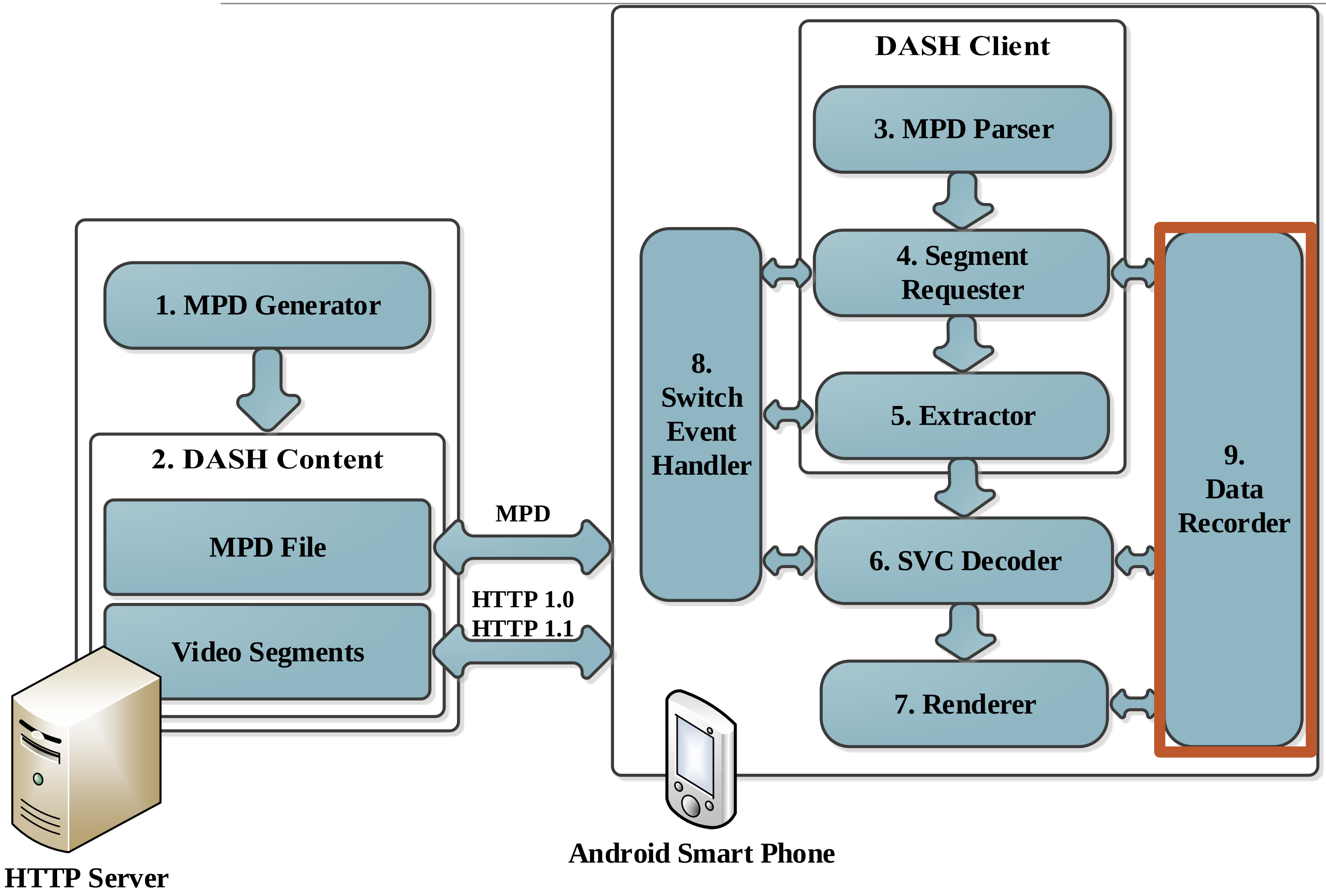
1. In charge of ISOBMFF
   1) Parse ISOBMFF
   2) Remove the boxes
   3) Obtain the media data
2. Reconstruct frame data
   1) Interleave frame data from multiple layers
   2) Make frame data compatible with SVC Decoder

Switch Event Handler



1. Triggered by
   - User preference
   - Algorithms
2. Send Notification to
   - **Segment Requester** to change the number of layers for requesting
   - **Extractor** to cancel the segments if layer number exceeds the selected layer number
   - **Decoder** to change the decoding parameters

Data Recorder



1. Measure the information
   - Throughput
   - Frame-rate
   - Delay
2. Used for
   - Switching algorithms
   - Performance analysis

Outline

Introduction

Background
- SVC
- MPEG-DASH

System Architecture

Implementations
- **Multi-Core SVC Decoder on Android**
- MPEG-DASH with SVC Decoder on Android

Experiments
- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

Port SVC Decoder to Android Devices

Single-thread SVC decoders are not real-time for mobile devices

- Ex: JSVM[1], OpenSVC[2]

We implemented multi-thread SVC decoder on Android devices

- Use OpenSVC as our SVC decoder library

OpenSVC does not support multi-thread

- We focus on avoiding race condition problem when invoking the decoder function

OpenSVC
Tools to scale

Compare JSVM with OpenSVC (running on OS X with Intel i5 2.3 GHz CPU)

Decoder	doc	jeux	soap	sport	talk
JSVM	17.75 (FPS)	20.36	19.44	17.44	19.11
OpenSVC	18.79	27.03	20.79	18.71	24.26

[1] http://www.hhi.fraunhofer.de/en/fields-of-competence/image-processing/research-groups/image-video-coding/svc-extension-of-h264avc/jsvm-reference-software.html

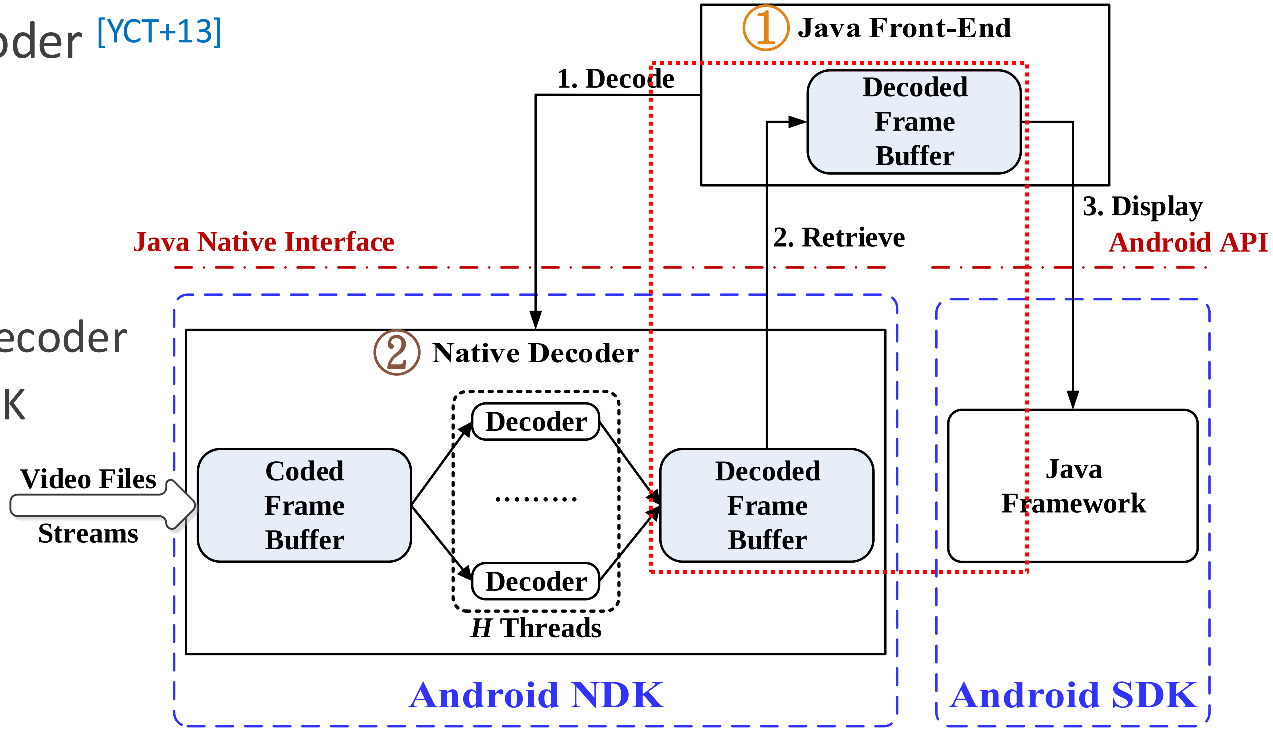
[2] http://www.opensvc.com/

Multi-Core SVC Decoder

Two main components of multi-core SVC decoder [YCT+13]

① **Java Front-End**

② **Native Decoder**



① Java Front-End

- Use JNI as interface between Java and Native Decoder
- Interact with Android framework by Android SDK

② Native Decoder

- Coded Frame Buffer (CFB)
  → stores packets from source
- Multiple decoder threads
- Decoded Frame Buffer (DFB)
  → stores decoded frames

[YCT+13] Y. Li, ***C. Chen***, T. Lin, C. Hsu, Y. Wang, and X. Liu. *An end-to-end testbed for scalable video streaming to mobile devices over http* (ICME'13)

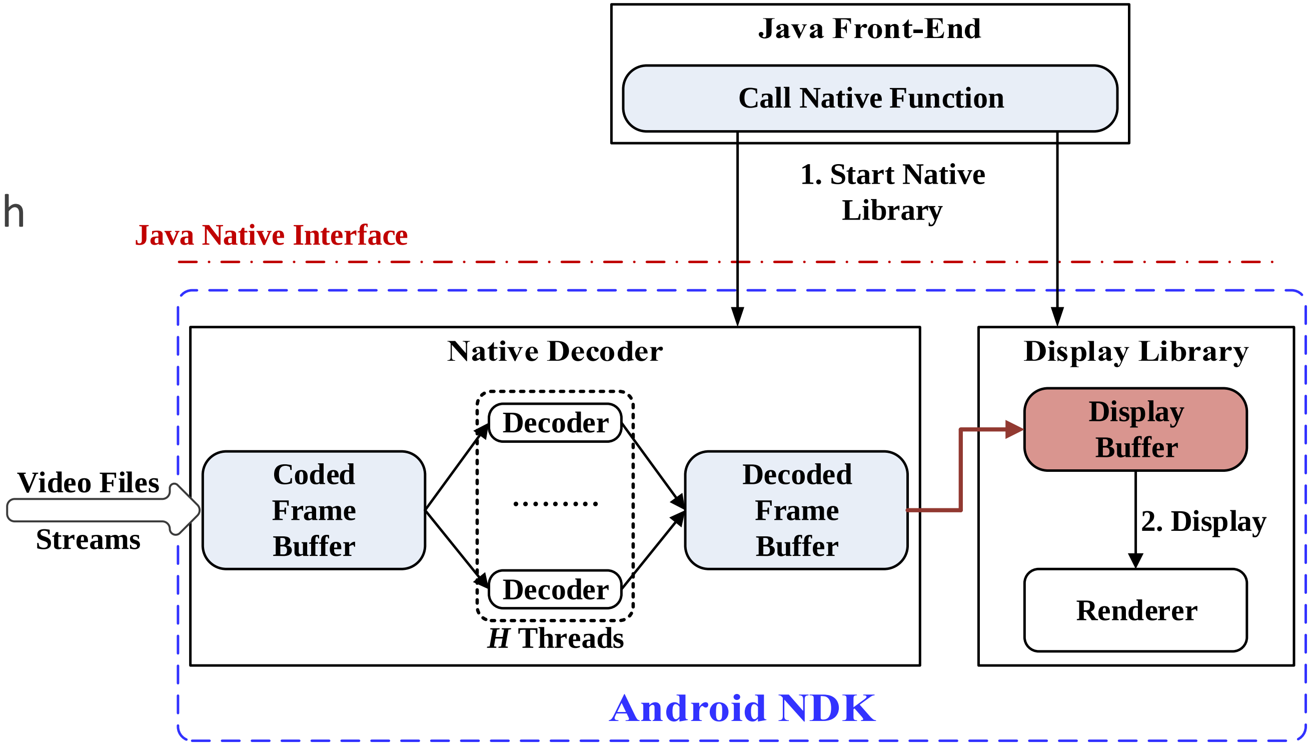
Replace Renderer with SDL Library

SDL[1] is a popular C/C++ library

- An open-source library
- Support hardware rendering
- Easier to handle events such as screen touch and video display than Android API

Integrate SDL into our SVC decoder

- Directly display frame data in native side
- To solve new race condition problem
- To avoid deadlock



[1] https://www.libsdl.org/

Outline

Introduction

Background
- SVC
- MPEG-DASH

System Architecture

Implementations
- Multi-Core SVC Decoder on Android
- **MPEG-DASH with SVC Decoder on Android**

Experiments
- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

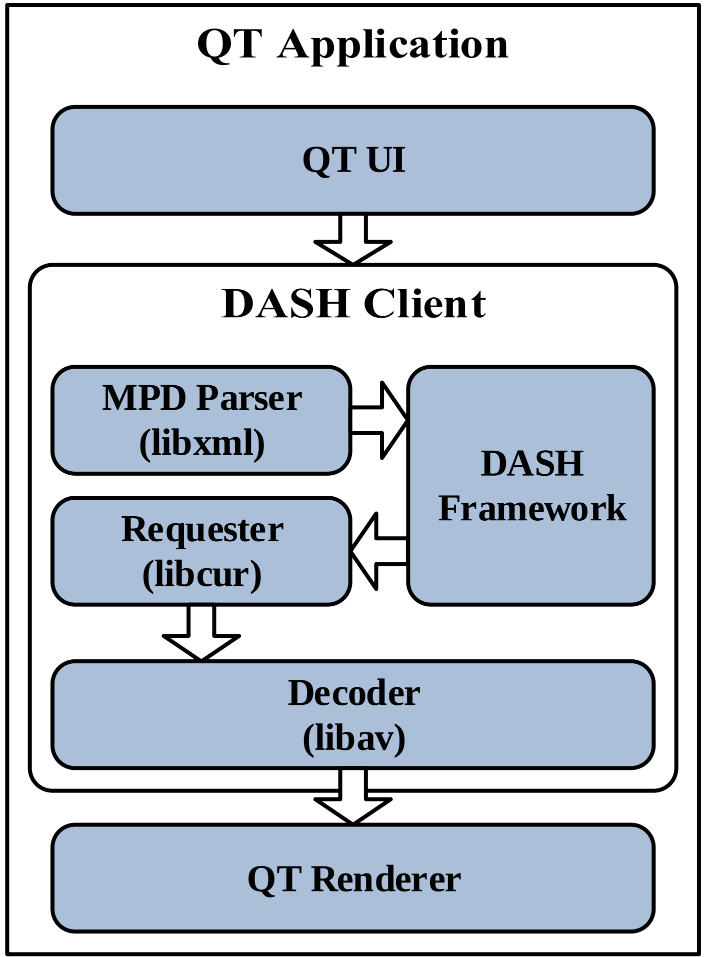
Conclusion and Future Work

MPEG-DASH Client

Our DASH Client is based on libdash[1]

- An open-source library
- Port libdash from PC to Android platform

- QT Application
  - Switch from QT to Android GUI
  - Design user interface for android
- MPD parser doesn't support SVC format
- Request segments from one of the multiple streams
- Extractor
  - ISOBMFF parser
  - Integrate with decoder



[1] https://github.com/bitmovin/libdash

Switch from QT to Android UI

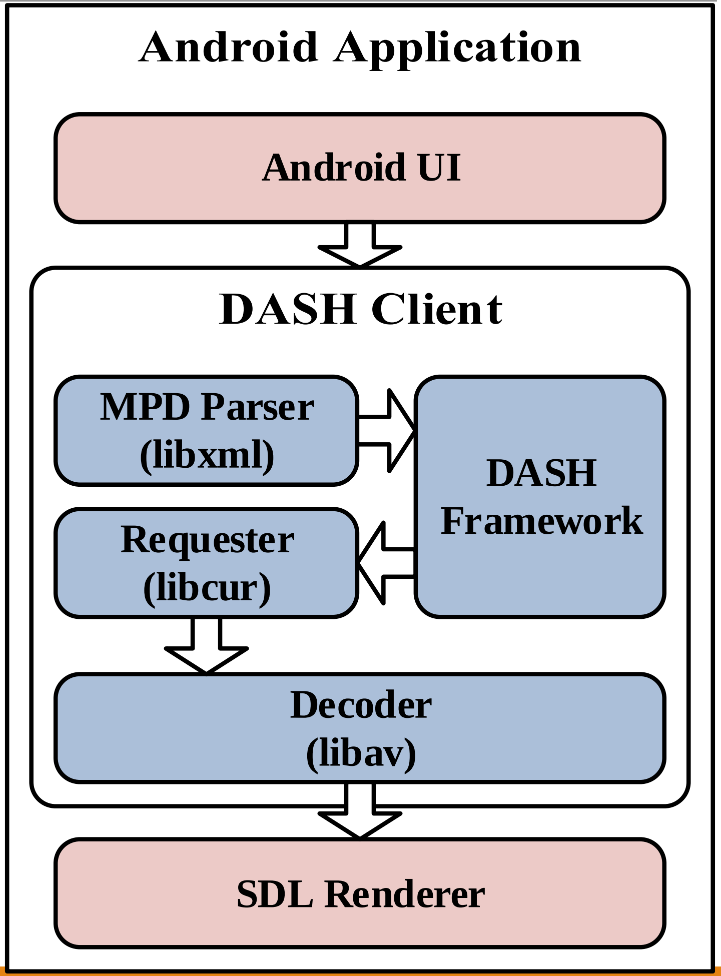
libdash is highly dependent on QT library

QT User Interface

- Switch QT UI to Android UI
- Design interface between UI and DASH Client

QT Renderer

- Replace QT Renderer with SDL Renderer
- Design interface between Decoder and Renderer



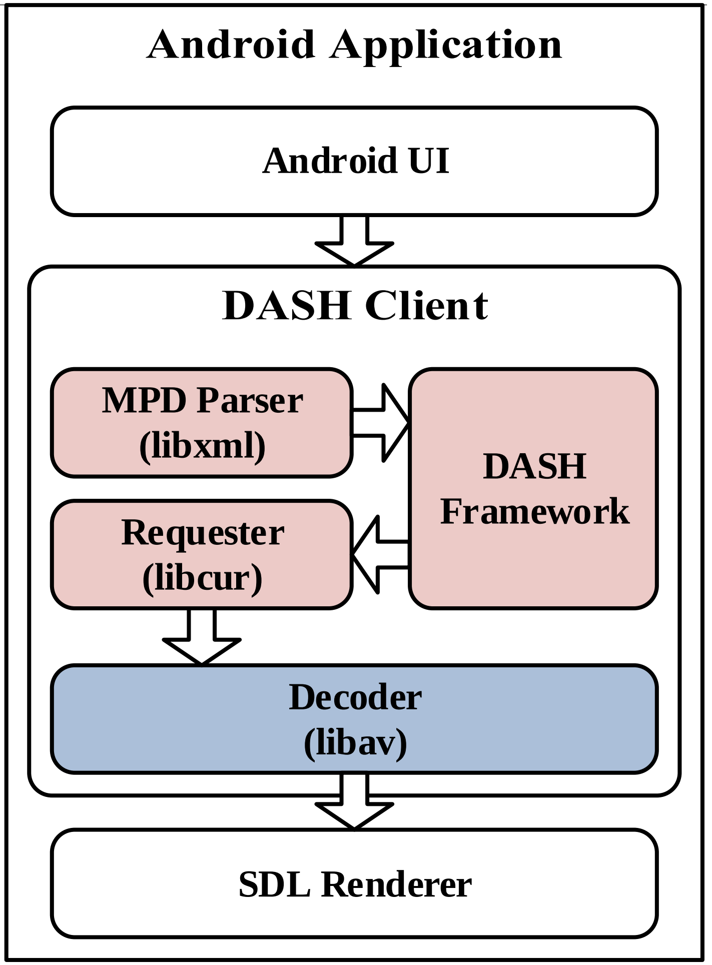
Make libdash Supportable to SVC

libdash doesn't support MPD structure for SVC

- This format is defined by MPEG
- Modify MPD parser to support this format

Multi-layer requester

- libdash only requests segments from one of the streams
- Support simultaneously requesting segments from multiple layers



Extractor Implementation

Segment is ISOMBFF

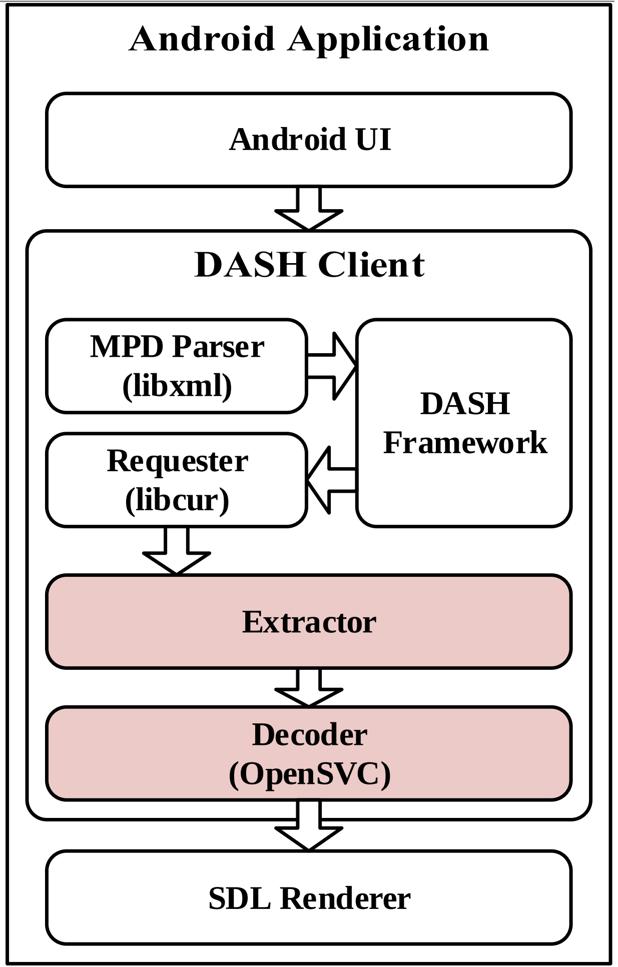
- Segment can’t be directly decoded

Parse ISOBMFF

- Trace GPAC source code
- Implement simple Extractor according to GPAC
- Remove boxes’ header and obtain media data

Reconstruction

- The dependency layers segments need to be reconstructed
- Decoder is able to decode reconstructed media data



Outline

Introduction

Background
- SVC
- MPEG-DASH

System Architecture

Implementations
- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

Experiments
- **Multi-Core SVC Decoder**
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

Experiment Setup for Multi-Core SVC Decoder

Video configuration

- Five HD videos: *doc*, *jeux*, *soap*, *sport*, and *talk*
- GOP: 16 frames
- Three spatial layers: 960x544, 480x272, and 240x144
- Repeat 5 times

Experiment Devices

1) Dual-core tablet with 1.4GHz CPU, 1 GB memory, and 1280x800 screen
2) Quad-core smart phone with 1.5 GHz CPU, 1 GB memory, and 1280x720 screen

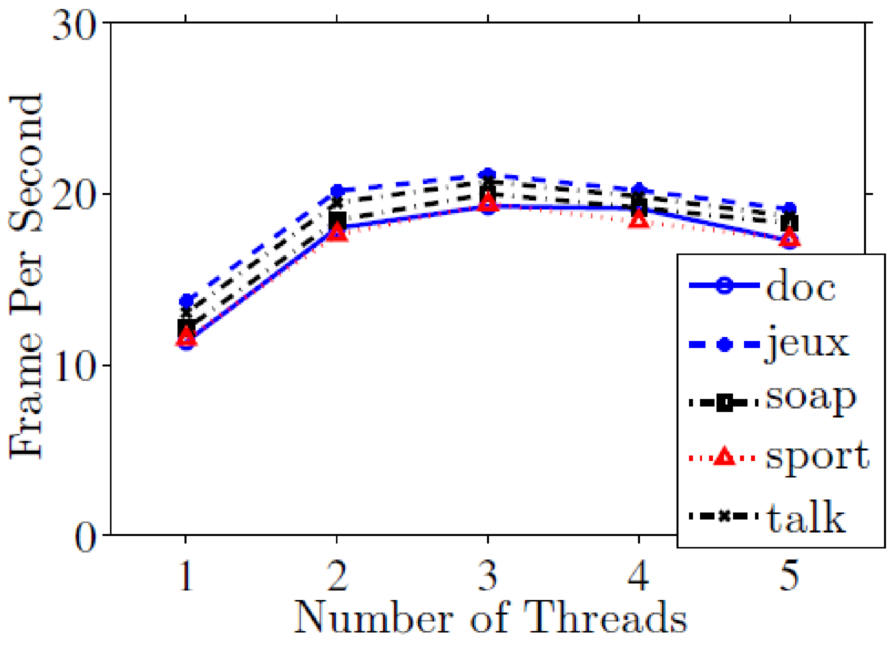
Video Description

Video	Description
`doc`	a documentary video talking about a woman who lost her house
`jeux`	a live show video about the guessing games
`soap`	an action style soap video
`sport`	a sports news video including volleyball, basketball, swimming, etc.
`talk`	a talk show video

Performance of Multi-core SVC Decoder

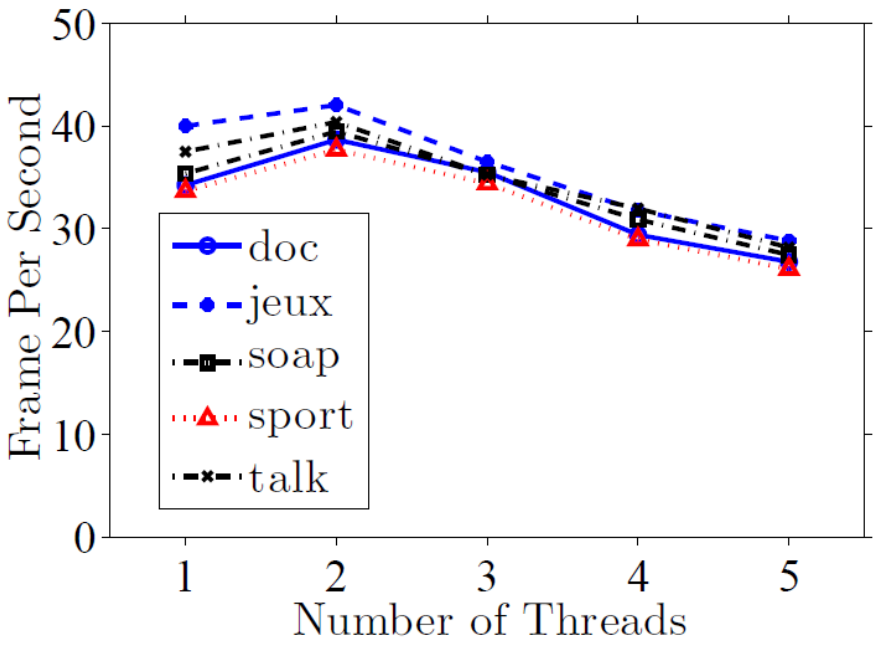
Using 960x544 spatial layer

- **Best FPS with 3 decoder threads**

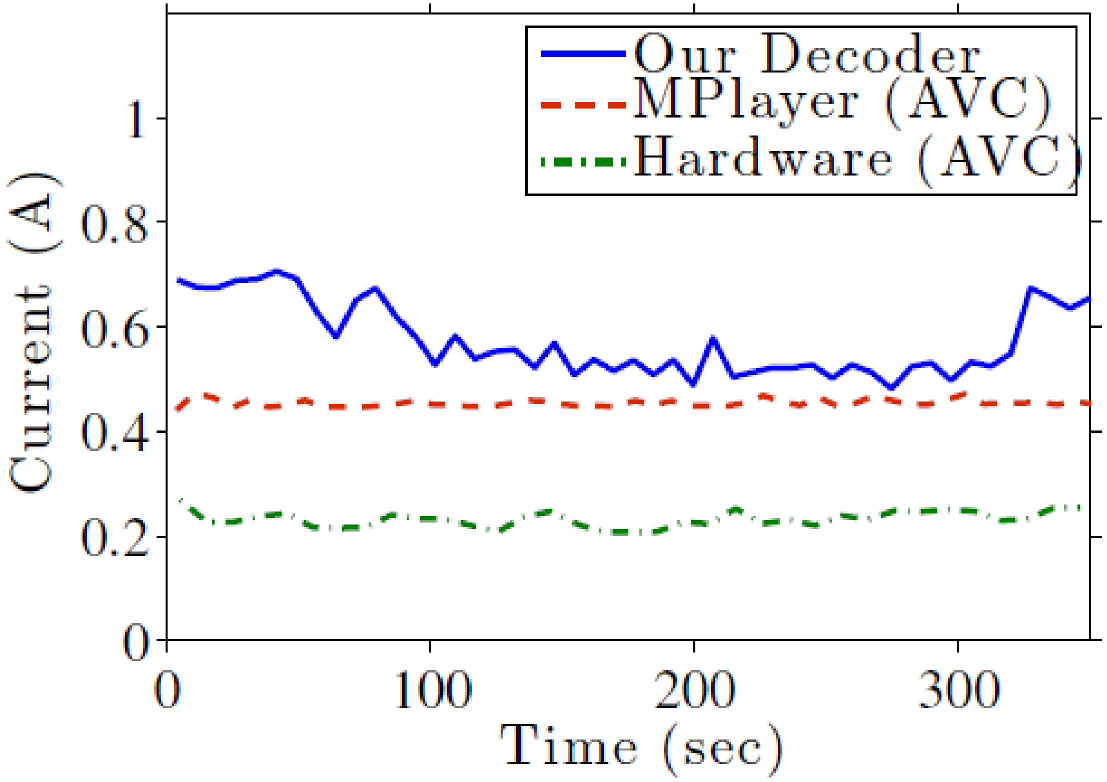


Using 480x272 spatial layer

- **Best FPS with 2 decoder threads**



Power Consumption



Use Agilent 66321D mobile communications DC

We measure the power consumption of our decoder, mplayer, and hardware decoder on the smart phone

Our SVC decoder only incurs small power overhead, **as low as 7%**, compared to mplayer

Outline

Introduction

Background
- SVC
- MPEG-DASH

System Architecture

Implementations
- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

Experiments
- Multi-Core SVC Decoder
- **MPEG-DASH with SVC Streaming**

Conclusion and Future Work

MPED-DASH with SVC Streaming Setup

Video Configuration

- Five HD videos: *doc*, *jeux*, *soap*, *sport*, and *talk*
- GOP size: 16 frames
- Each segment has 8 GOPs
- Three spatial layers: 320x180, 640x360, and 1280x720
- Repeat 5 times

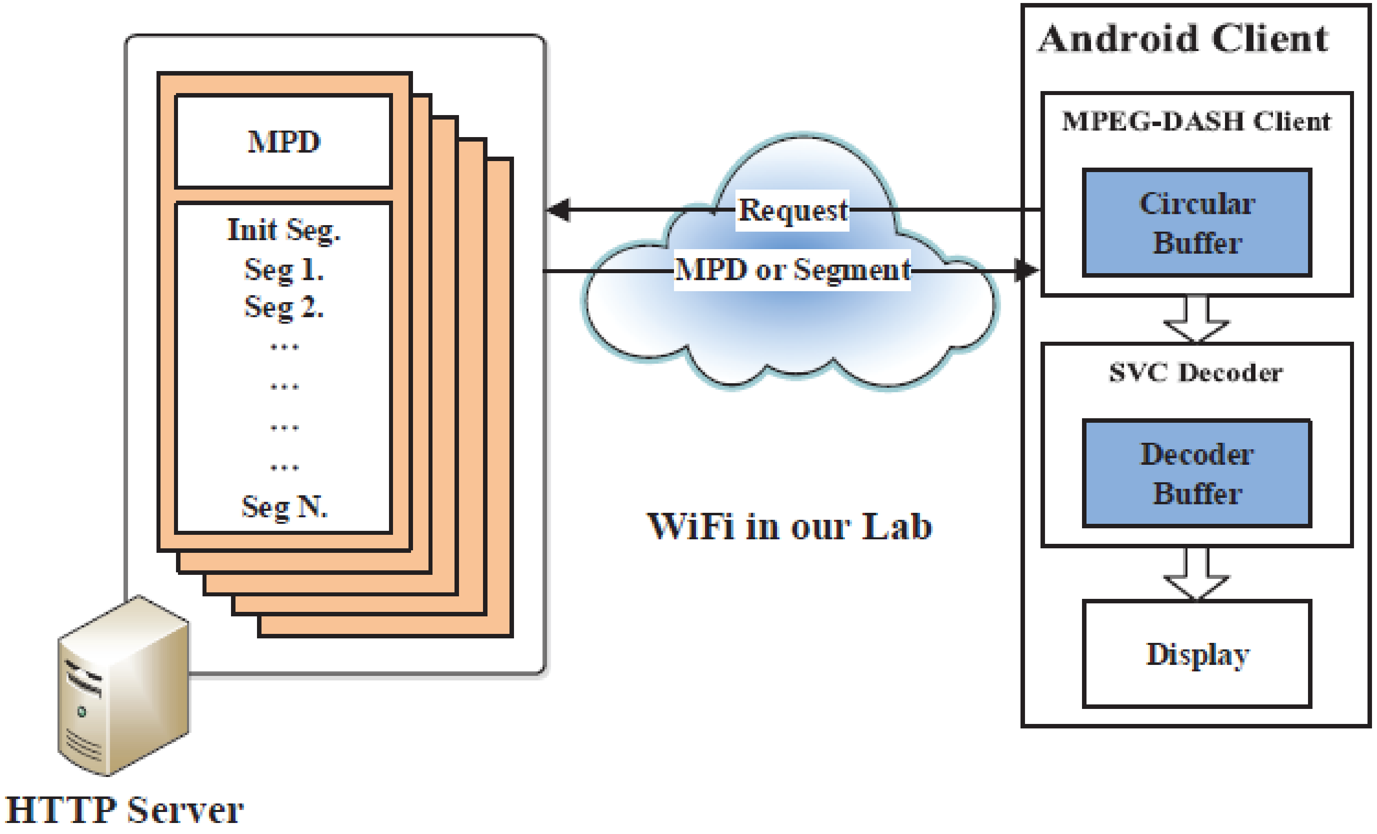
Experiment Device

- Quad-core smart phone with 1.5 GHz CPU, 1 GB memory, and 1280x720 screen

Performance Metrics

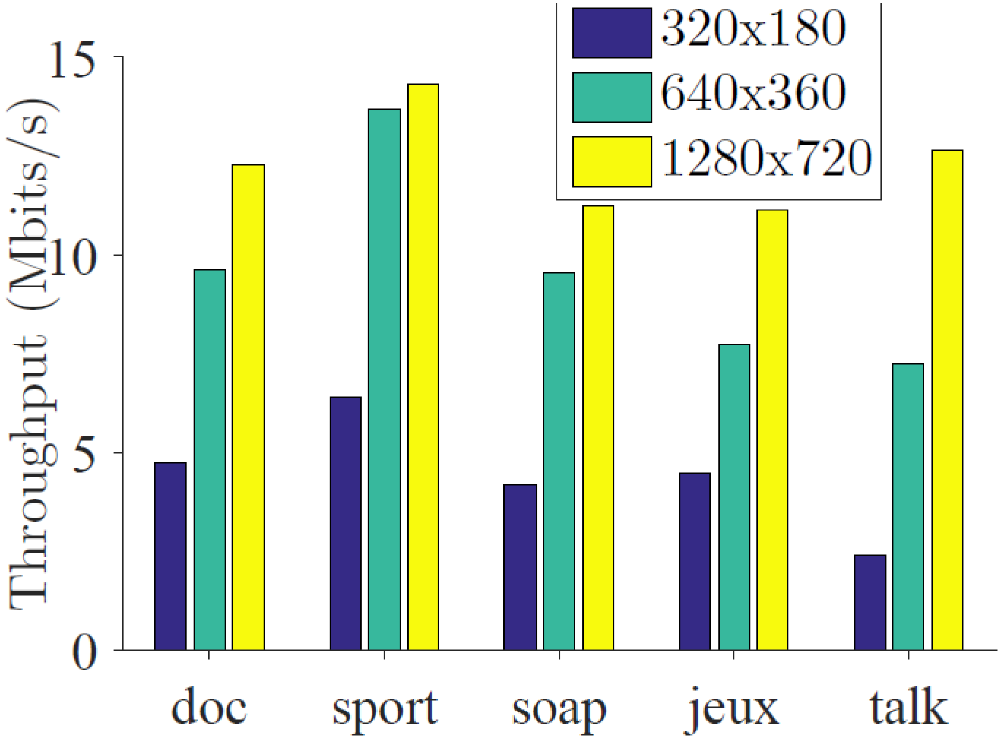
- Frame Per Second (FPS)
- Throughput

Architecture of Experiment Setup



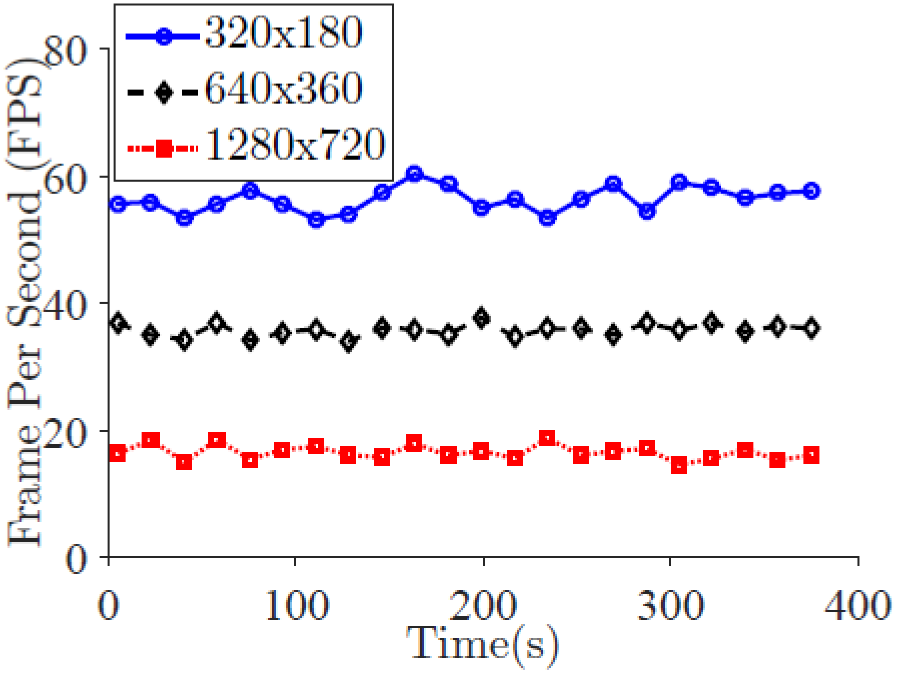
Throughput of MPEG-DASH Client

MPEG-DASH Client provides high bandwidth requirement streaming

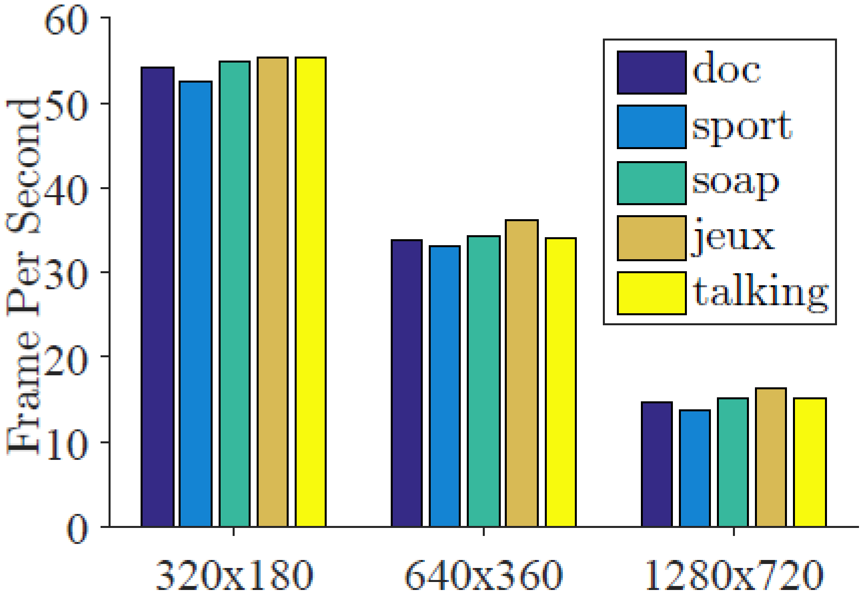


FPS Performance of SDL Renderer

Decoder achieves **50+**, **30+**, and **15+** FPS for 320x180, 640x360 and 1280x720, respectively



Instantaneous sample points of sport



Outline

Introduction

Background
- SVC
- MPEG-DASH

System Architecture

Implementations
- Multi-Core SVC Decoder on Android
- MPEG-DASH with SVC Decoder on Android

Experiments
- Multi-Core SVC Decoder
- MPEG-DASH with SVC Streaming

Conclusion and Future Work

Conclusion

We implemented the pioneering software-based H.264/SVC decoder for Android mobile device

- As low as 7% power consumption overhead compared to mplayer
- Leverage multi-threading to enhance decoding performance

We also integrate MPEG-DASH standard and SDL library into our client

The experimental results demonstrate the benefits in real testbeds

- DASH client achieves high throughput and supports high bandwidth requirement streaming
- SVC decoder achieves 50+ FPS and 30+ FPS for 320x180 and 640x360 streaming, respectively

Future Work

Our testbed can be used for large-scale user study to provide flexible Quality of Experience (QoE) model

- To Dynamically select video quality according to
  - Current bandwidth, delay jitter, packet loss rate, buffer state, etc.
- To choose the best quality for user
  - PNSR, Mean Opinion Score (MOS)

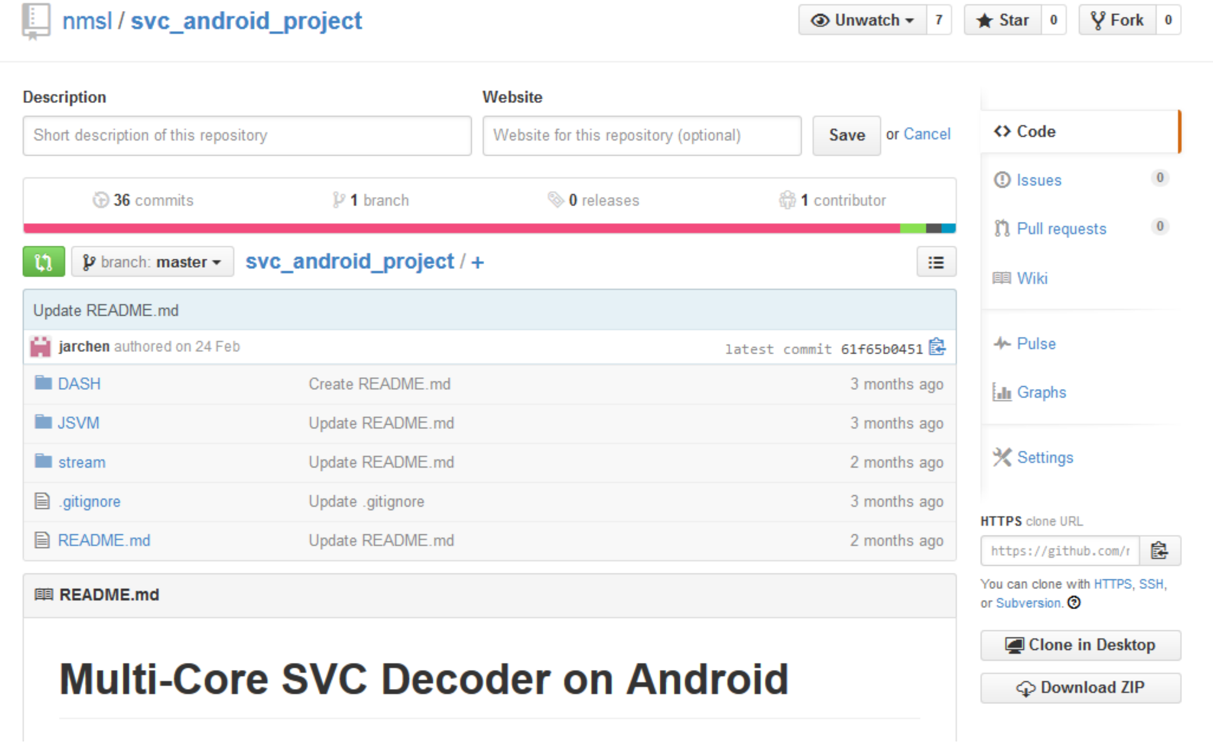
Enhance our SVC decoder

- Use NEON instructions to speed up decoding
- Design more efficient multi-threading decoding structure

Leverage more efficient scalable video coding standards such as H.265/SHVC

Our SVC Decoder is Public on Github

Available on: https://github.com/nmsl/svc_android_project



nmsl / svc_android_project

Description: Short description of this repository

Website: Website for this repository (optional)

36 commits | 1 branch | 0 releases | 1 contributor

branch: master | svc_android_project / +

Update README.md

jarchen authored on 24 Feb | latest commit 61f65b0451

DASH	Create README.md	3 months ago
JSVM	Update README.md	3 months ago
stream	Update README.md	2 months ago
.gitignore	Update .gitignore	3 months ago
README.md	Update README.md	2 months ago

HTTPS clone URL: https://github.com/r

You can clone with HTTPS, SSH, or Subversion.

README.md

Multi-Core SVC Decoder on Android

End