STREAMING ANALYTICS USING CMCD AND CMSD

Second Milestone Presentation

Daniel Yermakov, Maximilian Roschlau, Neha Shrestha

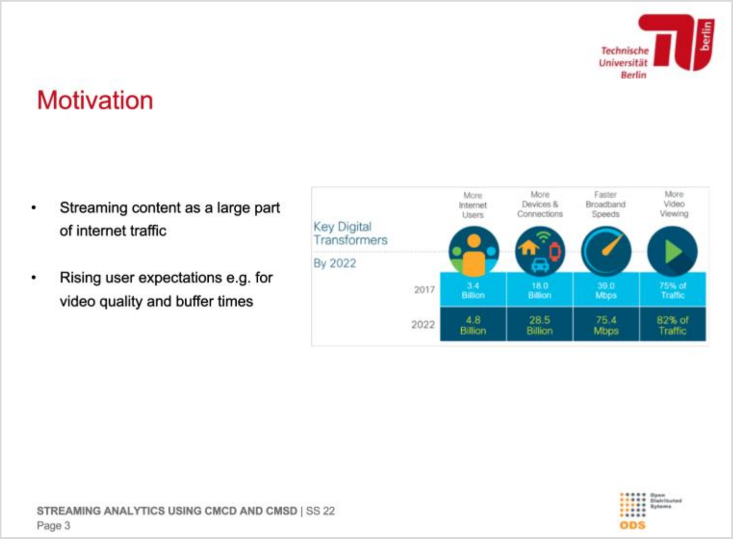
Open Distributed System | SS 22

Content

1. Problem Statement
2. NUStreaming – CMCD
3. NUStreaming – CMSD
4. Unified Streaming
5. NUStreaming-CMSD vs. Unified Streaming
6. Challenges
7. Schedule and Next Steps
8. References

Problem Statement

- Recap on problem Statement



Technische Universität Berlin

Motivation

- Streaming content as a large part of internet traffic
- Rising user expectations e.g. for video quality and buffer times

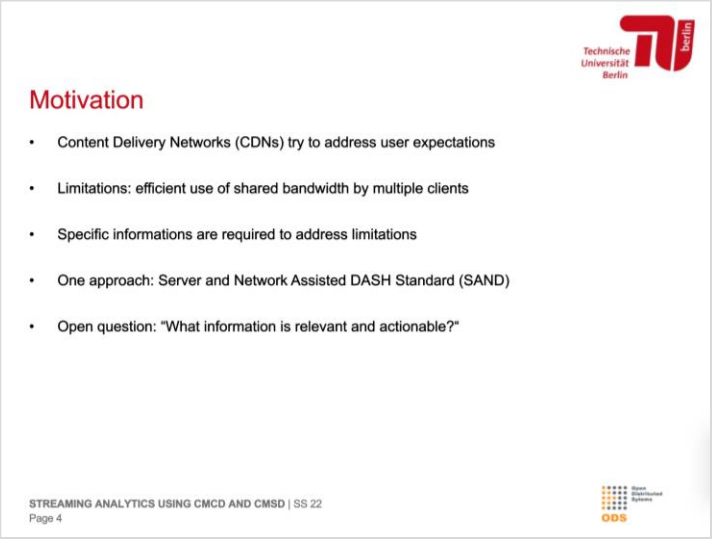
Key Digital Transformers

By 2022

	More Internet Users	More Devices & Connections	Faster Broadband Speeds	More Video Viewing
2017	3.4 Billion	18.0 Billion	39.0 Mbps	75% of Traffic
2022	4.8 Billion	28.5 Billion	75.4 Mbps	82% of Traffic

STREAMING ANALYTICS USING CMCD AND CMSD | SS 22
Page 3

Open Distributed Sytems ODS



Technische Universität Berlin

Motivation

- Content Delivery Networks (CDNs) try to address user expectations
- Limitations: efficient use of shared bandwidth by multiple clients
- Specific informations are required to address limitations
- One approach: Server and Network Assisted DASH Standard (SAND)
- Open question: "What information is relevant and actionable?"

STREAMING ANALYTICS USING CMCD AND CMSD | SS 22
Page 4

Open Distributed Sytems ODS

- Approached solution: Use of CMCD and CMSD specification

NUStreaming – CMCD

- Investigated the feasibility of CMCD and test its capabilities in the context of video delivery
- Prerequisite software

NGINX
git
node JS
dash.js
FFmpeg

NUStreaming CMCD – Project Structure

- CMCD – Aware Client
- CMCD – Aware Server
  - Request processing and parsing
  - Bandwidth allocation logic
  - Decision execution

Parameter	**Key**
Encoded bitrate	br
Buffer length	bl
Buffer starvation	bs
Deadline	dl
Measured throughput	mtp
Requested max. throughput	rtp
Object type	ot
Max buffer	com.example-bmx
Min buffer	com.example-bmn

NUStreaming – CMSD

- Extends proof-of-concept system that conforms with the client and server-side CMCD specification to support CMSD functions

- Incoming requests are scheduled based on their CMCD information request processing and parsing

- Requirements
  - All prerequisite software for CMCD
  - Additional nginx modules

NUStreaming CMSD – System Components

- Streaming Clients
  - CmcdModel.js
  - HTTPLoader.js (extended)
  - ThroughputHistory.js (modified)
- HTTP Server NGINX
  - CMCD Request processing and parsing
  - Response scheduling algorithm
  - CMSD response generation
  - Decision execution
- NetEm Network Emulator

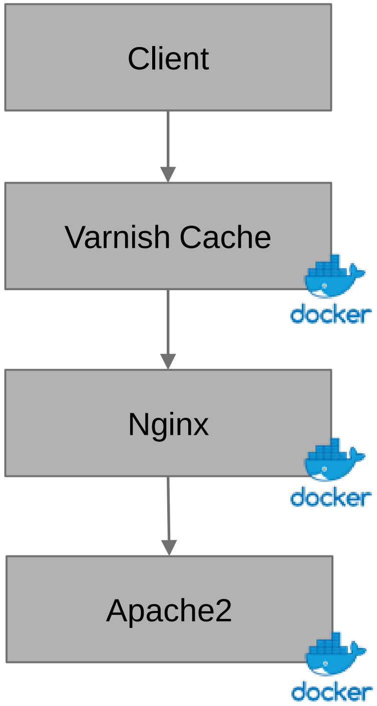
Parameter	**Key**
Encoded bitrate	br
Buffer length	bl
Buffer based delay	**com.example-dl**
Deadline	dl
Measured throughput	mtp
Requested max. throughput	rtp
segment duration	d
Max buffer	com.example-bmx
Min buffer	com.example-bmn

Unified Streaming – Origin CMSD

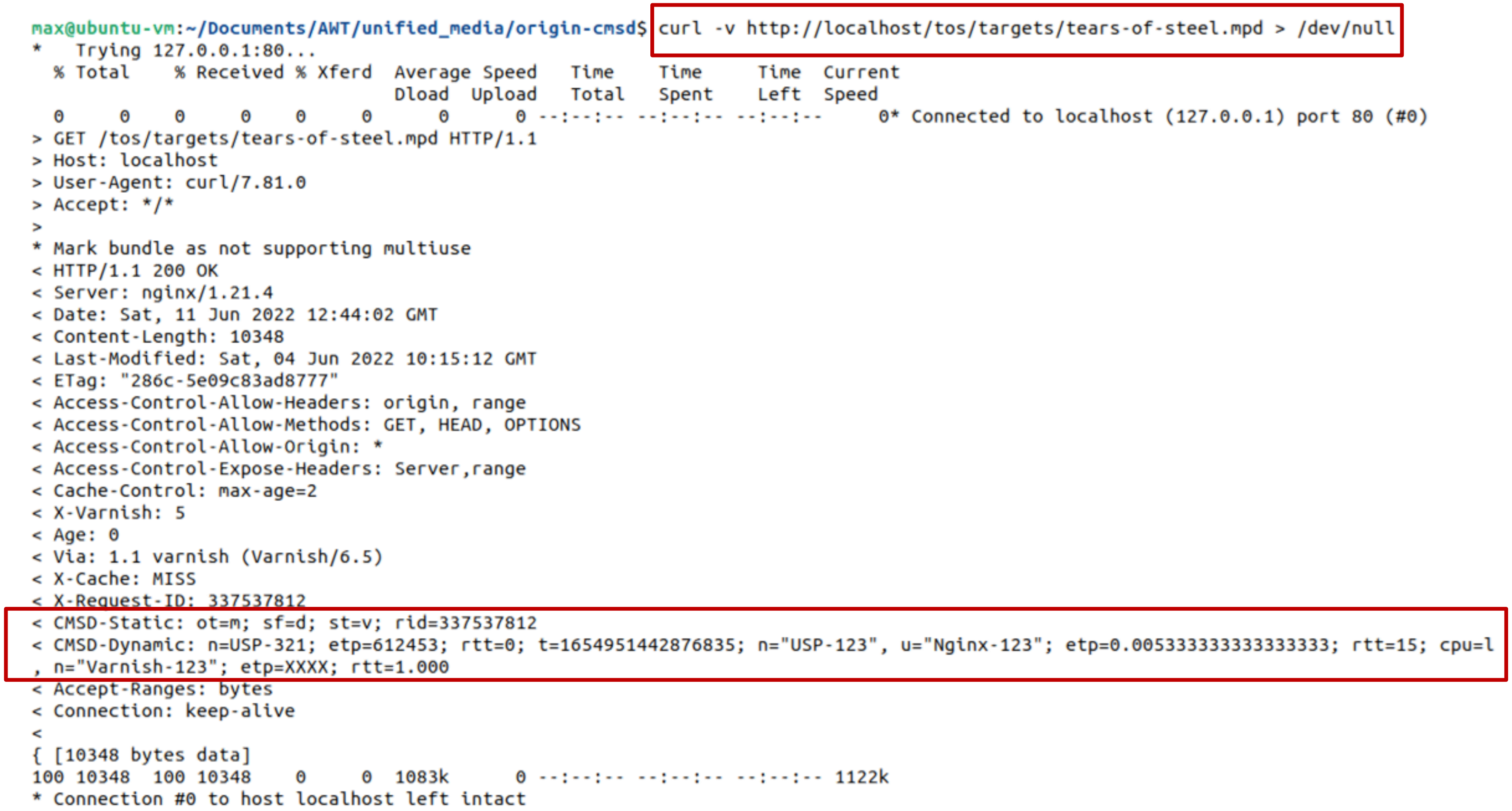
- Unified Streaming – video delivery technology company
- Contains pre-packed MPEG-DASH media fragments
- Experimental project for testing CMSD
- Requirements
  - Docker
  - Docker Compose
  - Git LFS (Large File Storage)

Unified Streaming – Project Structure

- Origin Server – Apache2
- Intermediate Server - Nginx
- Intermediate Server (edge) – Varnish Cache



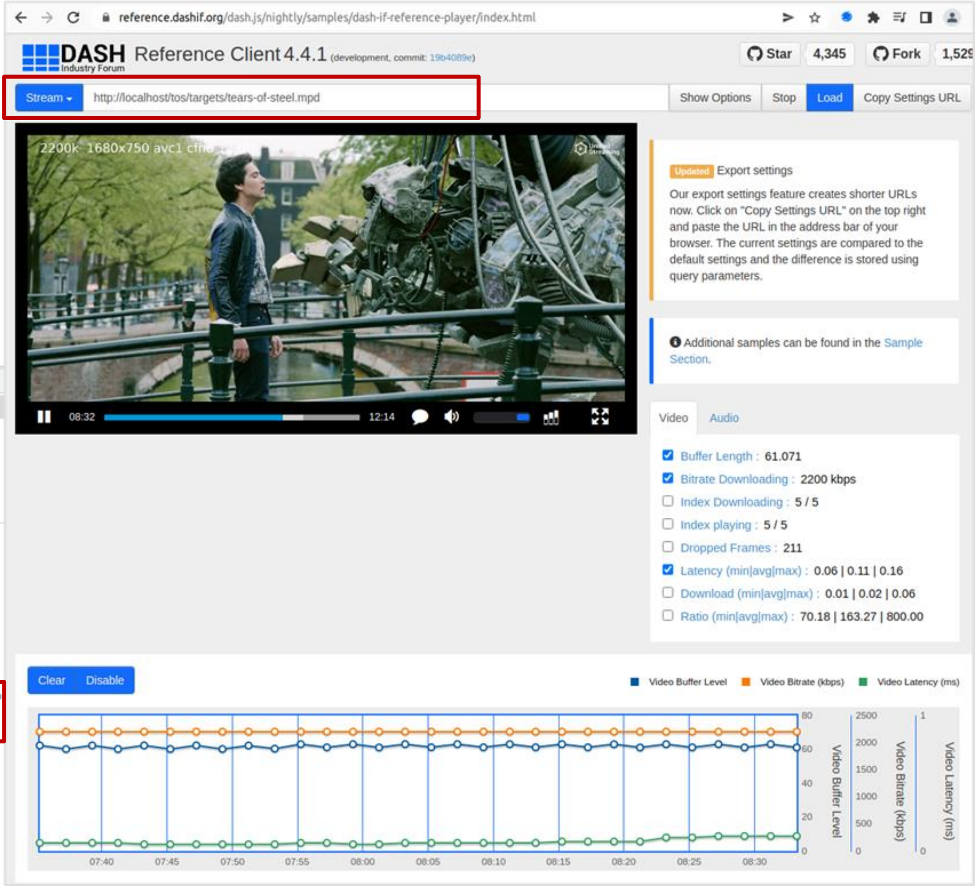
Unified Streaming – What is possible?



```
max@ubuntu-vm:~/Documents/AWT/unified_media/origin-cmsd$ curl -v http://localhost/tos/targets/tears-of-steel.mpd > /dev/null
*   Trying 127.0.0.1:80...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /tos/targets/tears-of-steel.mpd HTTP/1.1
> Host: localhost
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: nginx/1.21.4
< Date: Sat, 11 Jun 2022 12:44:02 GMT
< Content-Length: 10348
< Last-Modified: Sat, 04 Jun 2022 10:15:12 GMT
< ETag: "286c-5e09c83ad8777"
< Access-Control-Allow-Headers: origin, range
< Access-Control-Allow-Methods: GET, HEAD, OPTIONS
< Access-Control-Allow-Origin: *
< Access-Control-Expose-Headers: Server,range
< Cache-Control: max-age=2
< X-Varnish: 5
< Age: 0
< Via: 1.1 varnish (Varnish/6.5)
< X-Cache: MISS
< X-Request-ID: 337537812
< CMSD-Static: ot=m; sf=d; st=v; rid=337537812
< CMSD-Dynamic: n=USP-321; etp=612453; rtt=0; t=1654951442876835; n="USP-123", u="Nginx-123"; etp=0.005333333333333333; rtt=15; cpu=l
, n="Varnish-123"; etp=XXXX; rtt=1.000
< Accept-Ranges: bytes
< Connection: keep-alive
<
{ [10348 bytes data]
100 10348  100 10348    0     0  1083k      0 --:--:-- --:--:-- --:--:-- 1122k
* Connection #0 to host localhost left intact
```

Unified Streaming – What is possible?

- Simulate client with DASH-IF Reference Client
- Possibility to adapt bitrate
- Display metrics



reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html

DASH Industry Forum Reference Client 4.4.1 (development, commit: 19b4089e)

Star 4,345 Fork 1,529

Stream ▾ http://localhost/tos/targets/tears-of-steel.mpd

Show Options | Stop | Load | Copy Settings URL

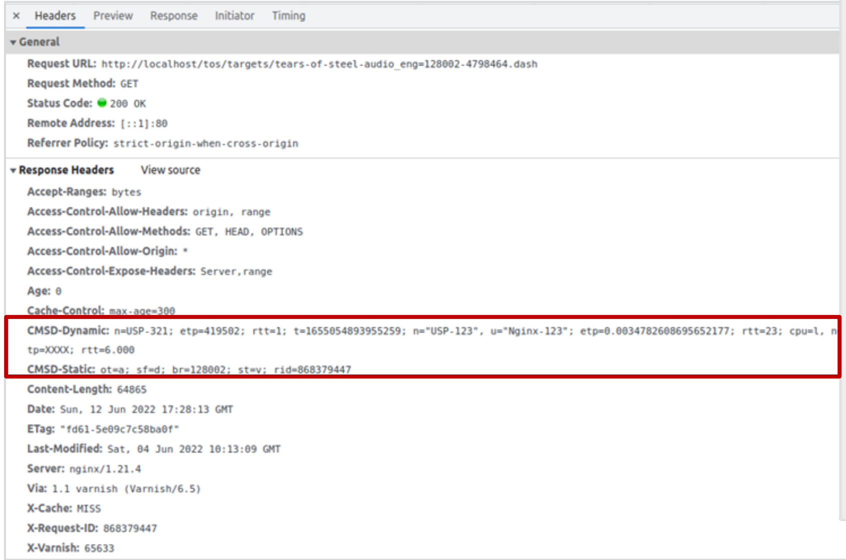
Updated Export settings

Our export settings feature creates shorter URLs now. Click on "Copy Settings URL" on the top right and paste the URL in the address bar of your browser. The current settings are compared to the default settings and the difference is stored using query parameters.

Additional samples can be found in the Sample Section.

Video | Audio

- ☑ Buffer Length : 61.071
- ☑ Bitrate Downloading : 2200 kbps
- □ Index Downloading : 5 / 5
- □ Index playing : 5 / 5
- □ Dropped Frames : 211
- ☑ Latency (min|avg|max) : 0.06 | 0.11 | 0.16
- □ Download (min|avg|max) : 0.01 | 0.02 | 0.06
- □ Ratio (min|avg|max) : 70.18 | 163.27 | 800.00



```
▾ General
  Request URL: http://localhost/tos/targets/tears-of-steel-audio_eng=128002-4798464.dash
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: [::1]:80
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers      View source
  Accept-Ranges: bytes
  Access-Control-Allow-Headers: origin, range
  Access-Control-Allow-Methods: GET, HEAD, OPTIONS
  Access-Control-Allow-Origin: *
  Access-Control-Expose-Headers: Server,range
  Age: 0
  Cache-Control: max-age=300
  CMSD-Dynamic: n=USP-321; etp=419502; rtt=1; t=1655054893955259; n="USP-123", u="Nginx-123"; etp=0.0034782608695652177; rtt=23; cpu=l, n
  tp=XXXX; rtt=6.000
  CMSD-Static: ot=a; sf=d; br=128002; st=v; rid=868379447
  Content-Length: 64865
  Date: Sun, 12 Jun 2022 17:28:13 GMT
  ETag: "fd61-5e09c7c58ba0f"
  Last-Modified: Sat, 04 Jun 2022 10:13:09 GMT
  Server: nginx/1.21.4
  Via: 1.1 varnish (Varnish/6.5)
  X-Cache: MISS
  X-Request-ID: 868379447
  X-Varnish: 65633
```

Unified Streaming – What is implemented?

- CMSD-Static header
- CMSD-Dynamic header
- 9/23 key-value pairs

Description	Key Name
Timestamp	t
Origin identifier	n
Object type	ot
Stream type	st
Encoded bitrate	br
Request ID	rid
CPU load	cpu
Estimated Throughput	etp
Round Trip Time	rtt

Unified Streaming – What is not implemented?

- Client part
- CMCD and CMSD interaction
- Key-value pairs

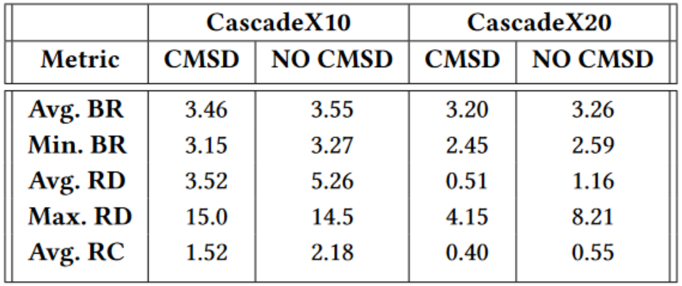
Description	Key Name
Max suggested bitrate	mb
Next Object Response	nor
Next Range Response	nrr
Object duration	d
Target latency	tl
…	…

NUStreaming-CMSD vs. Unified Streaming

Criteria	NUStreaming-CMSD	Unified Media
CMCD-Keys	6/18	Not implemented
CMSD-Keys	1*/23	9/23
CMSD Header-Types	Dynamic	Dynamic, Static
Features	Experiment**	

*com.example-dl (buffer-based delay) - not mentioned in working draft

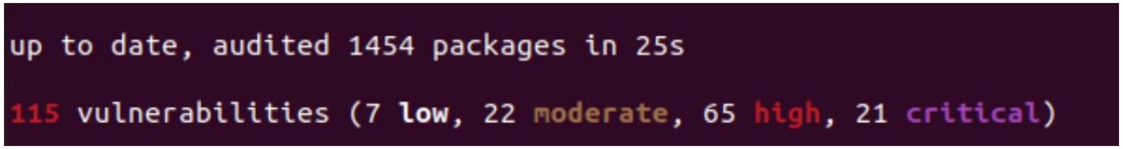
**performance measurement, which proofs positive impact of CMSD on rebuffering duration



	CascadeX10		CascadeX20	
Metric	**CMSD**	**NO CMSD**	**CMSD**	**NO CMSD**
Avg. BR	3.46	3.55	3.20	3.26
Min. BR	3.15	3.27	2.45	2.59
Avg. RD	3.52	5.26	0.51	1.16
Max. RD	15.0	14.5	4.15	8.21
Avg. RC	1.52	2.18	0.40	0.55

Challenges faced during the project

- NUStreaming setup procedure
  - Many requirements
  - Version conflicts

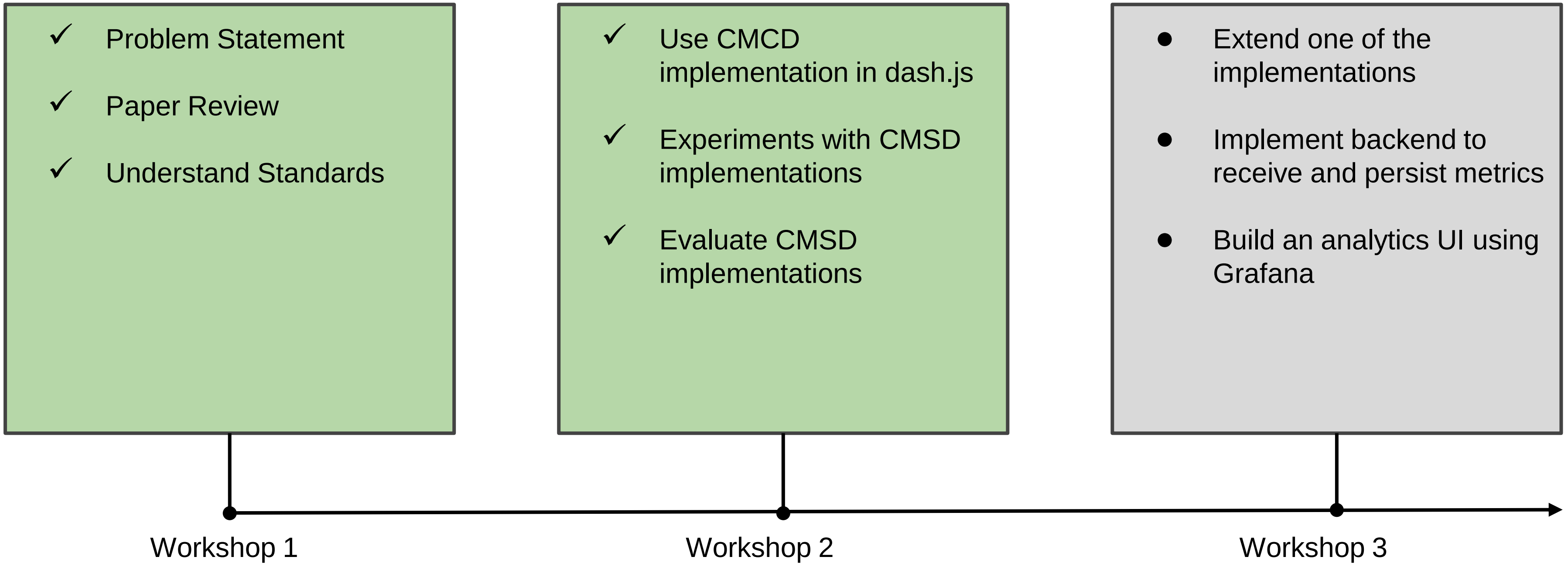


```
up to date, audited 1454 packages in 25s

115 vulnerabilities (7 low, 22 moderate, 65 high, 21 critical)
```

- Our setup instruction
  - https://github.com/karmatothex/awt-pj-ss22-streaming-analytics-using-cmcd-and-cmsd-1/wiki/Setup-Ubuntu-CMSD-Implementation

Schedule and Next Steps



✓ Problem Statement

✓ Paper Review

✓ Understand Standards

✓ Use CMCD implementation in dash.js

✓ Experiments with CMSD implementations

✓ Evaluate CMSD implementations

- Extend one of the implementations
- Implement backend to receive and persist metrics
- Build an analytics UI using Grafana

Thanks for your attention!

References

- Bentaleb, A., Lim, M., Akcay, M. N., Begen, A. C., & Zimmermann, R. (2021, July). Common media client data (CMCD) initial findings. In Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (pp. 25-33).
- DASH-IF Reference Client. Retrieved 12 June 2022, from https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html
- Lim, M., Akcay, M. N., Bentaleb, A., Begen, A. C., & Zimmermann, R. (2022, March). The benefits of server hinting when DASHing or HLSing. In Proceedings of the 1st Mile-High Video Conference (pp. 52-55).
- NUStreaming. CMCD-DASH. Retrieved 12 June 2022, from https://github.com/NUStreaming/CMCD-DASH
- NUStreaming. CMSD-DASH. Retrieved 12 June 2022, from https://github.com/NUStreaming/CMSD-DASH
- Unified Streaming. Origin CMSD. Retrieved 12 June 2022, from https://github.com/unifiedstreaming/origin-cmsd

References

- Unpublished Working Draft. Common Media Server Data (CMSD). Retrieved 12 June 2022, from https://docs.google.com/document/d/1BlTHfbF2VGSlA4vLx1fMssWqiGWYuzhmTfQ8VeyxF8g/edit#heading=h.w4dwbs4gi4x