

Clever Monkeys Communicating Discreetly

Will Law

Chief Architect

Akamai Edge Technology Group

October 2022



CMCD is a bridge between the world of **players** and the world of **content delivery**



Player sees a 65min playback session with 2 rebuffering events.

```
•172.232.14.71 r 1604275203.195 - 1 0 0 0 0 359 127.0.0.1 GET
/L/16382/760549/1m30s/mc.xxxxnews-live-
i.akamaihd.net/.redirect./724634/724634/ts/xxxxNLive1/20201008T111910/master_12118/m
aster_1200_ts/1468?file-type=ts&pt-delivery-format-qs=hls 200 - mjeJV - ipa.xxxxnews-live-
i.akamaihd.net - - - - - 1 - 348 - - 0 0 - 698f523 16382. - - - - - 90 - 116 116 22666 - - - - OW
- r7f9bcdd84bc0.265801729.p.g 464 TLj8 - 8a935940 - - - - 127.0.0.1 622 - - - - -
mtse=g|265801729;aid=490621;pin=TLj-;netp=;tff=0 - - - 706:20110:0:0 L1:NS:i:NS:s -
vcd=918;cpu=//thd@pht6/959 - - - - - 79 620 - -
•172.232.14.71 r 1604275209.331 - 1 0 0 0 0 359 127.0.0.1 GET
/L/16382/760549/1m30s/mc.xxxxnews-live-
i.akamaihd.net/.redirect./724634/724634/ts/xxxxNLive1/20201008T111910/master_12118/m
aster_1200_ts/1469?file-type=ts&pt-delivery-format-qs=hls 200 - mjAeJV - ipa.xxxxnews-live-
i.akamaihd.net - - - - - 1 - 348 - - 0 0 - 698fef9 16382. - - - - - 90 - 116 116 22666 - - - - OW
- r7f9bcdd84bc0.265801729.p.g 464 TLj8 - 8a935940 - - - - 127.0.0.1 652 - - - - -
mtse=g|265801729;aid=490621;pin=TLj-;netp=;tff=0 - - - 948:21420:0:0 L1:NS:i:NS:s -
vcd=918;cpu=//thd@pht6/1040 - - - - - 121 683 - -
```

CDN logs show 1,256 requests for various binary objects.



CDNs are blind to the media characteristics of each request

HTTP/1.1 GET /customer/segment3246.m4s

- Is this a 2s or a 6s media segment?
- Is this HLS or DASH?
- Is this at startup or steady state?
- What is the bitrate of this object?
- Is it a video or audio segment?
- Live or VOD stream?
- Is this player healthy or is it rebuffering?
- If this file is delivered in 2000ms, is that good, average or terrible performance?





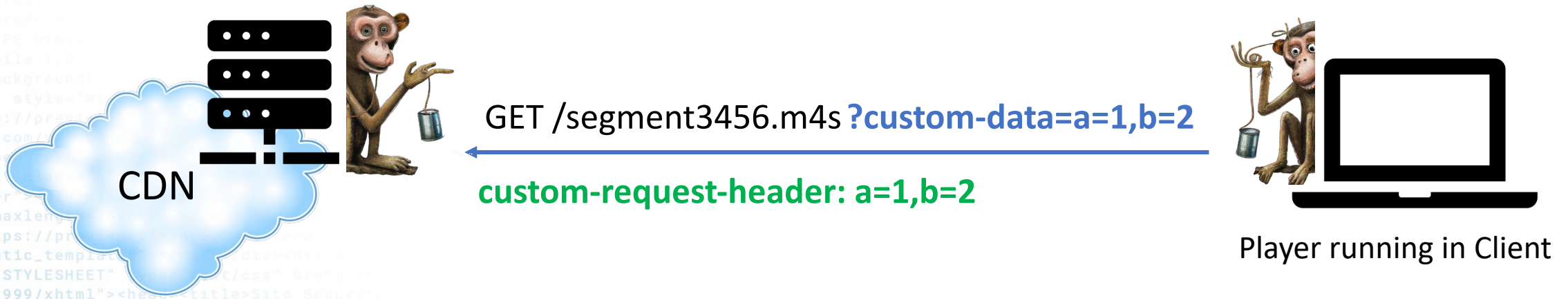
From the player's perspective

- Why doesn't the CDN know there is poor performance in a certain network before I do?
- How do I tie together my QoE metrics (Convivia, NPAW, Mux etc) with my CDN logs?
- How do I consistently inform all CDNs to prefetch the next media segment?
- CDN tells us we had "23,456 hits" but do those hits represent 10, 100 or 10,000 videos being played?



So there is a need for media players and CDNs to be able to exchange mutually beneficial information with one another.

- How to do it?
- Luckily, players already exchange messages every few seconds with each CDN, as they request playlists and media segments.



Who did the work?



- CTA WAVE project

- Akamai
- Apple
- BBC
- Comcast
- Disney Streaming Services
- Dolby
- Ericsson
- Eurofins
- Fraunhofer
- Harmonic
- Hulu
- JW Player
- Qualcomm
- Sony

Consumer
Technology
Association™

CTA Specification

Web Application Video Ecosystem - Common
Media Client Data

CTA-5004

September 2020



Common Media Client Data is ...

- A defined set of structured key/value pairs
- Communicating mutually beneficial media-related information
- From a player to a CDN via either
 - A set of custom headers
 - A query arg
 - A JSON object
- Common because the same data structure can be used across all players and all CDNs.
- Available for free at <https://tinyurl.com/cta5004spec>



What keys are sent?

- Encoded bitrate - **br**
- Buffer length - **bl**
- Buffer starvation - **bs**
- ContentID - **cid**
- Object duration - **d**
- Deadline - **dl**
- Measured throughput - **mtp**
- Next object request - **nor**
- Next range request - **nrr**
- Object type - **ot**
- Playback rate - **pr**
- Requested max tput - **rtp**
- Streaming format - **sf**
- Session ID - **sid**
- Stream type - **st**
- Startup - **su**
- Top bitrate - **tb**
- Version - **v**



What do these look like crossing the wire?



HEADER

```
CMCD-Request mtp=25400
CMCD-Object  r=3200,d=4004,ot=v,tb=6000
CMCD-Status  s,rtp=15000
CMCD-Session sid="6e2fb550-c457-11e9-bb97-0800200c9a66"
```

QUERY ARG

```
?CMCD=br%3D3200%2Cbs%2Cd%3D4004%2Cmtp%3D25400%2Cot%3Dv%2Crtp%3D15000%2Csid%3D%226e2fb550-c457-11e9-bb97-0800200c9a66%22%2Ctb%3D6000
```

JSON

```
{"br": 3200, "bs": true, "d": 4004, "mtp": 25400, "ot": "v",
"rtp": 15000, "sid":
"6e2fb550-c457-11e9-bb97-0800200c9a66", "tb": 6000}
```



Use-case #1: Prefetching content with non-monotonically increasing segment numbers

#EXTM3U

.....

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/00_000.mp4

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/08_007.mp4

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/16_016.mp4

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/24_023.mp4

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/32_032.mp4

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/40_039.mp4

#EXTINF:8.008,

2c8846c2-eb7a-4ae9-a32e-e9120803172c/5f84adac9b/48_048.mp4

.....

CMCD-Request:nor="08_007.mp4"

CMCD-Request:nor="16_016.mp4"

CMCD-Request:nor="24_023.mp4"

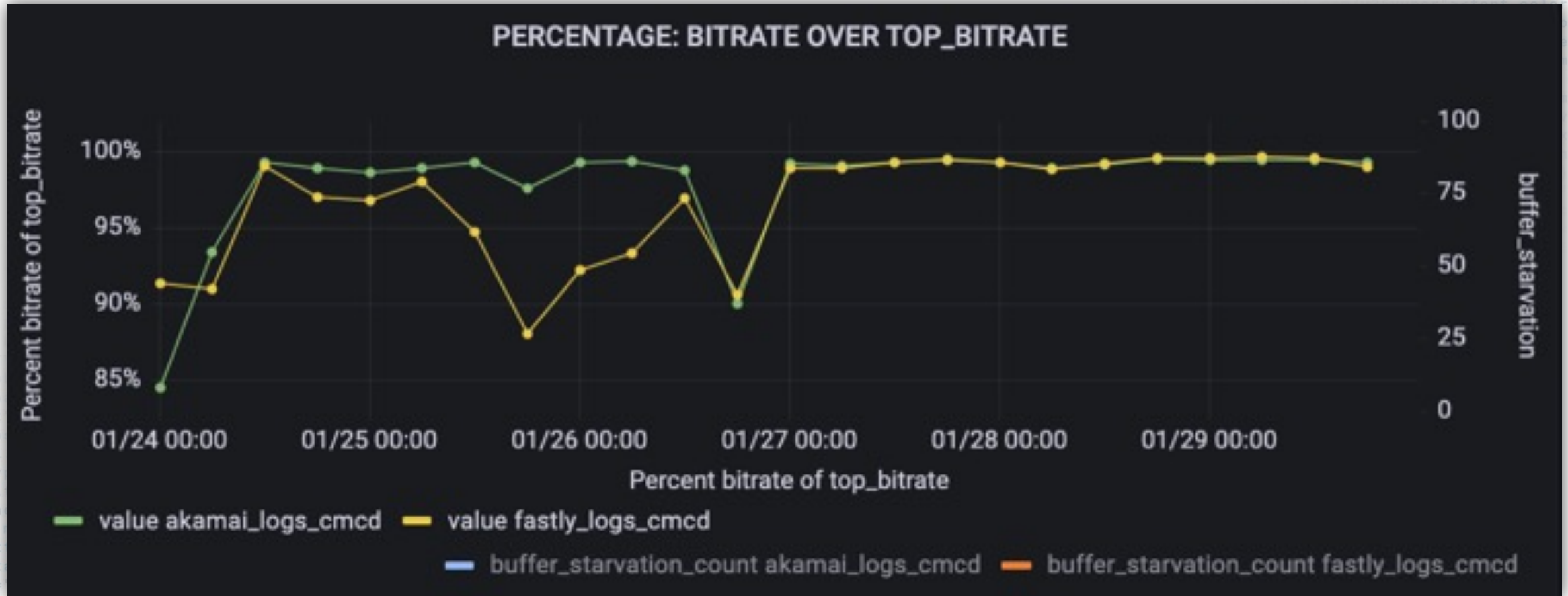
CMCD-Request:nor="32_032.mp4"

CMCD-Request:nor="40_039.mp4"

CMCD-Request:nor="48_048.mp4"

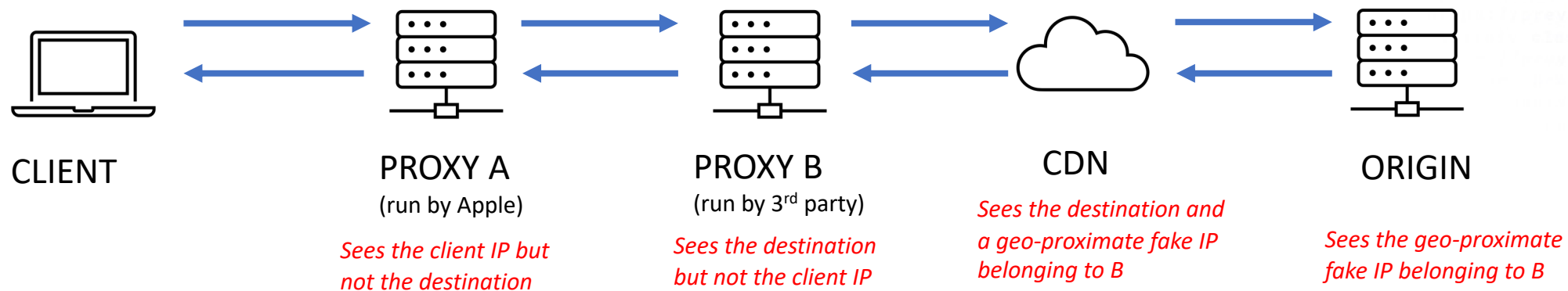
CMCD-Request:nor="56_055.mp4"

Use-case #2: Multi-CDN Client steering use-case:



Example from CBS/Viacom/Paramount CMCD monitoring

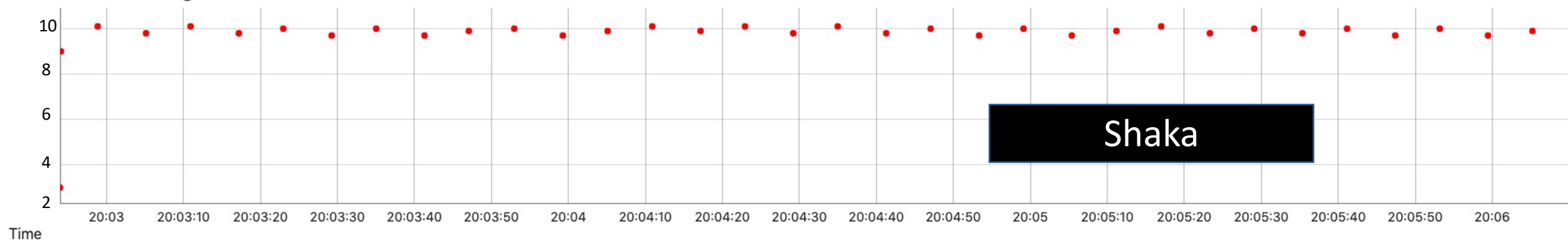
Use-case #3: Session identification through Apple's Private Relay



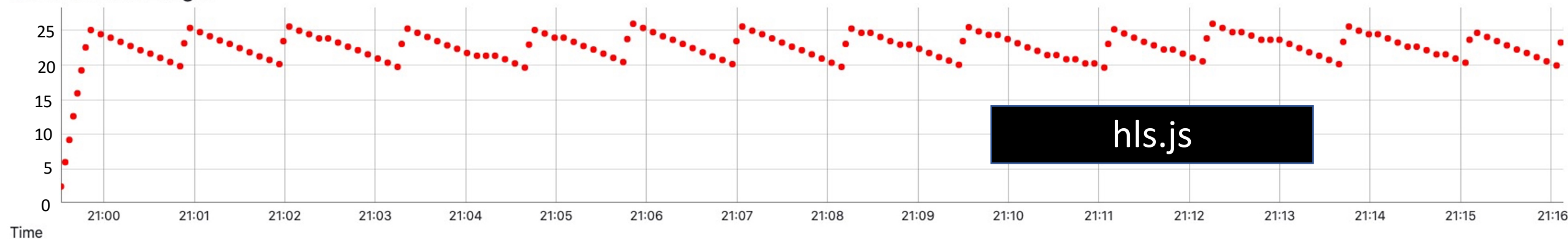
1. Apple's Private Relay intentionally obscures the client IP from the CDN and origin that is serving the content.
2. Using the combination of clientIP/user-agent to identify sessions in CDN will break.
3. Using CMCD sid parameter solves this problem.

Use-case #4: Get real time views into player health

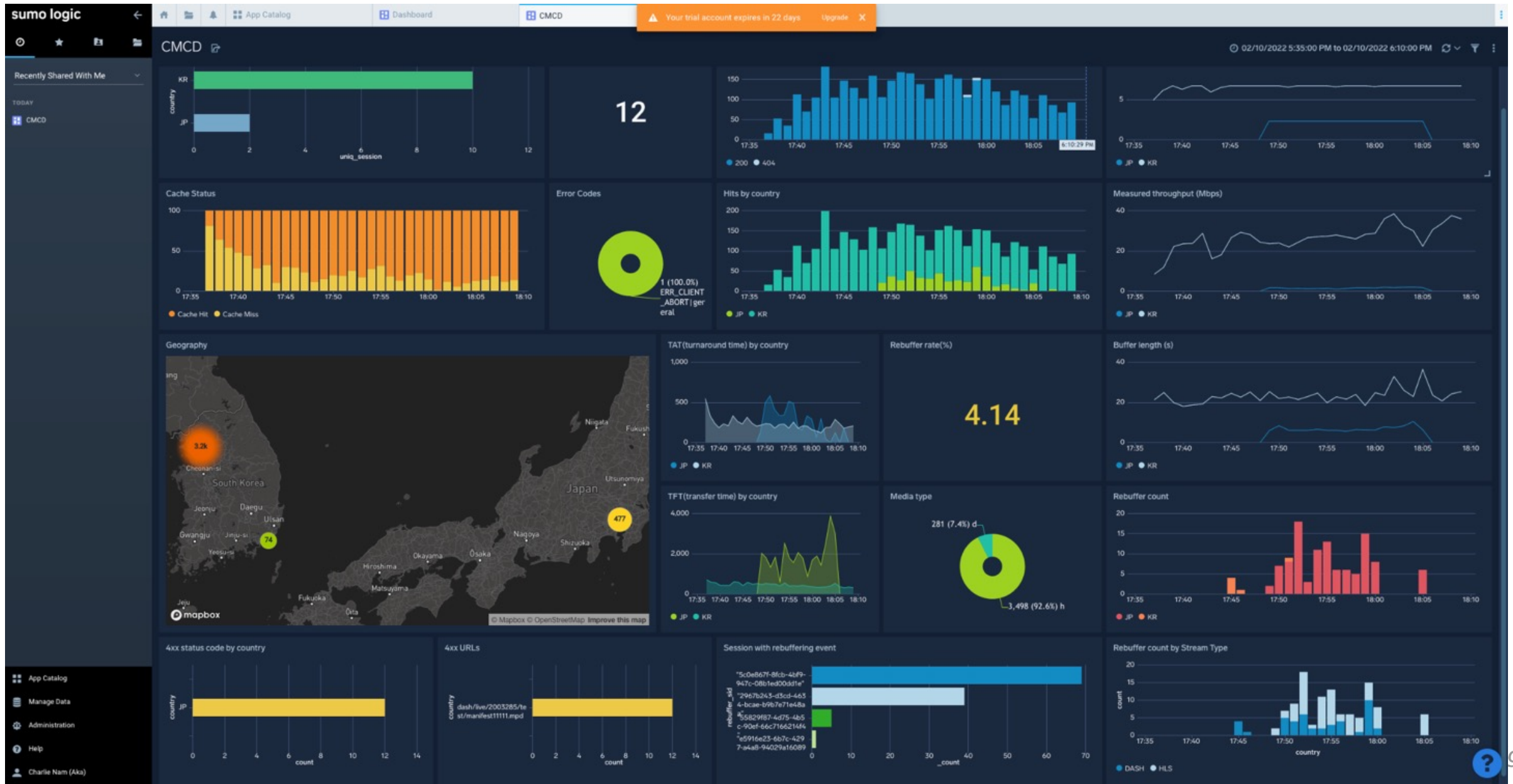
CMCD Buffer Length



CMCD Buffer Length



Use-case #5: Build yourself a QoE monitoring solution

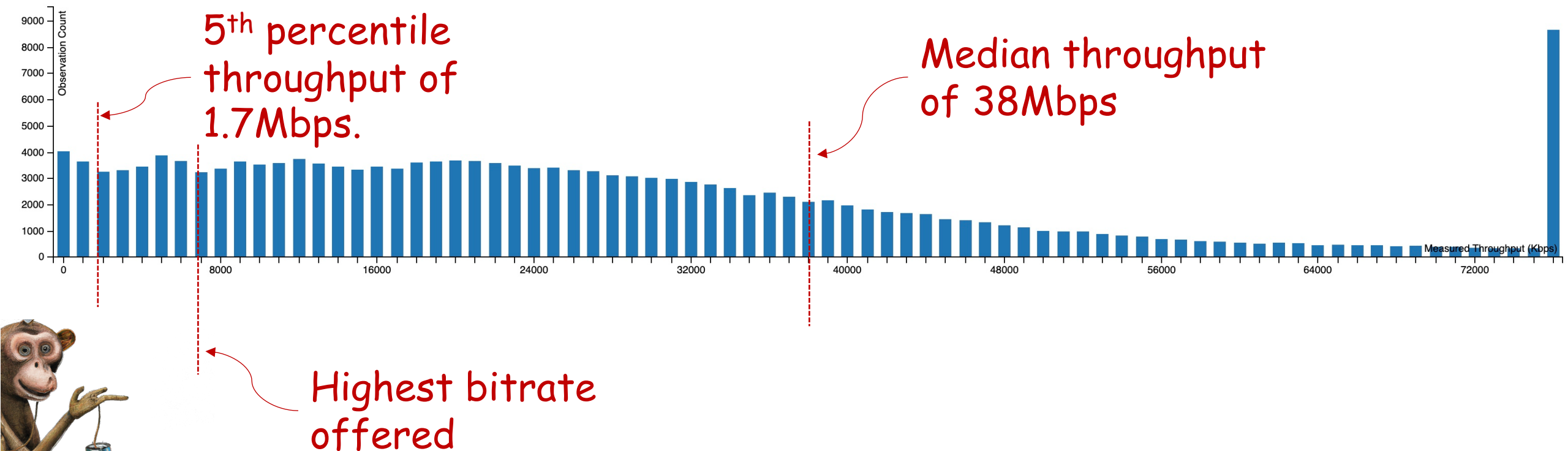




Use case #6: Dig out interesting data

18.9 Million requests, 181K sessions, 2 min of major USA OTT provider.

CMCD Median Measured Throughput

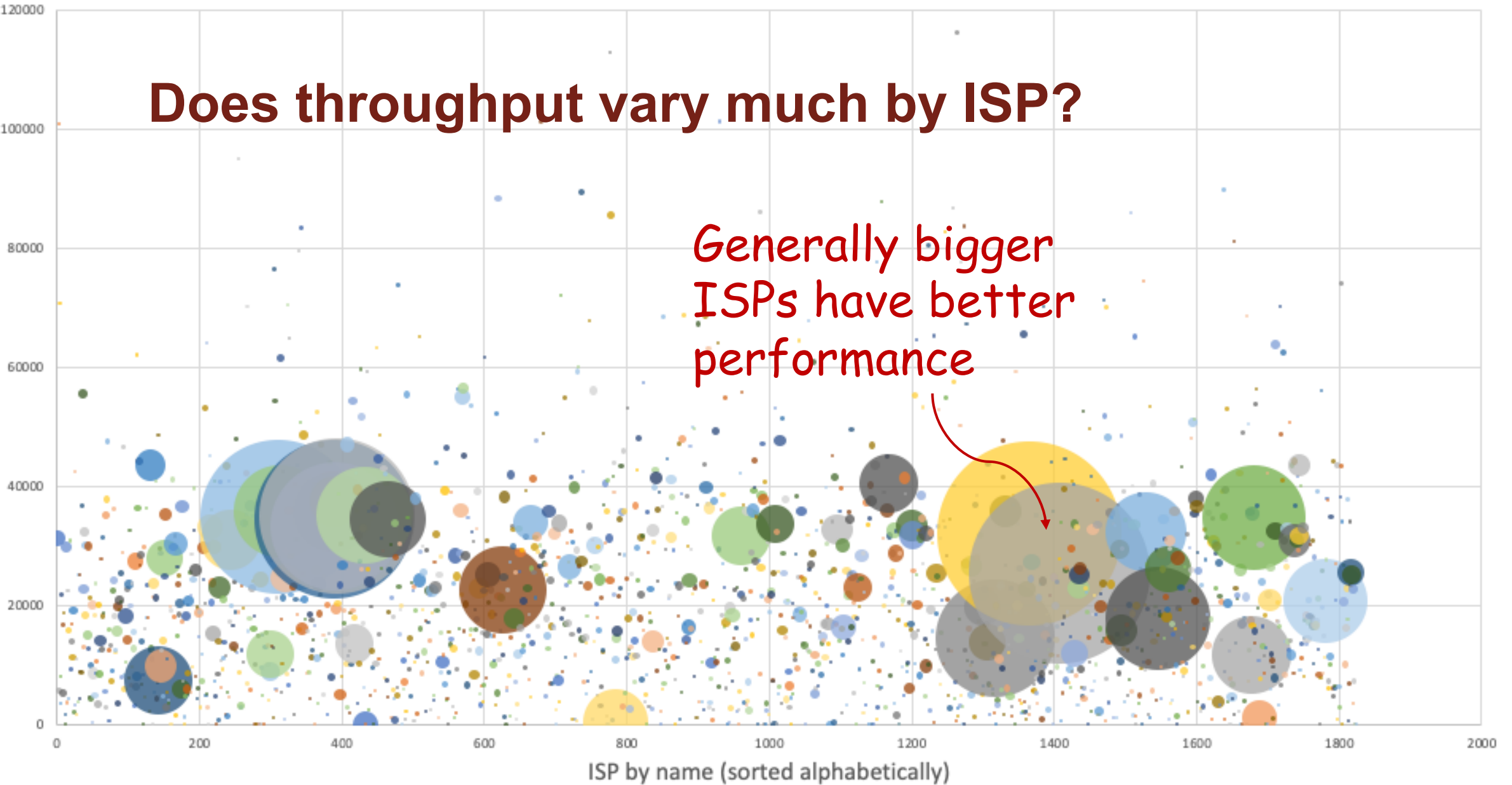


Average CMCD Measured Throughput by ISP

Does throughput vary much by ISP?

CMCD Measured Throughput in kbps

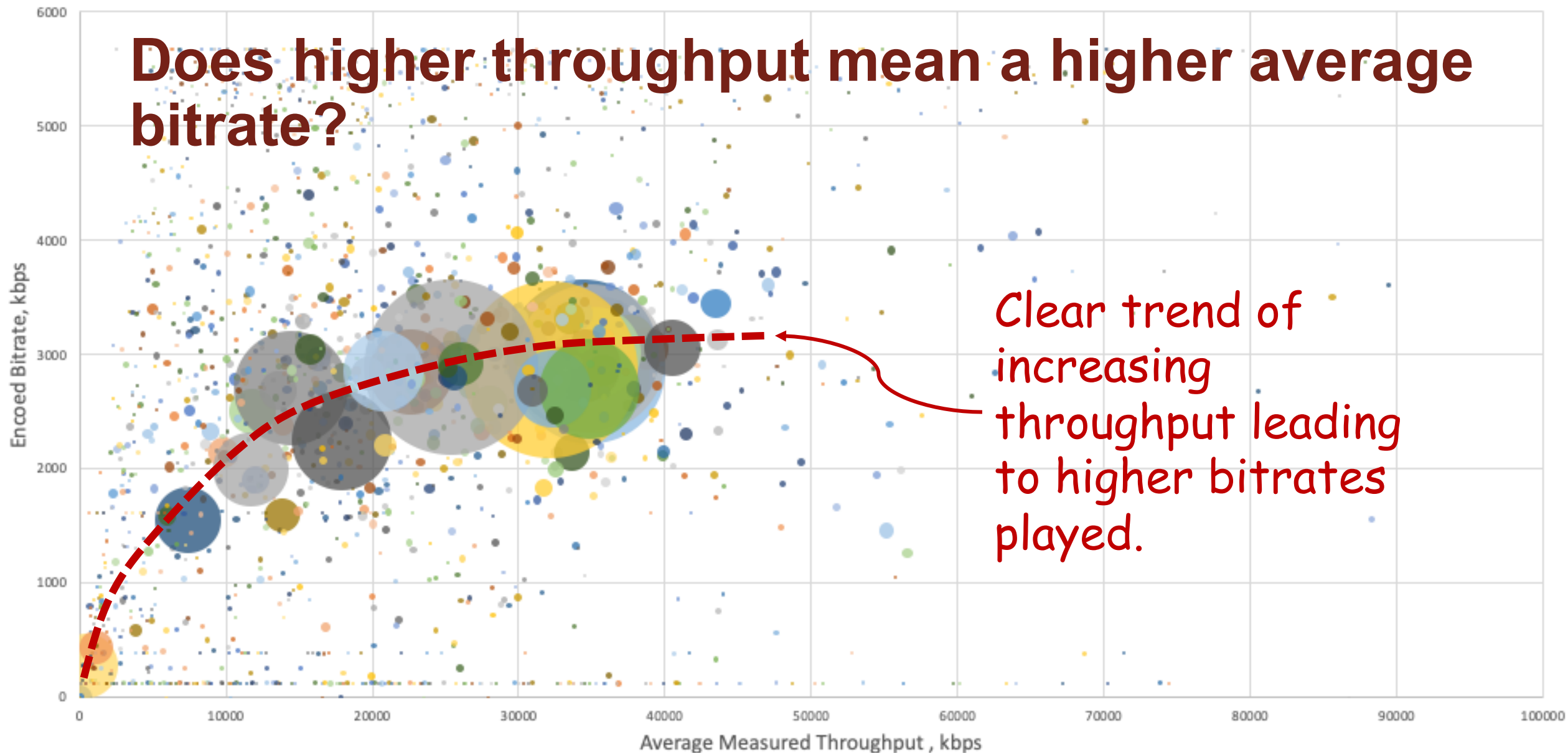
*Generally bigger
ISPs have better
performance*



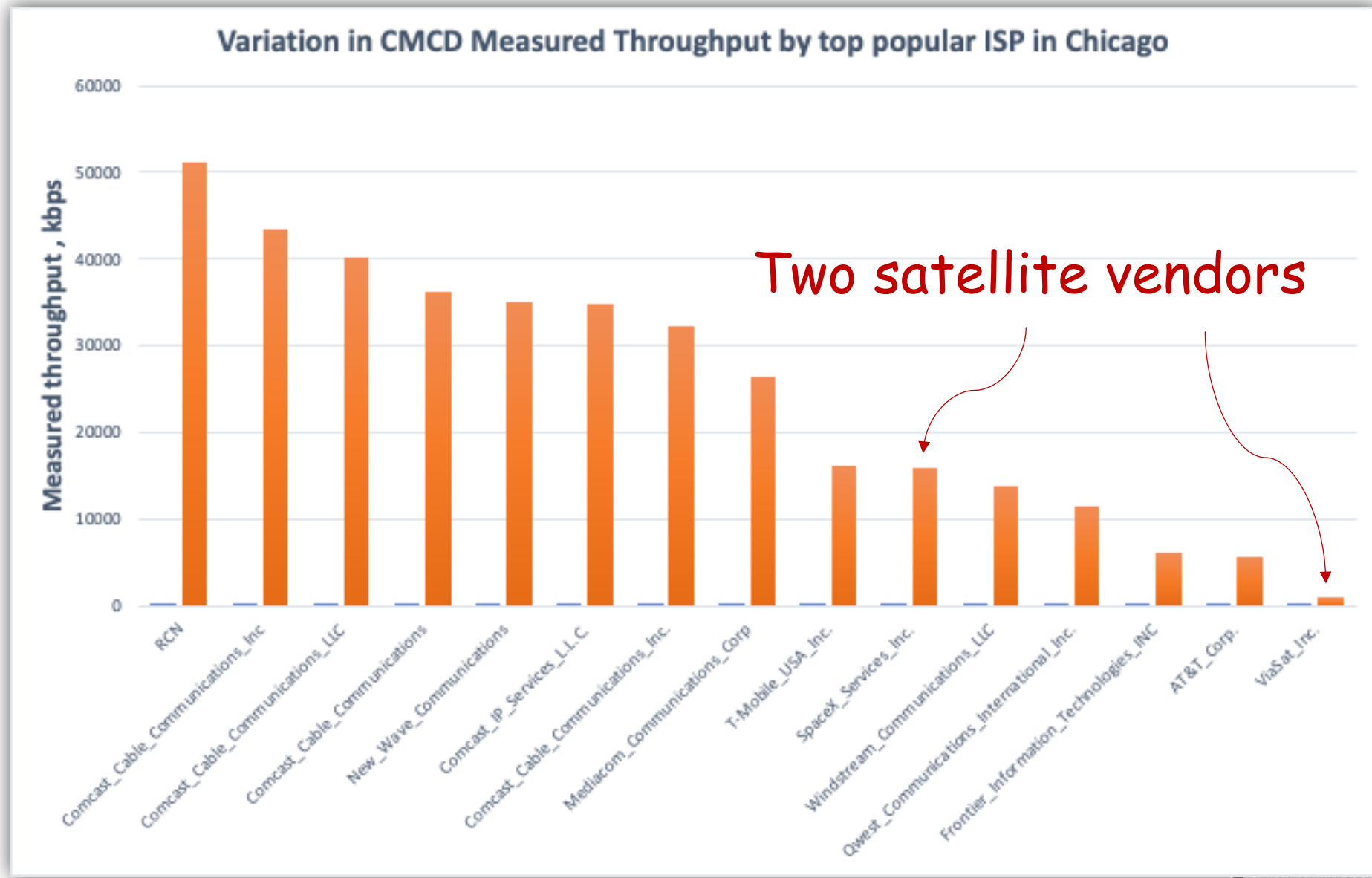
Variation in CMCD average Encoded Bitrate with CMCD measured throughput, by ISP

Does higher throughput mean a higher average bitrate?

Clear trend of increasing throughput leading to higher bitrates played.



How about variation within a city?

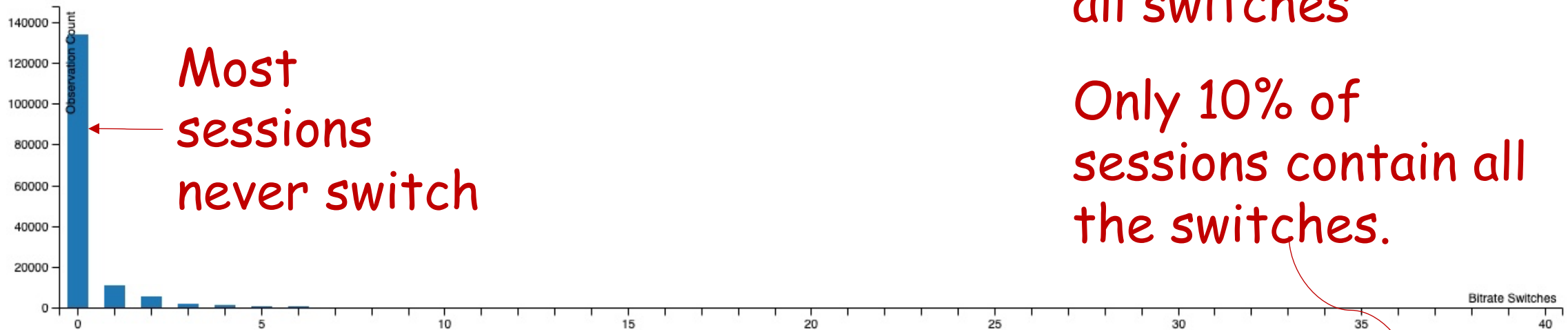


How often do OTT sessions switch bitrates?

The worst 1% contained 2/3 of all switches

Only 10% of sessions contain all the switches.

CMCD Bitrate Switches



Most sessions never switch

Count	1th Percentile	5th Percentile	10th Percentile	25th Percentile	Median	75th Percentile	90th Percentile	95th Percentile	99th Percentile
157,612	0.00	0.00	0.00	0.00	0.00	0.00	1.00	2.00	6.00

Which companies are using CMCD?



Vimeo has been actively working with CMCD data in our web playback



Our goal is to offer a standardized data set for operations teams to troubleshoot playback issues. A subset of our players now have the rich CMCD data set which makes debugging last-mile and CDN issues more timely and enables us to get to root cause faster



We're actively engaged on enabling CMCD across platforms in our ecosystem



**WARNER BROS.
DISCOVERY**



Broad support for CMCD is important



Player Support

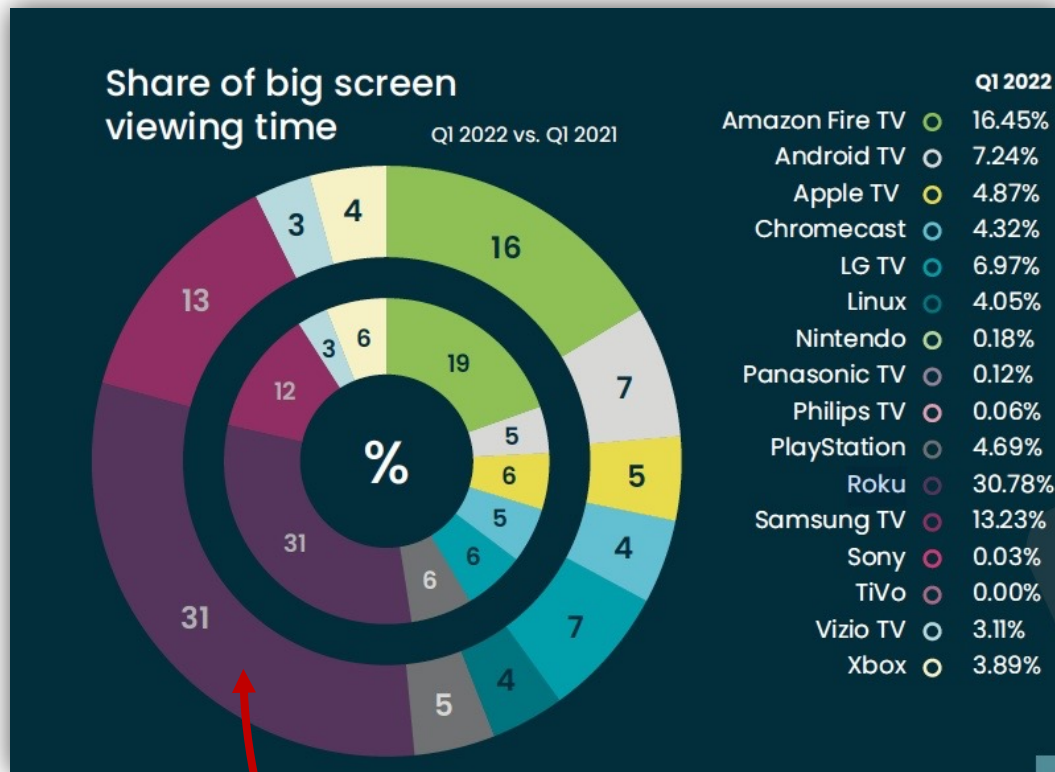


- CMCD can be added to any existing player by modifying outbound requests.
- Multiple players have added CMCD support as a first party feature, meaning you can enable it by setting a single property:
 - Dash.js – v3.0.3+, March 20
 - Hls.js – v1.1.0+ , Nov
 - Shaka - v3.3+, Jan 22
 - Roku – v11+, April 22
 - RDK (Comcast) 5.9s3+, Oct 22



King of the CMCD players

ROKU



Source: Conviva's State of Streaming Q1 2022

Roku delivers 31% of North American minutes viewed.

Since v11.0, all Roku players support CMCD, including with the upcoming v11.5, advanced fields such as nor, nrr which enable prefetching.



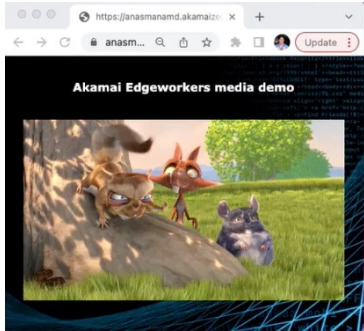
What if we don't have player support?

- Leverage the hierarchical and sequential nature of segmented playback
- Insert a proxy between the player and the origin and inject CMCD data on each request

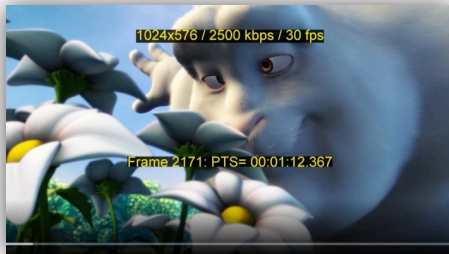


Meet Anders
who had a
bright idea

Test bed for CMCD injection POC



Hls.js player without CMCD enabled



Safari playing a m3u8 directly

```
buffer at 29600 so setting a delay of 351ms (total of 28802ms)
buffer at 28700 so setting a delay of 3125ms (total of 23927ms)
buffer at 28900 so setting a delay of 351ms (total of 24278ms)
buffer at 28500 so setting a delay of 3125ms (total of 27403ms)
buffer at 28600 so setting a delay of 3130ms (total of 30883ms)
buffer at 28600 so setting a delay of 351ms (total of 31234ms)
buffer at 29200 so setting a delay of 3134ms (total of 34368ms)
buffer at 29100 so setting a delay of 351ms (total of 34719ms)
buffer at 29000 so setting a delay of 3132ms (total of 37851ms)
buffer at 28900 so setting a delay of 351ms (total of 38202ms)
buffer at 28600 so setting a delay of 3113ms (total of 41315ms)
buffer at 28400 so setting a delay of 351ms (total of 41666ms)
buffer at 28900 so setting a delay of 3113ms (total of 44779ms)
buffer at 29000 so setting a delay of 351ms (total of 45130ms)
buffer at 31800 so setting a delay of 3897ms (total of 48226ms)
```

EdgeWorker Javascript



Akamai CDN and origin server

Datastream
real-time logs



Grafana
dashboard



Proxy playlist processing



#EXTM3U

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1054000

stream-1-9900/index.m3u8 ?CMCD=sid%3D422h4b3v6b%22%26mf%3Dh%26ob%3064
Dm%26b%3D1054%26tb%3D6064

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1864000

stream-2-18000/index.m3u8 ?CMCD=sid%3D422h4b3v6b%22%26mf%3Dh%26ob%3064
Dm%26b%3D1864%26tb%3D6064

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=6064000

stream-4-60000/index.m3u8 ?CMCD=sid%3D422h4b3v6b%22%26mf%3Dh%26ob%3064
Dm%26b%3D6064%26tb%3D6064

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3064000

stream-3-30000/index.m3u8 ?CMCD=sid%3D422h4b3v6b%22%26mf%3Dh%26ob%3064
Dm%26b%3D3064%26tb%3D6064



Variant playlist processing



Loading stream-2-18000/index.m3u8?CMCD=sid="h4k3v6b"&sf=h&ot=m&b=1864&tb=6064

#EXT-X-TARGETDURATION:4

#EXT-X-VERSION:7

#EXT-X-MEDIA-SEQUENCE:1

#EXT-X-PLAYLIST-TYPE:VOD

#EXT-X-INDEPENDENT-SEGMENTS

#EXT-X-MAP:URI="bbb_1800_0.m4v?CMCD=sid="h4k3v6b"&sf=h&b=1864&tb=6064&st=v&ot=v&su&d=4004

#EXTINF:4.004000 %3D1864%26tb%3D6064%26st%3Dv%26ot%3Di

bbb_1800_0.m4v?CMCD=sid="h4k3v6b"&sf=h&b=1864&tb=6064&st=v&ot=v&su&d=4004

#EXTINF:3.966000 CMCD=sid%3D%22h4k3v6b%22%26sf%3Dh%26b%3D1864%26tb%3D

bbb_1800_2.m4v?CMCD=sid="h4k3v6b"&sf=h&b=1864&tb=6064&st=v&ot=v&su&d=3960

#EXTINF:3.986000

bbb_1800_3.m4v?CMCD=sid="h4k3v6b"&sf=h&b=1864&tb=6064&st=v&ot=v&su&d=3986

6064%26st%3Dv%26ot%3Dv%26su%26d%3D3986

...



Server side processing

(Remember we receive CDN delivery logs muxed with the CMCD data)



Number of concurrent sessions = $\sum (\text{Unique sid})$

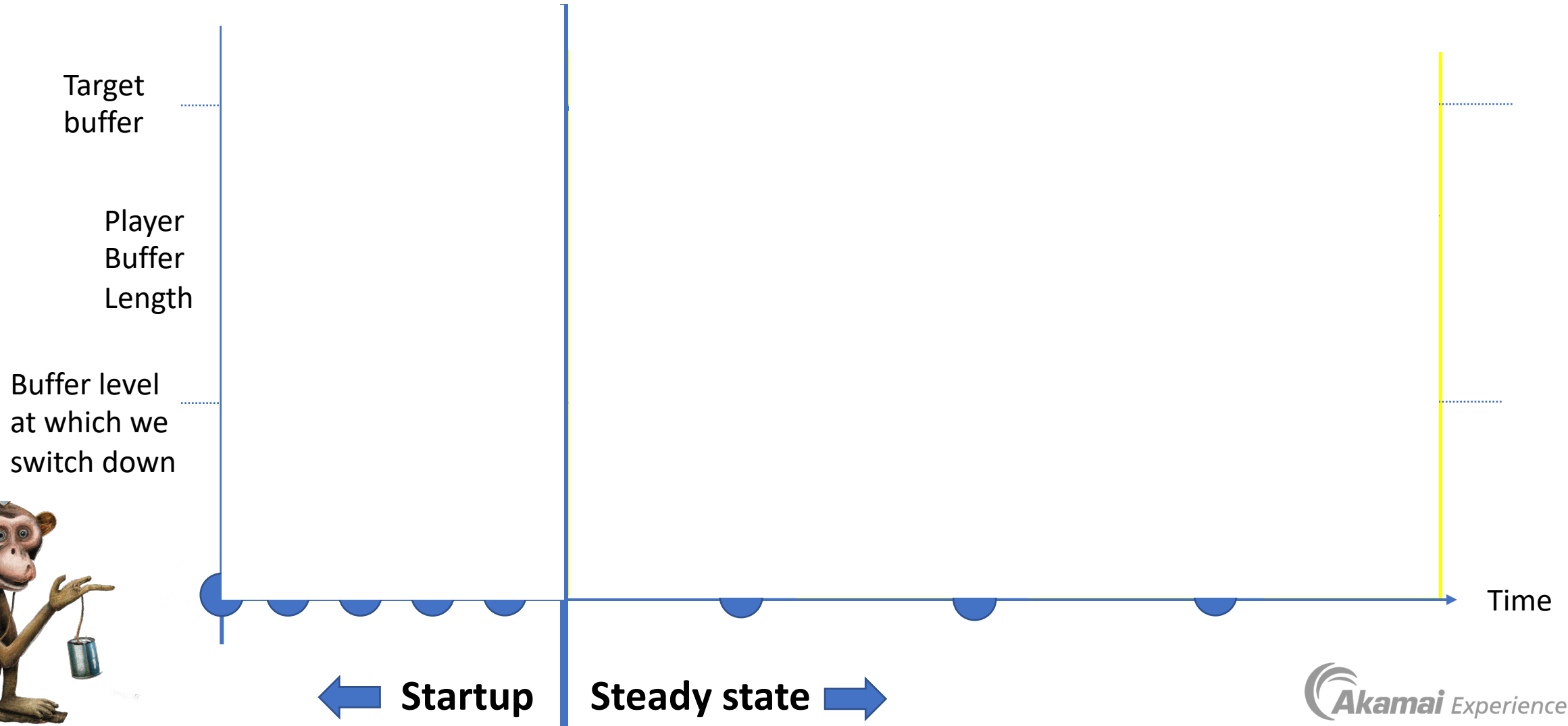
Estimated player buffer length for a given session =
 $\sum (\text{media duration} - \text{delivery duration})$

Estimated player video start time (VST) for a given session =

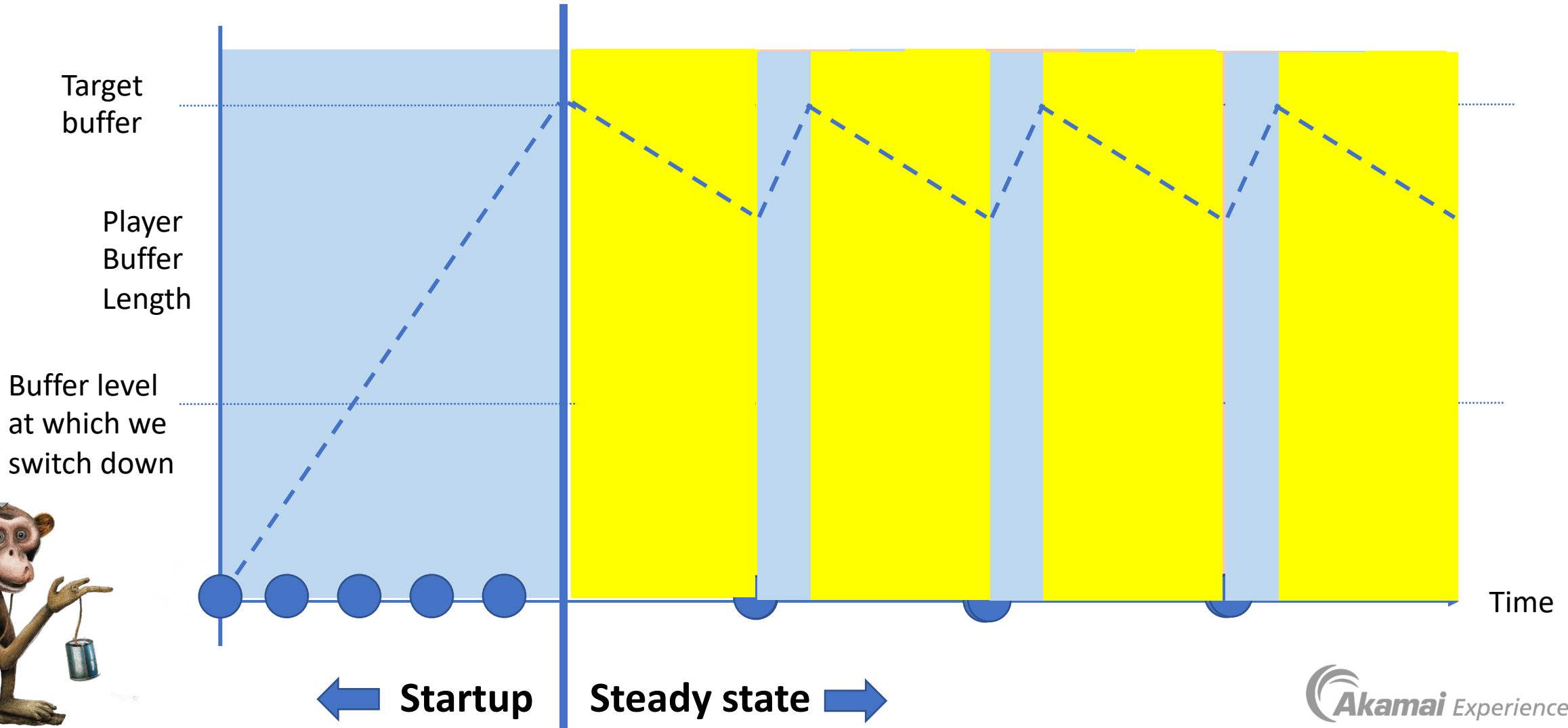
Timestamp of 3rd video segment request + 3rd video segment transfer time - timestamp of first playlist request - first playlist transfer time



The curious opportunities to be found in steady state.



The curious opportunities to be found in steady state.



Test bed for TTFB experiments

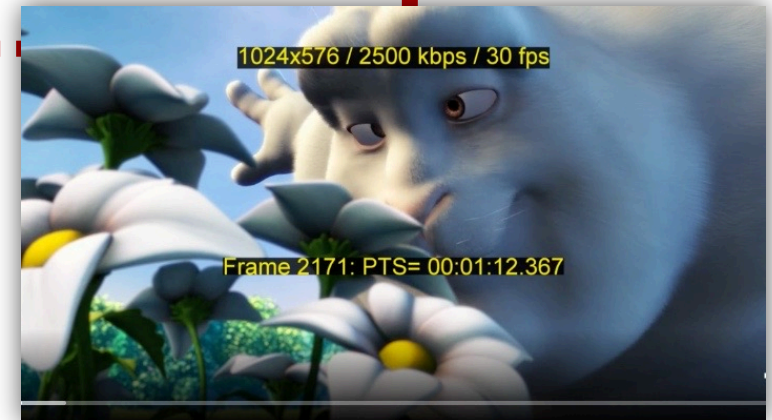
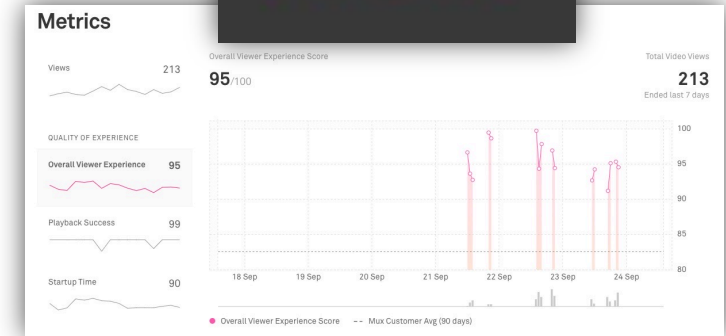
MUX Data for
media analytics



Akamai CDN and
origin server

```
nodejs-startup-proxy — node • node /usr/local/bin/nodemon index.js — 83x17
buffer at 29600 so setting a delay of 351ms (total of 20802ms)
buffer at 28700 so setting a delay of 3125ms (total of 23927ms)
buffer at 28900 so setting a delay of 351ms (total of 24278ms)
buffer at 28500 so setting a delay of 3125ms (total of 27403ms)
buffer at 28500 so setting a delay of 351ms (total of 27753ms)
buffer at 28600 so setting a delay of 3130ms (total of 30883ms)
buffer at 28600 so setting a delay of 351ms (total of 31234ms)
buffer at 29200 so setting a delay of 3134ms (total of 34368ms)
buffer at 29100 so setting a delay of 351ms (total of 34719ms)
buffer at 29000 so setting a delay of 3132ms (total of 37851ms)
buffer at 28900 so setting a delay of 351ms (total of 38202ms)
buffer at 28600 so setting a delay of 3113ms (total of 41315ms)
buffer at 28400 so setting a delay of 351ms (total of 41666ms)
buffer at 28900 so setting a delay of 3113ms (total of 44779ms)
buffer at 29000 so setting a delay of 351ms (total of 45130ms)
buffer at 31800 so setting a delay of 3097ms (total of 48226ms)
```

Node.js proxy
running on
localhost



dash.js player with CMCD enabled

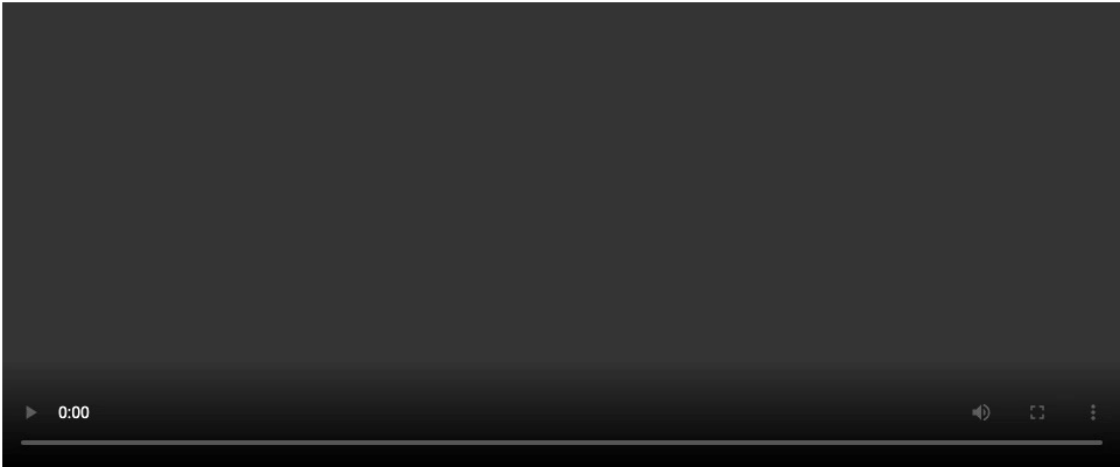


Startup test player

localhost/test/startup-test.html?player=test1&delay=2000

DASH-IF JS player Akamai Contacts Akamai Request T... Cracker 3.0 WebTransport Co...

START



Elements Console Performance insights Recorder Sources Network Performance Memory

Preserve log Disable cache No throttling

Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies

Blocked Requests 3rd-party requests

Name	Method	Status	Protocol	Type	Size	Time	Connecti...	C.	Waterfall
startup-test.html?player=test1&delay...	GET	304	http/1.1	document	247 B	8 ms	1127573	K...	
mux.js	GET	200	http/1.1	script	(memory cache)	0 ms	971213	k...	
dash.all.min.js	GET	200	http/1.1	script	(memory cache)	0 ms	959282	k...	

3 requests | 247 B transferred | 733 kB resources | Finish: 31 ms | DOMContentLoaded: 84 ms | Load: 84 ms

nodejs-startup-proxy -- -zsh -- 81x48

```
wilaw@sjc-mpavu nodejs-startup-proxy % nodemon index.js
```





Effect on QoE metrics of intentional TTFB delay – fast connection

MUX QoE metrics	0 ms
Overall viewer experience	98
Startup time score	95
Video start time (median seconds)	0.40
Weighted average bitrate (median Mbps)	15

Test conditions :

- VOD MPD with 10 representations from 320x180 @ 250kbps to 3840x2160 @ 15Mbps
- 4000ms segment duration
- 30 target buffer with unthrottled Chrome browser
- Akamai CDN via localhost node.js proxy



Effect on QoE metrics of intentional TTFB delay – slow connection



MUX QoE metrics	0 ms
Overall viewer experience	94
Startup time score	86
Video start time (median seconds)	1.37
Weighted average bitrate (median Mbps)	2.36

Test conditions :

- VOD MPD with 10 representations from 320x180 @ 250kbps to 3840x2160 @ 15Mbps
- 4000ms segment duration
- 12s target buffer, with Chrome browser throttled to 5Mbps
- Akamai CDN via localhost node.js proxy



How much can you increase TTFB without reducing player QoE?

- Target buffer of b seconds
- Segment duration of d seconds
- Switching safety ratio R
- Estimated throughput T Mbps
- Bitrate of object b Mbps

$$\int_0^{\infty} \frac{\ln x}{1+x^2} dx = ?$$

$$\text{Now } \int_0^1 \frac{\ln x}{1+x^2} dx = \int_{\infty}^1 \frac{\ln \frac{1}{t}}{1+\frac{1}{t^2}} \left(-\frac{1}{t^2} dt\right) \left\{ \text{Substitute } x = \frac{1}{t} \right\}$$

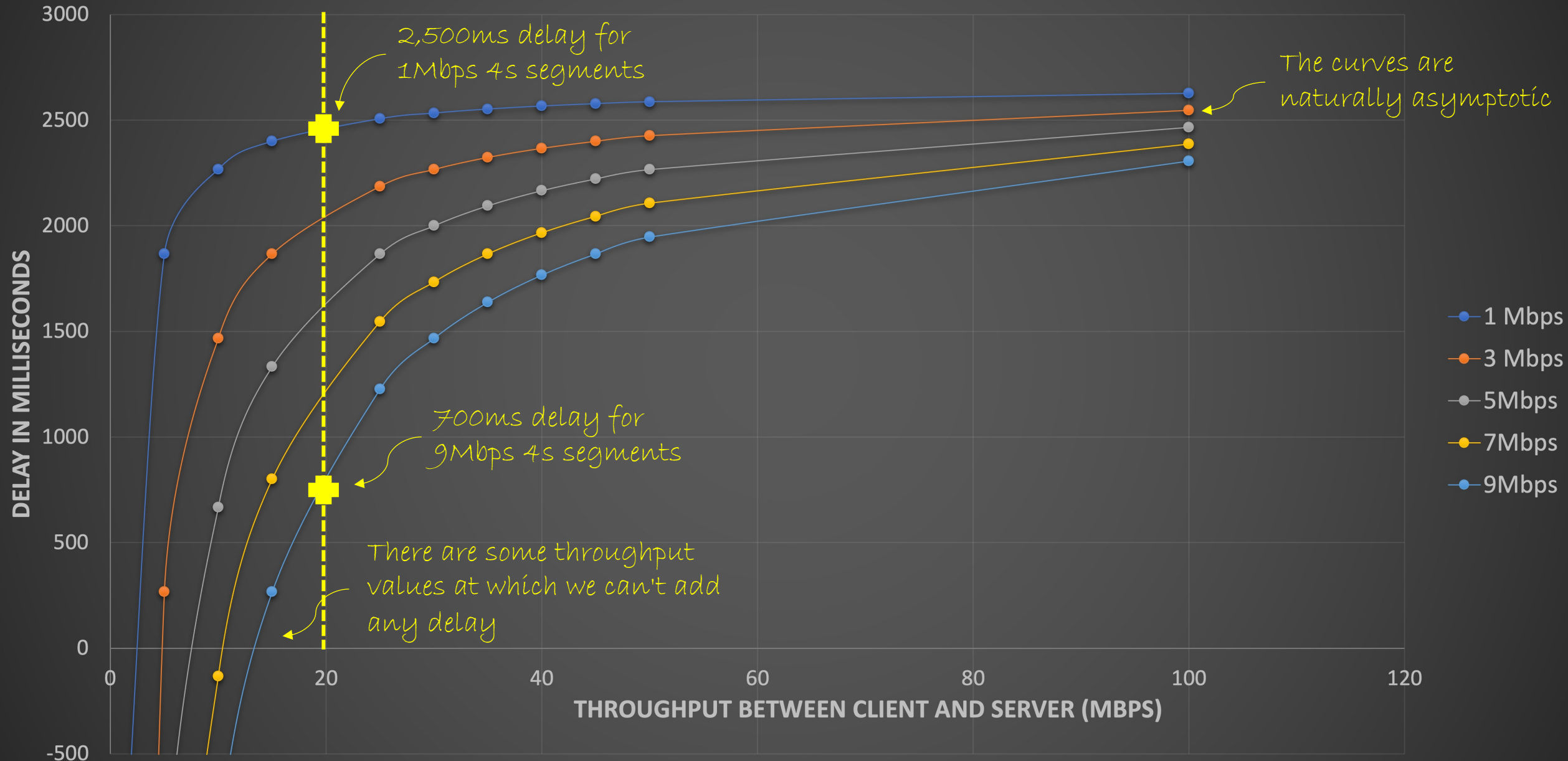
$$\text{Max delay in seconds} = \left(\frac{d}{R} \right) - \left(b \times \frac{d}{T} \right)$$

$$\Rightarrow \int_0^1 \frac{\ln x}{1+x^2} dx = - \int_{\infty}^1 \frac{(-\ln t)}{\left(1+\frac{1}{t^2}\right)t^2} dt = - \int_1^{\infty} \frac{\ln t}{1+t^2} dt$$

$$\Rightarrow \int_0^1 \frac{\ln x}{1+x^2} dx + \int_1^{\infty} \frac{\ln t}{1+t^2} dt = 0 \quad \{\text{Then change dummy variable}\}$$

$$\Rightarrow \int_0^1 \frac{\ln x}{1+x^2} dx + \int_1^{\infty} \frac{\ln x}{1+x^2} dx = 0 \Rightarrow \int_0^{\infty} \frac{\ln x}{1+x^2} dx = 0$$

Maximum non-QoE-impacting induced TTFB delay
as function of available throughput and encoded bitrate





Can we build a smart delay engine?

One that extracts maximum delay on the server –side without reducing player QoE?

Use 1.25

$$\text{Max delay} = \underbrace{(d/R)}_{\text{CMCD 'd'}} - \underbrace{(b \times d/T)}_{\text{CMCD 'b' CMCD 'mtp'}}$$



Startup test player

localhost/test/startup-test.html?player=comp-no-delay

DASH-IF JS player Akamai Contacts Akamai Request T... Cracker 3.0 WebTransport Co-...

START

0:00

Elements Console Performance Insights Recorder Sources Network Performance Memory

Filter ☐ Invert ☐ Hide data URLs ☒ All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies

☐ Blocked Requests ☐ 3rd-party requests

Name	Method	Status	Protocol	Type	Size	Time	Connecti...	C.	Waterfall
startup-test.html?player=comp-no-de...	GET	304	http/1.1	document	247 B	7 ms	1047229	K...	
mux.js	GET	200	http/1.1	script	(memory cache)	0 ms	971213	k...	
dash.all.min.js	GET	200	http/1.1	script	(memory cache)	0 ms	959282	k...	

nodejs-startup-proxy — node • node /usr/local/bin/nodemon index.js — 107x23

```
buffer at 28900 so setting a delay of 0 (total of 0s)
buffer at 29500 so setting a delay of 0 (total of 0s)
buffer at 29400 so setting a delay of 0 (total of 0s)
buffer at 29000 so setting a delay of 0 (total of 0s)
buffer at 28600 so setting a delay of 0 (total of 0s)
buffer at 29800 so setting a delay of 0 (total of 0s)
buffer at 29400 so setting a delay of 0 (total of 0s)
buffer at 29600 so setting a delay of 0 (total of 0s)
buffer at 29100 so setting a delay of 0 (total of 0s)
buffer at 29700 so setting a delay of 0 (total of 0s)
buffer at 29000 so setting a delay of 0 (total of 0s)
buffer at 29400 so setting a delay of 0 (total of 0s)
buffer at 28800 so setting a delay of 0 (total of 0s)
buffer at 29300 so setting a delay of 0 (total of 0s)
buffer at 28900 so setting a delay of 0 (total of 0s)
buffer at 29500 so setting a delay of 0 (total of 0s)
buffer at 29100 so setting a delay of 0 (total of 0s)
buffer at 29800 so setting a delay of 0 (total of 0s)
buffer at 29400 so setting a delay of 0 (total of 0s)
buffer at 30200 so setting a delay of 0 (total of 0s)
[nodemon] restarting due to changes...
[nodemon] starting node index.js
```

Startup test player

localhost/test/startup-test.html?player=comp-no-delay

DASH-IF JS player Akamai Contacts Akamai Request T... Cracker 3.0 WebTransport Co-...

START

0:00

Elements Console Performance Insights Recorder Sources Network Performance Memory


Filter ☐ Invert ☐ Hide data URLs ☒ All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other ☐ Has blocked cookies

☐ Blocked Requests ☐ 3rd-party requests

Name	Method	Status	Protocol	Type	Size	Time	Connecti...	C.	Waterfall
startup-test.html?player=comp-no-de...	GET	304	http/1.1	document	247 B	39 ms	1062521	K...	
mux.js	GET	200	http/1.1	script	(memory cache)	0 ms	971213	k...	
dash.all.min.js	GET	200	http/1.1	script	(memory cache)	0 ms	959282	k...	

3 requests | 247 B transferred | 733 kB resources | Finish: 53 ms | DOMContentLoaded: 133 ms | Load: 132 ms

```
buffer at 29200 so setting a delay of 3097 (total of 17239ms)
buffer at 30000 so setting a delay of 3116 (total of 20355ms)
buffer at 29600 so setting a delay of 351 (total of 20706ms)
buffer at 29200 so setting a delay of 3113 (total of 23819ms)
buffer at 29100 so setting a delay of 351 (total of 24169ms)
buffer at 28800 so setting a delay of 3113 (total of 27282ms)
buffer at 28800 so setting a delay of 351 (total of 27633ms)
buffer at 28600 so setting a delay of 3113 (total of 30746ms)
buffer at 28400 so setting a delay of 351 (total of 31097ms)
buffer at 28900 so setting a delay of 3097 (total of 34194ms)
buffer at 28700 so setting a delay of 351 (total of 34545ms)
[nodemon] restarting due to changes...
buffer at 29100 so setting a delay of 3097 (total of 37641ms)
[nodemon] starting 'node index.js'
buffer at 28000 so setting a delay of 3097ms (total of 3097ms)
buffer at 28800 so setting a delay of 351ms (total of 3448ms)
buffer at 28700 so setting a delay of 3116ms (total of 6564ms)
buffer at 28600 so setting a delay of 351ms (total of 6915ms)
buffer at 29000 so setting a delay of 3113ms (total of 10028ms)
buffer at 28500 so setting a delay of 351ms (total of 10379ms)
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
```





Why is the useful?

- **Optimized caching**
 - We don't need to cache all the content at the edge.
 - In theory I can cache only 2% of a VOD content without harming player QoE.
- **Optimized prefetching**
 - Why prefetch if the player has a high tolerance for TTFB?
- **Serve content from further away.**
 - Not all files need be served from the edge. Serve from a location which is cheaper, or which has a lower carbon footprint.
- **Throttle for ABR stability**
 - Faster for individuals is not necessarily better for the aggregate good.
 - Slow down delivery of individual streams. Save resources for other streams. Lower peak congestion.



Throttling for ABR stability


The yellow player estimates that throughput is entirety of channel, and switches up

However, at next segment load it only has 1/3 throughput and risks rebuffering.

By throttling delivery as much as possible without causing a switchdown, consistent B/W estimation can be achieved.




Time



Time



What's next? - CMSD v2

- Work will start at CTA WAVE in 1H-23  **WAVE**
WEB APPLICATION VIDEO ECOSYSTEM
- New features to be added (amongst many)
 - **Video start time** – round out the QoE metrics
 - **Target buffer** - allows you to normalize buffer health across large populations of diverse players
- File your issues and requests at <https://github.com/cta-wave/common-media-client-data/issues>



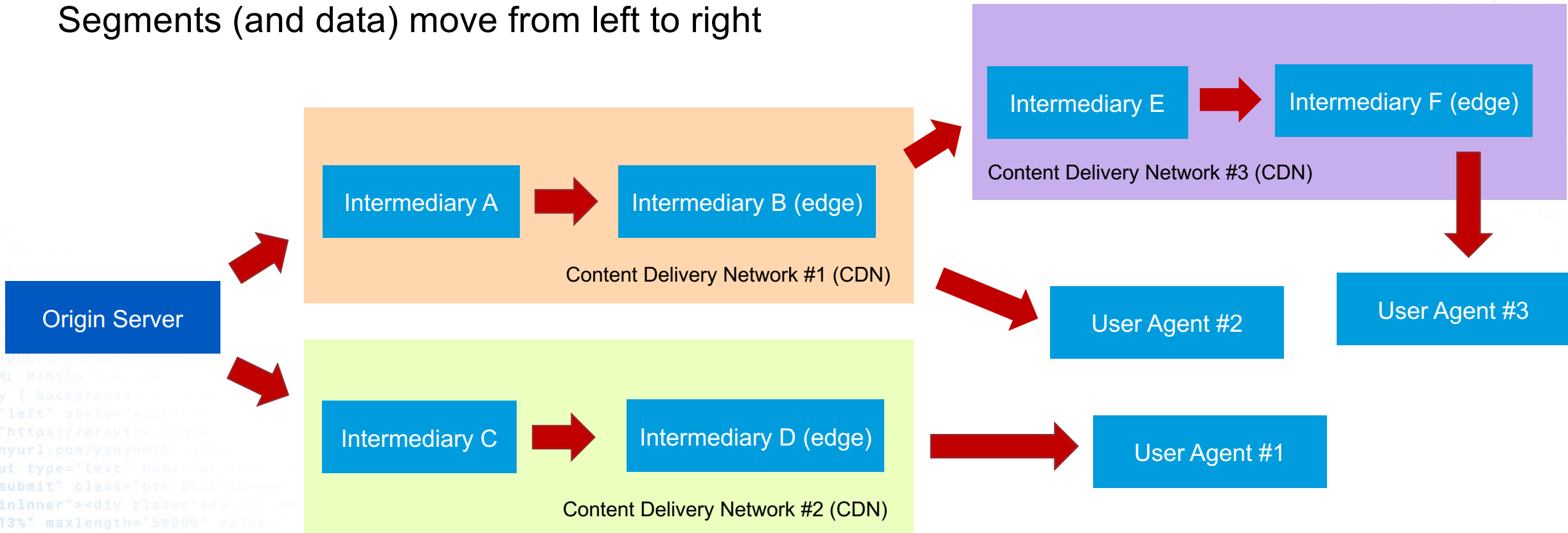
Common Media Server Data (CMSD)

Will Law, Dec 2022
Chair – CMSD WG



CMSD – A bridge between origins, CDNs and players

Segments (and data) move from left to right



Sharing component specific knowledge

Things the origin knows:

- Related objects - such as next segment, or init files and starting segments.
- Object type - audio/video/init/key/manifest
- Availability time
- Held time
- Object duration
- Object encoded bitrate
- Target latency
- Server-induced delay
- Whether the segment is involved in startup VOD or LIVE

Things the CDN knows

- Estimate link throughput
- Link RTT
- Whether the intermediary server is under duress or not
- TTFB on a forward request
- Max suggested bitrate that the player should play

How can data be transmitted?

Simply via **Response Headers** which are a robust and well understood data transfer system

Two headers are defined, in order to limit HPACK/QPACK table entries

CMSD-Static – data about the request that does not change over each link

CMSD-Dynamic - data about the request that changes with each link

CMSD-Static:

n="OriginProviderA";ot=v;sf=h;st=v;ht=437;d=500;br=2000

CMSD-Dynamic: "CDNB-3ak1";etp=96;rtt=8

CMSD-Dynamic: "CDNB-w35k";etp=76;rtt=32

CMSD-Dynamic: "CDNA-987.343";etp=48;rtt=30

CMSD-Dynamic: "CDNA-312.663";etp=115;rtt=16;mb=5000

What type of data is transmitted?

Description	Key	Header Name	Type & Unit	Value definition
Timestamp	t	CMSD-Dynamic	Integer ms	Milliseconds since Unix Epoch in UTC. This value MUST be NTP synchronized [15] and MUST represent the time at which the server began the response.
Entity identifier	n	CMSD-Static	String	An identifier for the processing server. Maximum length of 64 characters. The value SHOULD identify both the organization and the intermediate server that is writing the key.
Estimated throughput	etp	CMSD-Dynamic	Integer Kbps	The throughput between the server and the client over the currently negotiated transport as estimated by the server at the start of the response.
RTT	rtt	CMSD-Dynamic	Integer milliseconds	Estimated round trip time between client and server.
Max suggested bitrate	mb	CMSD-Dynanic	Integer Mbps	The maximum bitrate value that the player SHOULD play in its Adaptive Bit Rate (ABR) ladder. If the player is playing a bitrate higher than this value, it SHOULD immediately switch to a bitrate lower than or equal to this value. This maximum suggested bitrate applies to all future requests made by the player for this playback session, until updated or the end-of-content is reached.

What type of data is transmitted? - 2

Description	Key	Header Name	Type & Unit	Value definition
Next object response	nor	CMSD-Static	Inner list of Strings	The relative path to one or more objects which can reasonably be expected to be requested by a media client consuming the current response. An intermediate server MAY use this key to perform a prefetch action.
Next range response	nrr	CMSD-Static	Inner List of Strings of the form "<range-start>-<range-end>"	If the next response will be a partial object request, then this string denotes the byte range to be returned. If the 'nor' field is not set, then the object is assumed to match the object currently being served. Formatting is similar to the HTTP Range header, except that the unit MUST be 'byte', the 'Range:' prefix is NOT required and specifying multiple ranges is NOT allowed. Valid combinations are: "<range-start>-", "<range-start>-<range-end>" and "-<prefix-length>". The range specified at a given position in the 'nrr' list will apply to the object specified at the same position in the 'nor' list. If the value is an empty string, then no range request is to be made for that object and the entire object should be fetched.
Stream type	st	CMSD-Static	Token - one of [v,l]	v = all segments are available – e.g., VoD. l = segments become available over time – e.g. live. state information should be trusted for no longer than the cache time of the object. Low latency streams can be signaled using the target latency 'tl' key.

What type of data is transmitted? - 3

Description	Key	Header Name	Type & Unit	Value definition
Object type	ot	CMSD-Static	Token - one of [m,a,v,av,i,c,tt,k,o]	The media role of the current object being returned: m = text file, such as a manifest or playlist a = audio only v = video only av = muxed audio and video i = init segment c = caption or subtitle tt = ISOBMFF timed text track k = cryptographic key, license or certificate. o = other
Stream type	st	CMSD-Static	Token - one of [v,l]	v = all segments are available – e.g., VoD. l = segments become available over time – e.g. live. state information should be trusted for no longer than the cache time of the object. Low latency streams can be signaled using the target latency 'tl' key.
Streaming format	sf	CMSD-Static	Inner list of tokens -([d h s o])	The streaming format that defines the current response. d = MPEG DASH h = HTTP Live Streaming (HLS) s = Smooth Streaming o = other

What type of data is transmitted? - 4

Description	Key	Header Name	Type & Unit	Value definition
Availability time	at	CMSD-Static	Integer milliseconds	The wallclock time at which the first byte of this object became available at the origin for successful request. The time is expressed as integer milliseconds since the Unix Epoch.
Held time	ht	CMSD-Static	Integer milliseconds	The number of milliseconds that this response was held back by an origin before returning. This is applicable to blocking responses under LL-HLS.
Object duration	d	CMSD-Static	Integer milliseconds	The playback duration in milliseconds of the object. If a partial segment is being served, then this value MUST indicate the playback duration of that part and not that of its parent segment.
Encoded bitrate	br	CMSD-Static	Integer Kbps	The encoded bitrate of the audio or video object being requested. If the instantaneous bitrate varies over the duration of the object, the peak value should be communicated.
Startup	su	CMSD-Static	Boolean	Key is included without a value if the object is needed for playback within the first 30 seconds of a VOD asset . This key MUST NOT be sent if it is FALSE.

What type of data is transmitted? - 5

Description	Key	Header Name	Type & Unit	Value definition
Session ID	sid	CMSD-Static	String	A tie to the CMCD Session ID request which spawned this response. This common attribute between CMCD and CMSD provides a join across both data sets.
Request ID	rid	CMSD-Static	String	A request ID, issued by the player or an upstream component, that is received by the origin when processing inbound content requests.
Time to First Byte	tfb	CMSD-Dynamic	Integer Milliseconds	The elapsed time between a request being received at the server and first byte of the response being sent.
Duress	du	CMSD-Dynamic	Boolean	TRUE if the server is under duress, due to cpu, memory, disk IO, network IO. The thresholds for signaling duress are left to the discretion of the server operator. The intent is that the client should use this signal to move away to an alternate server if possible. This key should only be sent if TRUE.
Target latency	tl	CMSD-Static	Integer milliseocnds	The target distribution latency of the stream, from origin to player.
Server-induced delay	sd	CMSD-Dynamic	Integer Milliseconds	The server may fetch the requested object from upstream or may process more urgent requests causing additional delays for this request. If the client knows about this extra delay, it can subtract it from the total response time when estimating the available bandwidth.

Use case #1: Smart prefetching

Master playlist

<http://example.com/content/index.m3u8>

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=150000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/content/low/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=240000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/content/lo\_mid/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=440000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/content/hi\_mid/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=640000,RESOLUTION=640x360,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/content/high/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=64000,CODECS="mp4a.40.5"
http://example.com/content/audio/index.m3u8
```

CMSD response headers on
<http://example.com/content/index.m3u8>

CMSD-Static: `nor=("low/index.m3u8"
"lo_mid/index.m3u8" "hi_mid/index.m3u8"
"high/index.m3u8" "audio/index.m3u8")`

Variant playlist

http://example.com/content/hi_mid/index.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:6
#EXT-X-MAP:URI="init.mp4"
#EXTINF:4.004,  
segment1.mp4  
#EXTINF:4.004,  
segment2.mp4  
#EXTINF:2.002,  
...
```

CMSD response headers on
http://example.com/content/hi_mid/index.m3u8

CMSD-Static:
`nor=("init.mp4"
"segment1.mp4"`

Use case #2: Starting bitrate selection

Player makes a request for a playlist to start a stream

<http://example.com/content/index.m3u8>

and receives the following CMSD headers in the response

CMSD-Static:ot="m",st="h",su, nor= ("720-3mbps.m3u8"
"1080-8mbps.m3u8")

CMSD-Dynamic:etp="34000",rtt="32"

Player recognizes the 34Mbps estimated throughput and decides to start with the 1080-8mbps.m3u8 variant.

Use case #3: DASH low latency timing

Player fetches a segment in a LL-DASH presentation at time 1651954053652

<https://example.com/live/lowlatency/segment64.mp4>

and receives the following CMSD header

`CMSD-Static:ot="v",st="d",tl="3500",at=1651954053353"`

It now knows it can make its next request 1651954053652 – 1651954053353 = 299ms earlier.

This will allow its buffer to increase by 299ms for the same latency, or its latency to decrease by 299ms for the same buffer.

Use case #4: preferential edge caching

CND edge sees the following CMSD headers on different objects

CMSD-Static: ot="v";st="h";d="6006";nor=("segment43.mp4")

CMSD-Static: ot="v";st="h";d="6006";nor=("segment167.mp4")

CMSD-Static: ot="v";st="h";su;d="6006";nor=("segment3.mp4")

CMSD-Static: ot="v";st="h";d="6006";nor=("segment67.mp4")

It caches the segment tagged with 'su', as that is one critical to player start-up performance.

Use case #5: dynamic timeout thresholds

Timeout values against origins and parent servers are nearly always set at a fixed value. However timeouts should be optimized around segment duration and target latency.

An intermediate CDN server sets a timeout value of 2s after receiving this response header:

```
CMSD-Static:ot="v",st="1";d="6006",t1="18000"
```

Against the same origin it sets a timeout value of 0.5s after receiving this response header:

```
CMSD-Static:ot="v",st="1";d="2002",t1="7500"
```


Use case #6: Network throughput limits to relieve peak congestion

A last mile network is suffering from peak congestion. They request the CDN to request all players to voluntarily reduce their top bitrate to less than 5Mbps so that all users can have

The CDN injects the following header into each media object response.

```
CMSD-Dynamic:mb="5000"
```

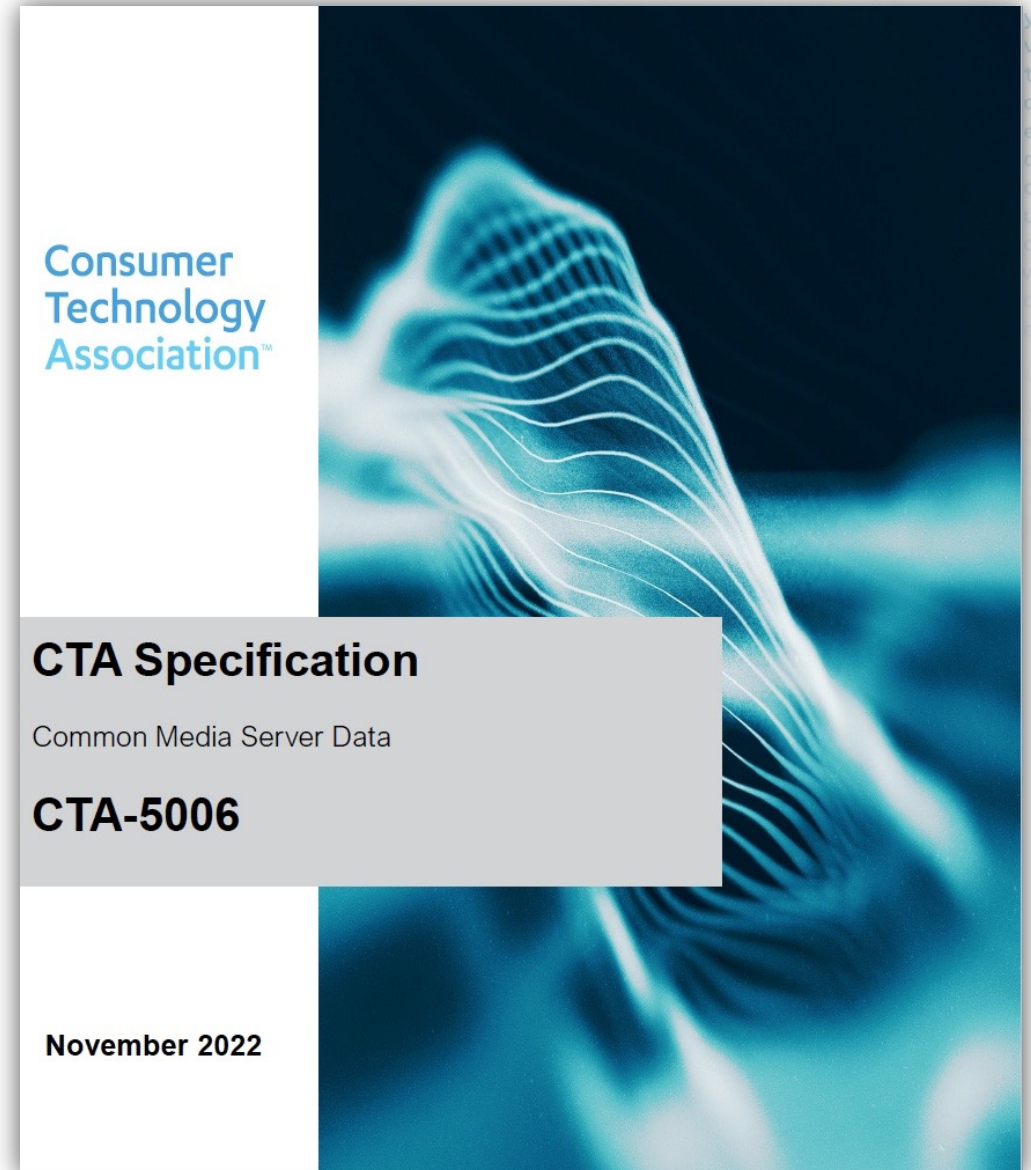
All players receiving this header will voluntarily switch down until they are playing a representation $< 5\text{Mbps}$. To remove the restriction once the congestion has passed, the CDN could send

```
CMSD-Dynamic:mb="50000"
```

CMSD status – published!

Published as **CTA-5006** in
November 2022.

You may download it for free from:
<https://shop.cta.tech/collections/standards/products/web-application-video-ecosystem-common-media-server-data-cta-5006>.





Thank you for your time.