ADAPTIVE STREAMING AT vimeo

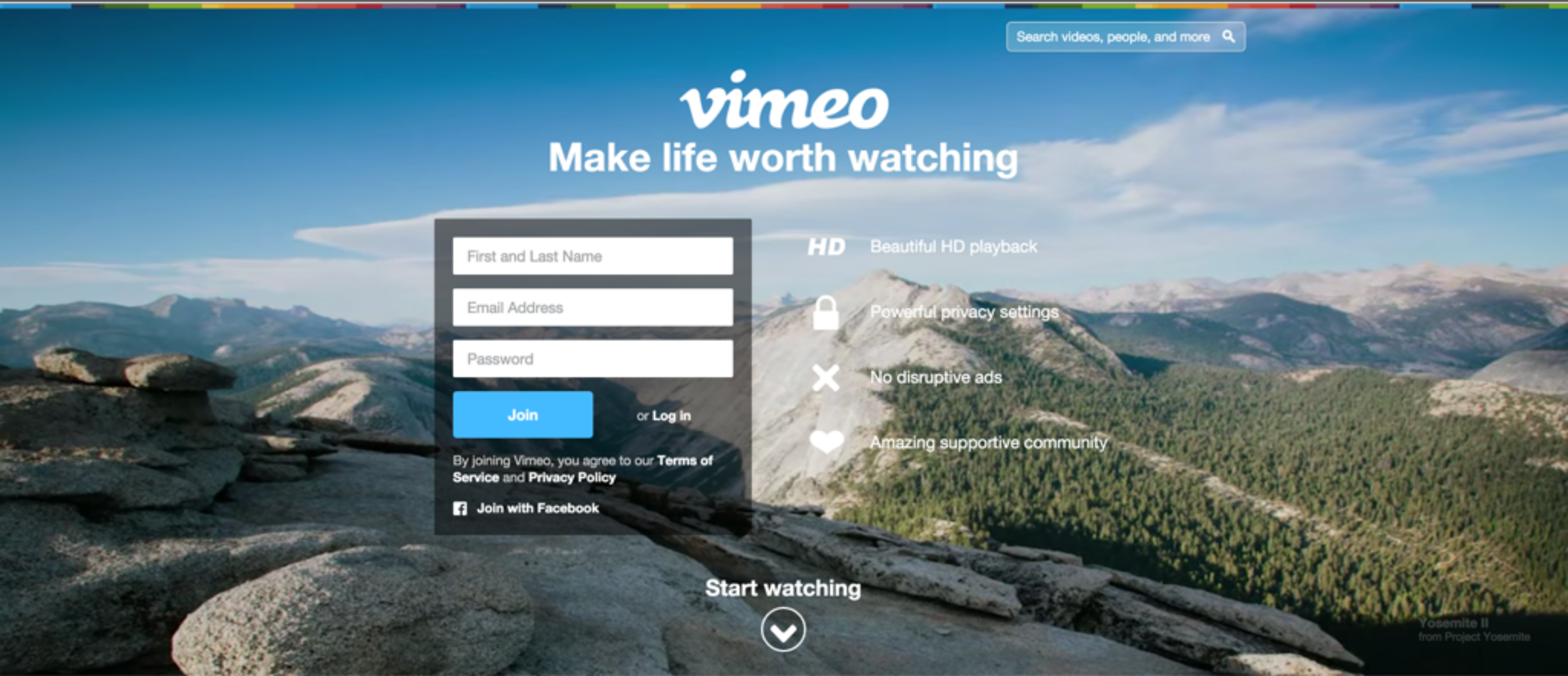
Justin Ruggles

Lead Engineer, Transcoding & Delivery

justinr@vimeo.com

ABOUT VIMEO

- Video hosting platform, founded in 2004, that allows creators to share their content in high quality, free of advertising
- Paid subscriptions give creators more storage and advanced tools
- Creators can sell their content, at any price, through Vimeo On Demand



HISTORY OF VIDEO DELIVERY AT VIMEO

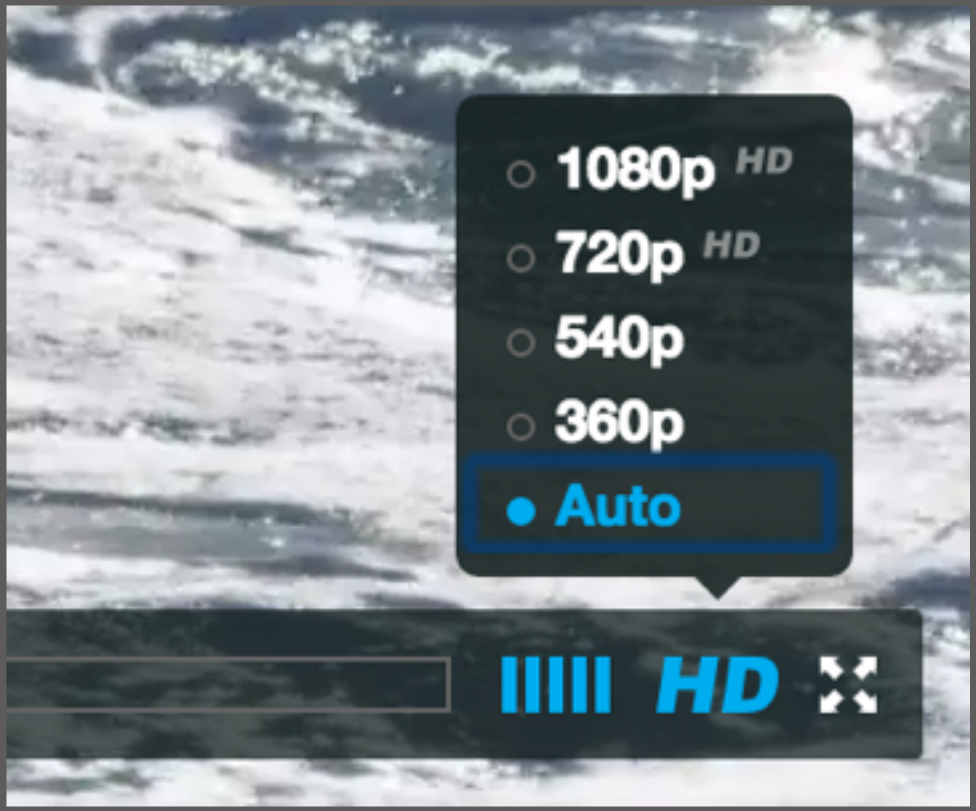
- Flash
  - FLV with VP6 video and MP3 audio (2004)
  - MP4 progressive “pseudo-streaming” with H.264 and AAC (2011)
- Apple’s HLS adaptive streaming for iPhone
  - Added a custom profile for iPhone 3G (2009)
  - CDN-provided on-the-fly HLS packaging from MP4 source
- HTML5 progressive MP4 (2010)
  - Moved away from Flash as browsers added support
- MPEG-4 DASH adaptive streaming (2015)
  - Using our own packager in front of multiple CDNs
  - Up to 10 profiles: 360p, 540p, 720p30, 720p60, 1080p30, 1080p60, 2K30, 2K60, 4K30, 4K60

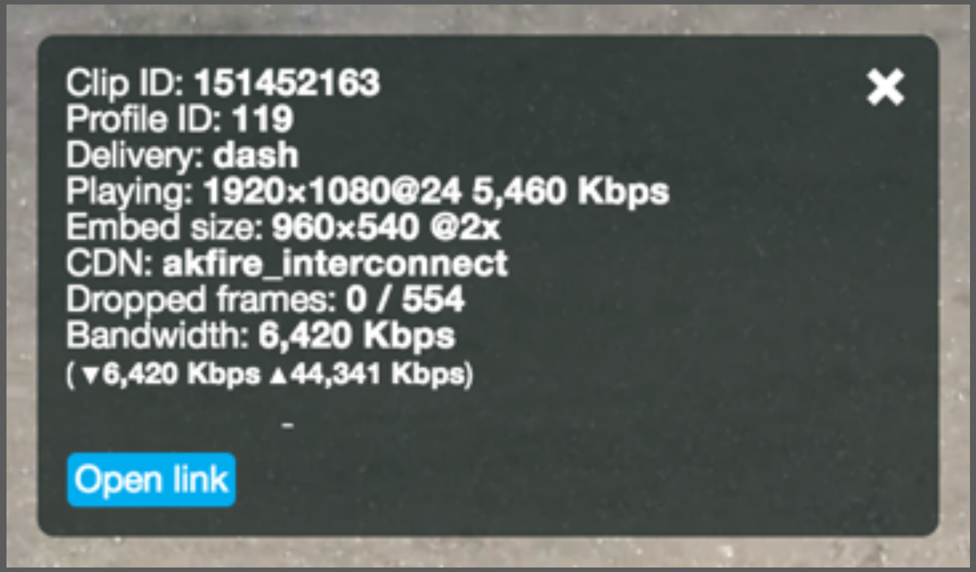
SKYFIRE

- At the end of 2014, the Video Group kicked off the Skyfire project to bring DASH to our player and write our own on-the-fly packaging backend
- The player needed to add MediaSource Extensions (MSE) support, implement an adaptive bitrate switching algorithm, and add a quality selector menu
- The backend needed to have a playlist server and video segmenter written from the ground up
- To facilitate faster delivery and multiple CDNs, this project also included migrating our storage from Akamai NetStorage to Google Cloud Storage

PLAYER

- Front-end code was refactored to be more modular and ported to ES6
- media-sorcerer : Simplified MediaSource API wrapper
- skyfire.js : Player code for DASH playback
  - MSE interaction
  - Adaptive stream switching
- Added HLS support to our Flash player
- Quality Selector
  - Allows users to select a specific quality
  - Defaults to Auto for adaptive switching

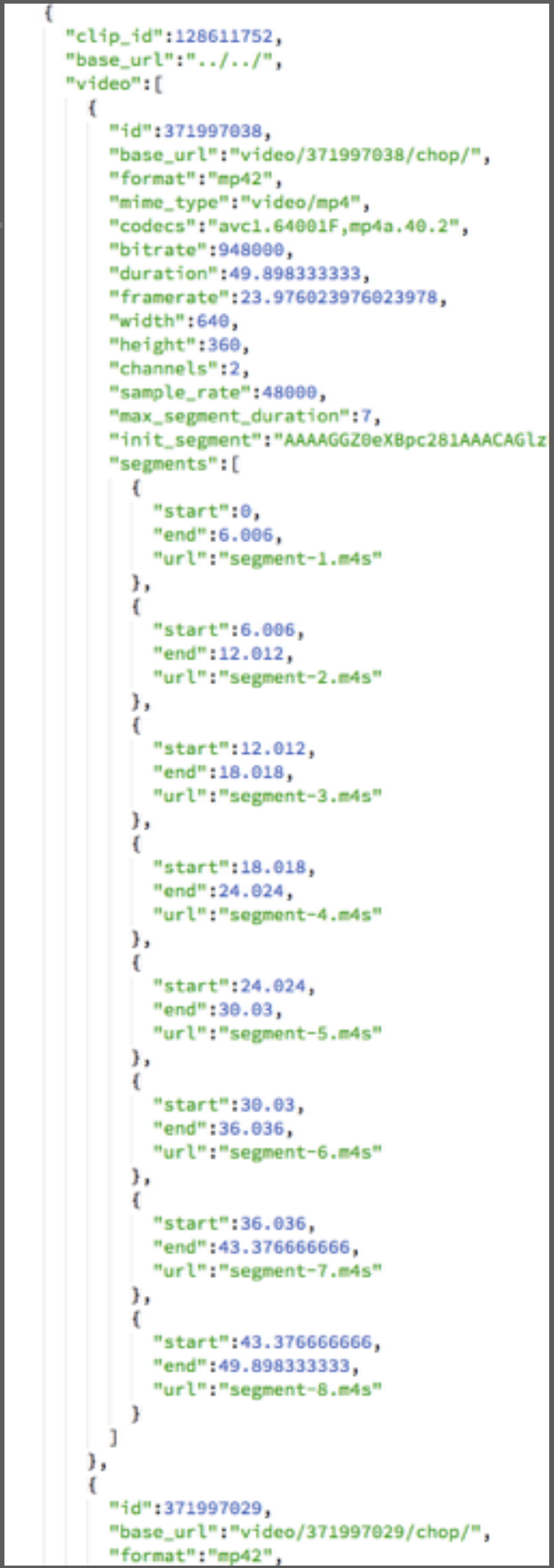



Clip ID: 151452163
Profile ID: 119
Delivery: dash
Playing: 1920×1080@24 5,460 Kbps
Embed size: 960×540 @2x
CDN: akfire_interconnect
Dropped frames: 0 / 554
Bandwidth: 6,420 Kbps
(▼6,420 Kbps ▲44,341 Kbps)

Open link

PLAYLIST SERVER

- Project Name: Grapple
- m3u8 format for HLS delivery
- JSON format for DASH delivery in Vimeo player
  - Custom format for simpler player integration and easy readability
  - Init segments are base64-encoded in the playlist to reduce the number of client calls
- MPD format for DASH delivery in 3rd-party players
  - Tested in dash.js, Shaka player, JW Player, bitdash, and MS Edge browser
- Supports separate or interleaved audio and video



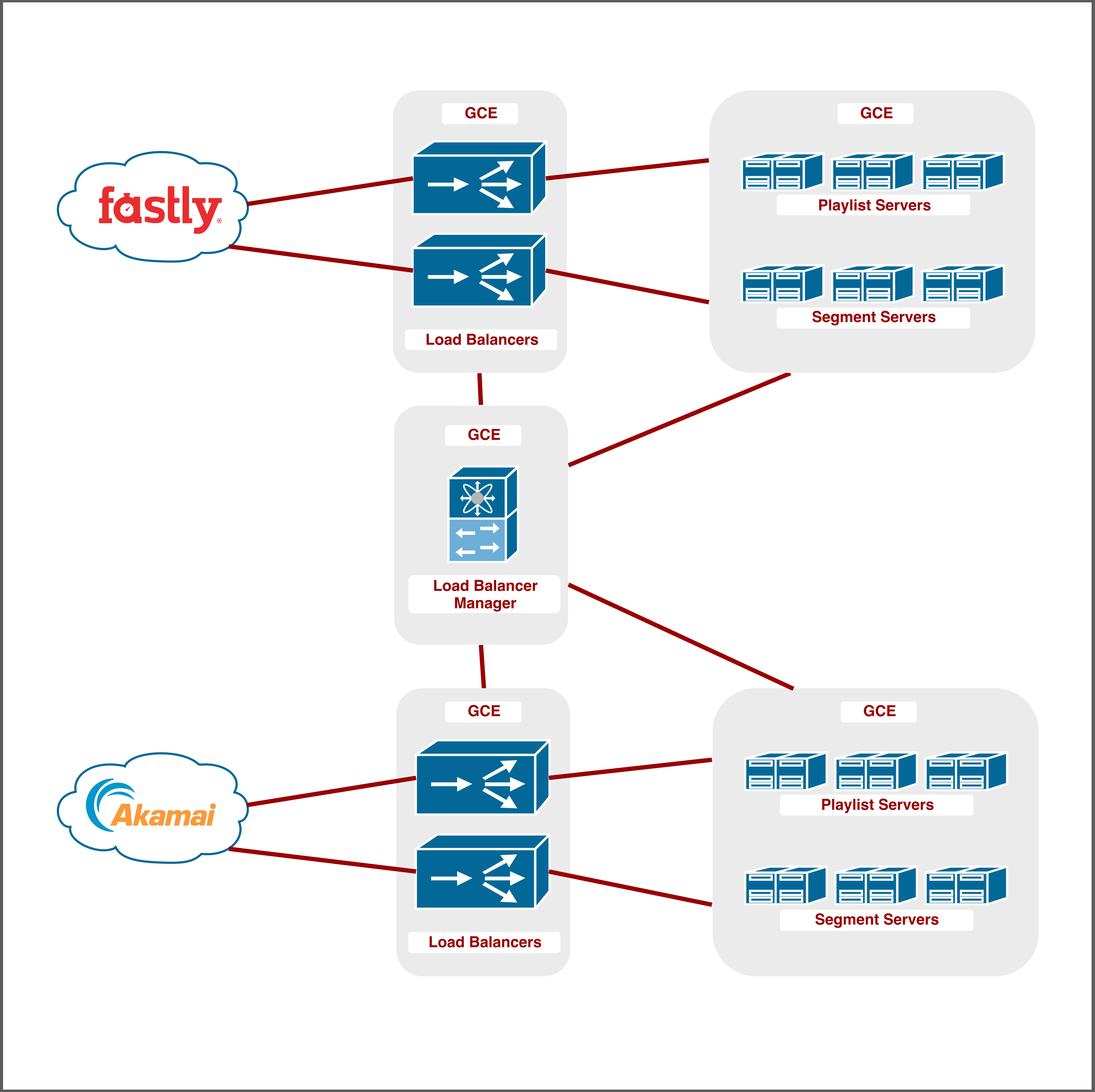
```
{
  "clip_id":128611752,
  "base_url":"../../",
  "video":[
    {
      "id":371997038,
      "base_url":"video/371997038/chop/",
      "format":"mp42",
      "mime_type":"video/mp4",
      "codecs":"avc1.64001F,mp4a.40.2",
      "bitrate":948000,
      "duration":49.898333333,
      "framerate":23.976023976023978,
      "width":640,
      "height":360,
      "channels":2,
      "sample_rate":48000,
      "max_segment_duration":7,
      "init_segment":"AAAAGGZ0eXBpc281AAACAGlz
      "segments":[
        {
          "start":0,
          "end":6.006,
          "url":"segment-1.m4s"
        },
        {
          "start":6.006,
          "end":12.012,
          "url":"segment-2.m4s"
        },
        {
          "start":12.012,
          "end":18.018,
          "url":"segment-3.m4s"
        },
        {
          "start":18.018,
          "end":24.024,
          "url":"segment-4.m4s"
        },
        {
          "start":24.024,
          "end":30.03,
          "url":"segment-5.m4s"
        },
        {
          "start":30.03,
          "end":36.036,
          "url":"segment-6.m4s"
        },
        {
          "start":36.036,
          "end":43.376666666,
          "url":"segment-7.m4s"
        },
        {
          "start":43.376666666,
          "end":49.898333333,
          "url":"segment-8.m4s"
        }
      ]
    },
    {
      "id":371997029,
      "base_url":"video/371997029/chop/",
      "format":"mp42",
```

SEGMENT SERVER

- Project Name: Chop Shop
- On-the-fly segmenter
- Converts progressive MP4 to DASH or HLS
- Each segment request is independent
- Flexible segment duration pattern
- MP4 analysis endpoint used internally and by the playlist server
- Two-tier cache (local + memcache) for analysis results and MP4 global headers

BACKEND ARCHITECTURE

- 2 CDNs
- HAProxy load balancers
- Our own load balancer manager (Highly Available)
- Skyfire servers in several Google Compute Engine autoscaled instance groups
- 15,000 req/s at CDN edges
- 2,500 req/s at the origin servers



OPTIMIZING CHOP SHOP : ORIGINAL IMPLEMENTATION

- FFmpeg : open source libraries for everything multimedia
  - Used the FFmpeg internal HTTP client
  - Makes open-ended byte range requests, similar to browser download for progressive playback
- L-SMASH: open source library for MP4 analysis and manipulation
  - Only supports file or custom I/O
  - Implemented custom HTTP I/O with similar behavior to FFmpeg's HTTP client
  - Used Go HTTP client with cgo callbacks from liblsmash

L-SMASHとは

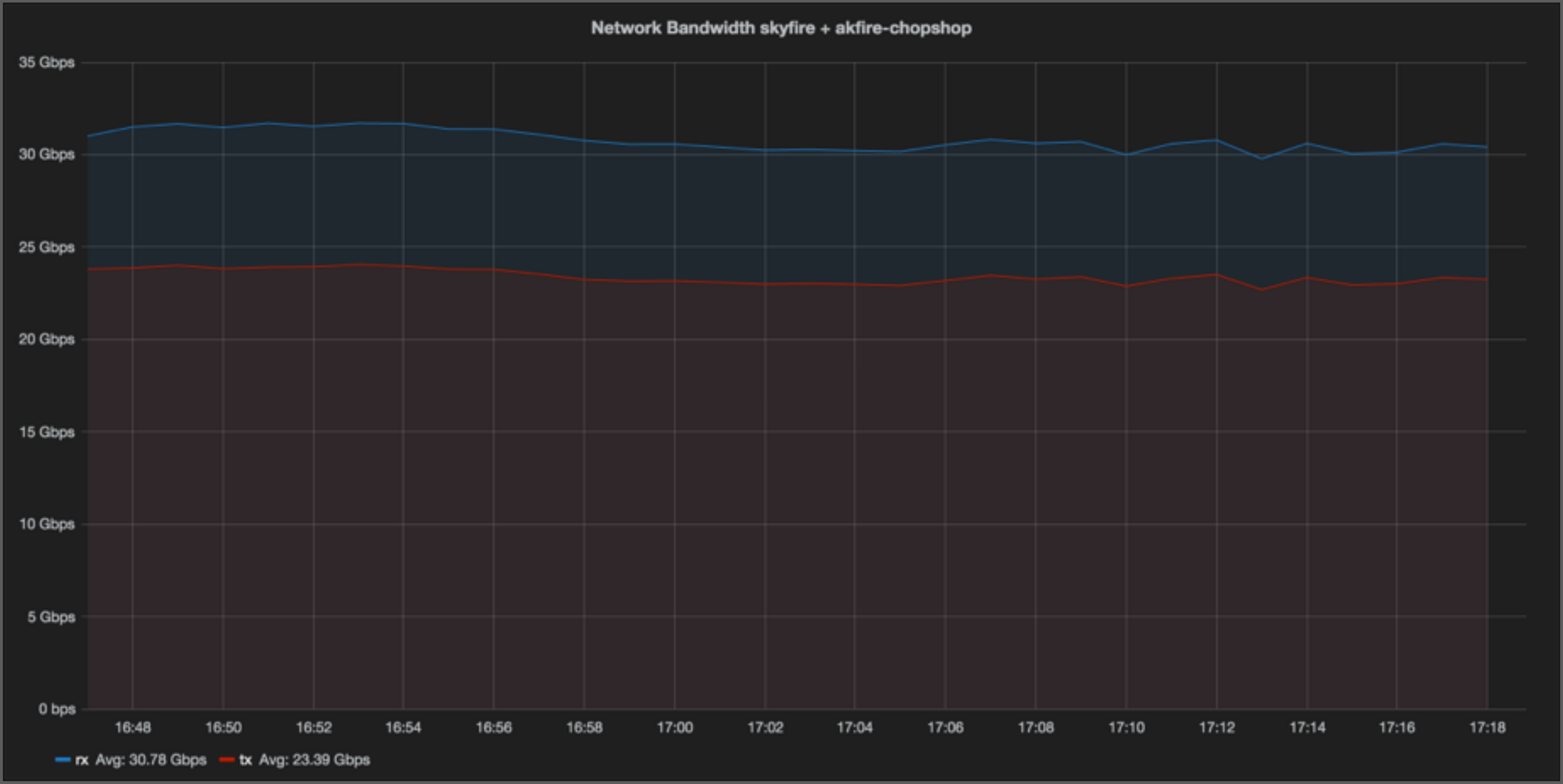
OPTIMIZING CHOP SHOP : ISSUE #1

- Issue : Sub-optimal HTTP requests
- Open-ended byte range requests led to significant overread
- Small average segment size (1 MB) and very fast throughput meant that even closing the connections quickly led to about twice as much data arriving to the client than what was needed by FFmpeg
- Resulted in approx. 2.5X inbound network I/O as compared to outbound

OPTIMIZING CHOP SHOP : ISSUE #1

- Solution : No open-ended byte range requests
- Moved off of FFmpeg's internal HTTP client
  - FFmpeg also allows custom I/O via C callbacks, so we reused the Go HTTP client code from the liblsmash-based analyzer
- Solved overread by using the MP4 analysis data to request exact ranges required by FFmpeg, in parallel
  - Implemented a sparse memory buffer to behave like the full file in I/O callbacks but return 0's for out-of-range data
- Inbound network I/O from GCS is now approx. 1.3X outbound (about even if you account for memcache traffic)

OPTIMIZING CHOP SHOP : ISSUE #1



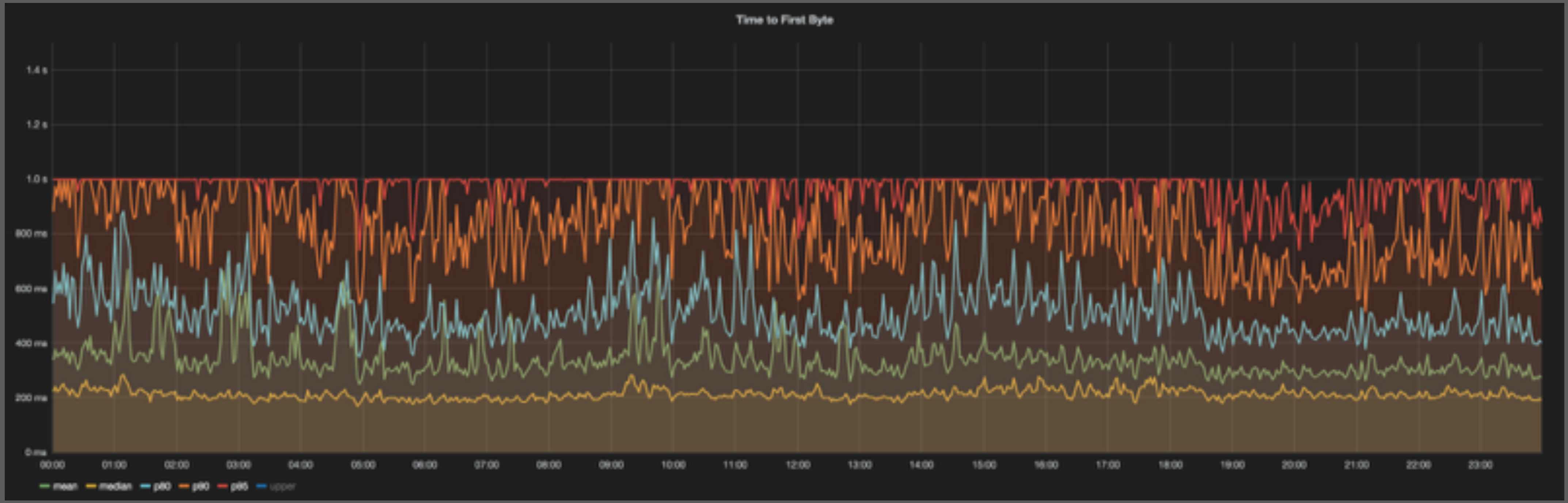
OPTIMIZING CHOP SHOP : ISSUE #2

- Issue : Suboptimal Google Cloud Storage response time
- GCS average and median TTFB was decent (180-200ms) but the tail of the distribution was very poor (5-10 sec or more)
- This was unacceptable for video delivery with 6s segments
- The FFmpeg internal HTTP client has no full request timeout option, no retries on error, and does not report the HTTP response status to the user

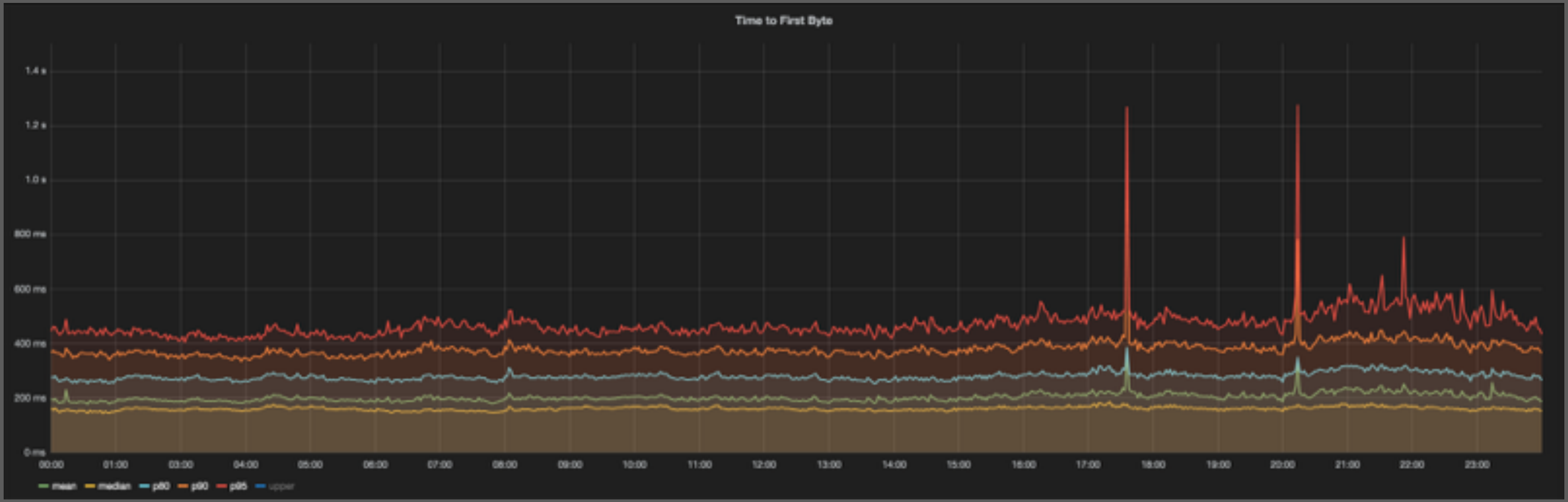
OPTIMIZING CHOP SHOP : ISSUE #2

- Solution : Aggressive timeouts and multiple retries
- We had already moved off of FFmpeg’s internal HTTP client because of the byte range request issue
- We modified our custom HTTP I/O package to have configurable retries with different timeouts for each attempt
    - Original timeouts : 4 attempts - [1s, 1s, 1s, 2s]
    - Changed to [1s, 1s, 2s, 5s] to reduce errors
    - GCS behavior is much more stable, so now just [2s, 5s]

OPTIMIZING CHOP SHOP : ISSUE #2



1 sec timeouts during bad GCS weather



Current behavior