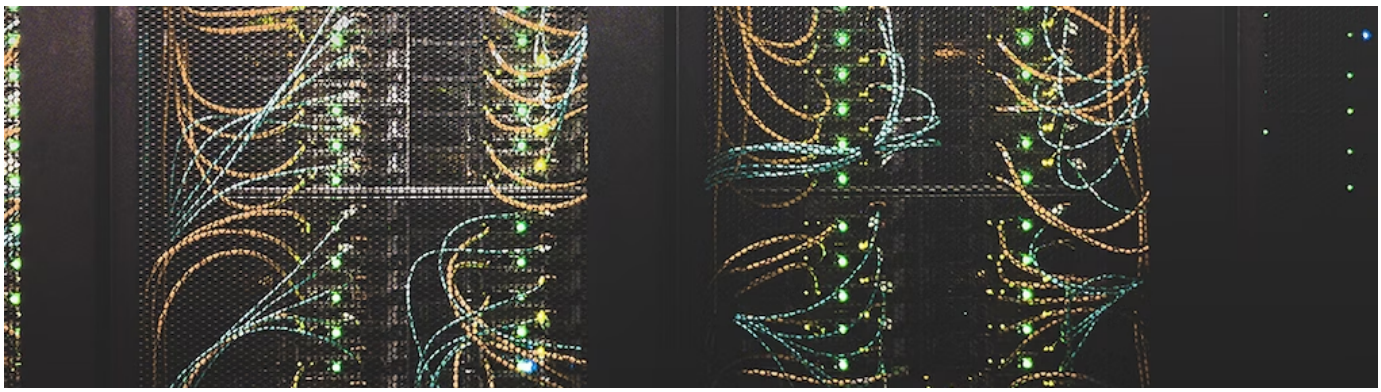
#STREAMING FEB 10, 2021

Keep the stream live, despite any infrastructure failure

PAVEL ŠVEJDA

CDN IOS APPLE ANDROID SMARTTV WEBSITE



At Showmax we deliver top-shelf subscription video on demand services. Our ultimate goal is to offer the right content, at the right time, on the right platform, that works...every time. Since no infrastructure is perfect, there always are issues. To maintain a high-quality stream, we need to effectively foresee and prepare to address all of the peaks and infrastructure failures. Nobody wants to watch a story that is constantly interrupted because of a server downtime.

This is obviously a very important issue for us, and we have managed to identify one of the major root causes. When the stream starts, the path to the content is fixed and kept for the duration of the playback. To keep the stream live in the event of server failure, we need a backup path — a so-called second backup stream.

Quick streaming introduction

At Showmax we use a lot of terms related to video streaming, but, to follow this blog post, it's enough to understand the basics. Let's quickly recap the most important ones.

Adaptive streaming (also known as adaptive bitrate streaming) is a technology designed to deliver video to the user over http, in the form of short video chunks that are continuously downloaded just a few moments before they are actually played. The player automatically selects the right video quality of the next video chunks based on current network throughput. More can be found **here**.

file, for example.

HLS is an HTTP-based adaptive bitrate streaming communications protocol developed by Apple Inc. More can be found **here**.

DASH also known as MPEG-DASH, is an adaptive bitrate streaming technique that enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers. More can be found **here**.

What is a second stream backup?

Second stream backup is a well-known feature of players that helps users to continue watching videos in case of a server failure. General practice is that the playlist users receive when they want to play an asset contains just one path to the desired content. More to what we have said above, second stream backup means that we sneak another path to the desired content and in case of failure of the first stream the player will switch to the second (backup) stream. The players then play from the backup. Some of them even periodically check the status of the primary source, and when it is available again, they switch back to it. This can really secure a good user experience even in case of an outage.

Our approach

Within the documentation for **DASH** and **HLS**, it is clearly-stated how the second stream should be configured in playlists, and it's not too complicated to follow.

In the beginning, we tried a quick modification with nginx and our locally modified playlist where we injected a second stream. We tested this on our website (Shaka Player) using Chrome, and it worked.

On Android devices we use ExoPlayer. Unfortunately, when we checked with the support the options we have, we received some bad news. ExoPlayer has a ~5 year old **request** for this feature that has not been resolved yet because, obviously, it's not a major feature.

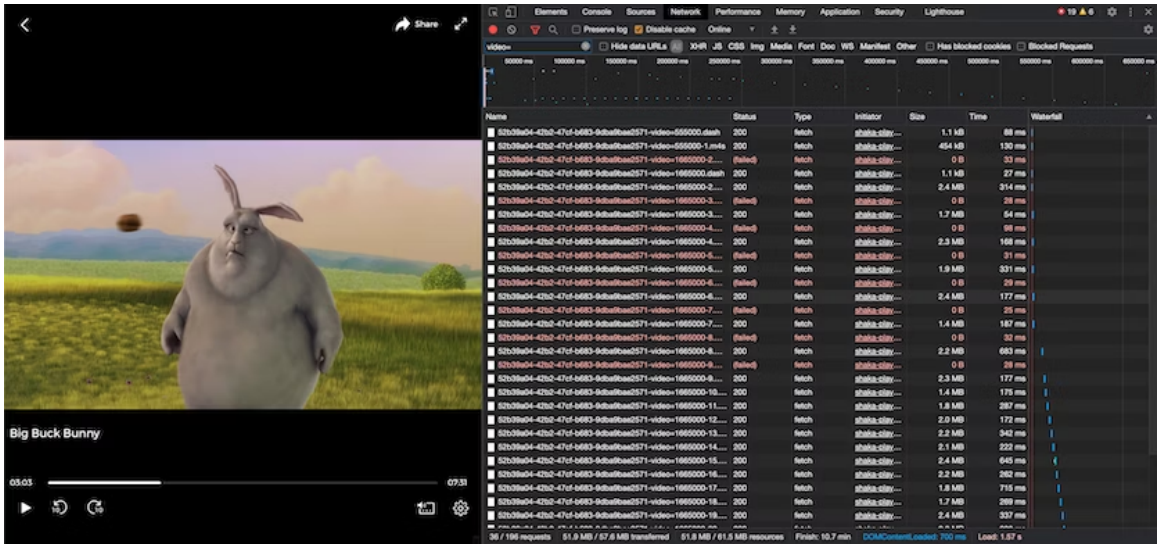
So, we had to get back to the beginning and test all the players on our platform to make sure they supported this feature. We implemented a proof of concept on our backends to test it. Our staging environment is very similar to our production environment, so we deployed the changes there, without putting production in danger.

How do you test server failure without actually breaking the server?

The short answer: Use a bunch of proxies you already have!

that we could test to see if the players would try another source and continue playing despite encountering a critical failure on the first stream.

On the picture below, you can see that the first nine chunks failed to load (the red lines) from the primary source so the player used the backup (white lines after the red ones) and loaded them from there.



Backwards compatibility

A big question we had was: “Do we break the players on the unsupported devices when we add the second stream backup feature to playlists?”

Luckily the answer is no because the second stream is either taken into account or ignored by the player. This means we have backward compatible changes (and yes we tested it).

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" type="static">
    <Period id="1" duration="2520635">
        <BaseURL>https://primary_stream/dash</BaseURL>      <-- Primary stream
        <BaseURL>https://backup_stream/dash</BaseURL>       <-- Backup stream used in case of primary stream failure
        <AdaptationSet audioSamplingRate="44100" mimeType="audio/mp4" contentType="audio" segmentAlignment="True">
            <AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
            <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc"/>
            <ContentProtection />
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <SegmentTemplate timescale="44100" duration="441000" media="id.m4s" initialization="id.dash"/>
            <Representation id="audio=128001" bandwidth="128001"/>
        </AdaptationSet>
        <AdaptationSet sar="1:1" mimeType="video/mp4" maxBandwidth="1353000" maxWidth="1280" maxHeight="720" segmentAlignment="True">
            <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011" value="cenc"/>
            <ContentProtection />
            <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
            <SegmentTemplate timescale="600" duration="6000" media="id.m4s" initialization="id.dash"/>
            <Representation width="256" height="144" scanType="progressive" id="video_1=67000" bandwidth="67000"/>
            <Representation width="384" height="216" scanType="progressive" id="video_1=98000" bandwidth="98000"/>
            <Representation width="512" height="288" scanType="progressive" id="video_1=175000" bandwidth="175000"/>
            <Representation width="640" height="360" scanType="progressive" id="video_1=261000" bandwidth="261000"/>
            <Representation width="768" height="432" scanType="progressive" id="video_1=446000" bandwidth="446000"/>
            <Representation width="1024" height="576" scanType="progressive" id="video_1=793000" bandwidth="793000"/>
            <Representation width="1280" height="720" scanType="progressive" id="video_1=1353000" bandwidth="1353000"/>
        </AdaptationSet>
```

The result

PlayerVersionSupport**ShakaPlayer**3.0.8**ExoPlayer**2.12.3**Apple native player**All Apple devicesSmartTV native playerNot Applicable, is vendor and version specific

Player	Version	Support
ShakaPlayer	3.0.8	Supported
ExoPlayer	2.12.3	Unsupported
Apple native player	All Apple devices	Supported
SmartTV native player	Not Applicable, is vendor and version specific	Unsupported

Instant improvements to UX

These results meant that our proof of concept could be polished and deployed to production. Without much engineering work on the clients, we were rapidly able improve user experience — not for everyone as we would like, but at least for devices that support the second stream.

For the devices without the backup stream support, we have to implement our own version of the second stream or help their official maintainers to add the support.

We eventually want this feature on all of our platforms, so stay tuned for what we will come up with!

You might be **interested**

Pytest Appium: iOS/Android Setup BrowserStack and Simulators

#APPS 14 DECEMBER 2023

DAVID ALDORF

Inspirations from Droidcon Lisbon

#APPS 06 DECEMBER 2023

MICHAL URSINY

Integration Tests

#APPS 21 NOVEMBER 2023

OLEH PSHENYCHNYI