

Свой плеер для DASH: вошли и вышли, приключение на 20 минут

Ольга Попова,
Yandex Infrastructure

Ольга Попова



Разработчик интерфейсов
в Yandex Infrastructure

Круг Хармона



План



1. MPEG-DASH и shaka-player

2. Что не так с shaka-player?

3. Наше решение – YaSP

4. Архитектура

5. Подводные камни

6. Что с метриками?

7. Эксперименты

8. Выводы

HLS vs DASH

HLS

DASH

master.m3u8 – основной плейлист
level.m3u8 – плейлист дорожки

.mpd - плейлист

Разработчик: Apple Inc.

Разработчик: DASH IF

Нативная поддержка

—

—

Ежеквартальная



MPEG-DASH

Популярные библиотеки для MPEG-DASH

Перечисли библиотеки для проигрывания MPEG-DASH видео в браузере с открытым исходным кодом

Video.js, Shaka Player, Dash.js, JW Player, MediaElement.js, Flowplayer, Kaltura Media Player, Cineast, Plex, Media Source Extensions (MSE).



video.js



dash.js

shaka-player

dash.js

rx-player

dash.js vs shaka-player vs rx-player

	dash.js	shaka-player	rx-player
Разработчик	DASH IF	Google	Canal+
Имплементация стандарта	Полная	Частичная	Частичная
Вес сборки*	214кБ	177кБ	149кБ
Поддерживаемые стандарты	DASH, MSS	DASH, HLS, MSS	DASH, MSS

shaka-player

Немного о Shaka

Умеет в

- DASH, HLS и MSS (VOD-only)
- VOD и Live
- DRM

Особенности

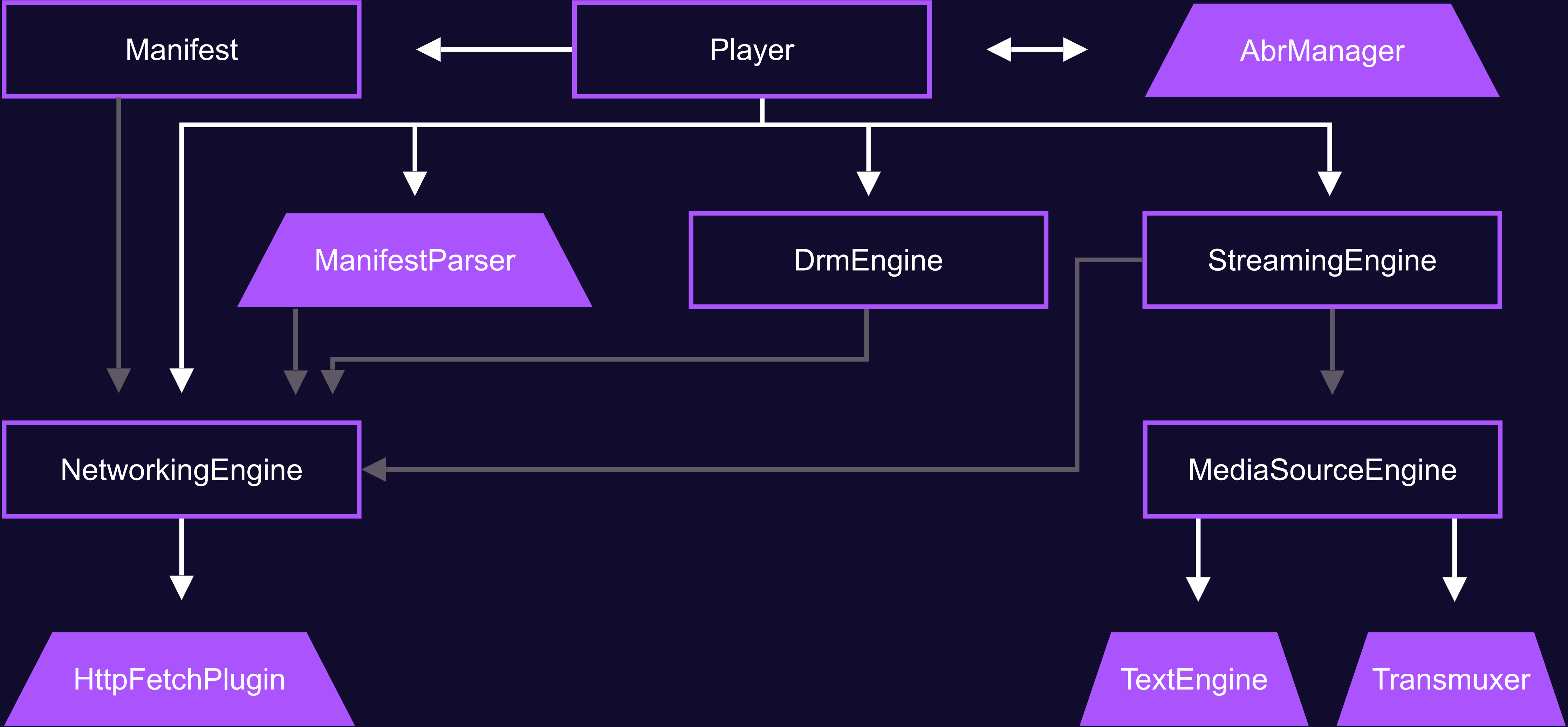
- Dependency injection
- Closure Compiler

Closure Compiler

```
goog.provide('shaka.Player');  
goog.require('goog.asserts');
```

```
/**  
 * @event shaka.Player.StateChangeEvent  
 * @description Fired when the player changes load states.  
 * @property {string} type  
 *   'onstatechange'  
 * @property {string} state  
 *   The name of the state that the player just entered.  
 * @exportDoc  
 */
```

Архитектура Shaka

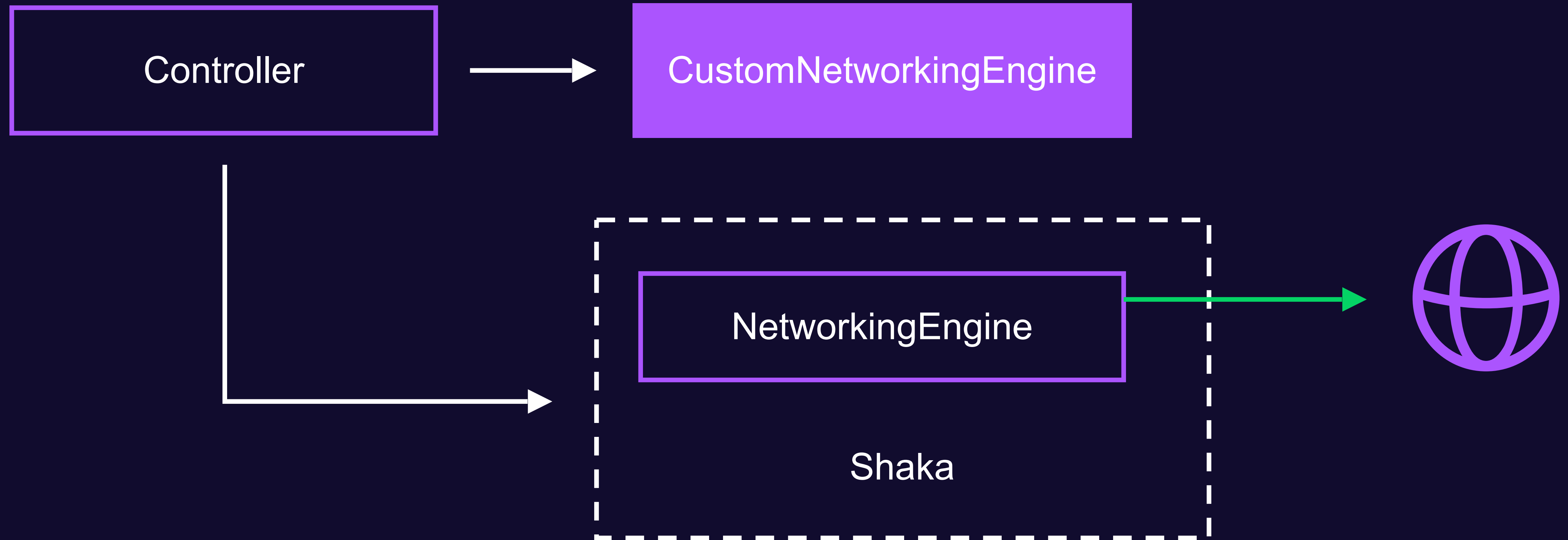






Наши доработки в shaka-player

Свой сетевой слой



Multi-BaseURL

```
<MPD xmlns:cenc="urn:mpeg:cenc:2013" xmlns:mspr="urn:microsoft:playready"
xmlns="urn:mpeg:dash:schema:mpd:2011" profiles="urn:mpeg:dash:profile:isoff-
live:2011" minBufferTime="PT4.000S" type="static"
mediaPresentationDuration="PT1H53M21.795S">
```

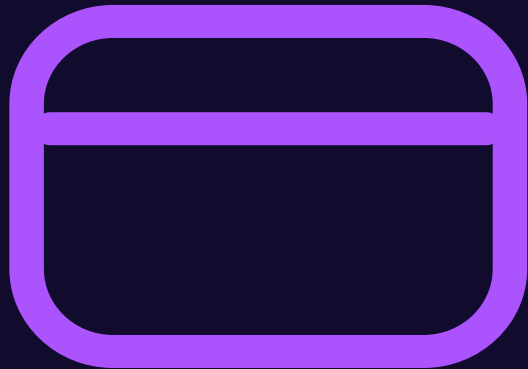
```
  <BaseURL>https://base-url-1/</BaseURL>
```

```
  <BaseURL>https://base-url-2/</BaseURL>
```

```
  <BaseURL>https://base-url-3/</BaseURL>
```

```
  <BaseURL>https://base-url-4/</BaseURL>
```

Multi-BaseURL



Player



CDN 1

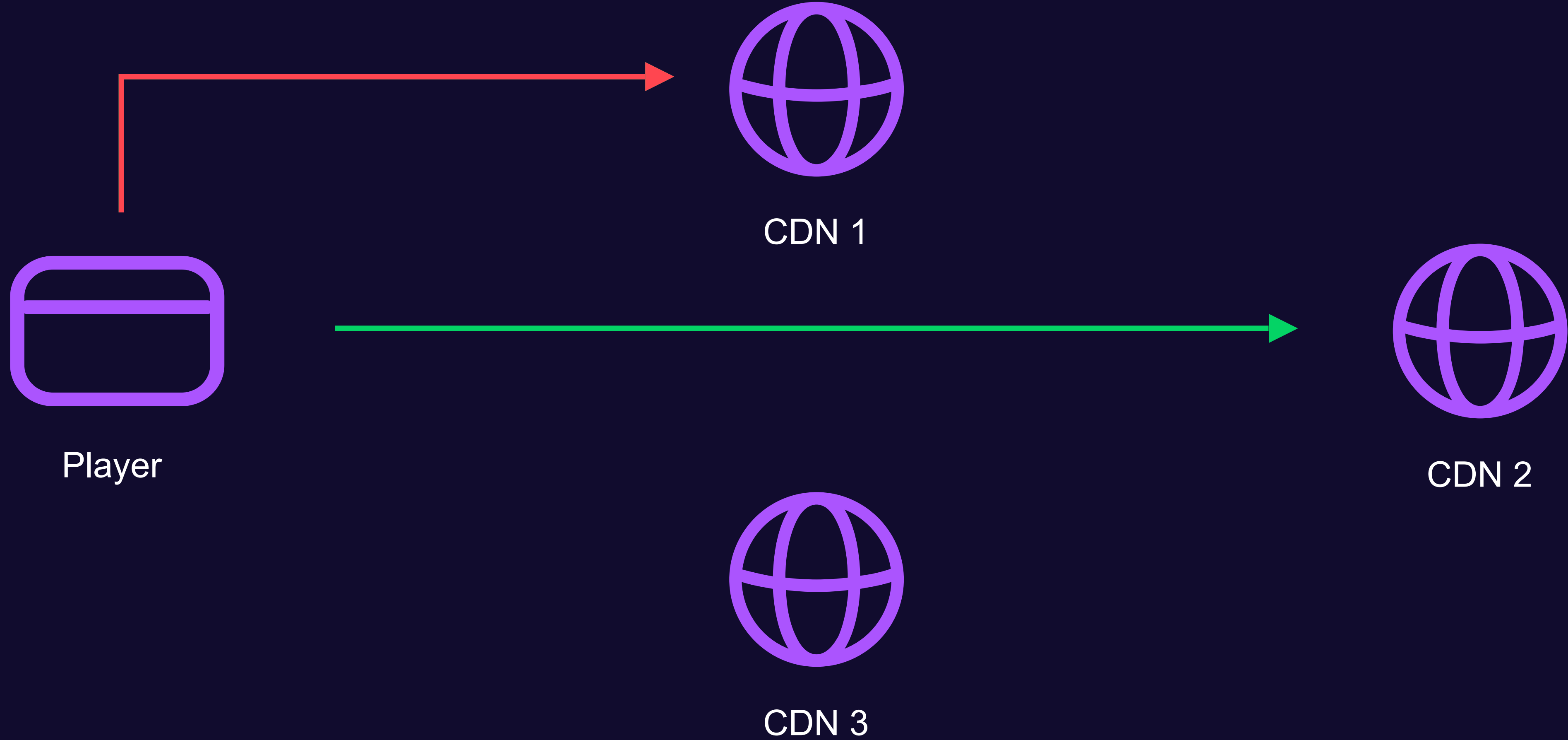


CDN 2

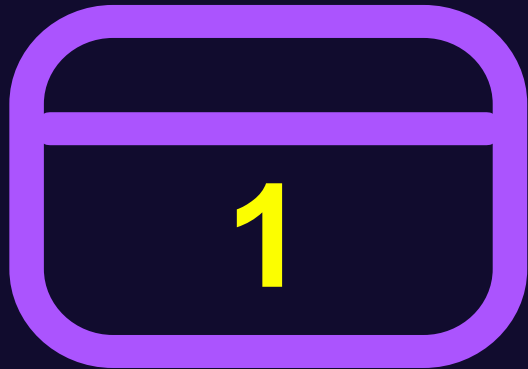


CDN 3

Multi-BaseURL



Multi-BaseURL



Запрашивает
сегмент №1



CDN 1

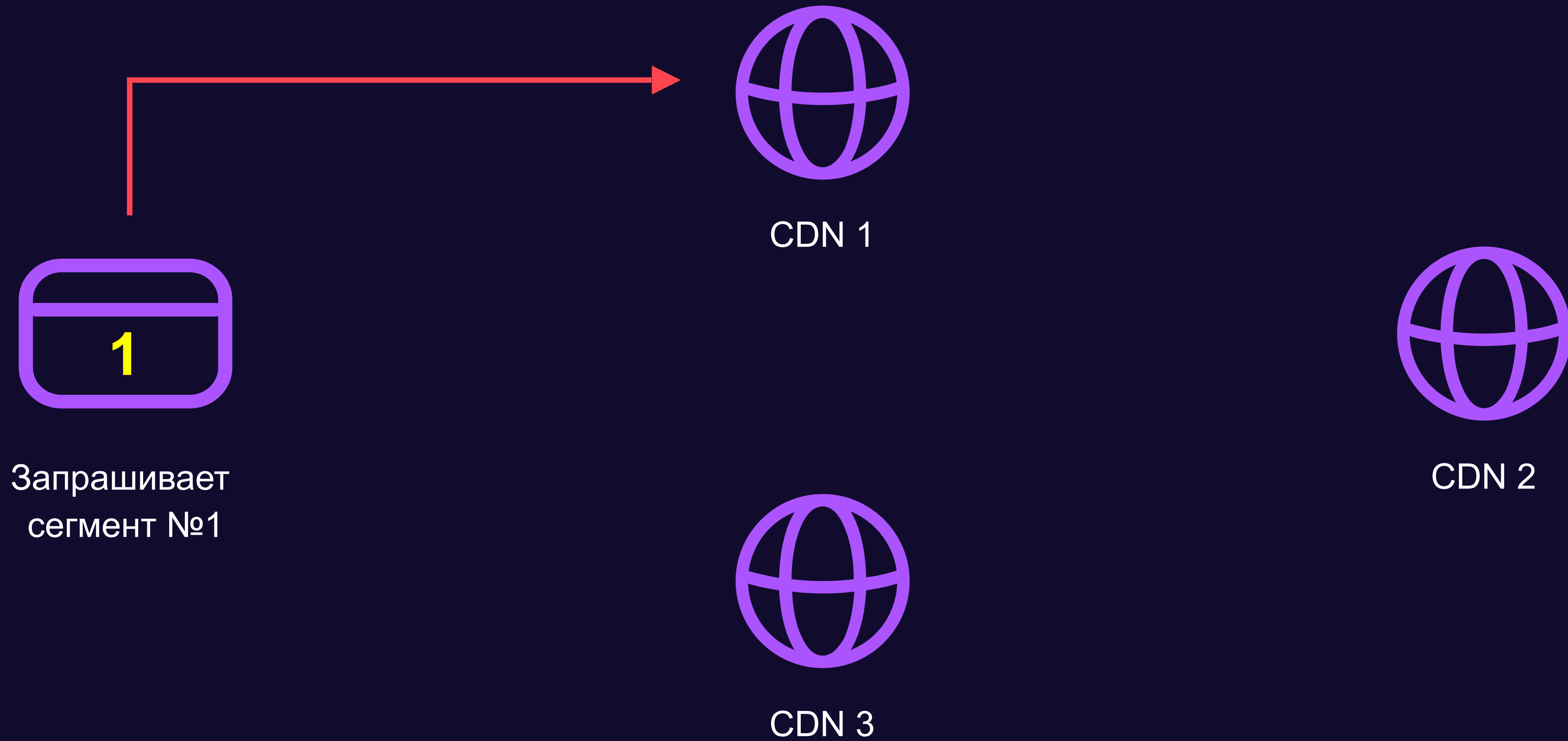


CDN 2

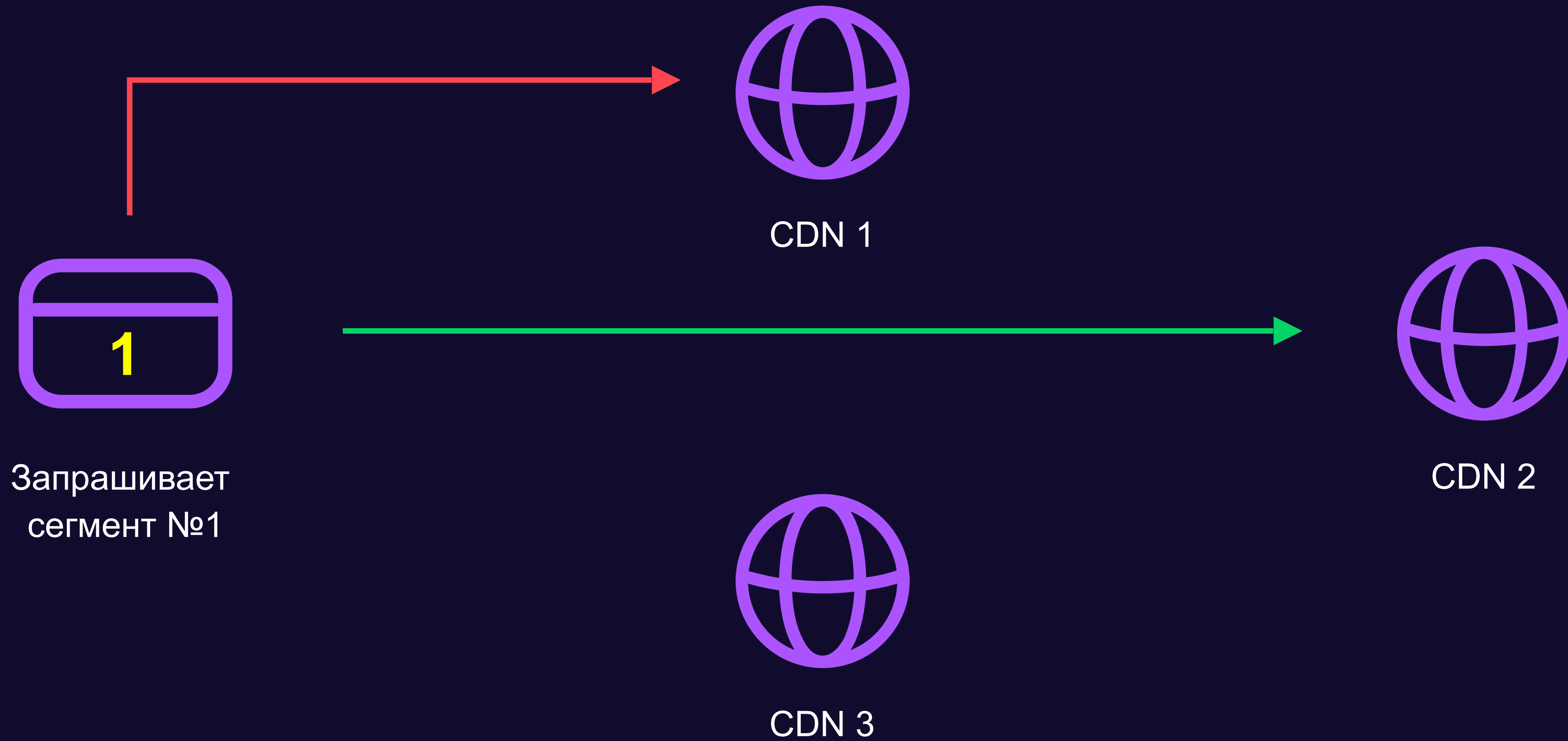


CDN 3

Multi-BaseURL



Multi-BaseURL



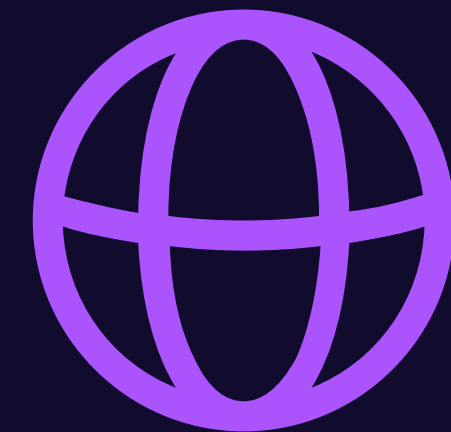
Multi-BaseURL



Запрашивает
сегмент №2



CDN 1

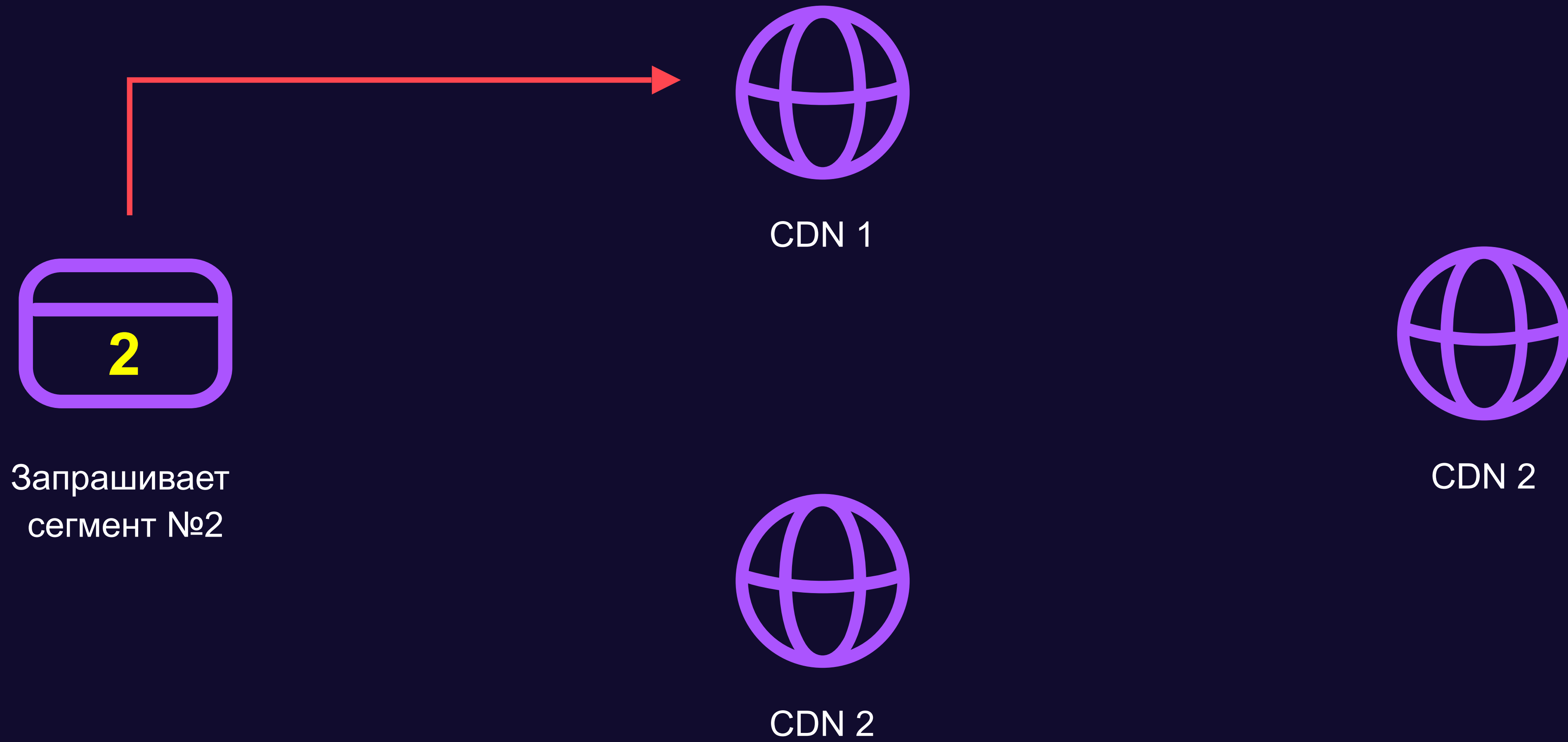


CDN 2

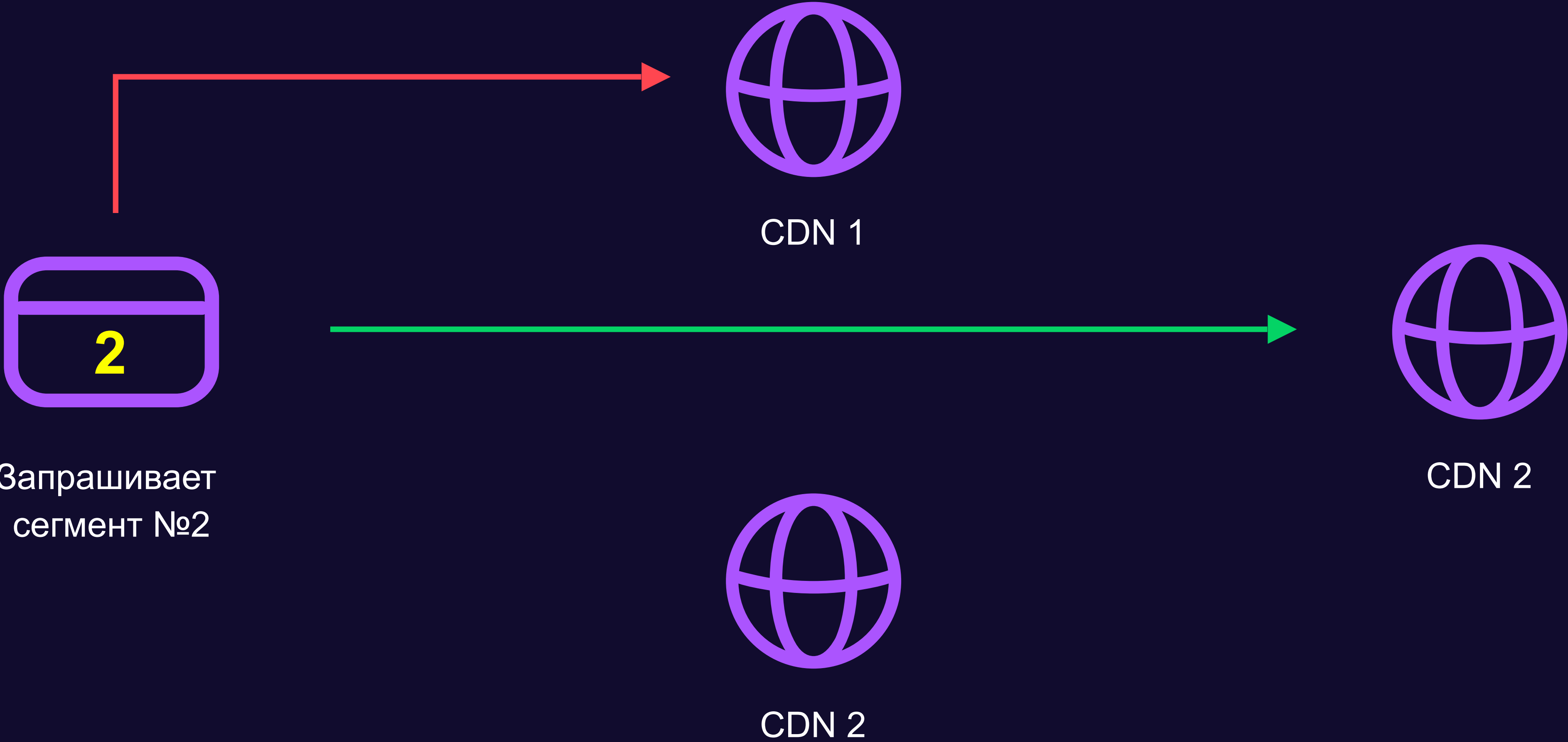


CDN 2

Multi-BaseURL



Multi-BaseURL



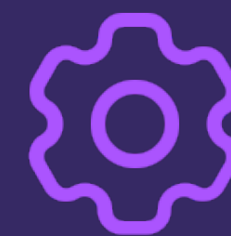
Content Steering



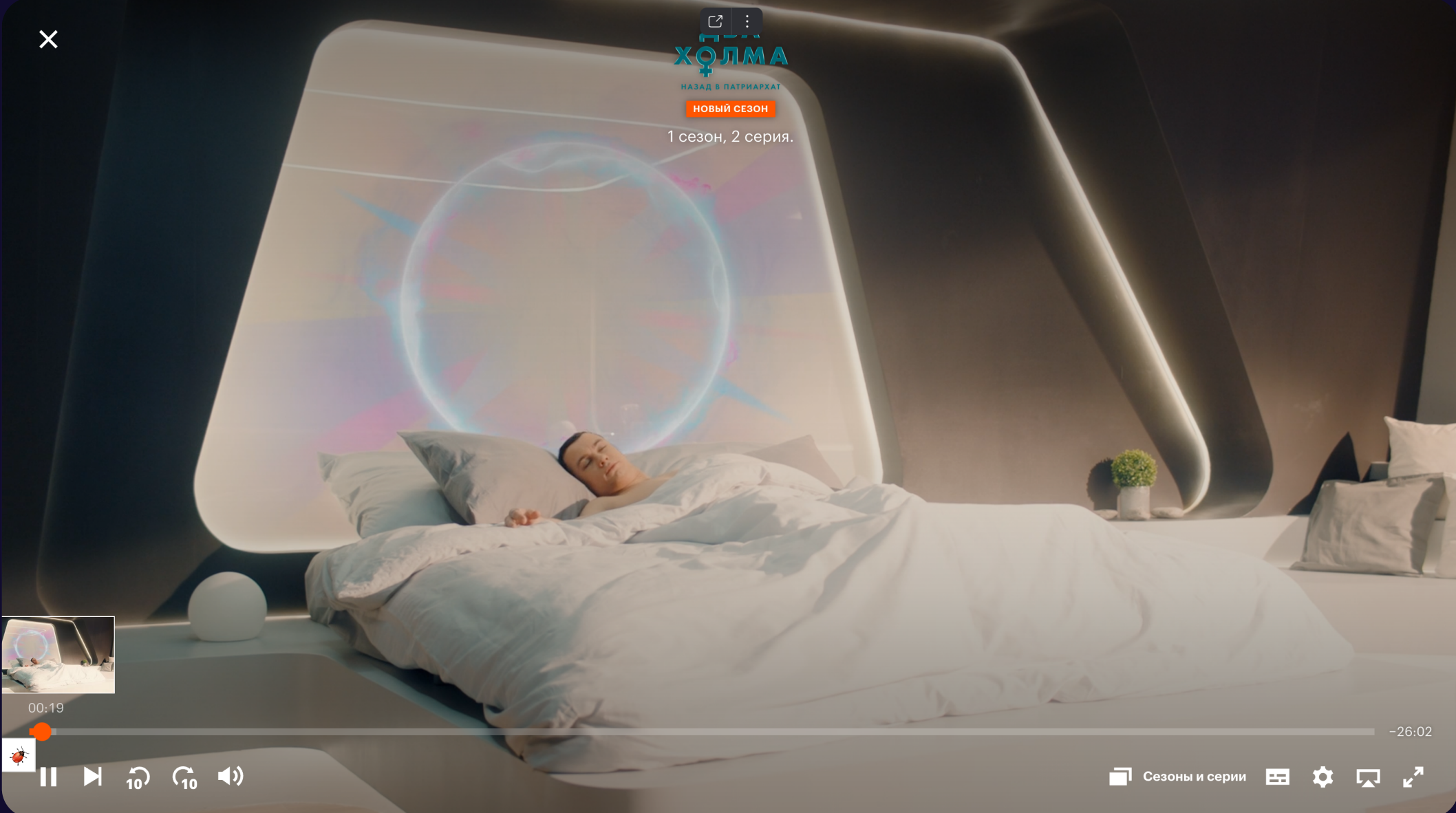
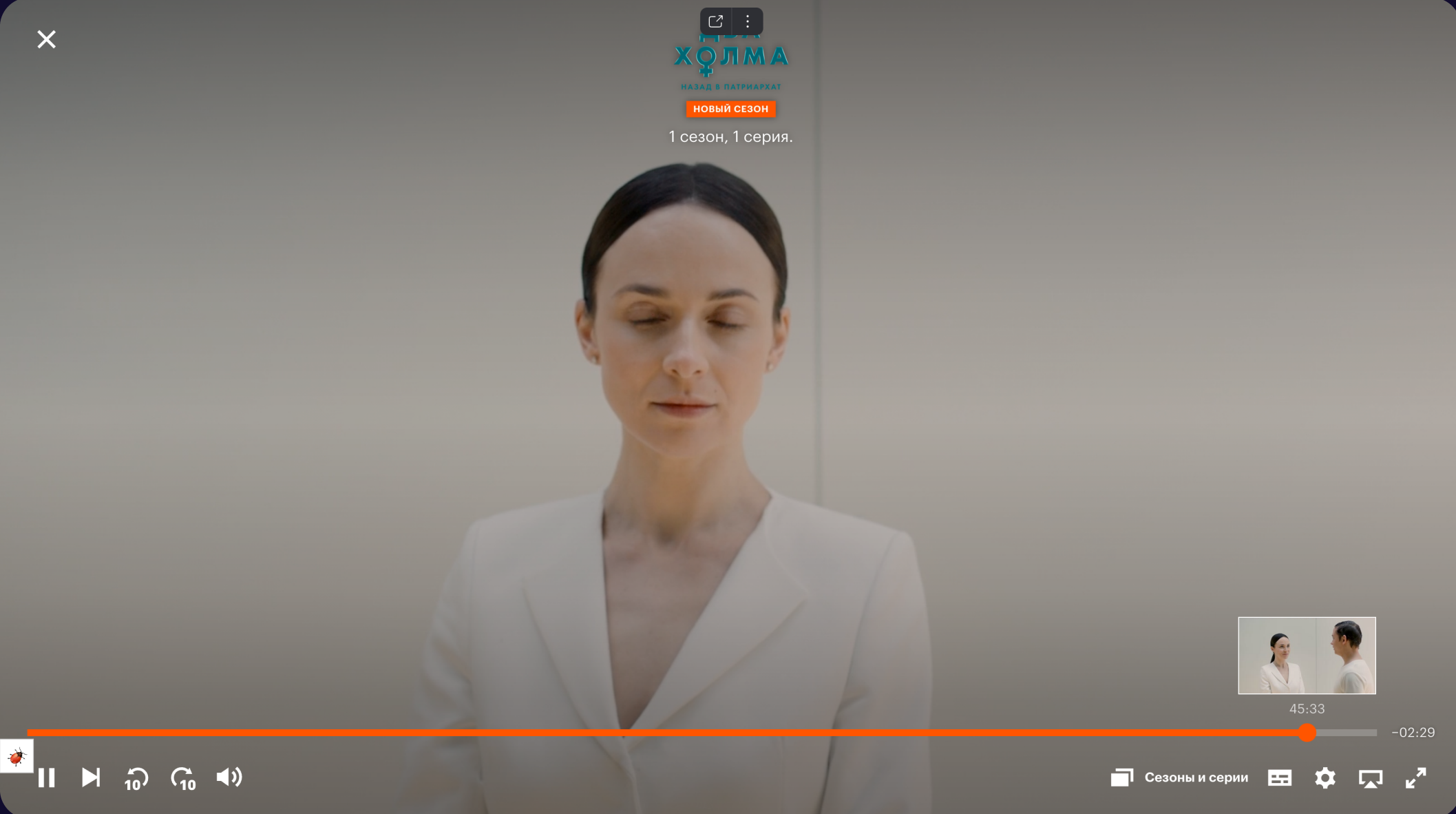
<Location>

Зачем?

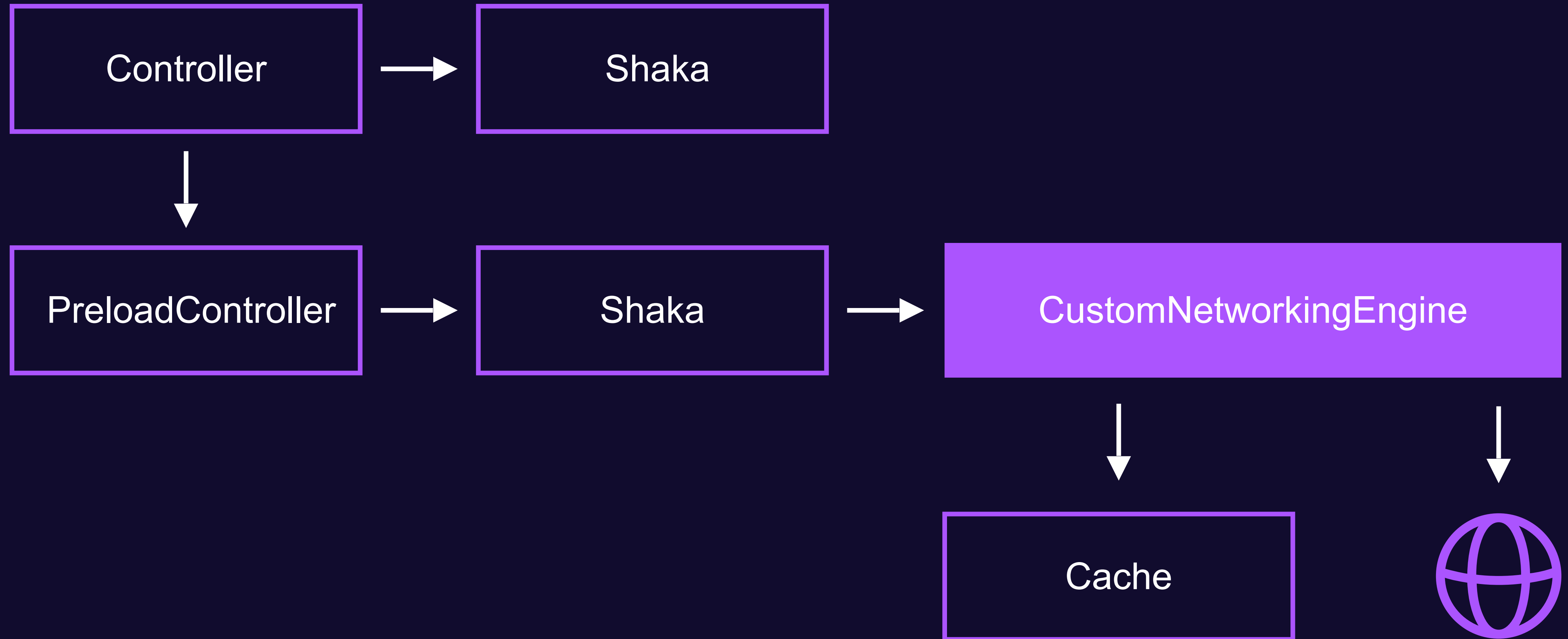
Фиксация параметров, с которыми ходим в бек за обновлением manifest



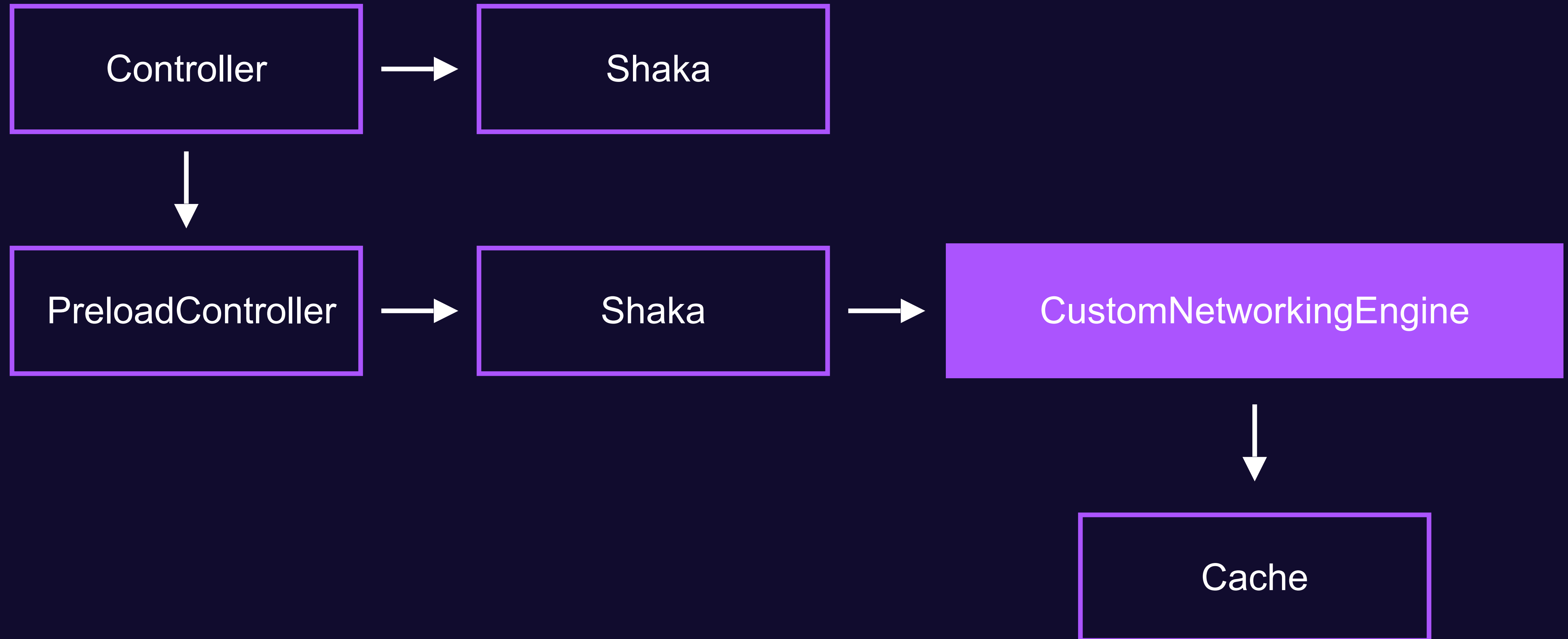
Предзагрузка



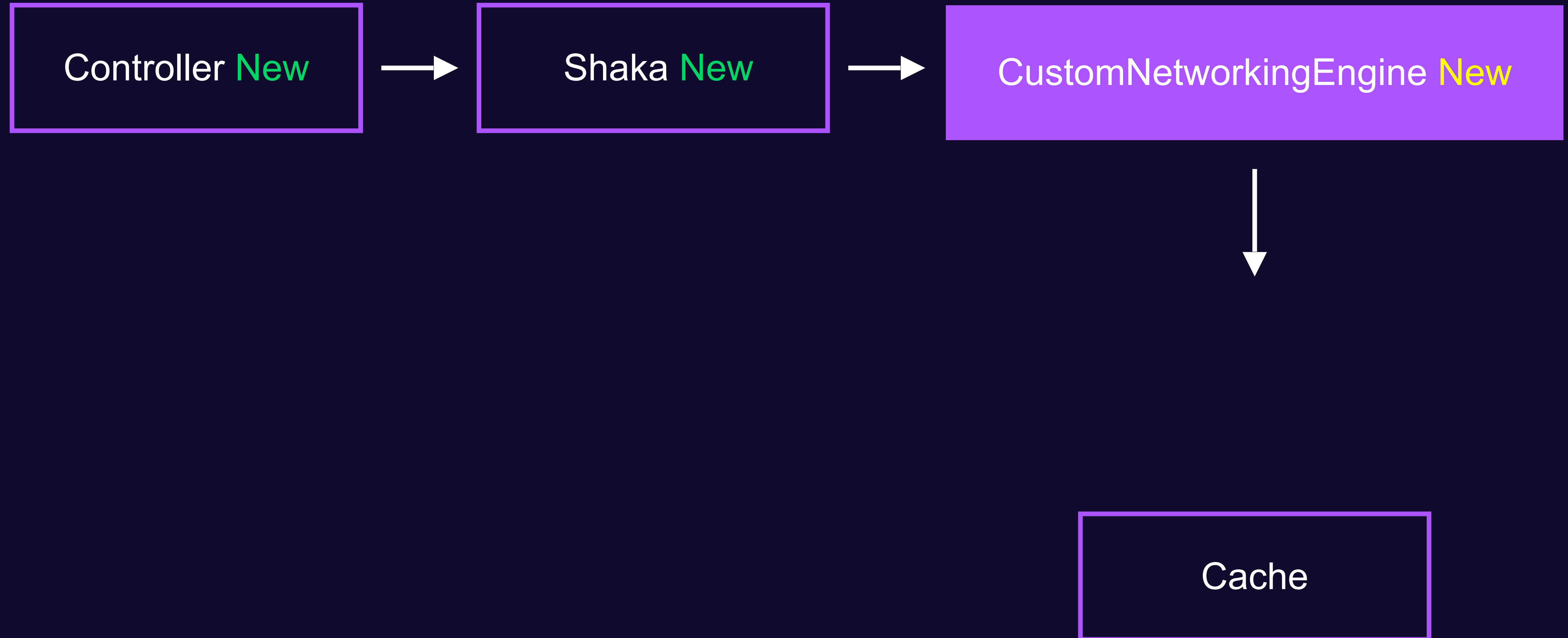
Предзагрузка



Переключение на предзагруженный source



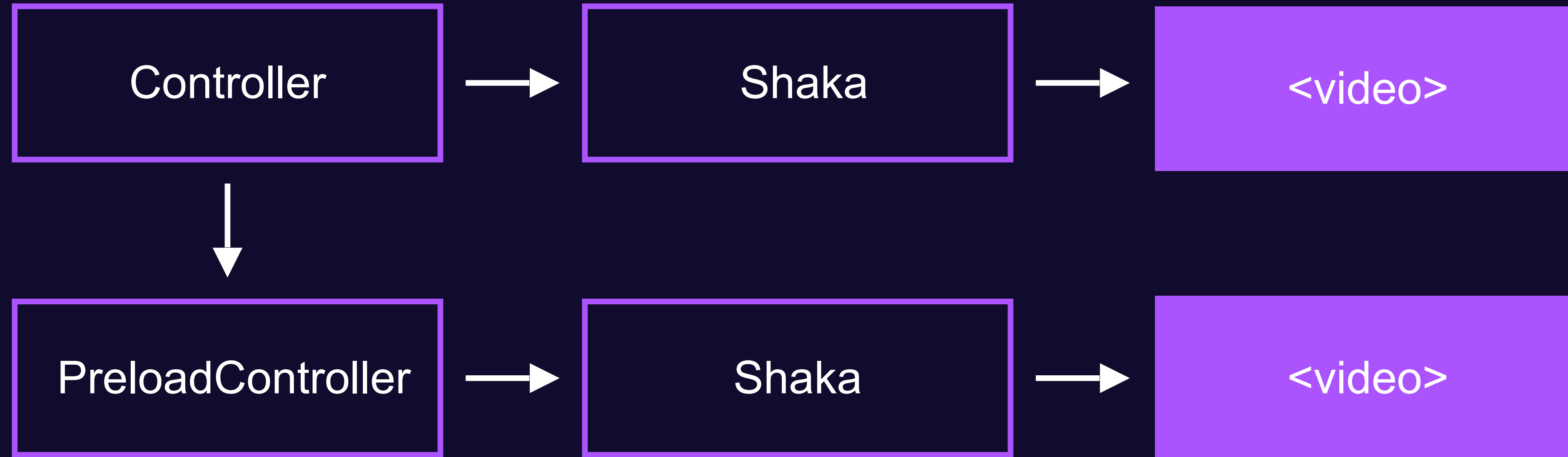
Переключение на предзагруженный source



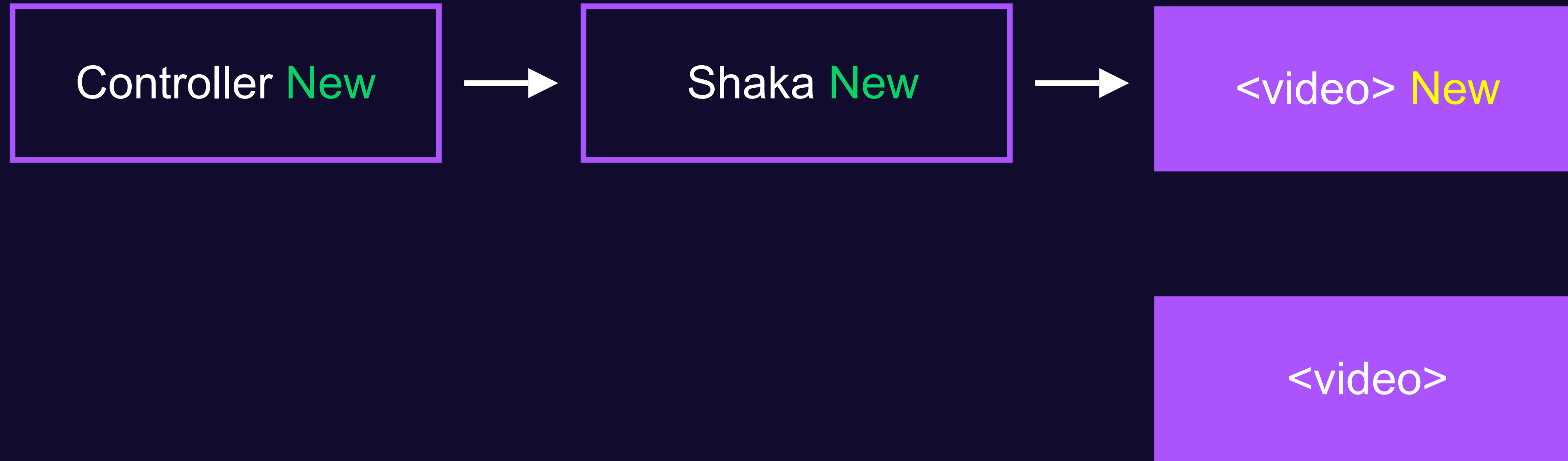
Shaka-player сразу складывает скачанные данные в MSE



Переключение на предзагруженный source

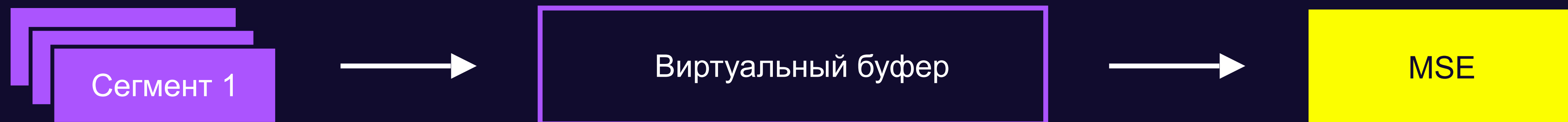


Переключение на предзагруженный source



Предзагрузка в hls.js

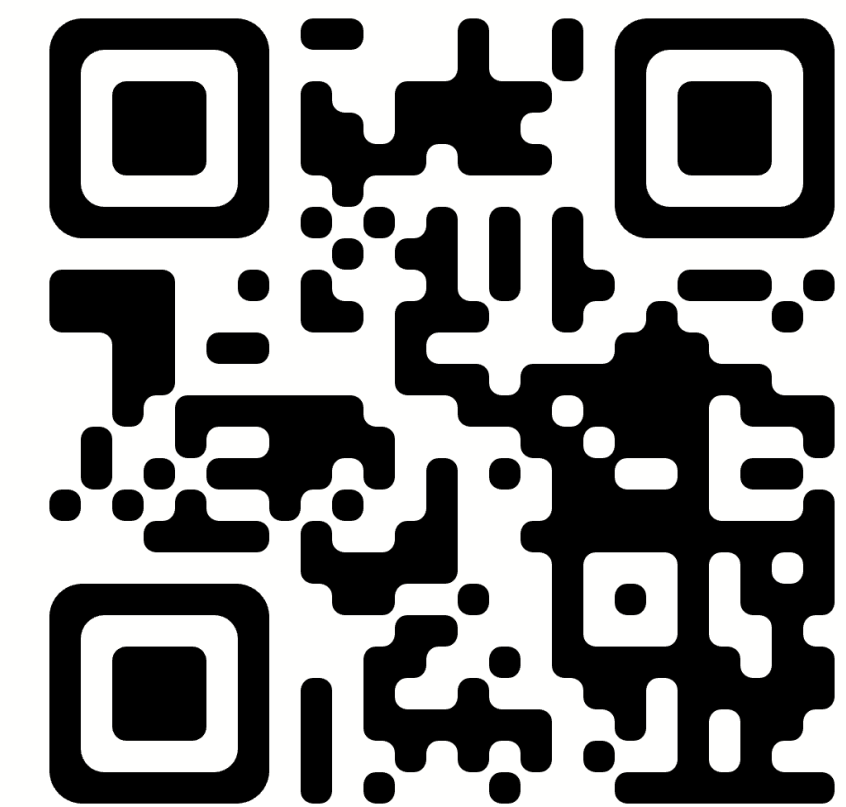
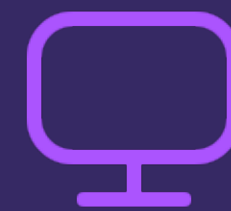
```
var hls = new Hls();  
hls.loadSource(videoSrc);  
hls.attachMedia(video);
```



ABR

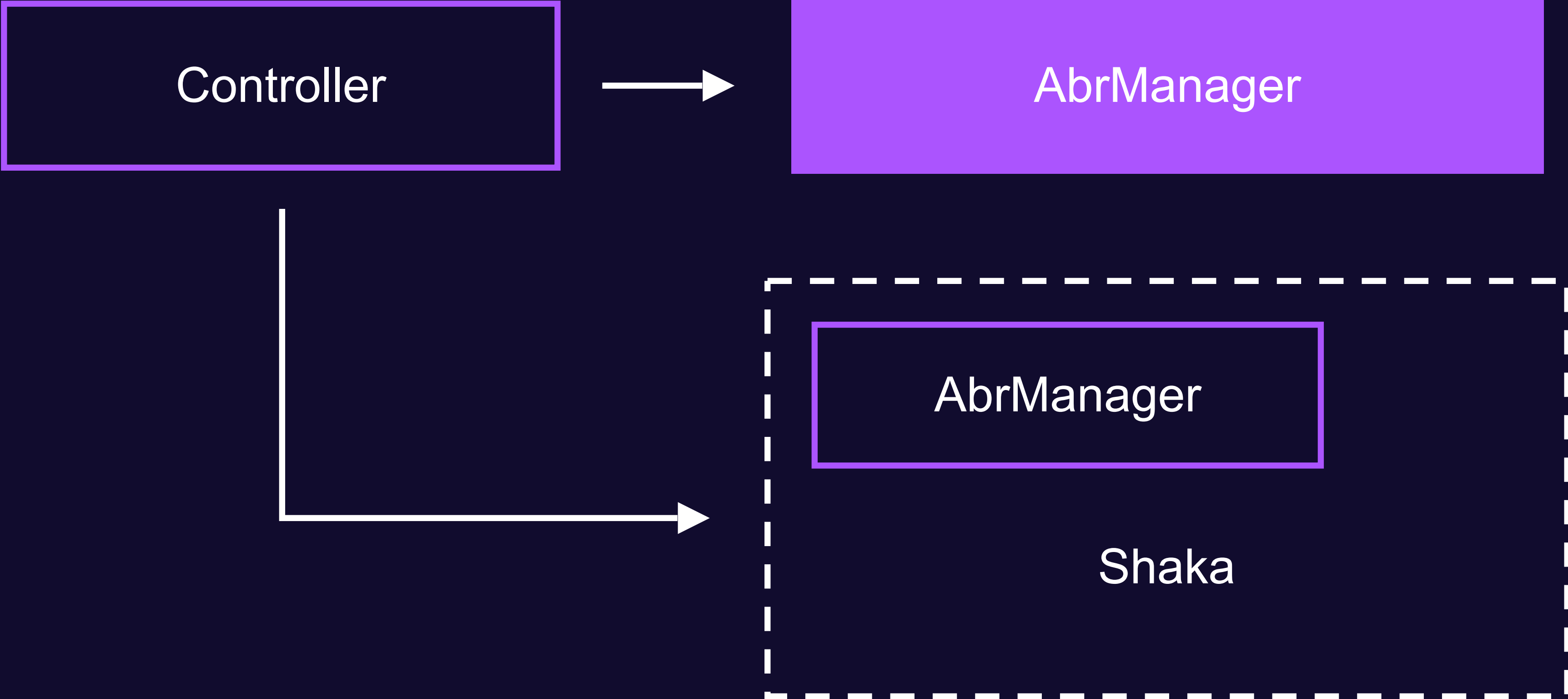
ABR (**A**daptive **B**it**R**ate)

Технология для динамического выбора качества на протяжении воспроизведения контента

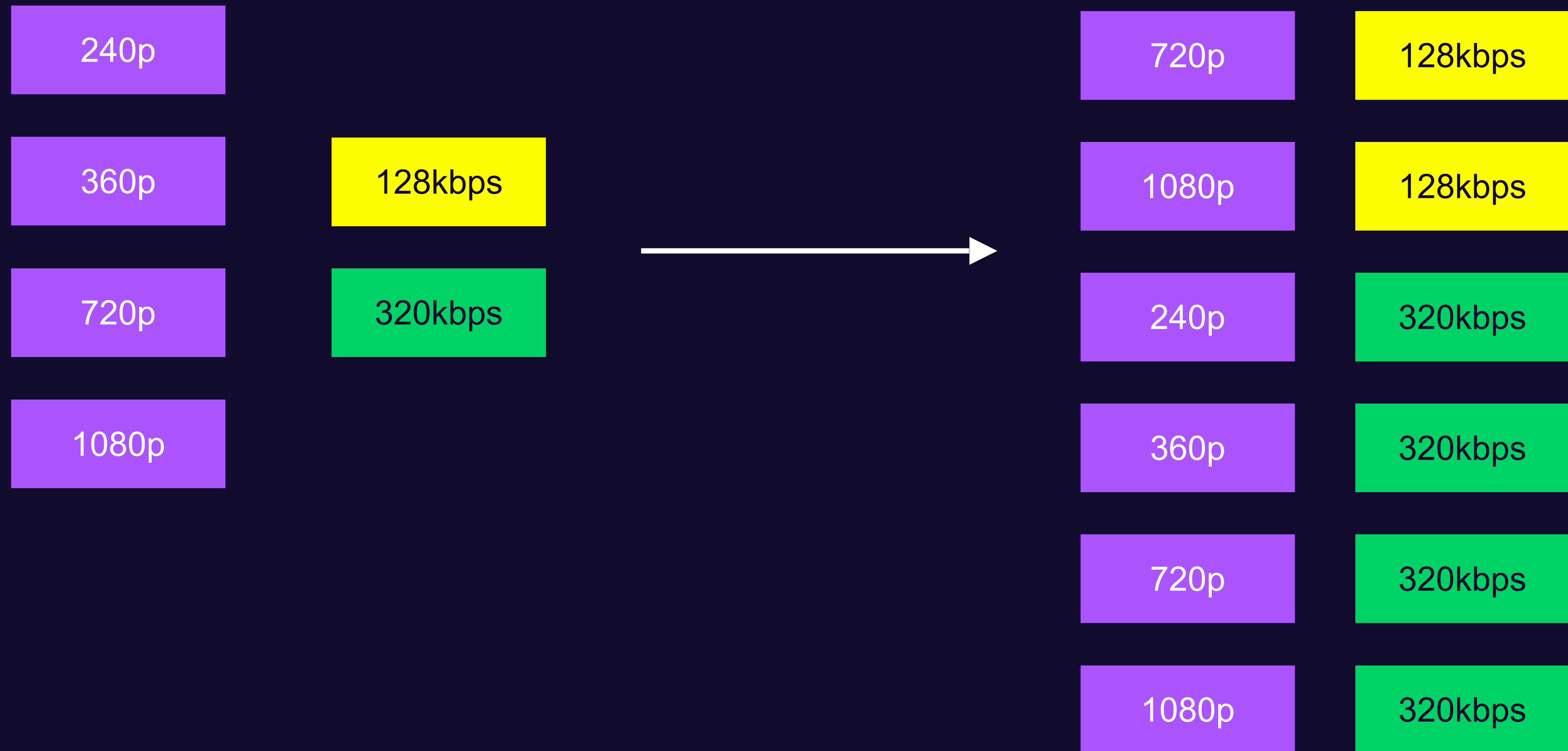


<https://youtu.be/ojlvKjSMq64>

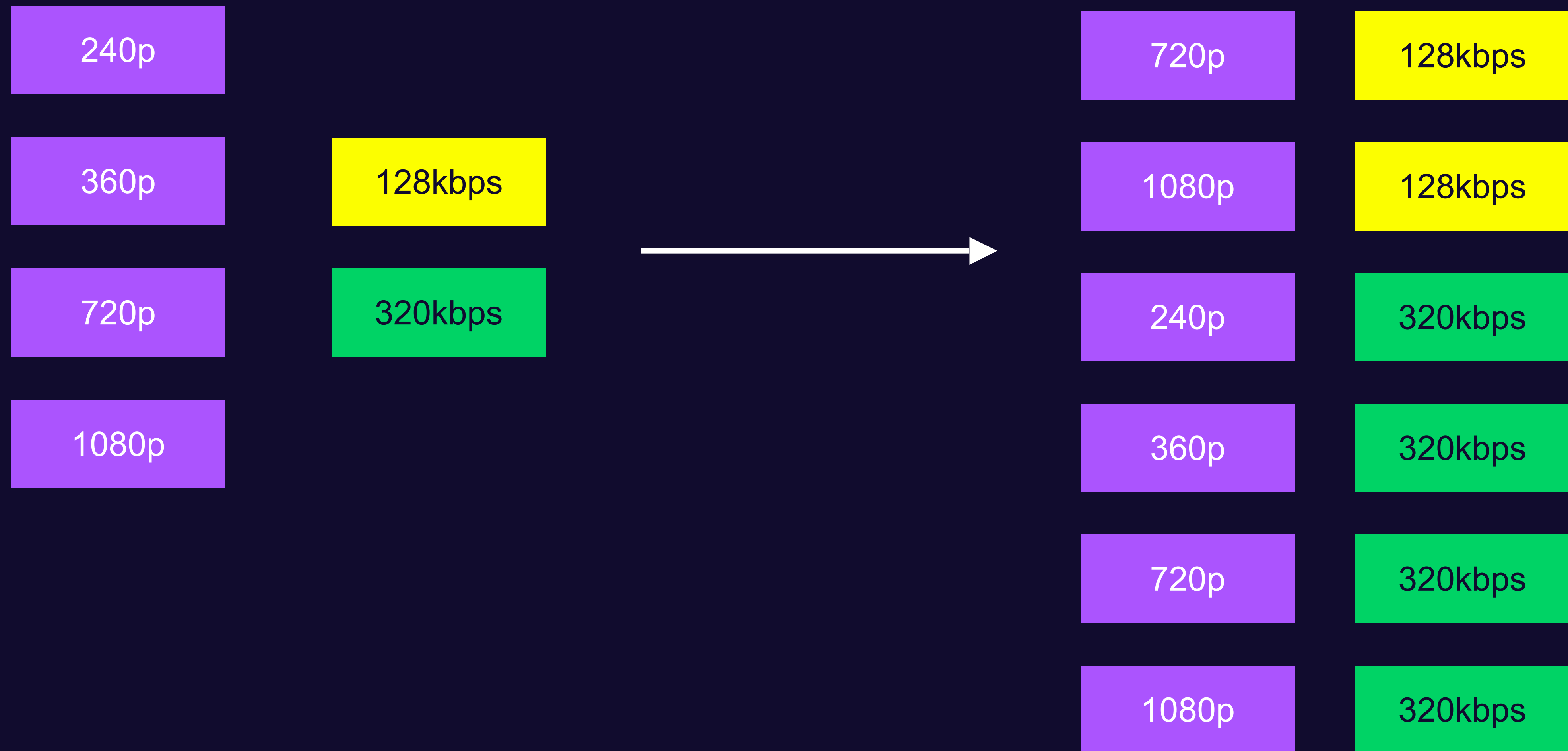
Свой ABR



Варианты



В нашем AbrManager



ABR

Варианты



Реконфигурация shaka-player сбрасывает ABR полностью



BandwidthEstimator



Вызывается в произвольный для нас момент времени



Вызов ABR

	shaka-player	Что нам надо было
Инициализация	✓	✓
Загрузка сегмента	✓	✓
Изменение конфига	✓	✗
Изменение аудиодорожки	✓	✓
Выбор видеодорожки	✓	✓
DRM	✓	✓
Таймаут загрузки	✓	?
В процессе загрузки	✗	✓

Low-latency

Low-latency (II)

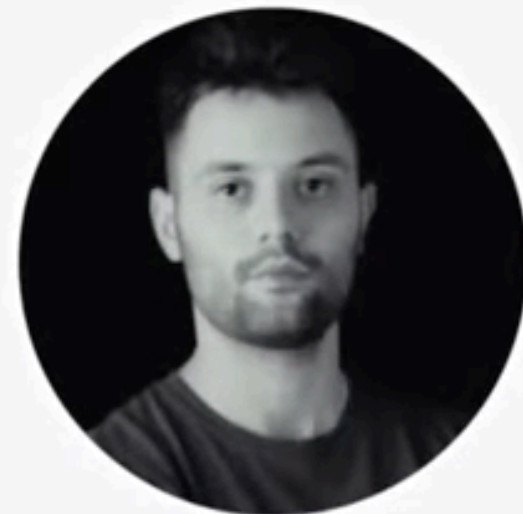
Это трансляции с низкой задержкой



Low-latency

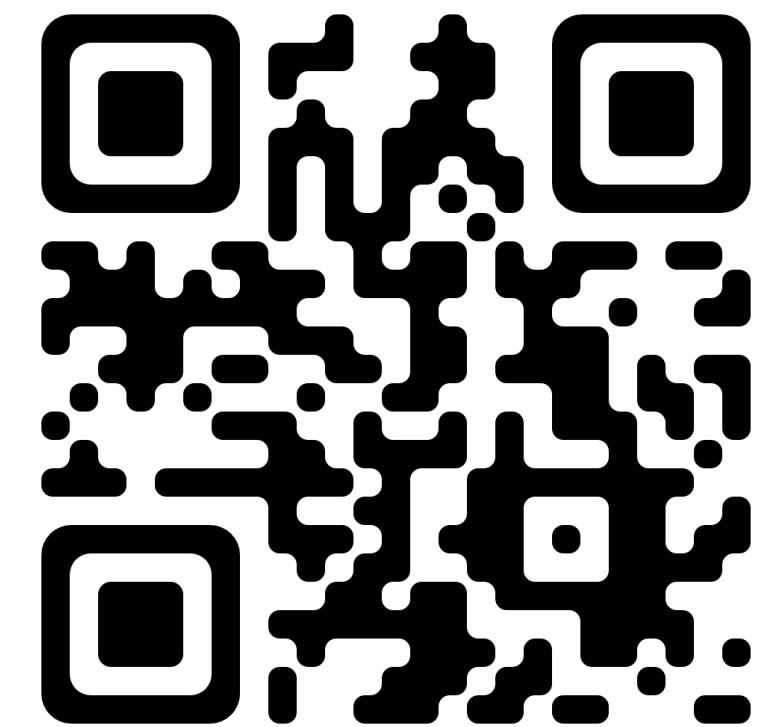


**Алексей
Гусев**
Яндекс



**Дмитрий
Кравцов**
Яндекс

Уменьшаем задержку в live-трансляциях



<https://youtu.be/GDt-xpmTeyU>

Low-latency



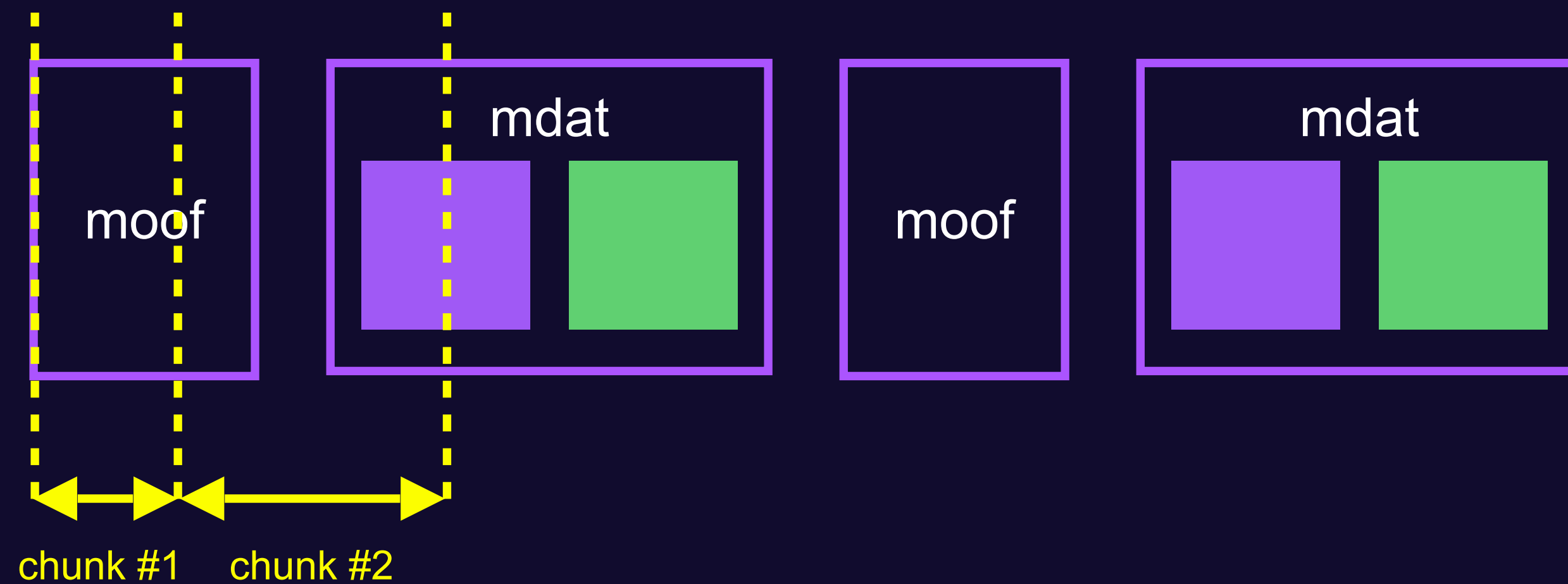
CMAF-сегмент



Наши доработки в Shaka

- Чтение данных из потока

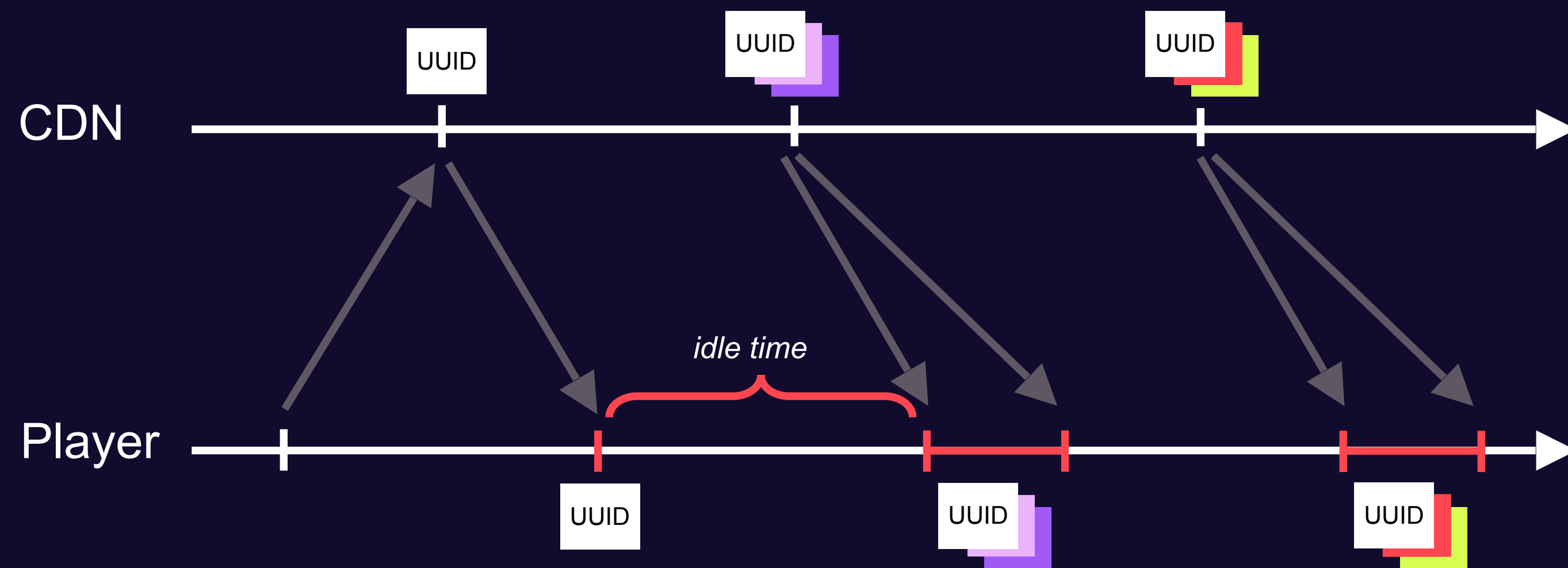
Low-latency



Наши доработки в Shaka

- Чтение данных из потока
- Парсинг mp4

Low-latency

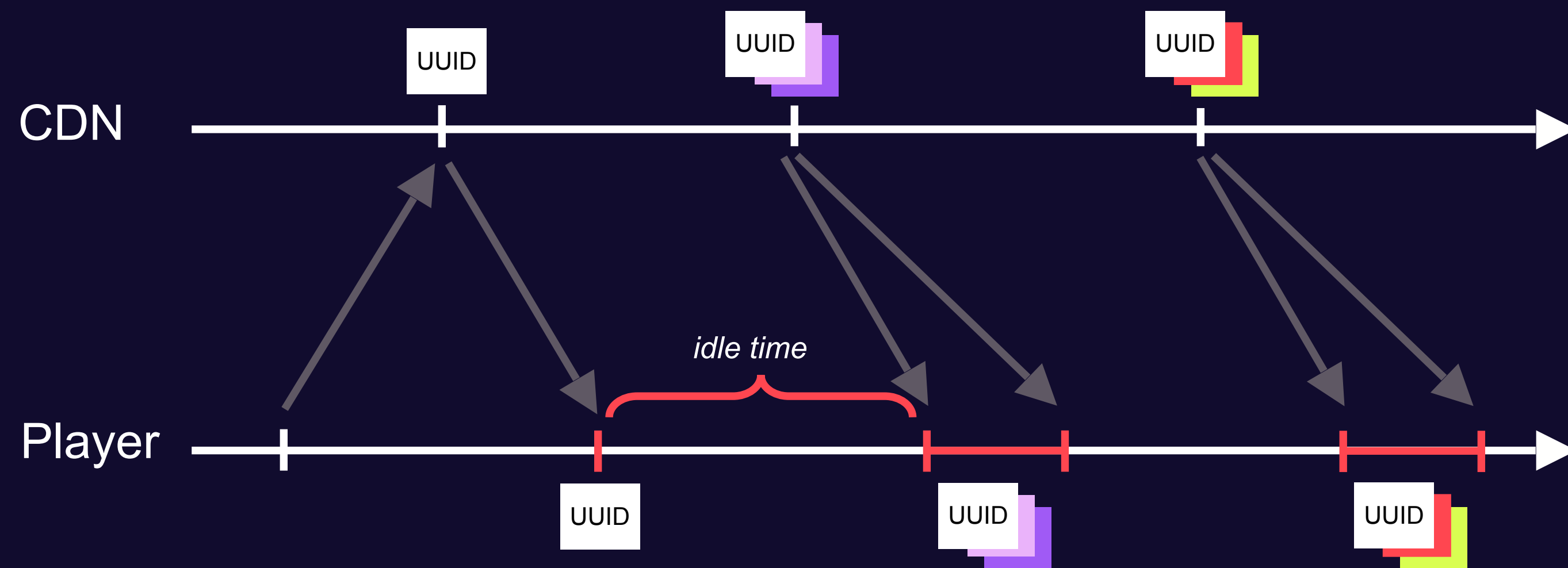


$$\text{idle time} = \text{UUID\#2} - \text{UUID\#1}$$

Наши доработки в Shaka

- Чтение данных из потока
- Парсинг mp4
- Парсинг серверного времени для правильного BandwidthEstimate

Low-latency



$$\text{idle time} = \text{UUID\#2} - \text{UUID\#1}$$

Наши доработки в Shaka

- Чтение данных из потока
- Парсинг mp4
- Парсинг серверного времени для правильного BandwidthEstimate
- Правки в ABR — дополнительные значения α для EWMA

Баги, которые мы породили

Пустые mdat-ы

01

Safari ломался

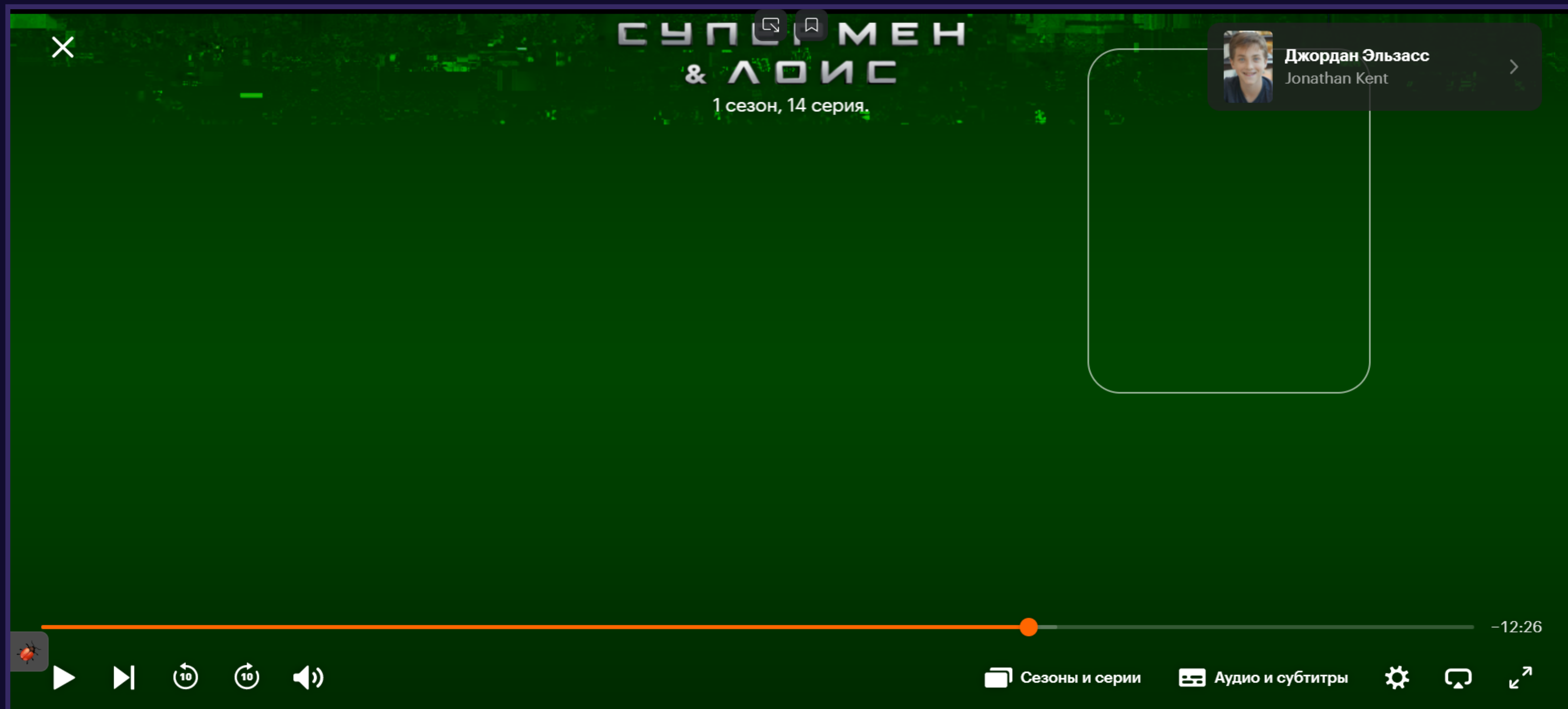
02

Снова попатчили
парсинг mp4

03



Баги, которые мы породили



Серверный capping

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:cenc="urn:mpeg:cenc:2013"
xmlns:mspr="urn:microsoft:playready" mediaPresentationDuration="PT1H54M30.000S"
minBufferTime="PT4.00S" profiles="urn:mpeg:dash:profile:isoff-live:2011"
type="static">
  <SupplementalProperty
    schemeIdUri="<id нашего серверного каппинга>"
    value="720"
  />
  <BaseURL>https://base-url-1/</BaseURL>
```

Серверный сарринг

Парсинг плейлиста:

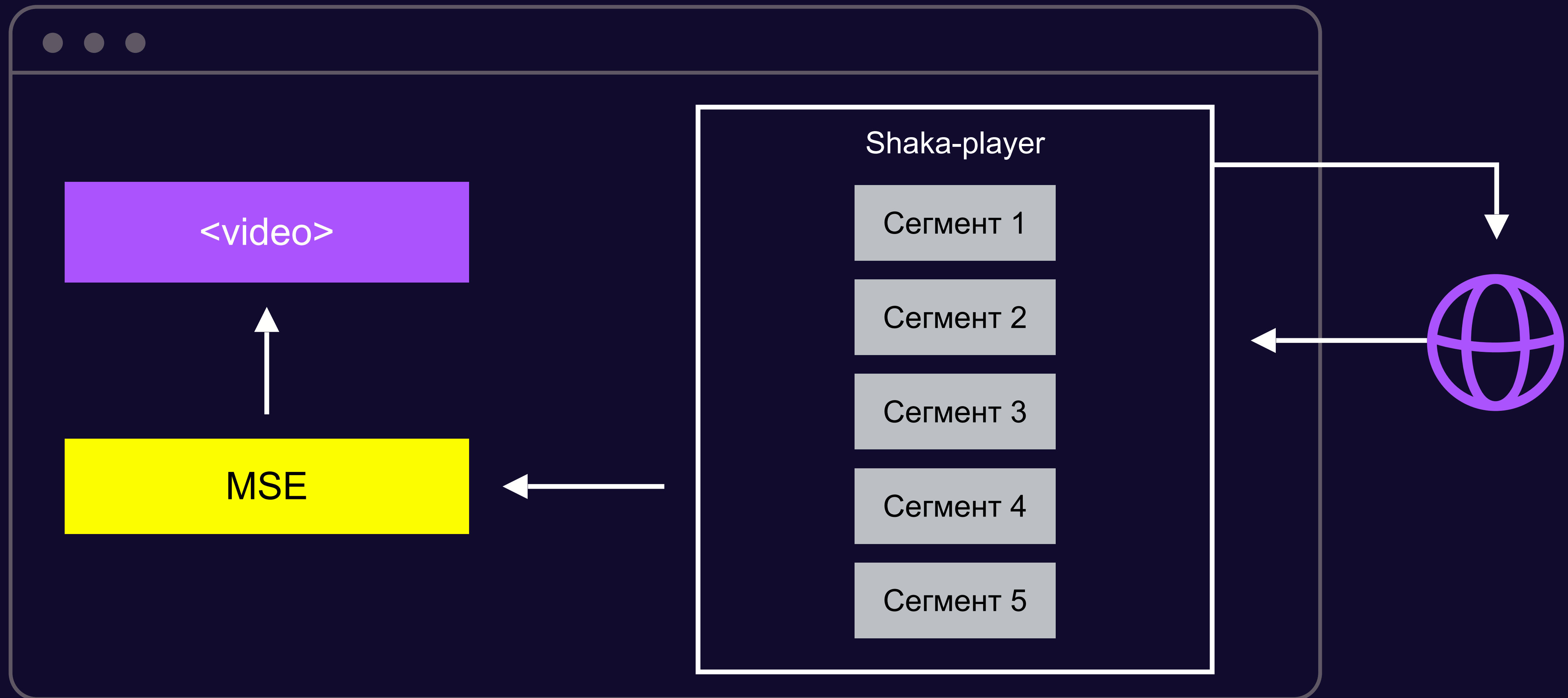
Shaka

AbrManager

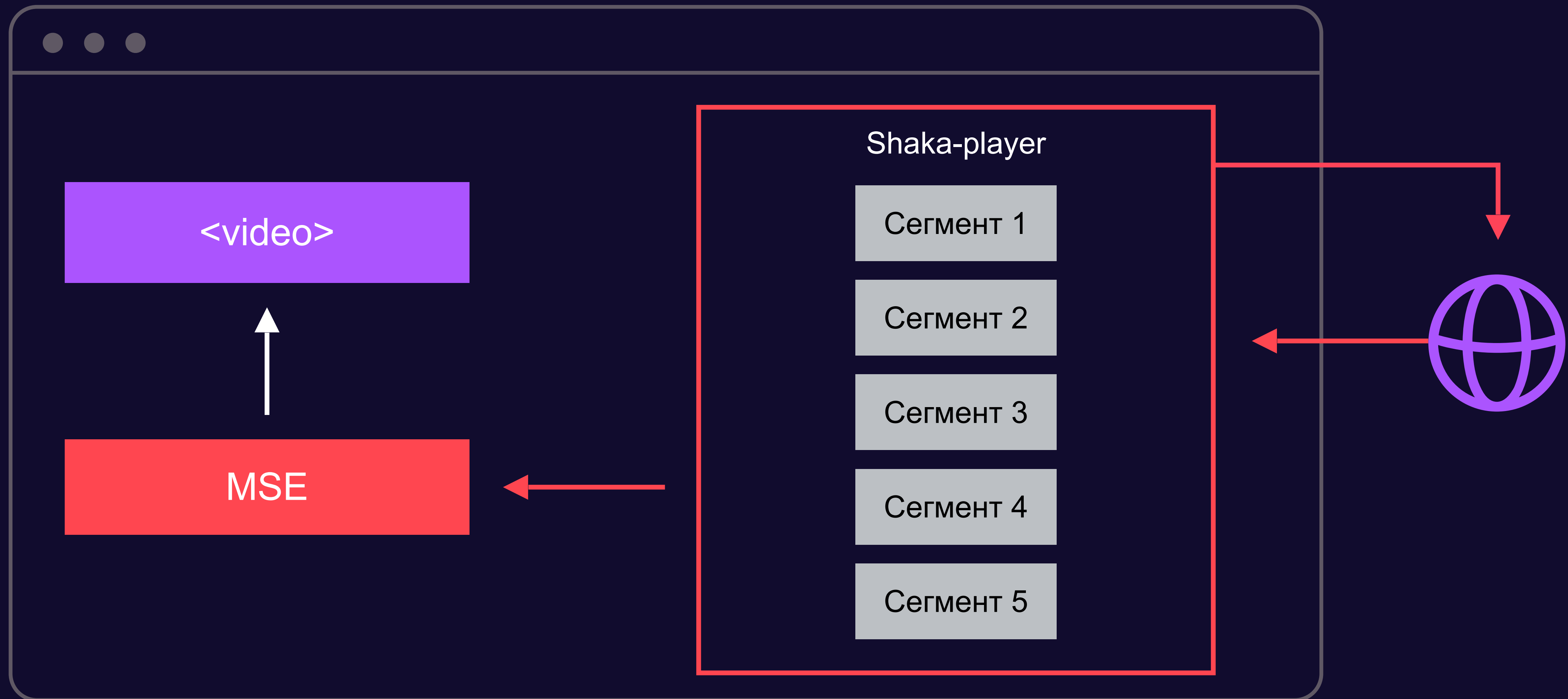


**Что мы не смогли
реализовать с shaka-player**

MSE в Web Worker



MSE в Web Worker



MSE в Web Worker

2020-2021

Разработка
концепта и demo

Август 2022

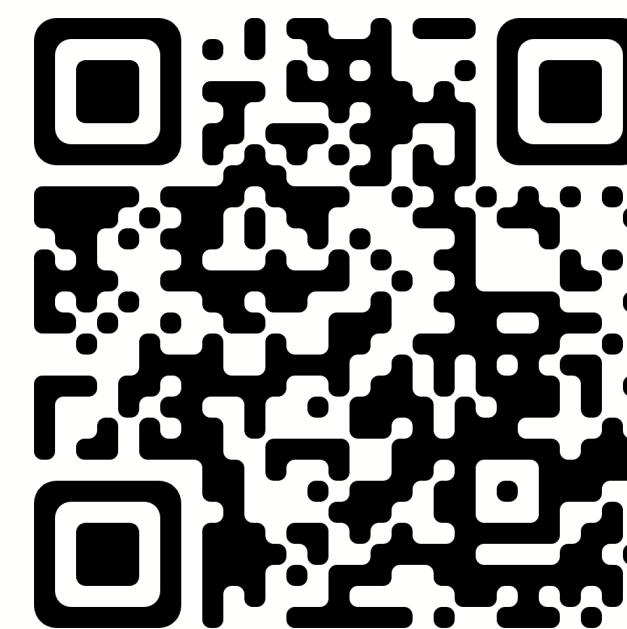
Chrome 105

Экспериментальный
флаг

Ноябрь 2022

С Chrome 108

Включён
по умолчанию



<https://clck.ru/35zykn>

MSE & Web Worker

← → ↻ wolenetz.github.io/mse-in-workers-demo/mse-in-workers-demo.html

MSE-in-Workers Demo

Use the button, below, to get started.

Demo details are at the [bottom of this page](#).

[Start Demo](#)

Awaiting Start

Demo parameters (Applied when "Start Demo" button is clicked)

Media URL (to fetch completely, then append in chunks) : test-5seconds.webm

Media type : video/webm; codecs="vp9"

Chunk append size : 2048 bytes 1024 bytes 512 bytes 256 bytes 1 byte

Busy-wait duration (milliseconds) : 50ms 100ms 200ms 400ms 800ms 1600ms

MSE usage on main thread	MSE usage in worker thread (video element remains on main thread)
--------------------------	---

Demo details

This is a demo of how usage of Media Source Extensions API from a dedicated worker context can avoid "buffering jank" when the main window context is very busy, even though the media element playing the buffered media is still on that main thread.

This demo presents a side-by-side comparison of two players, one fetching and buffering to its media element solely on the main window context, and the other player relying on a dynamically created dedicated worker to fetch and buffer the same media into a MediaSource owned by that worker's context, attached to the main window context's media element via a transferred-to-main MediaSourceHandle object set on the media element's srcObject attribute.

A scenario where the main thread is under heavy contention is achieved by frequently busy-waiting on it while both players are fetching, buffering and playing. Furthermore, asynchronous appendBuffer operations are performed (on each of the main and worker MSE demos) in small chunks to enable rapid appearance of "buffering jank" on the player being buffered via MSE on the main thread versus a much better experience on the player being buffered via MSE on the worker thread, even if the media stream is short.

In practice, main thread (aka Window context) contention can result from many sources, though commonly it is from complex and frequent task execution demands made by the application on the Window execution context, and is made worse when the platform has less execution capacity available. The DOM and associated application javascript operate in the Window execution context. DedicatedWorkers run in a concurrent execution context with respect to the Window execution context. Even with concurrent buffering and smoother playback by using MSE on a dedicated worker context, observe that the video element for the MSE-in-Worker player can still have poor controls response time when the Window context is under high contention. This is because that element (and its controls) can only execute on the Window context and their event handlers can have high scheduling latency.

To try in Chrome, use 105.0.5180.0 or greater (earlier versions since 88.0.4300.0 experimentally supported worker MediaSource attachment using legacy object URLs, but that functionality has been removed from both the implementation and specification).

- Chrome has MSE-in-Workers enabled by default beginning with version 108.0.5334.0.
- Chrome 105.0.5180.0 through 108.0.5333.0: enable this feature by starting with cmdline option `--enable-blink-features=MediaSourceInWorkers,MediaSourceInWorkersUsingHandle` (preferred way) or with `chrome://flags/#enable-experimental-web-platform-features` enabled.

If the main thread MSE buffering playback is too slow (it might take a long while to render even the first frame on some machines), reduce the amount of contention on the main thread by stopping the demo, selecting a smaller busy-wait duration (to handle other main thread tasks more frequently) or a larger append-size (to do more real buffering work when scheduled), and restarting the demo.

See [the demo's github page for more information](#).

MSE в Web Worker в shaka-player не ожидается

The screenshot shows a GitHub issue page for the repository 'shaka-project / shaka-player'. The issue title is 'Running Shaka in a Web Worker #5008', which is marked as 'Closed'. It was opened by IsaacSNK on February 16 and has 5 comments. The issue is categorized as a 'type: question'.

Issue Details:

- Repository:** shaka-project / shaka-player
- Issue Title:** Running Shaka in a Web Worker #5008
- Status:** Closed
- Opened by:** IsaacSNK
- Opened on:** Feb 16
- Comments:** 5
- Label:** type: question

Comments:

- IsaacSNK (Contributor) commented on Feb 16:** Hi! Is there any plan on taking advantage of the MSE in Web Workers for Shaka? This could have great performance benefits when the main thread is busy. I see MSE in Web Workers is available for Chrome 108...Any concern, ideas you may have?
- IsaacSNK added the 'type: question' label on Feb 16.**
- avelad (Collaborator) commented on Feb 28:** @joeyparrish can you review it? Thanks!
- github-actions (bot) commented on Mar 4:** @IsaacSNK Does this answer all your questions? If so, would you please close the issue?

Issue Metadata (Right Side):

- Assignees:** No one assigned
- Labels:** type: question
- Projects:** None yet
- Milestone:** No milestone
- Development:** No branches or pull requests
- Notifications:** You're not receiving notifications from this thread. [Subscribe](#)

Виртуальный буфер и кеширование

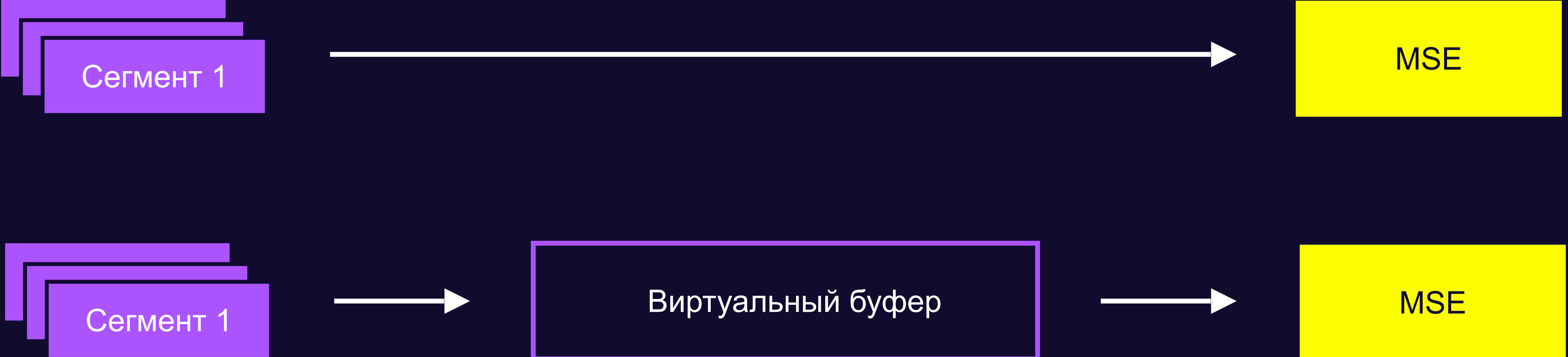
Shaka-player скачивает сегменты сразу в MSE



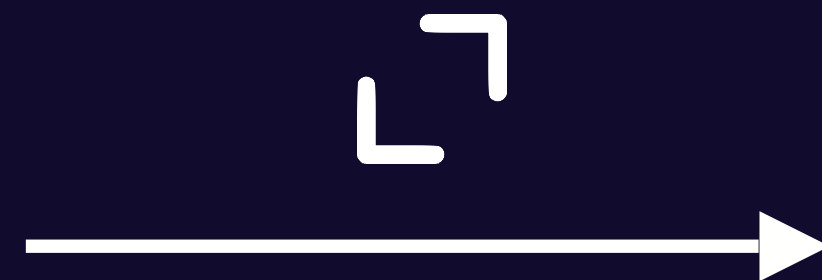
Все ограничения MSE — наши ограничения

	Chrome	Chromecast*	Firefox	Safari
Video	150MB	30MB	100MB	290MB
Audio	12MB	2MB	15MB	14MB

Виртуальный буфер и кеширование



Перекачка буфера



Перекачка буфера

Shaka-player скачивает
сегменты сразу в MSE



Требуется патчинг сетевого
слоя с неизвестным
результатом

Некоторые правила в ABR

Связанные с сетью

Например, учёт TTFB
(Time To First Byte)



Связанные с наполненностью буфера

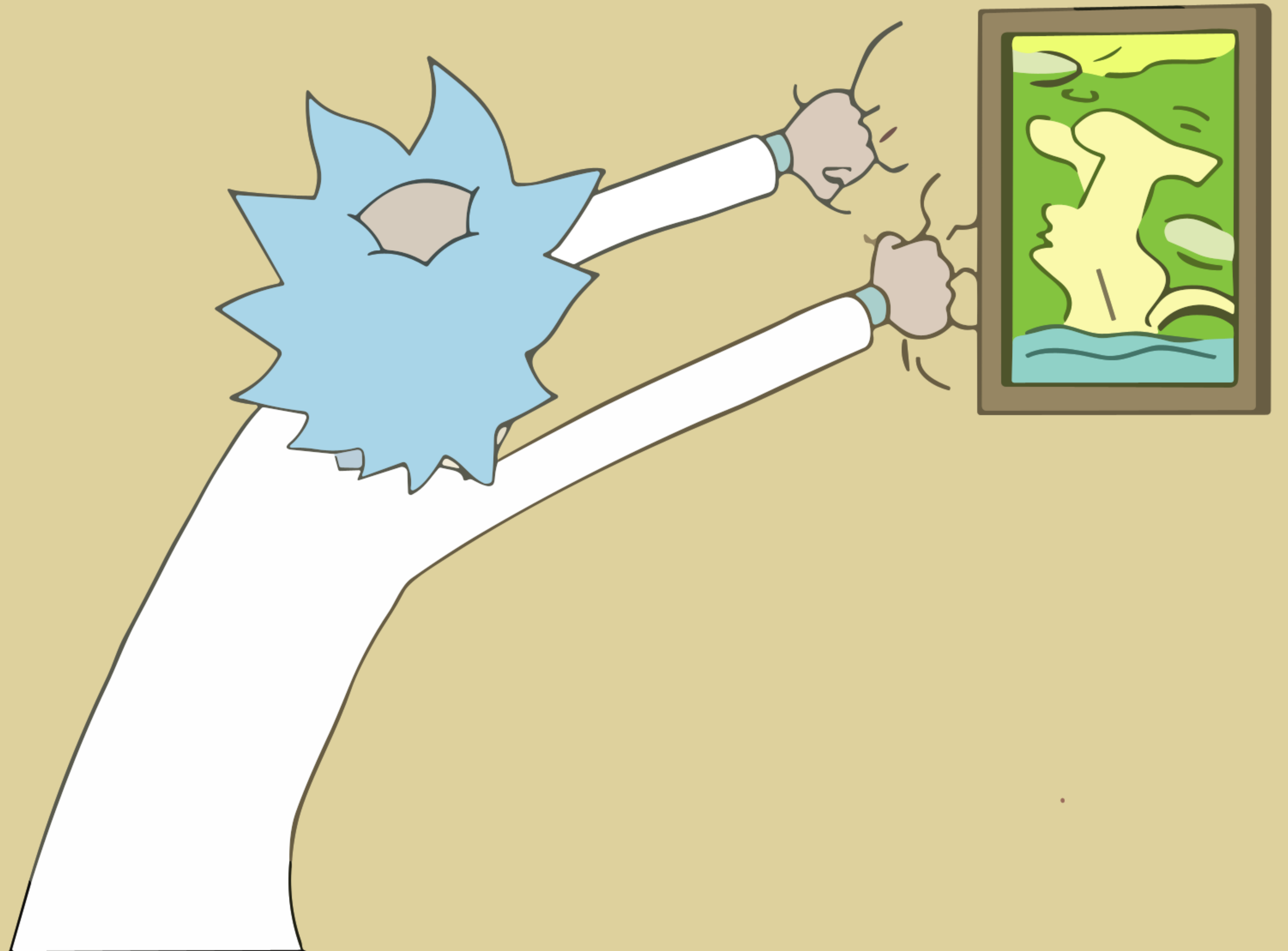


shaka-player стала нам «мала»

shaka-player

- Оперирует вариантами
- Качает сразу в MSE
 - Ограничения MSE
 - Невозможна перекачка буфера
 - Невозможны виртуальный буфер и кеширование
- Не поддерживает MSE-in-WebWorker
- Сетевой слой
- Обновления — боль







YaSP —
Yet another StreamPlayer

Проектирование

Что мы хотели заложить в YaSP

Техническое решение

Стриминг и abr в instream-рекламе

Расширение <video>

Предзагрузка

Перенос большей части кода в отдельный тред

WebWorker (MSE-in-WebWorker)

Виртуальный буфер

Перекачка буфера

Override <video>

Потоковое видео в рекламе



Унификация кода работы с
native HLS и MSE

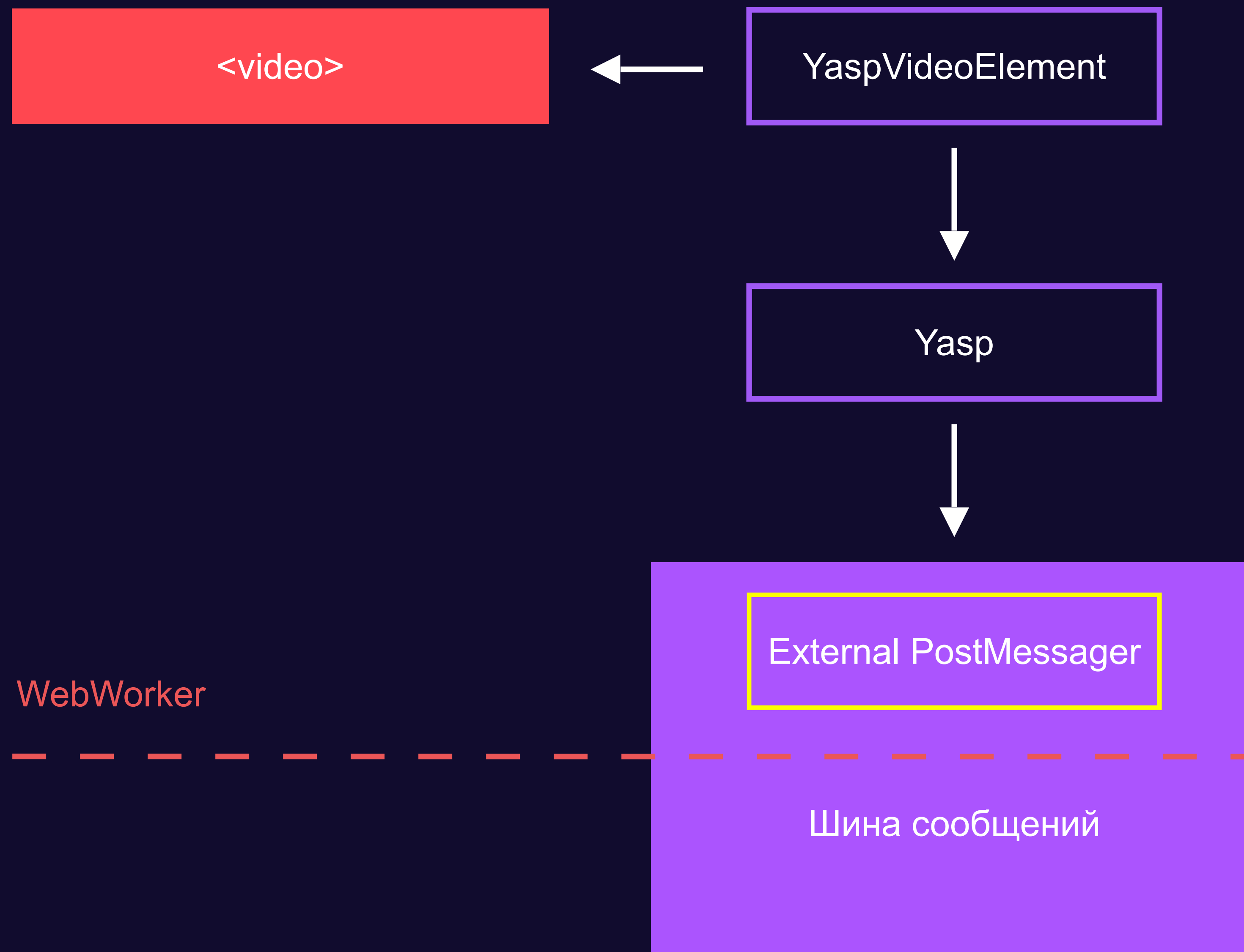




Архитектура YaSP

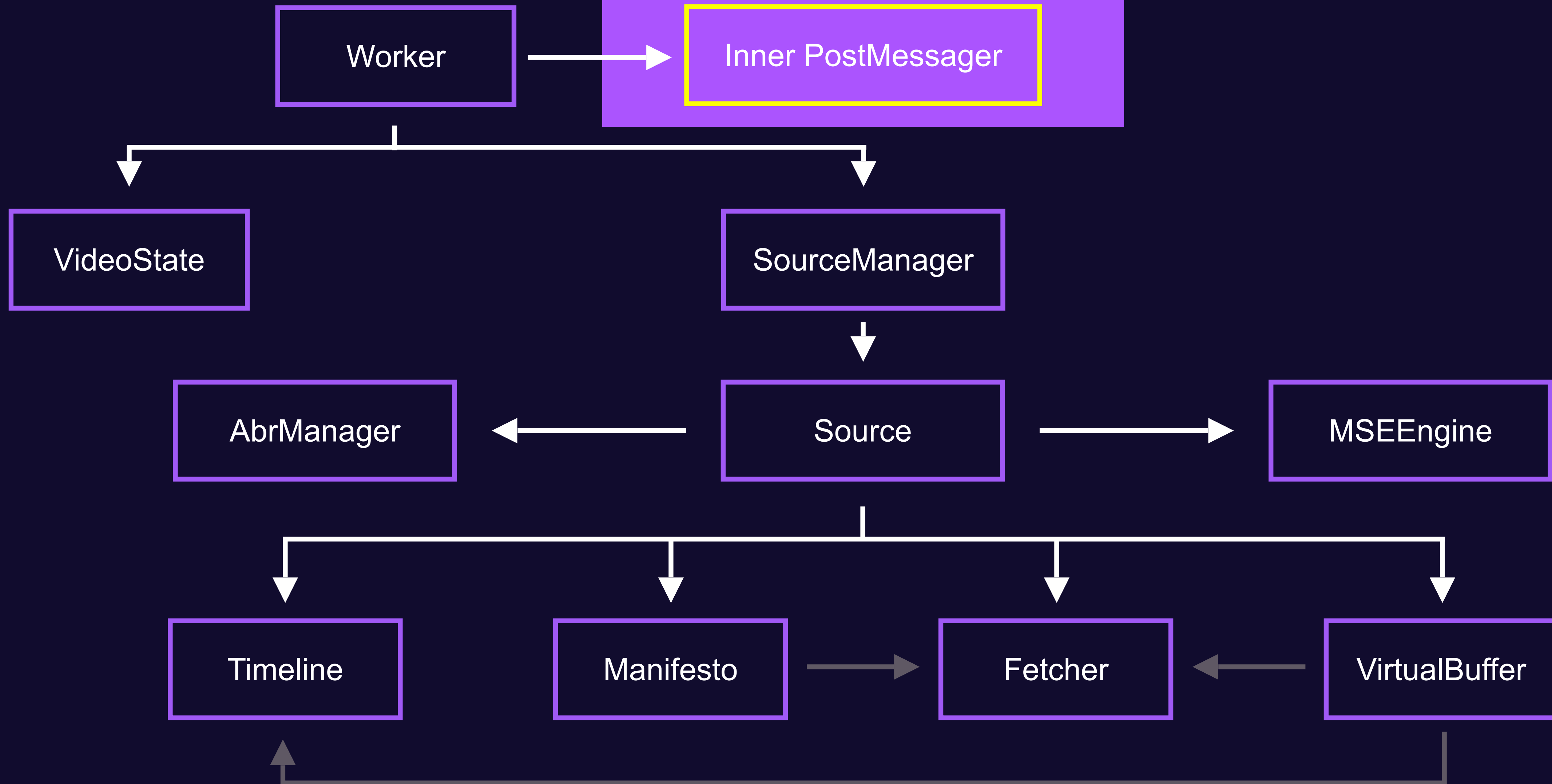


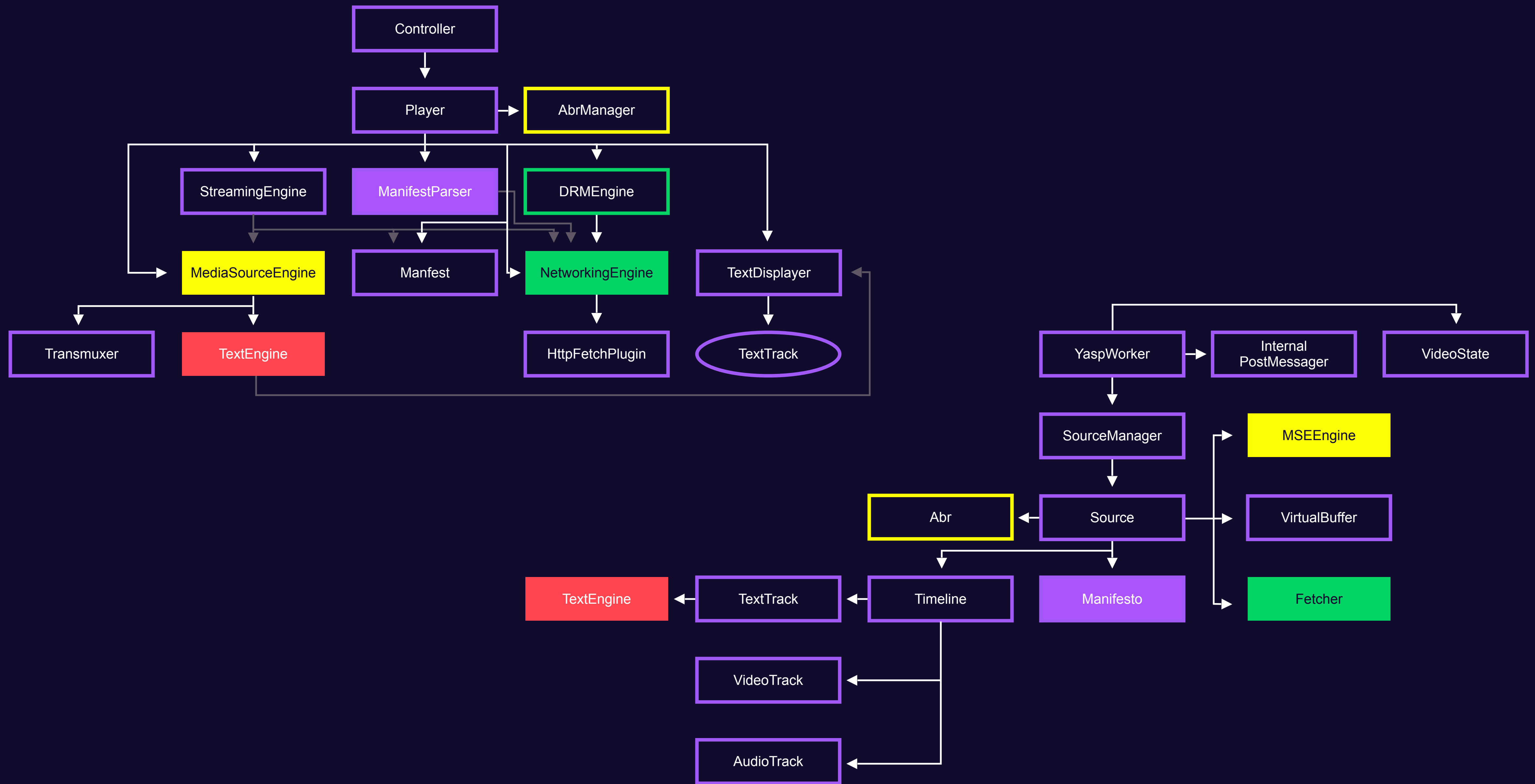
Архитектура YaSP



WebWorker

Шина сообщений

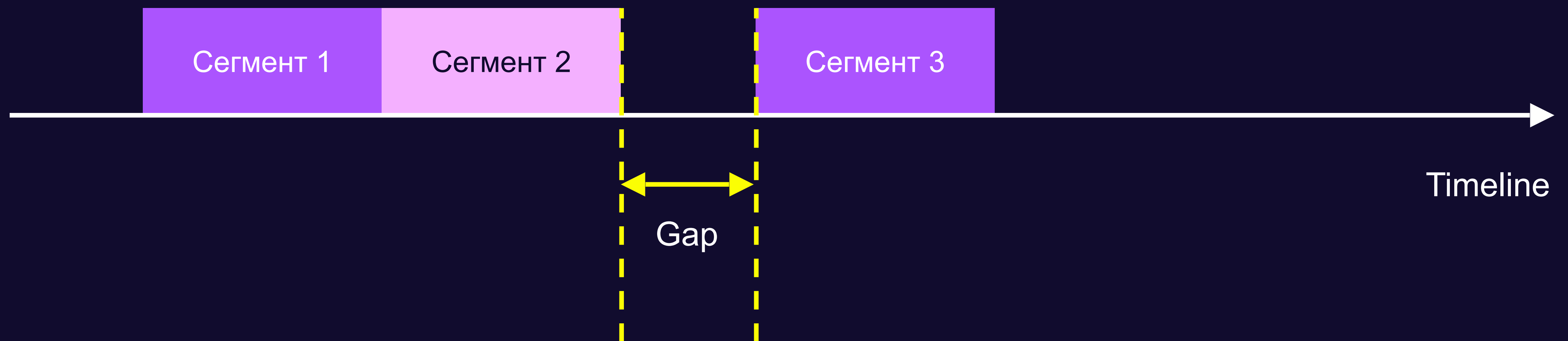




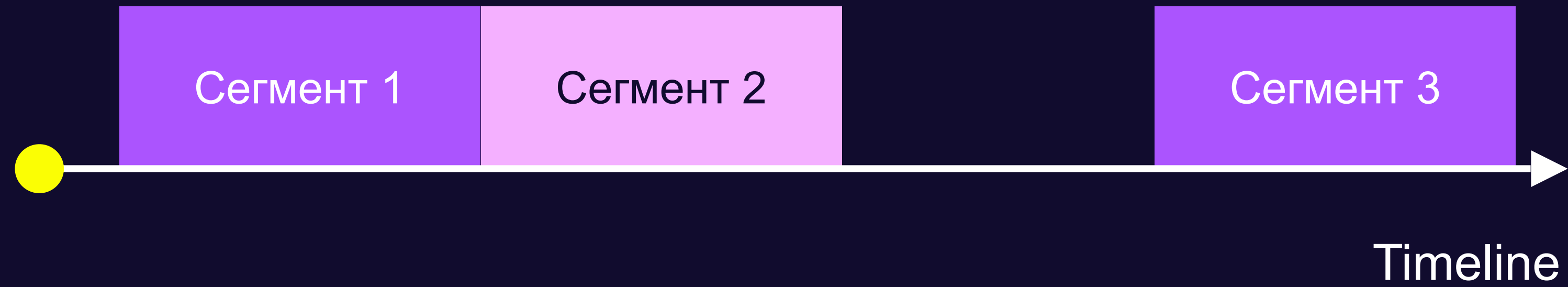
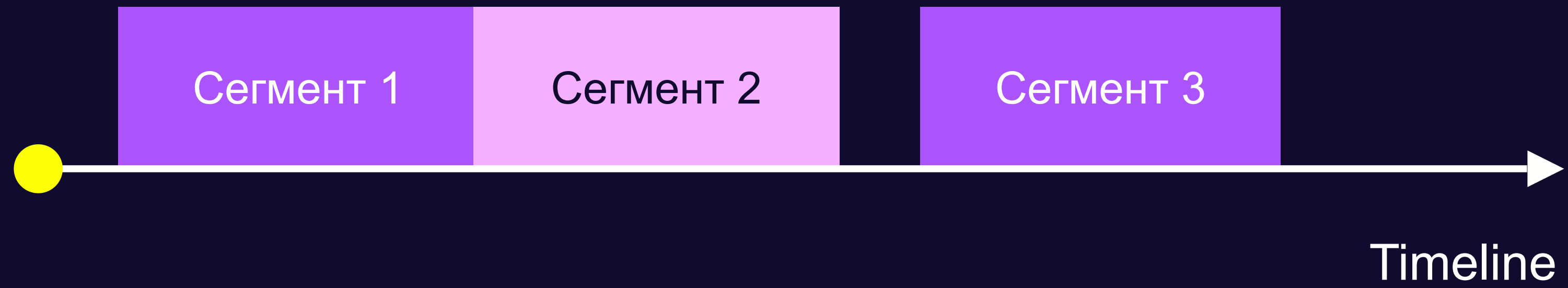


Дырки в потоке

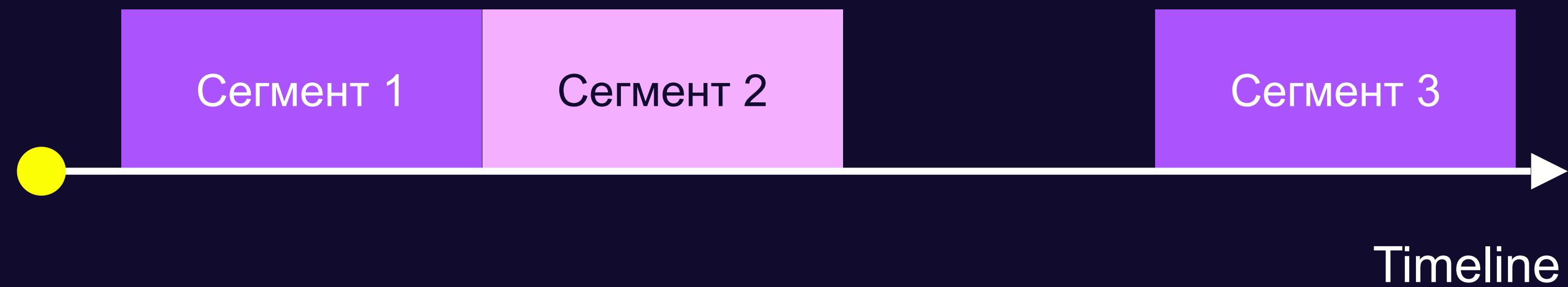
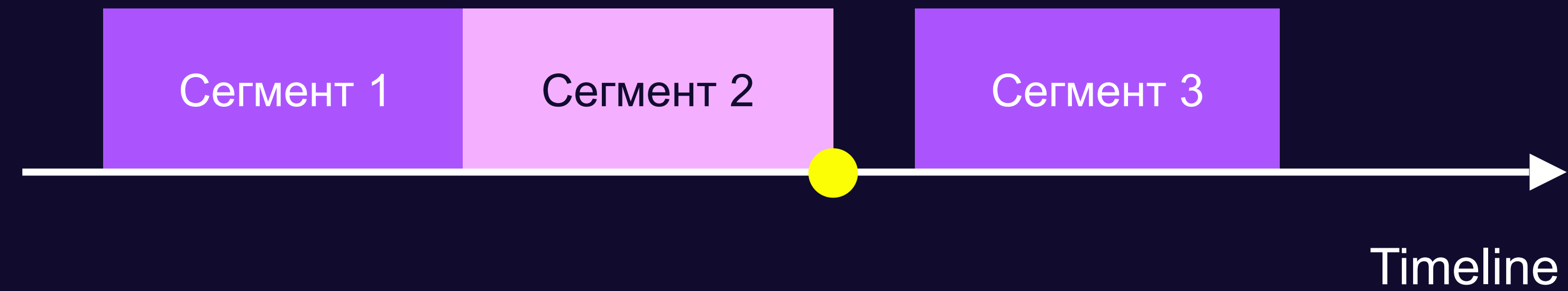
Если конец текущего сегмента не совпадает с началом следующего, то мы говорим, в потоке есть **дырка (Gap)**



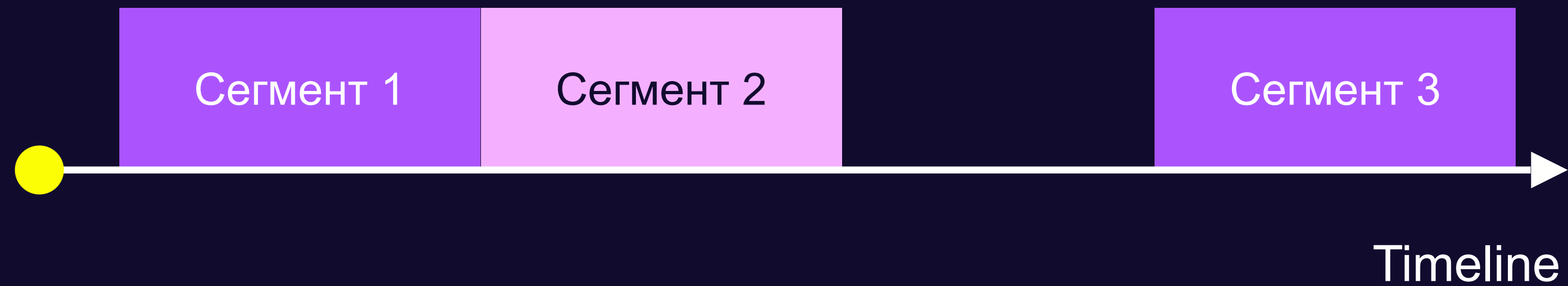
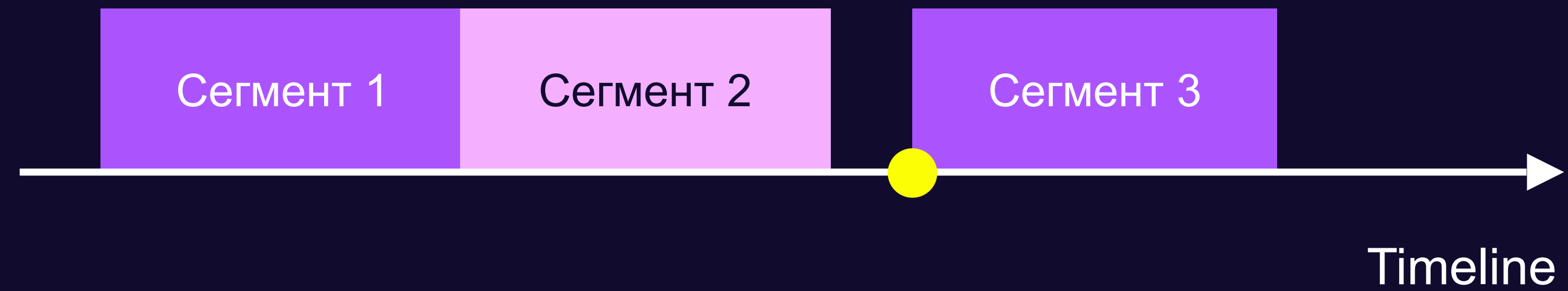
Дырки в потоке



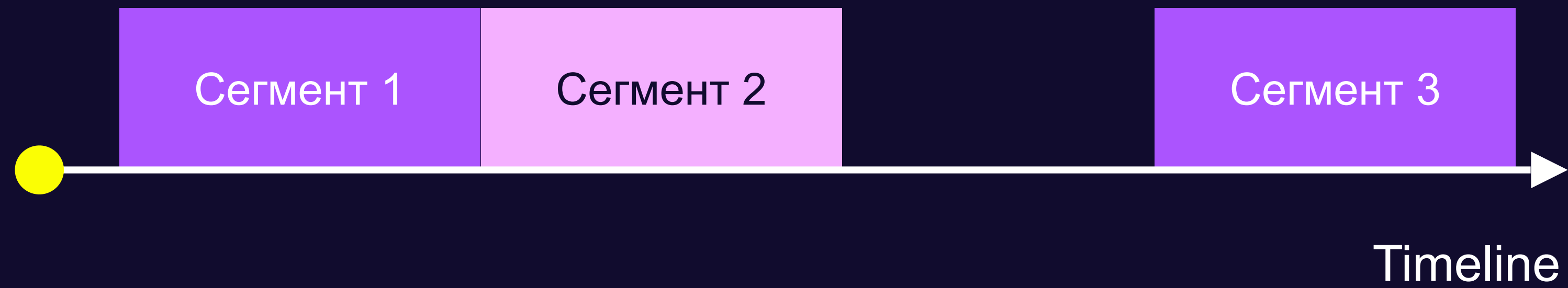
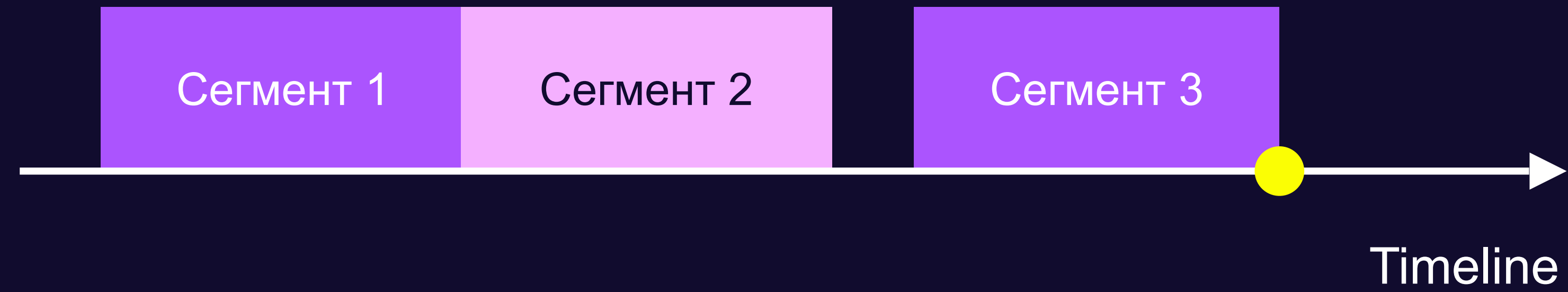
Дырки в потоке



Дырки в потоке



Дырки в потоке



Причины дырок

В потоках с несколькими периодами не выровнены границы между периодами

01

Суммарная длина контента в сегменте меньше заявленной в манифесте

02

GapJumper

Условия для прыжка:

<video>
в буферизации

01

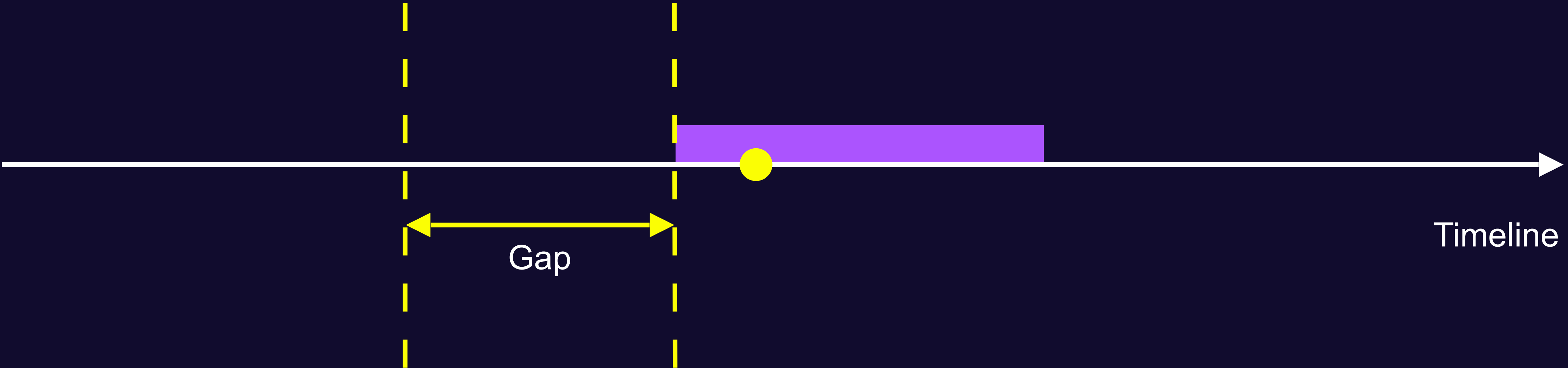
В BufferedRanges
есть данные впереди

02

Нет перемотки
в процессе

03

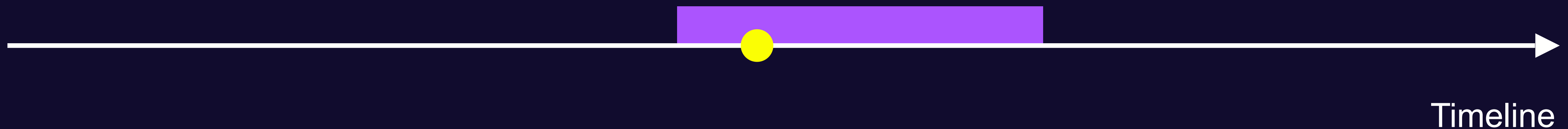
GapJumper и перемотка назад



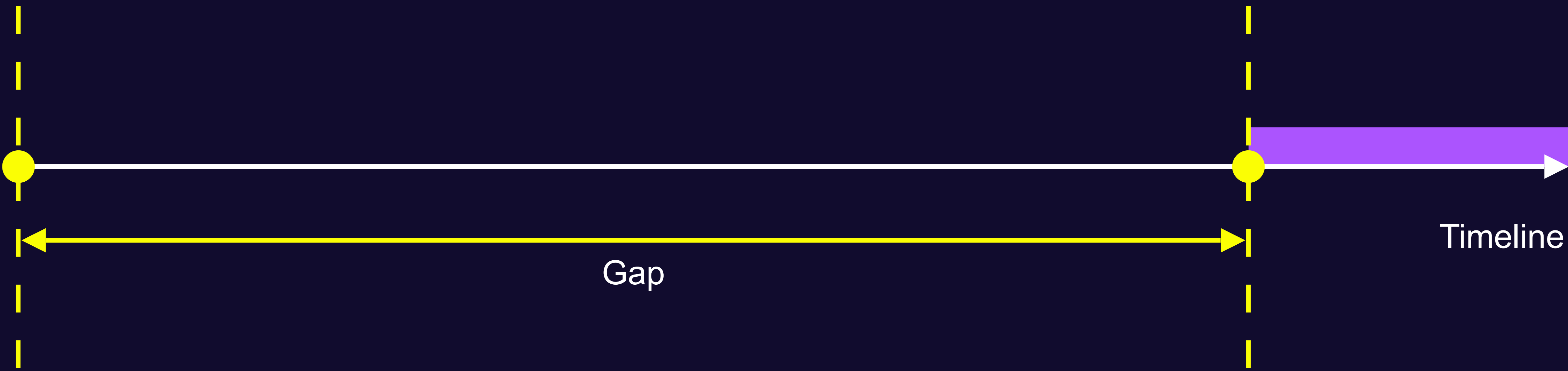
GapJumper и перемотка назад

Решение

Очищать накаченный буфер при перемотке



GapJumper и зацикленные видео

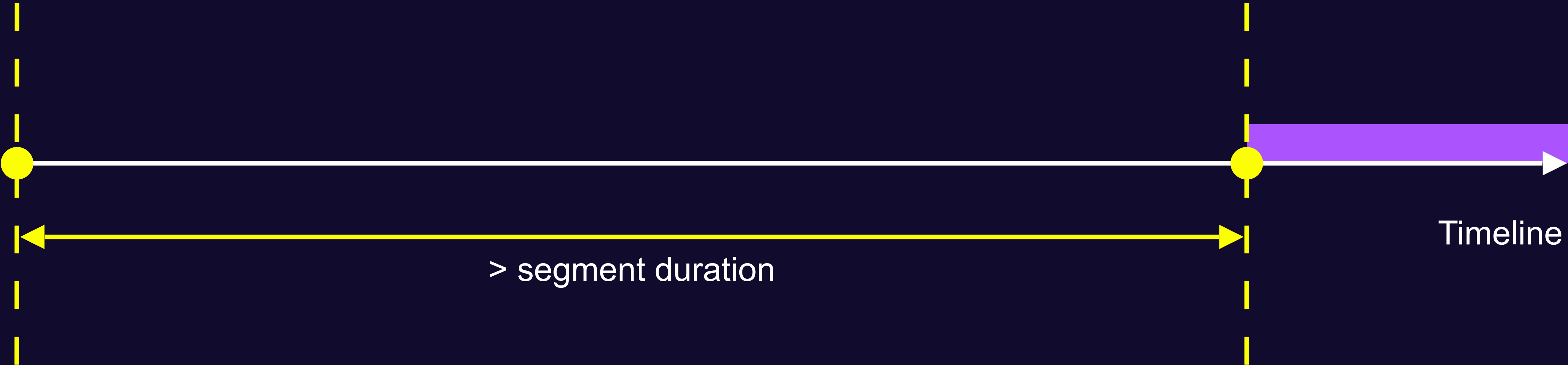


GapJumper и зацикленные видео



Решение

Ограничить размер прыжка длиной сегмента



Рассинхрон скачивания аудио и видео

Audio



Video



DRM

Неполная спецификация EME
(Encrypted Media Extensions)

01

Непонятно, как правильно
дестроить CDM

02

Непонятно, как правильно
переключаться между разными
DRM-системами

03

Пытались сделать синхронным

04

Синхронное DRM API



Синхронное DRM API

```
if (buffered.length > 0) {  
    buffered.end(0) // exception: buffered length === 0  
}
```

Разработка
своего движка



ОЖИДАНИЕ



РЕАЛЬНОСТЬ



Что с метриками?

Ошибки: фатальные
и нефатальные

01

Количество и длина
буферизаций — помогло
понять, что у нас
проблемы с дырками

02

Скорость запуска

03

TVT (Total View Time)

04

QoE

05

Трафик (traffic / TVT)

06



Немного яндексовой терминологии

Эксперимент

Красный

Серый

Зеленый

↑ Фаталов
↓ TVT

Без изменений метрик

↓ Фаталов
↑ TVT

Как принимались?

По серому относительно
пропатченной shaka-player

01

В несколько этапов

02

Таймлайн экспериментов

I этап

VOD

II этап

VOD + DRM

III этап

Live

IV этап

LL Live

Тесты

Диктатура покрытия
unit-тестами



Интеграционные тесты





А стоило ли оно того?



Улучшение QoE
за счёт MSE-in-
Webworker



Заложено
много мест для
оптимизаций:

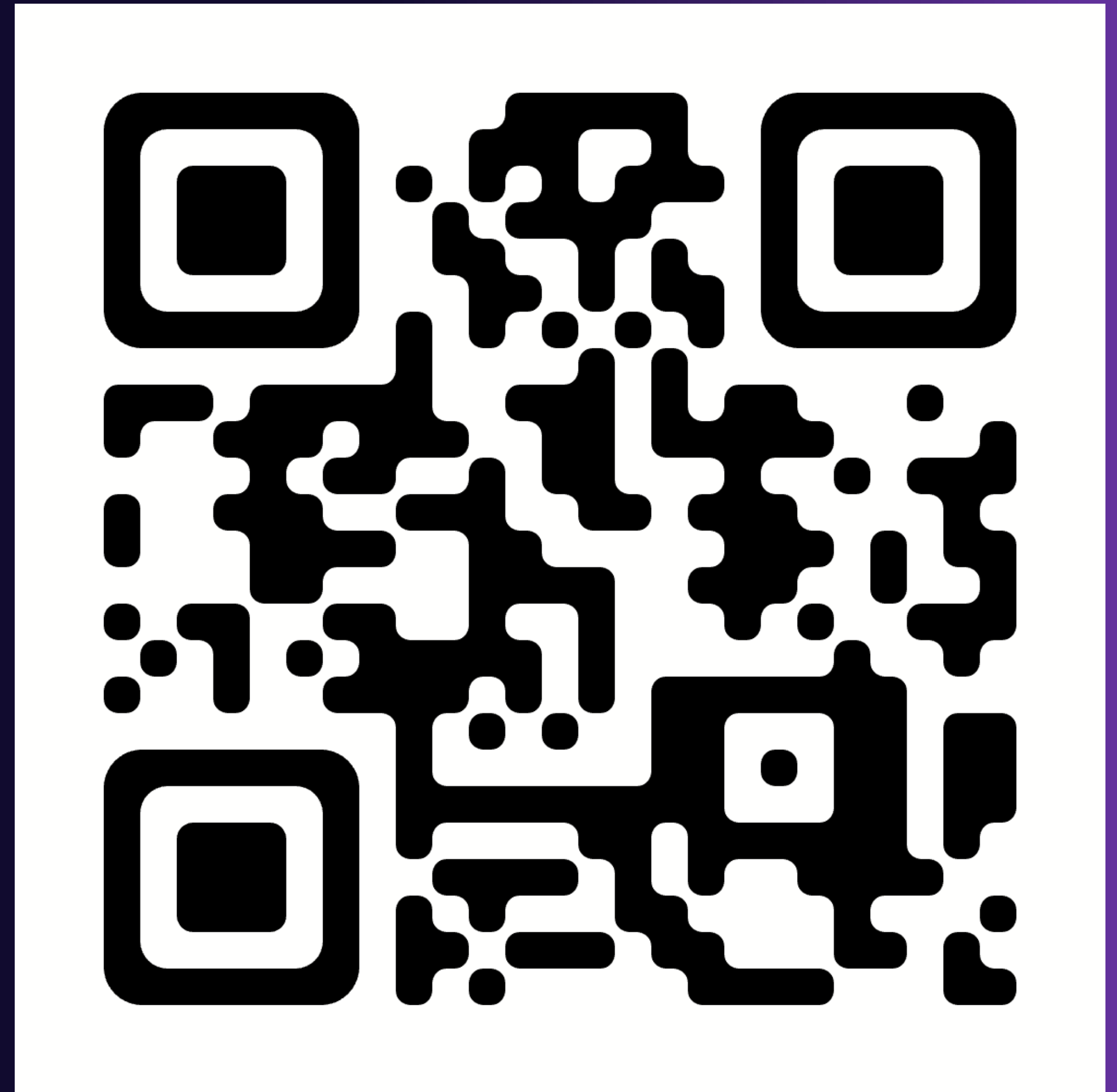
- Новые правила ABR
- Виртуальный буфер



2 года



Наш канал



<https://t.me/+wYjjXgFm8UUyZjQ6>

Вопросы?

Ольга Попова,
Yandex Infrastructure