videotech

Уменьшаем задержку в live-трансляциях

Дмитрий Кравцов,
Яндекс.Видеостриминг

Алексей Гусев,
Яндекс.Дзен

Где можно посмотреть видео через нас?

КиноПоиск

Яндекс ТВ

Яндекс Дзен

Яндекс Алиса

Яндекс Спорт

Яндекс Музыка

Яндекс Погода

Яндекс Маркет

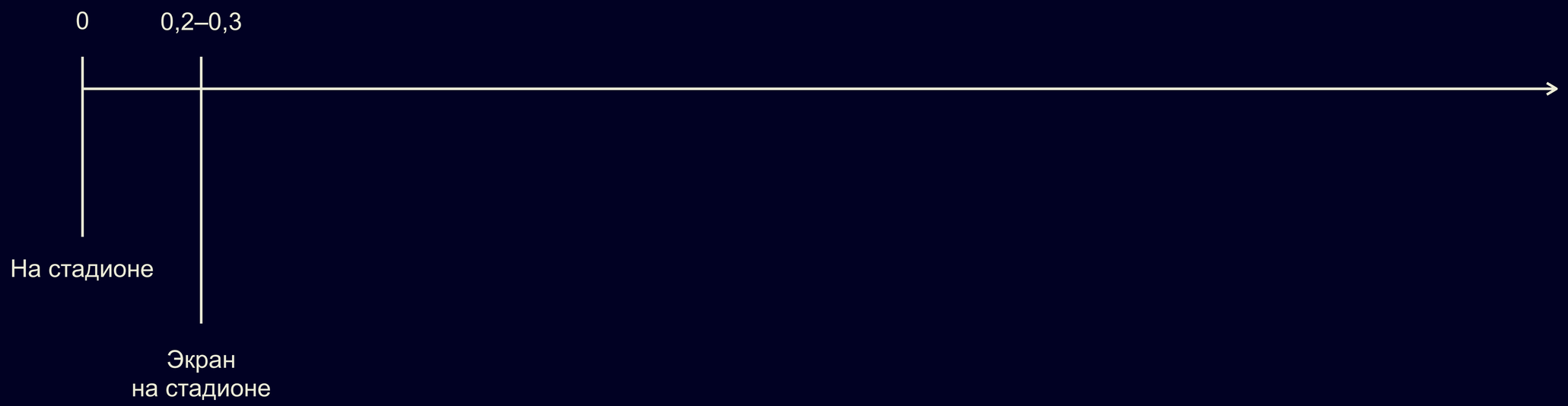
Спортивные трансляции

Стример и подписчики

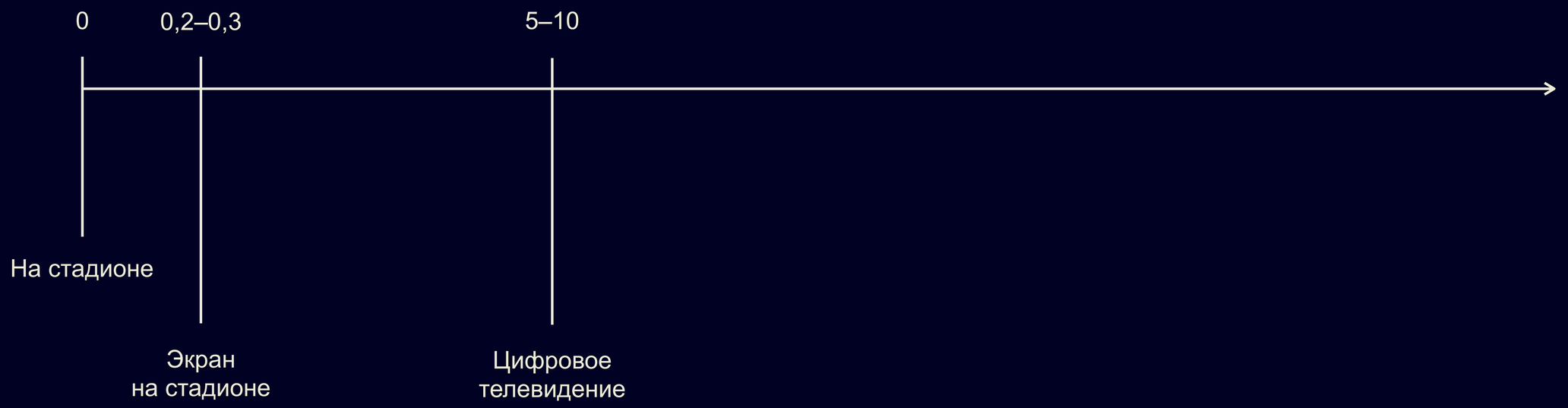
Как можно смотреть футбольный матч?

Как можно смотреть футбольный матч?

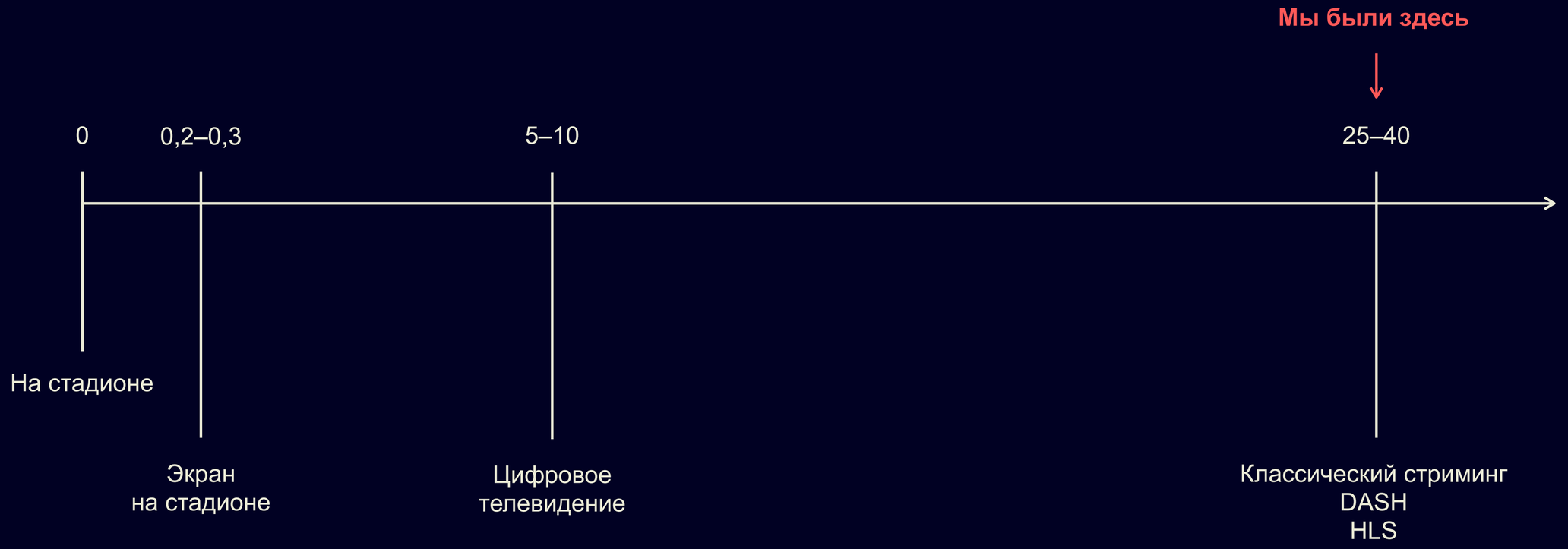
Как можно смотреть футбольный матч?



Как можно смотреть футбольный матч?



Как можно смотреть футбольный матч?



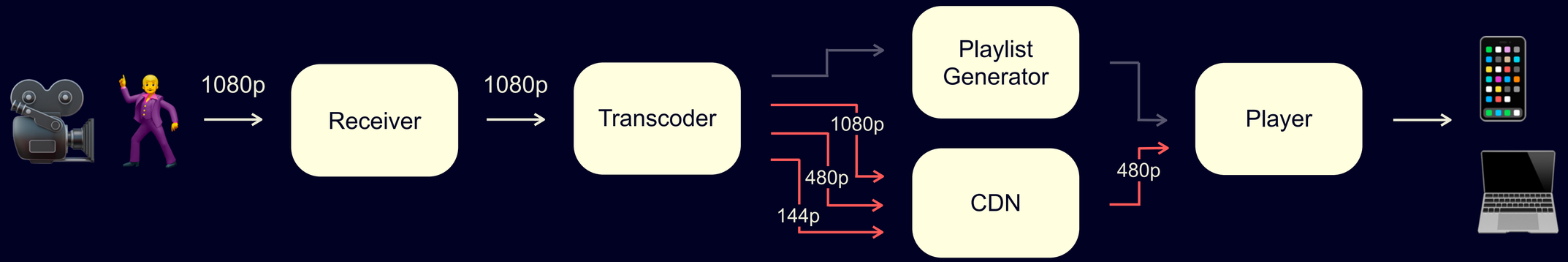
Бюджет задержки

Во что мы целились?

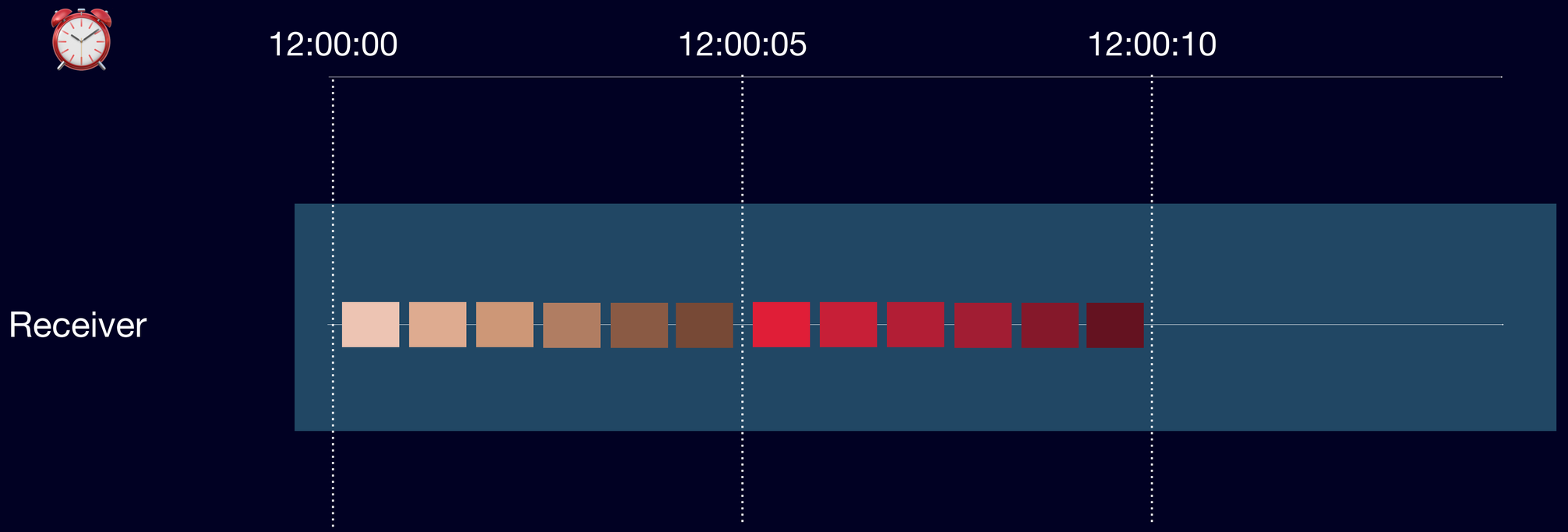
10 секунд

Старая архитектура

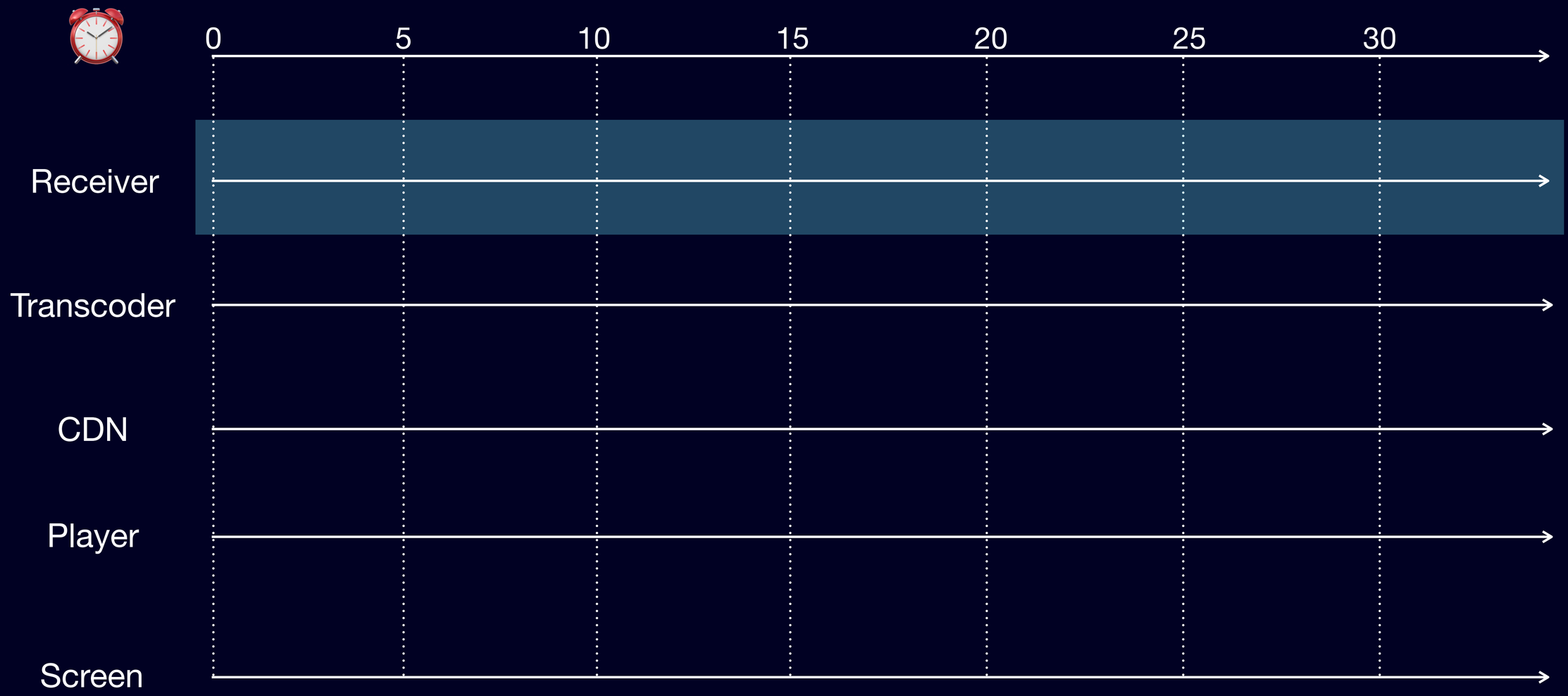
Как все работало раньше?



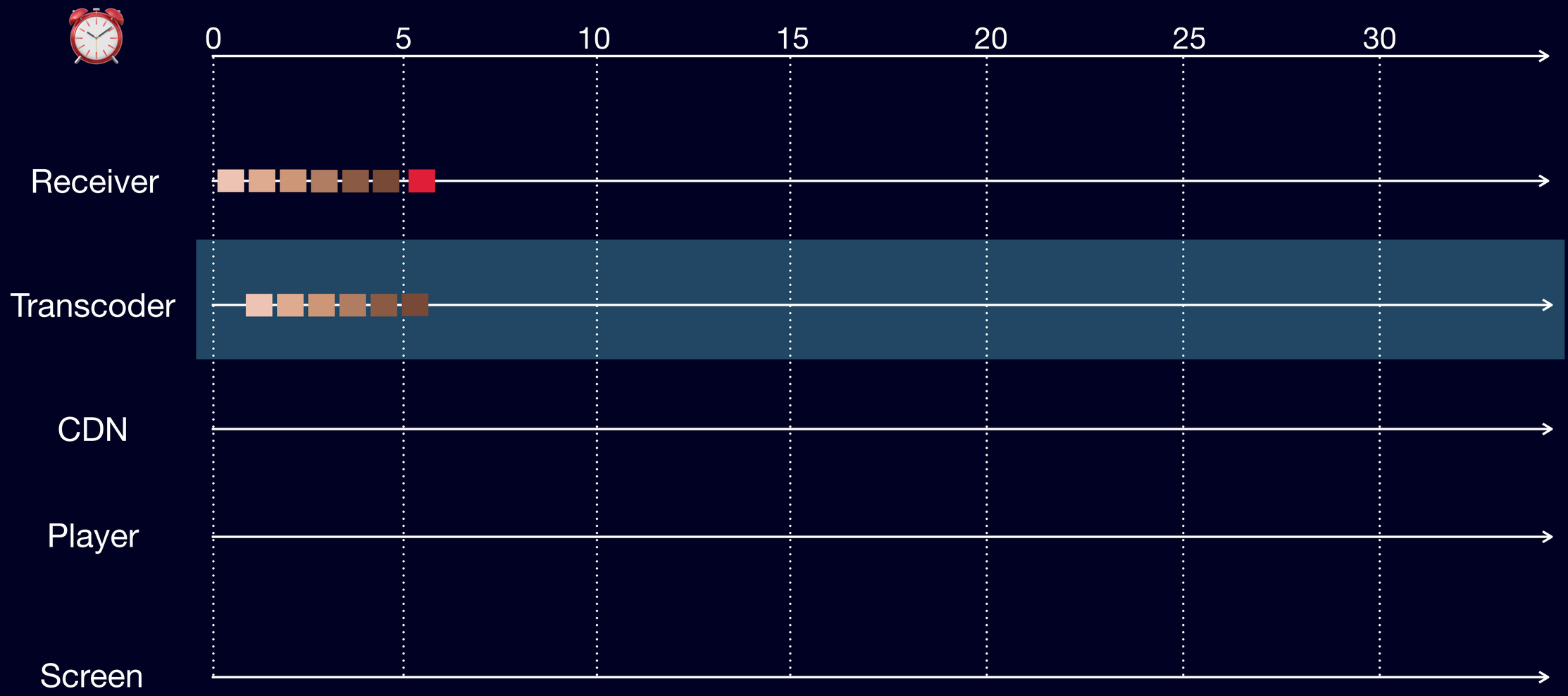
Прием сигнала



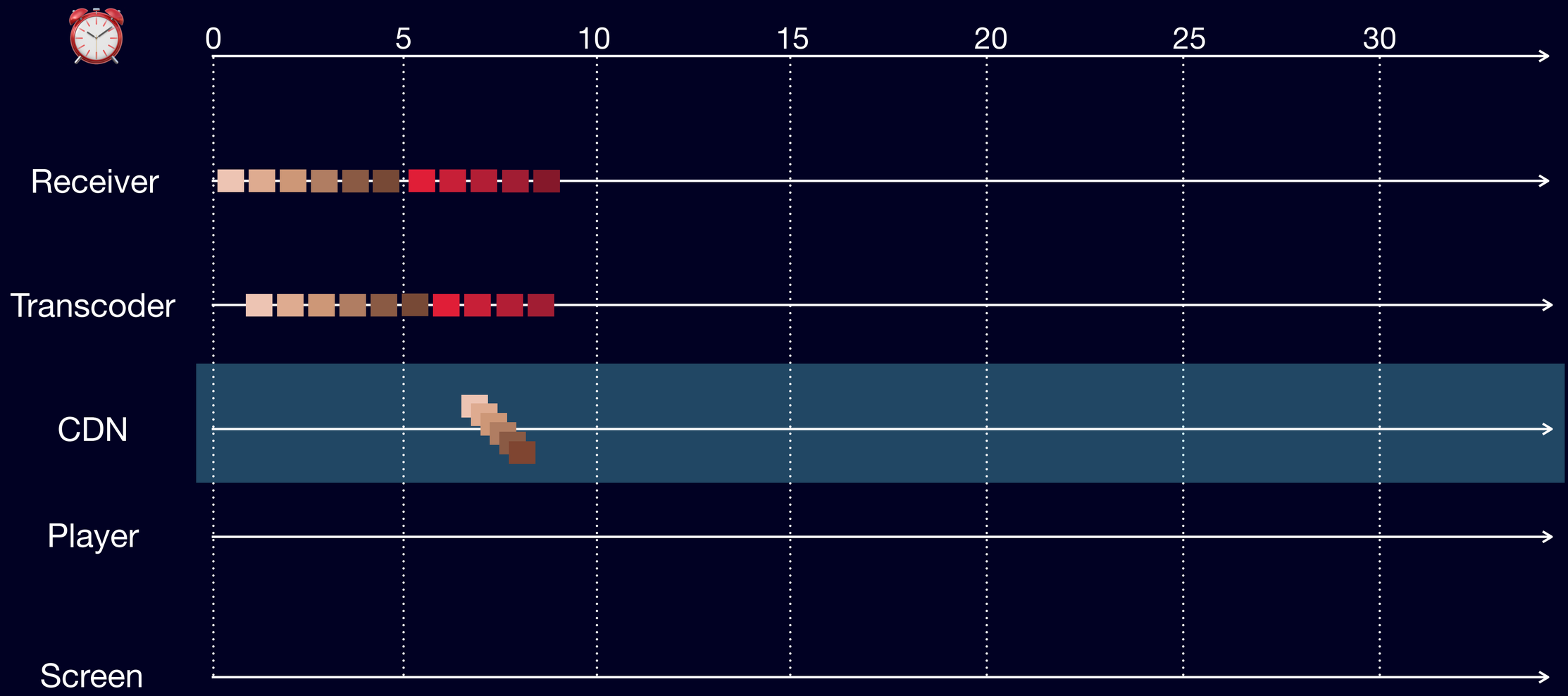
Откуда берется задержка?



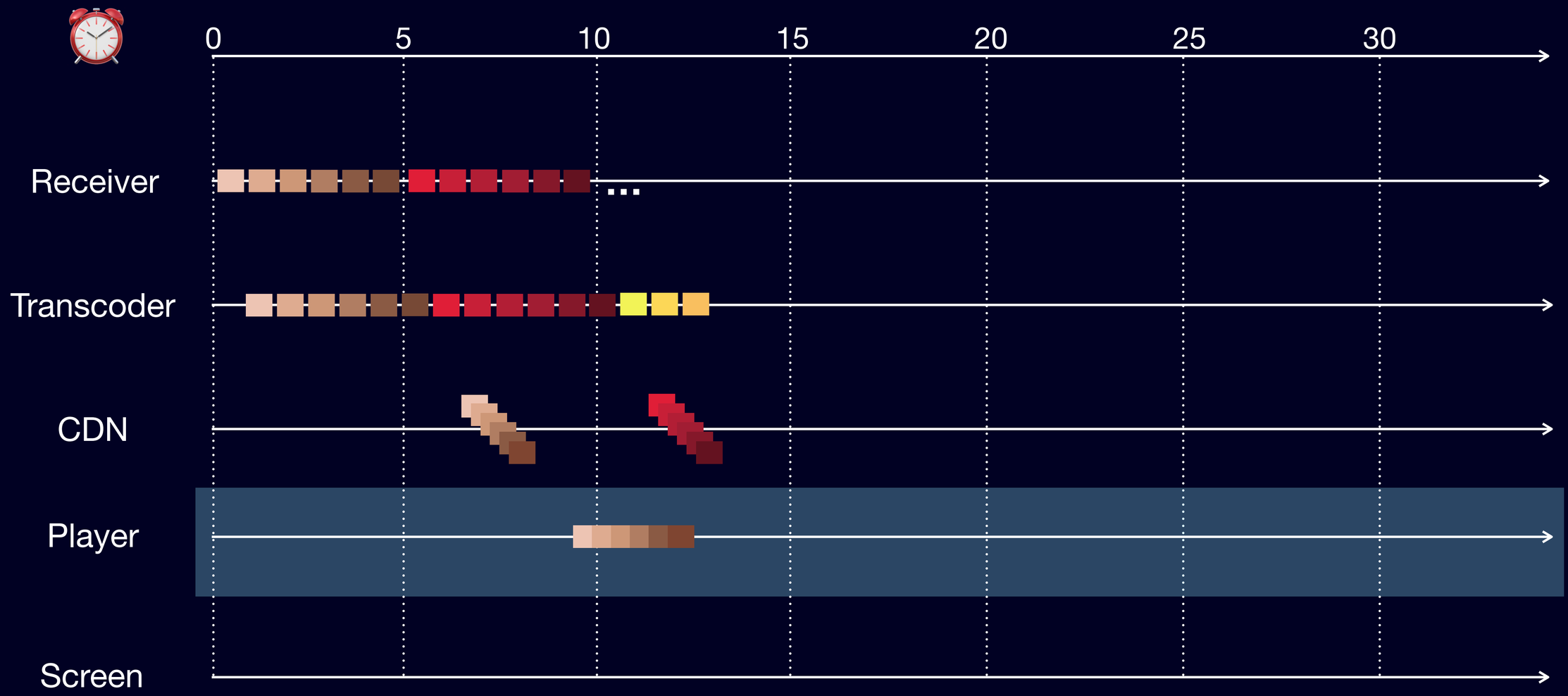
Откуда берется задержка?



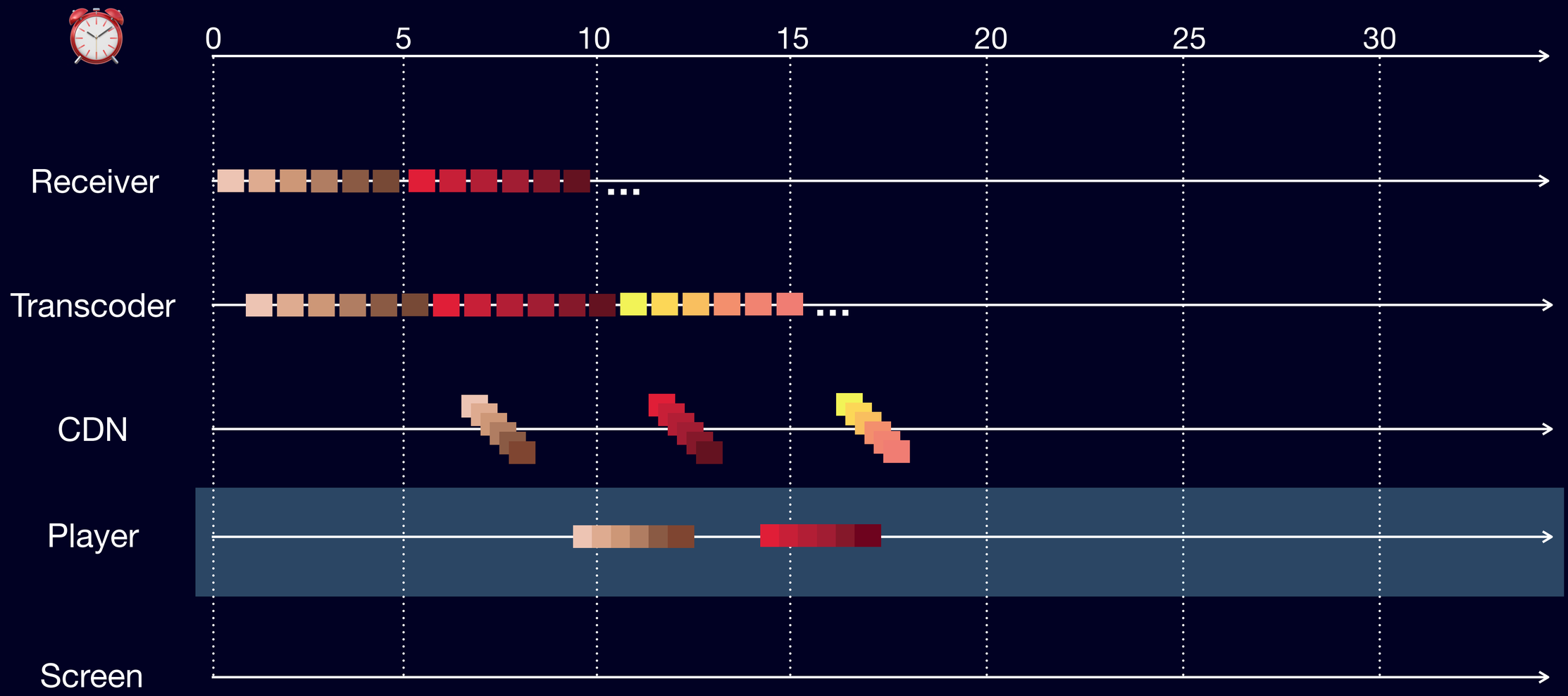
Откуда берется задержка?



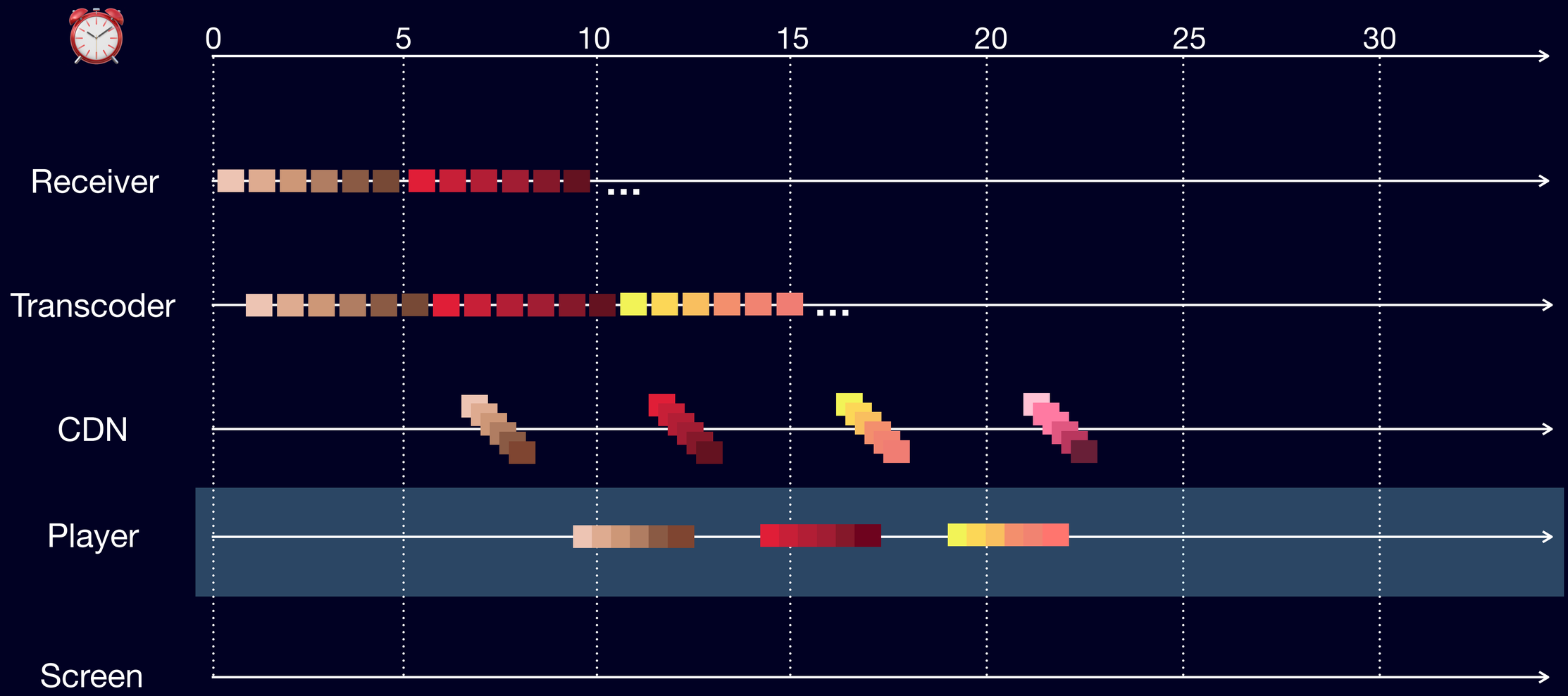
Откуда берется задержка?



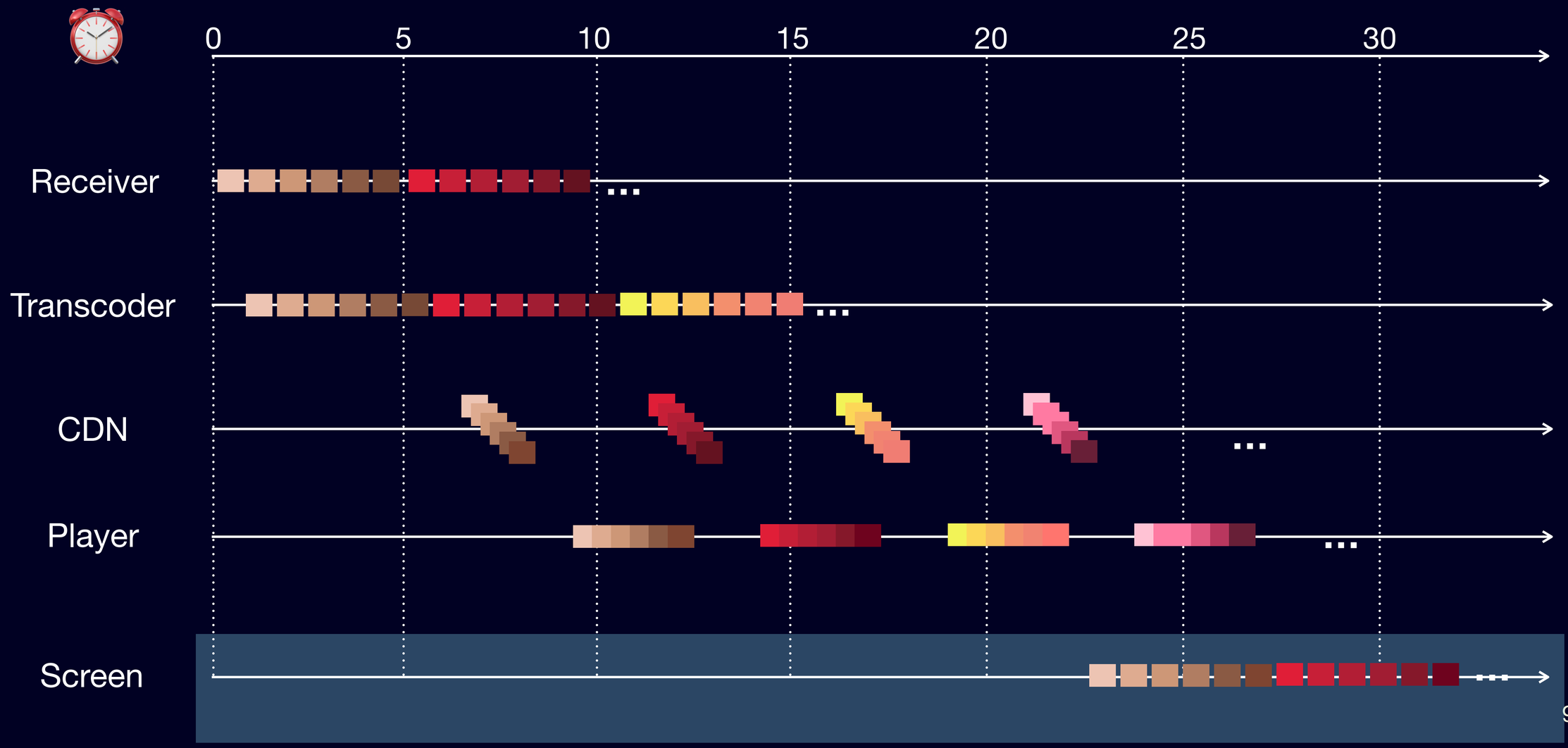
Откуда берется задержка?



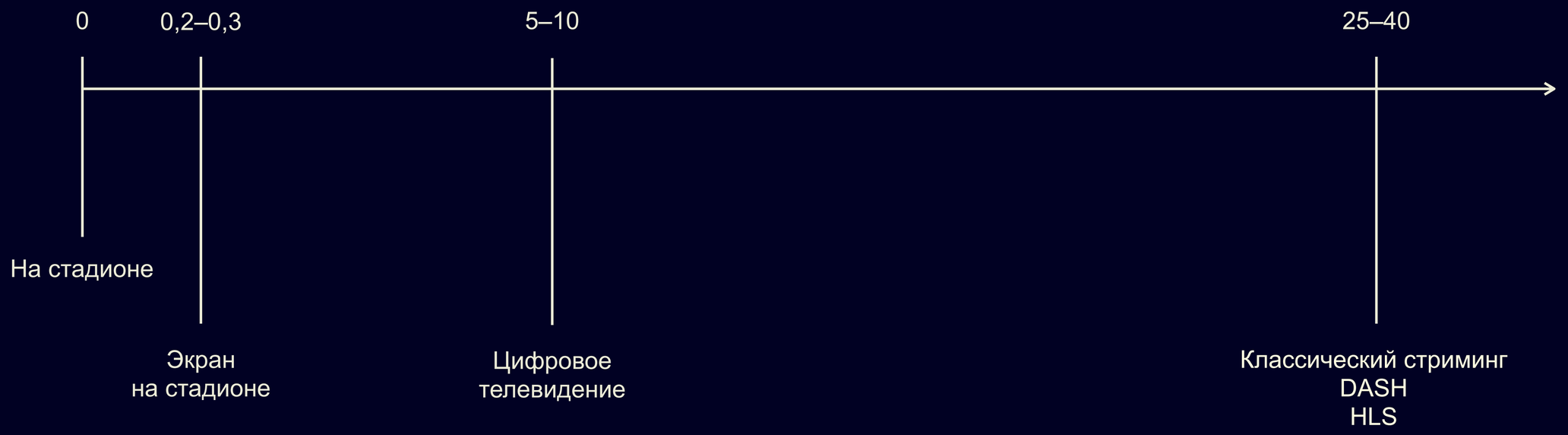
Откуда берется задержка?



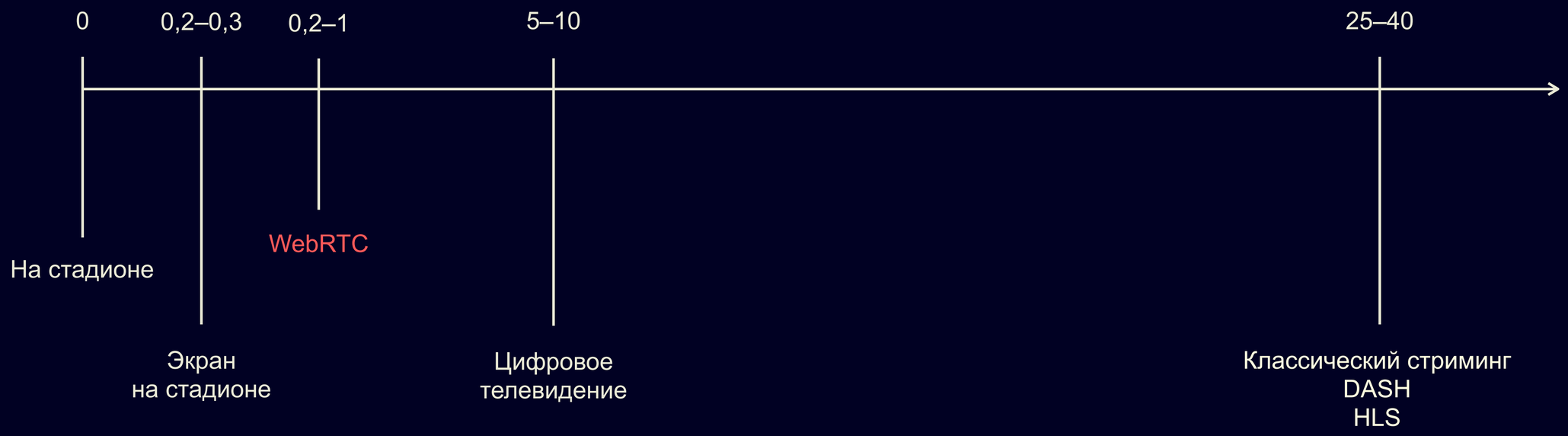
Откуда берется задержка?



Куда можно было двигаться



Куда можно было двигаться



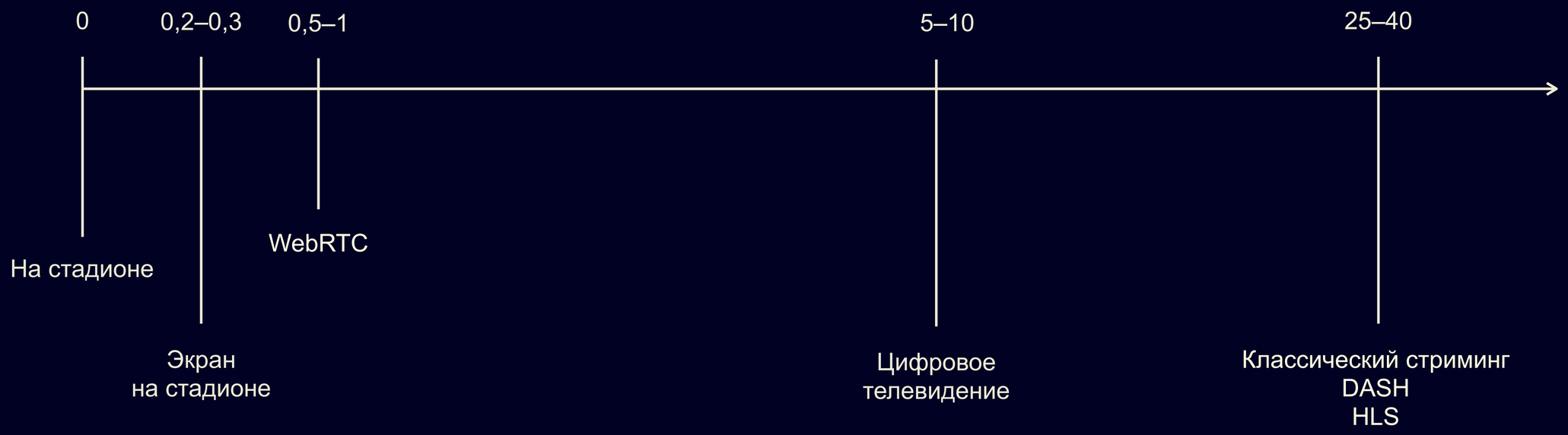
WebRTC

\+ Супер-маленькая задержка

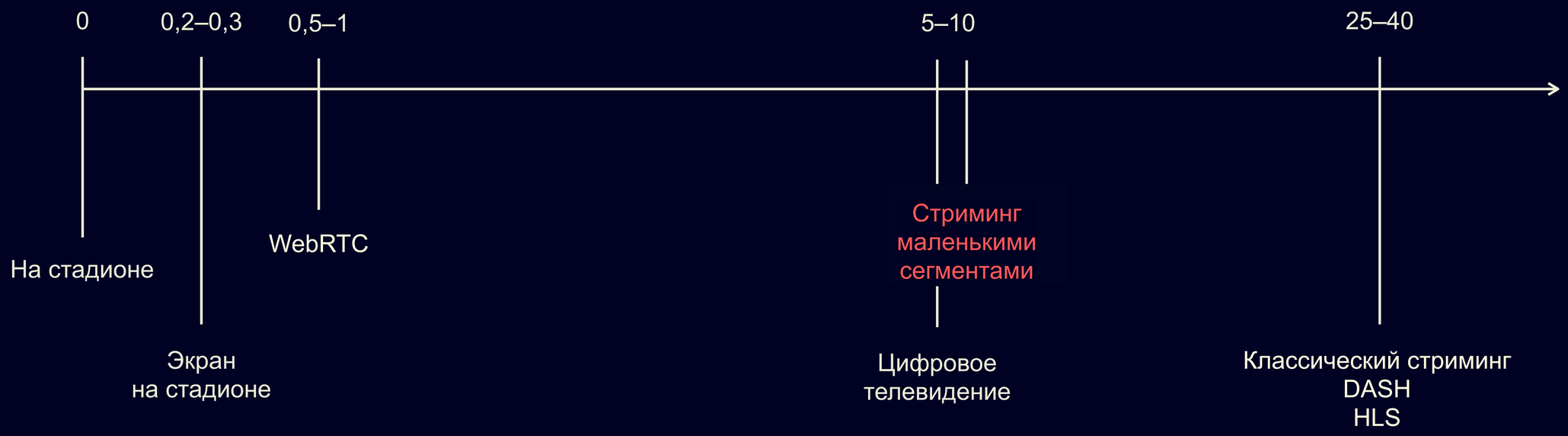
\- Отсутствует гарантия на доставку

\- Не все клиенты поддерживают

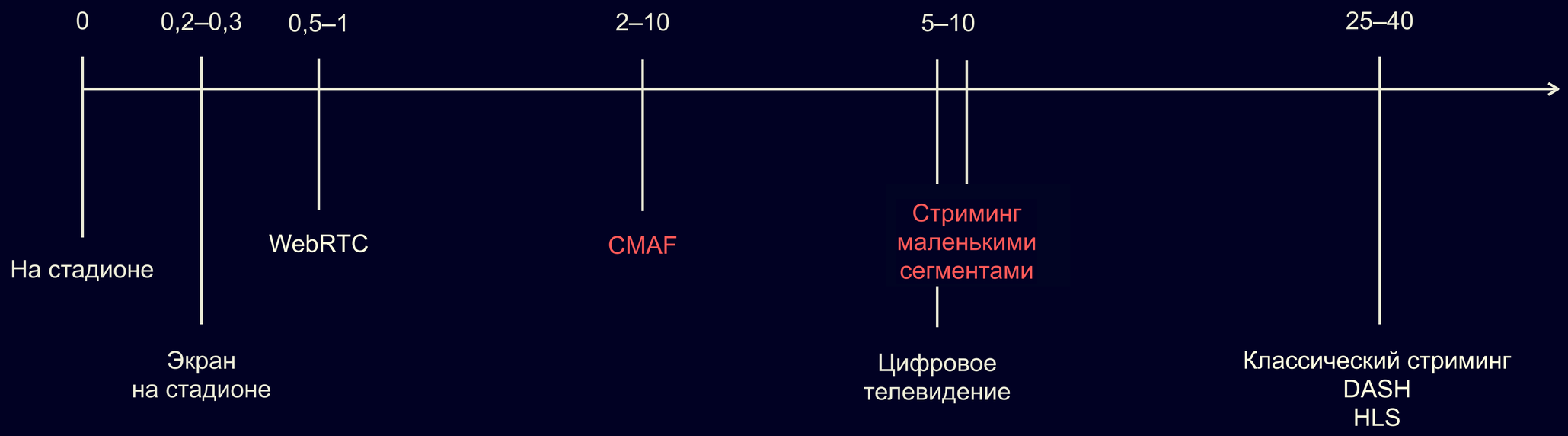
Куда можно было двигаться



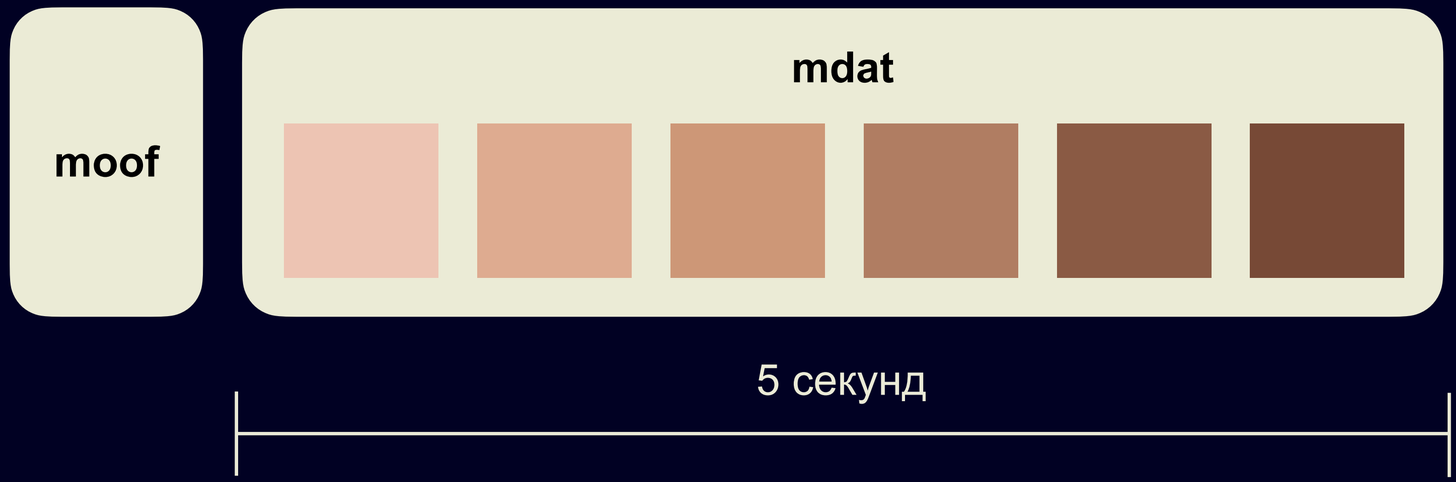
Куда можно было двигаться



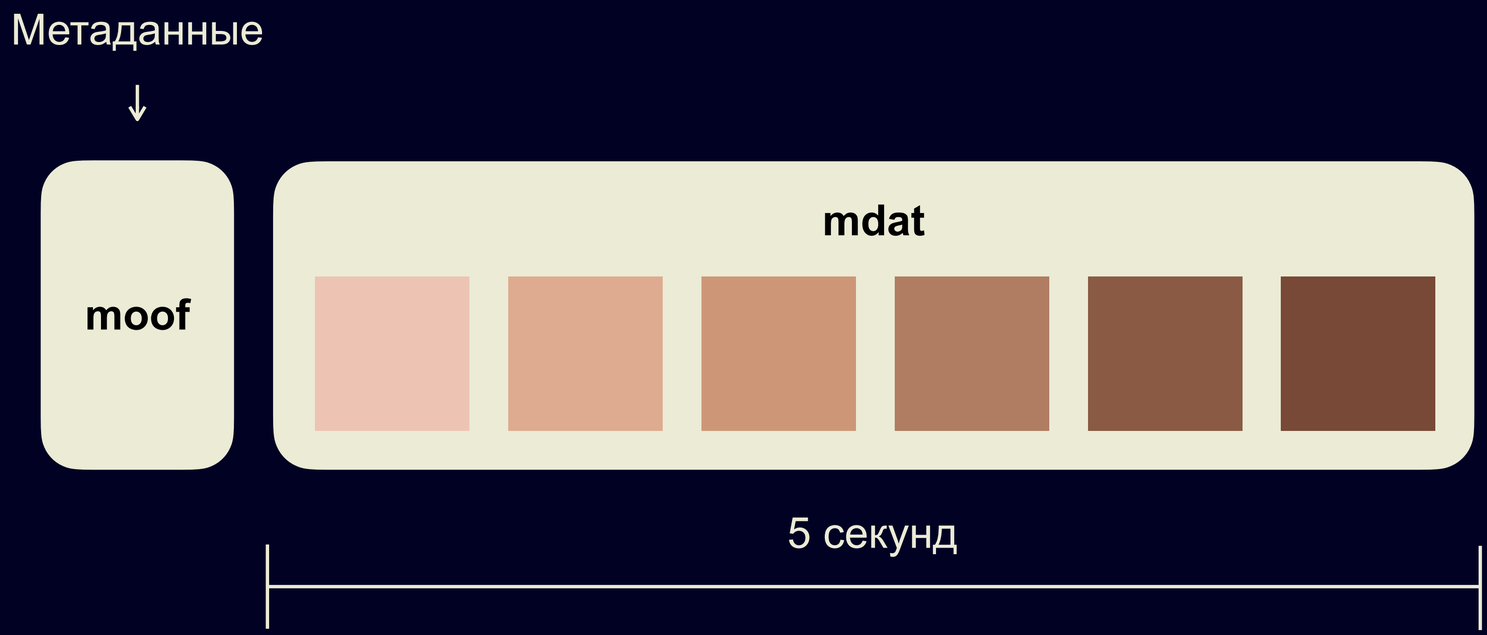
Куда можно было двигаться



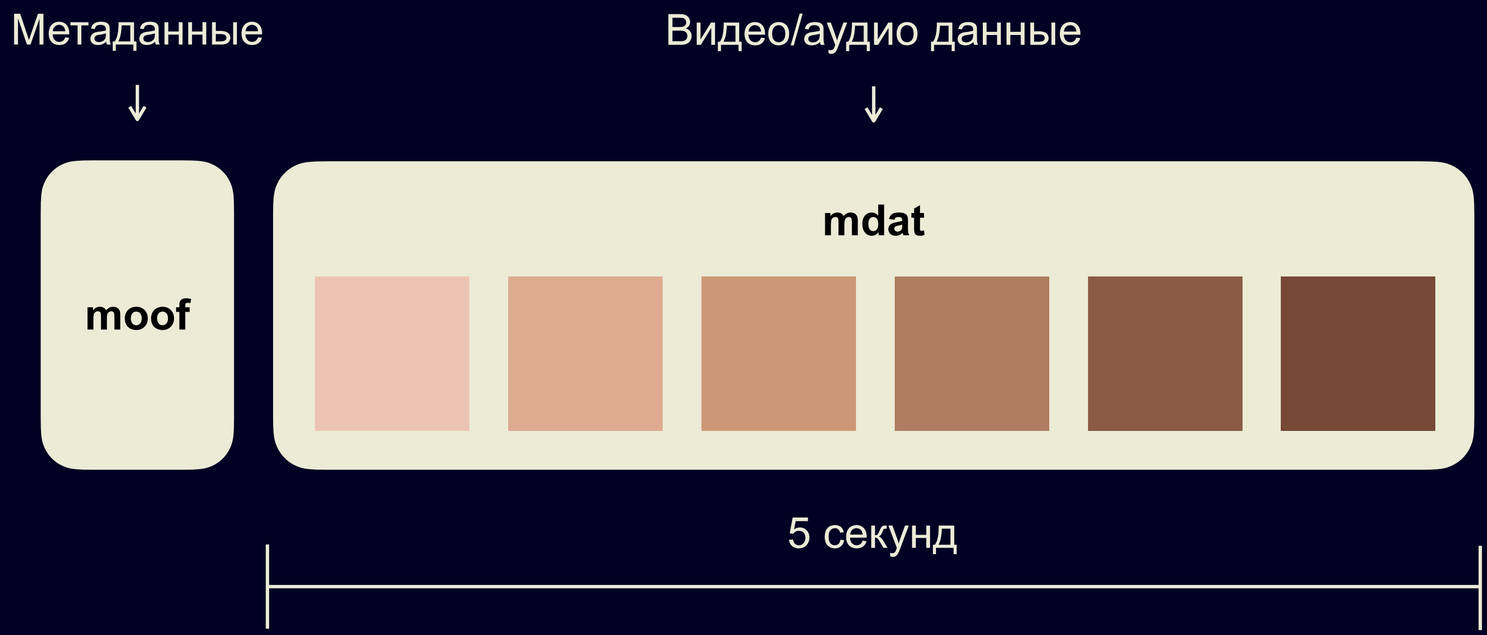
Как устроен сегмент?



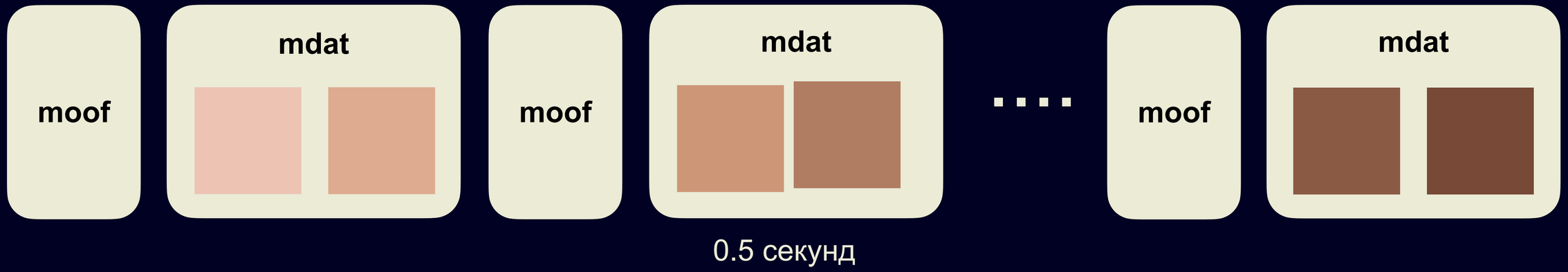
Как устроен сегмент?



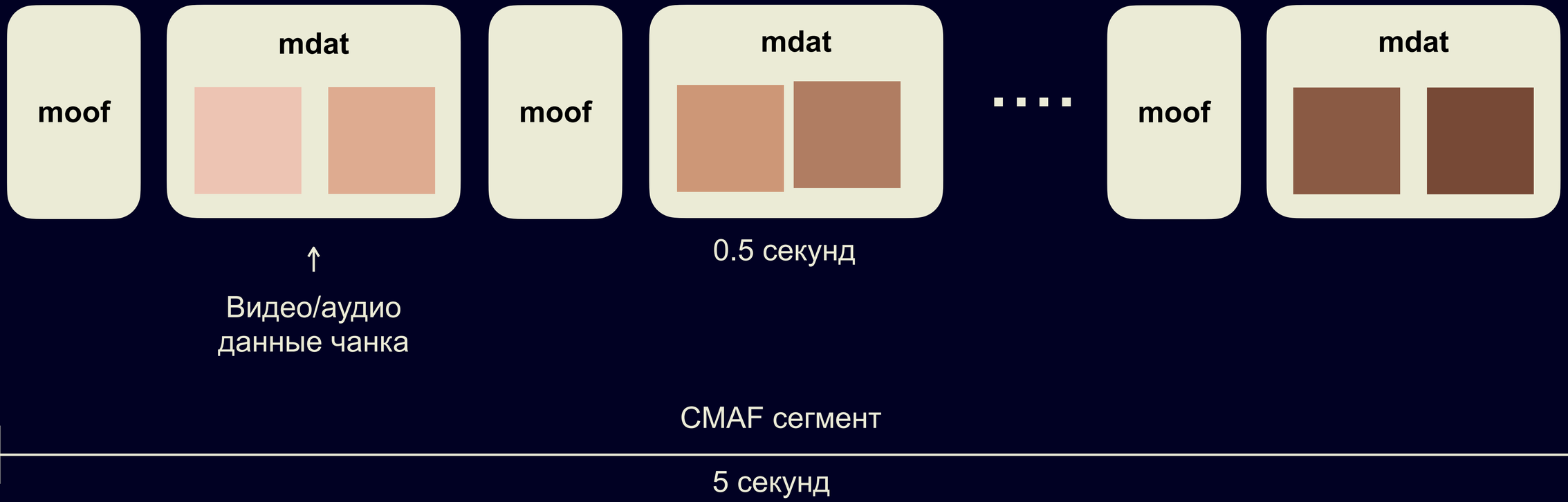
Как устроен сегмент?



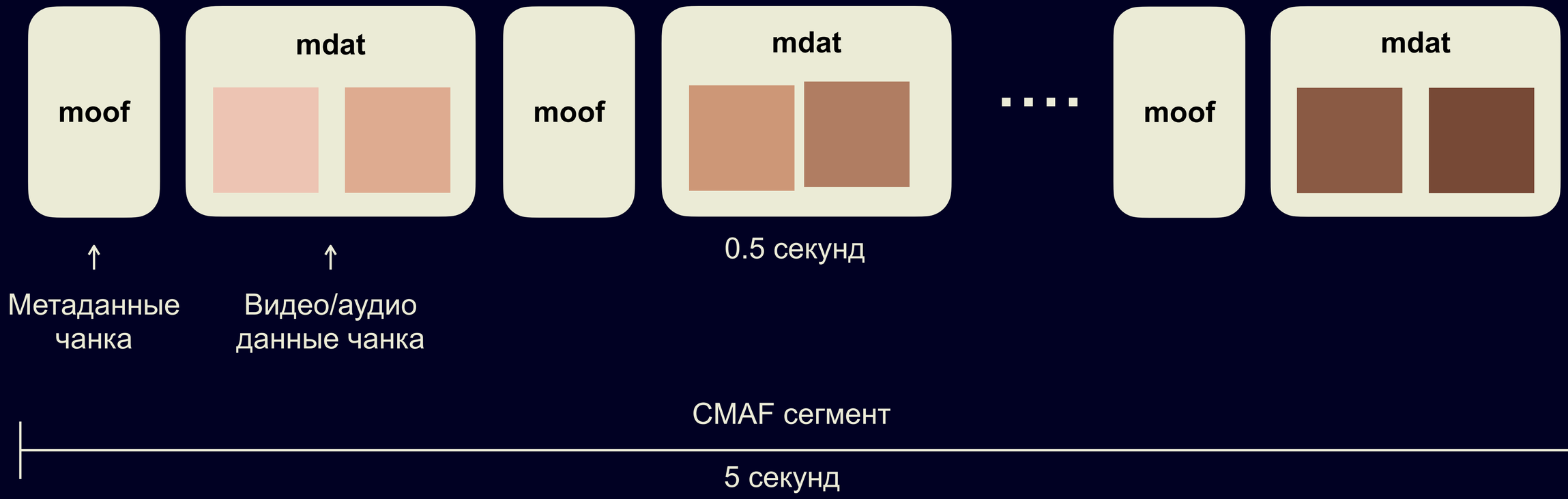
Как устроен CMAF-сегмент?



Как устроен CMAF-сегмент?



Как устроен CMAF-сегмент?



Как раздать неполный сегмент?

Сегмент

Как раздать неполный сегмент?

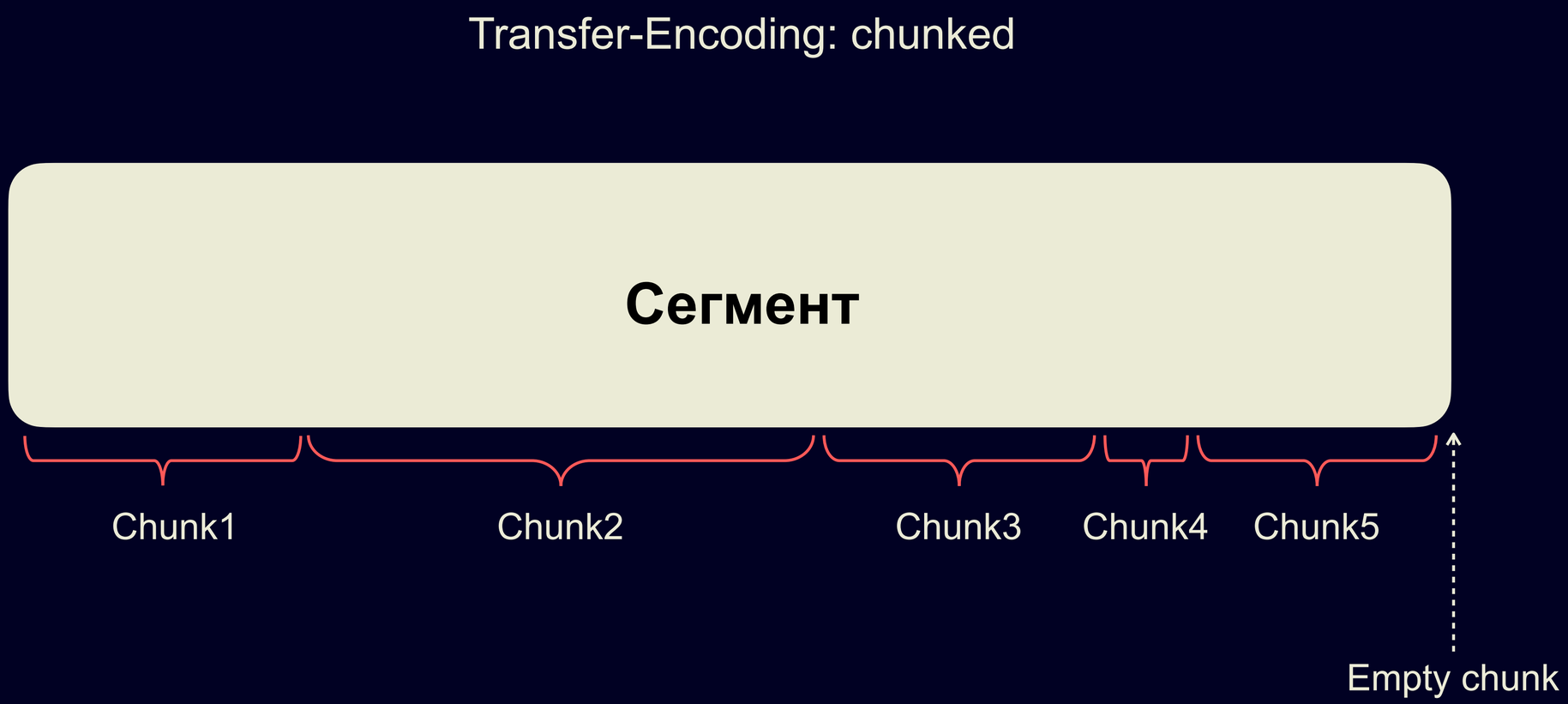
Transfer-Encoding: chunked

Сегмент

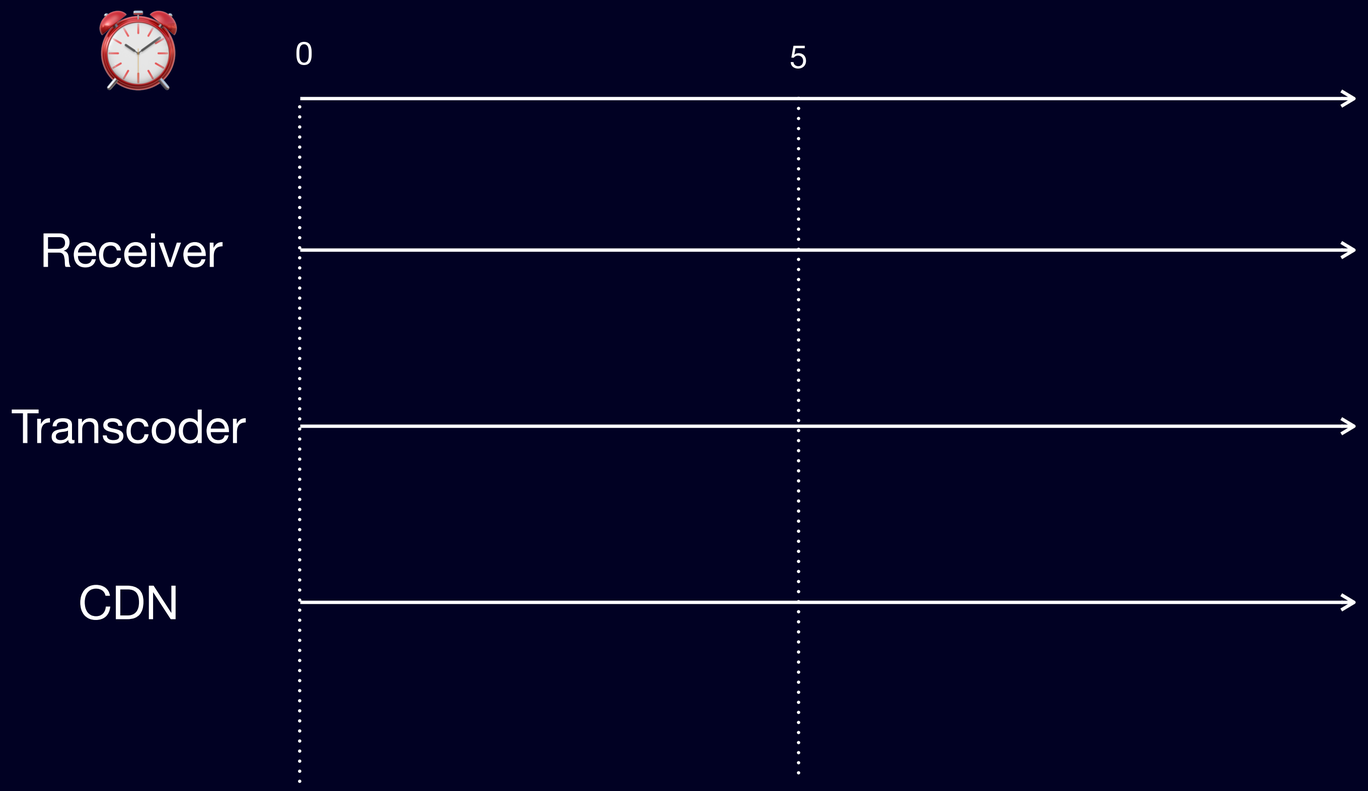
Как раздать неполный сегмент?

Transfer-Encoding: chunked

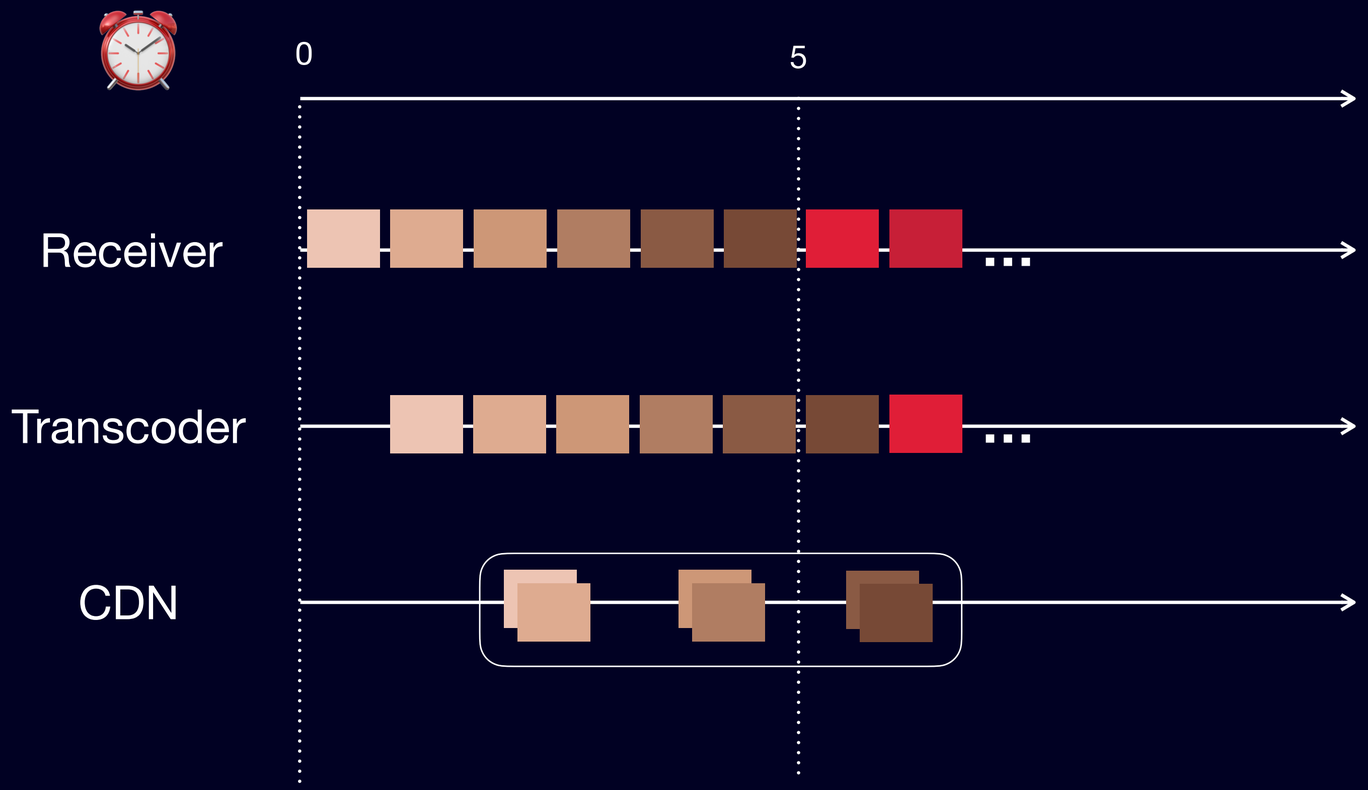
Как раздать неполный сегмент?



Задержка для CMAF

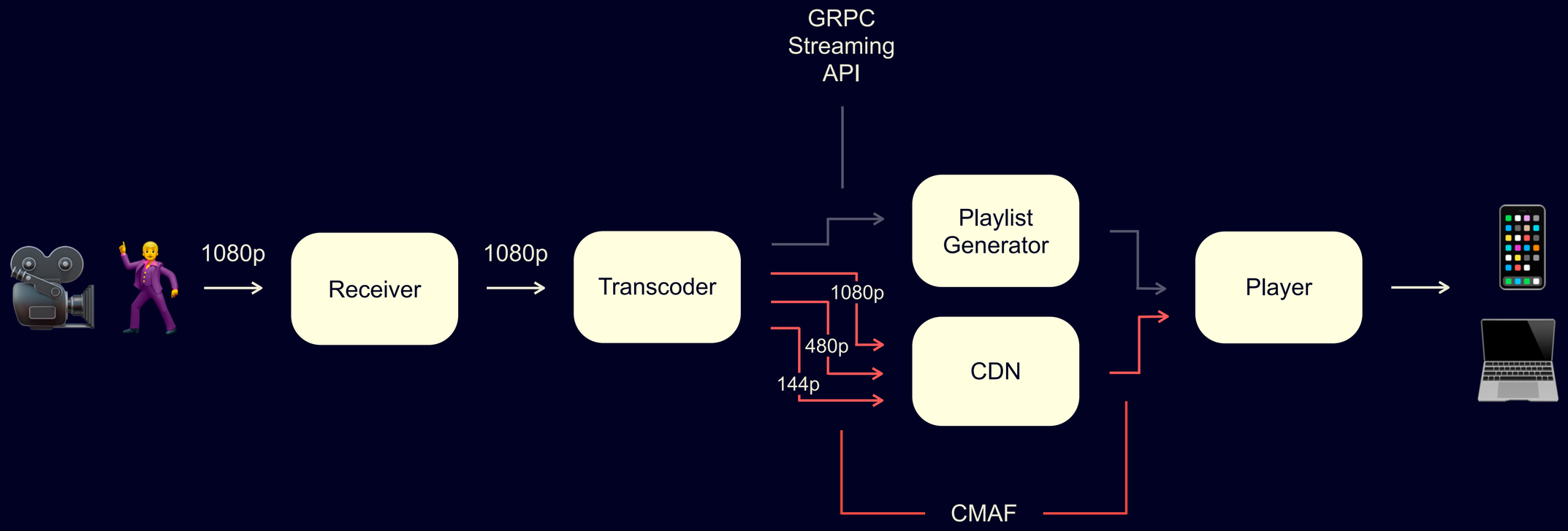


Задержка для CMAF

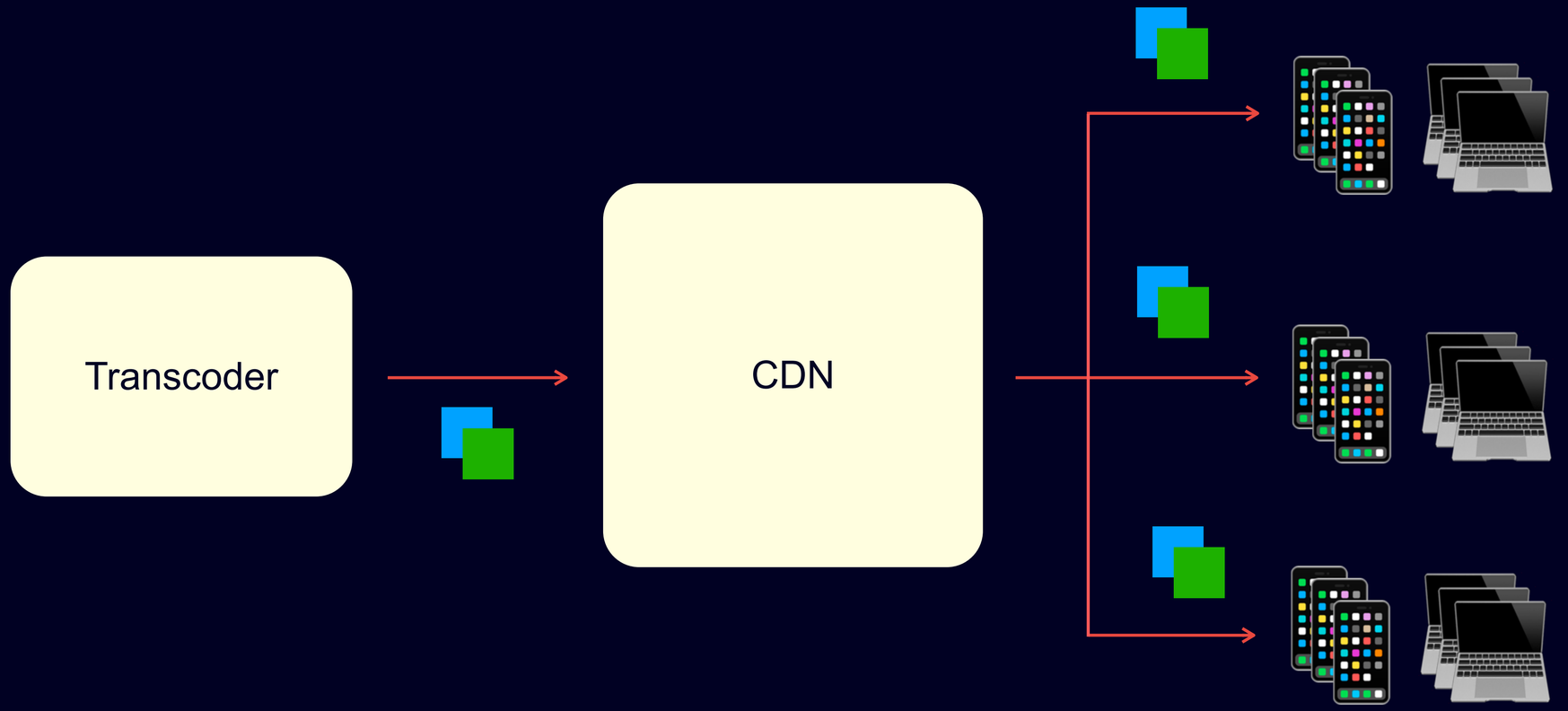


Новая архитектура

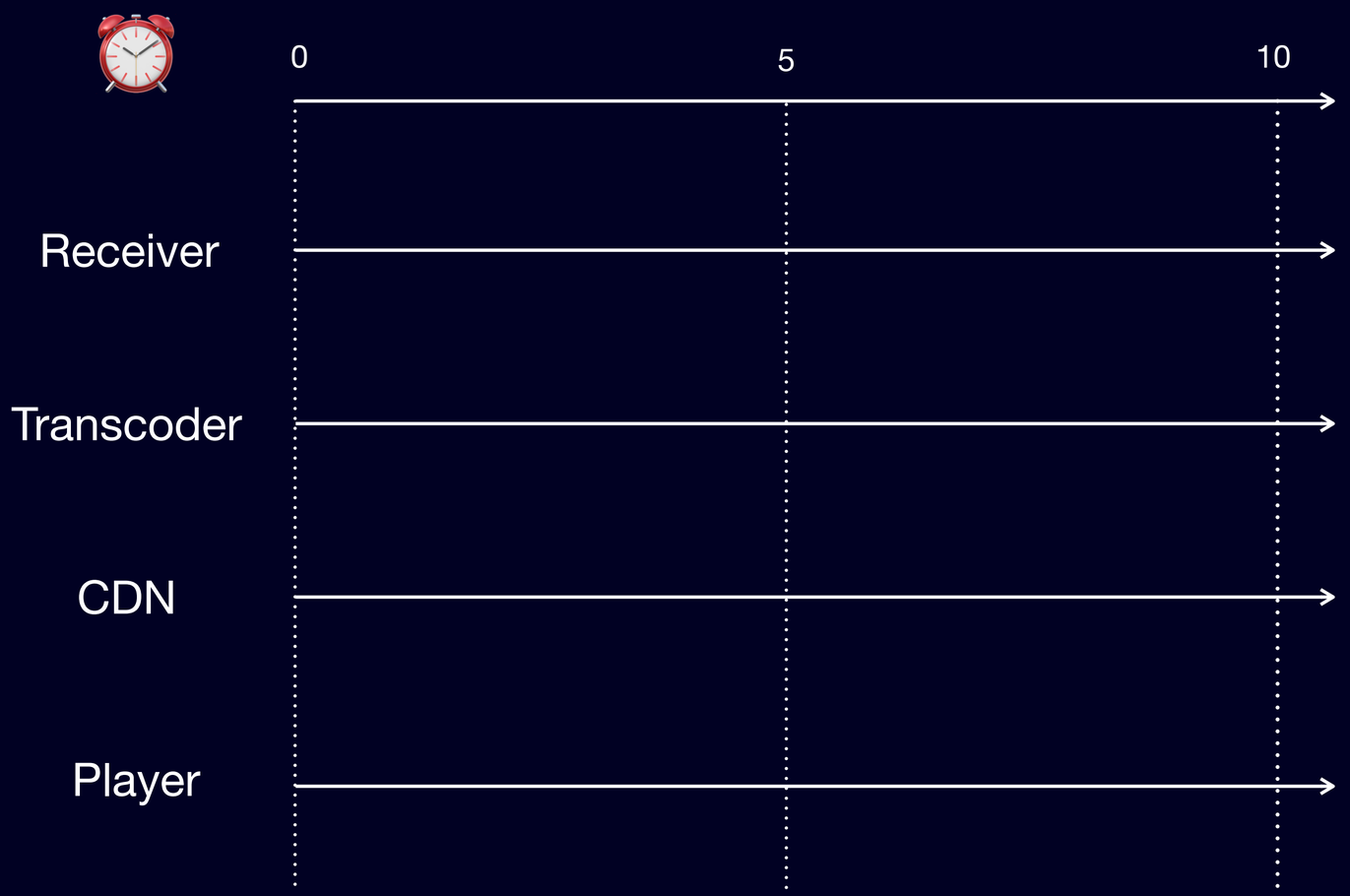
А что поменялось?



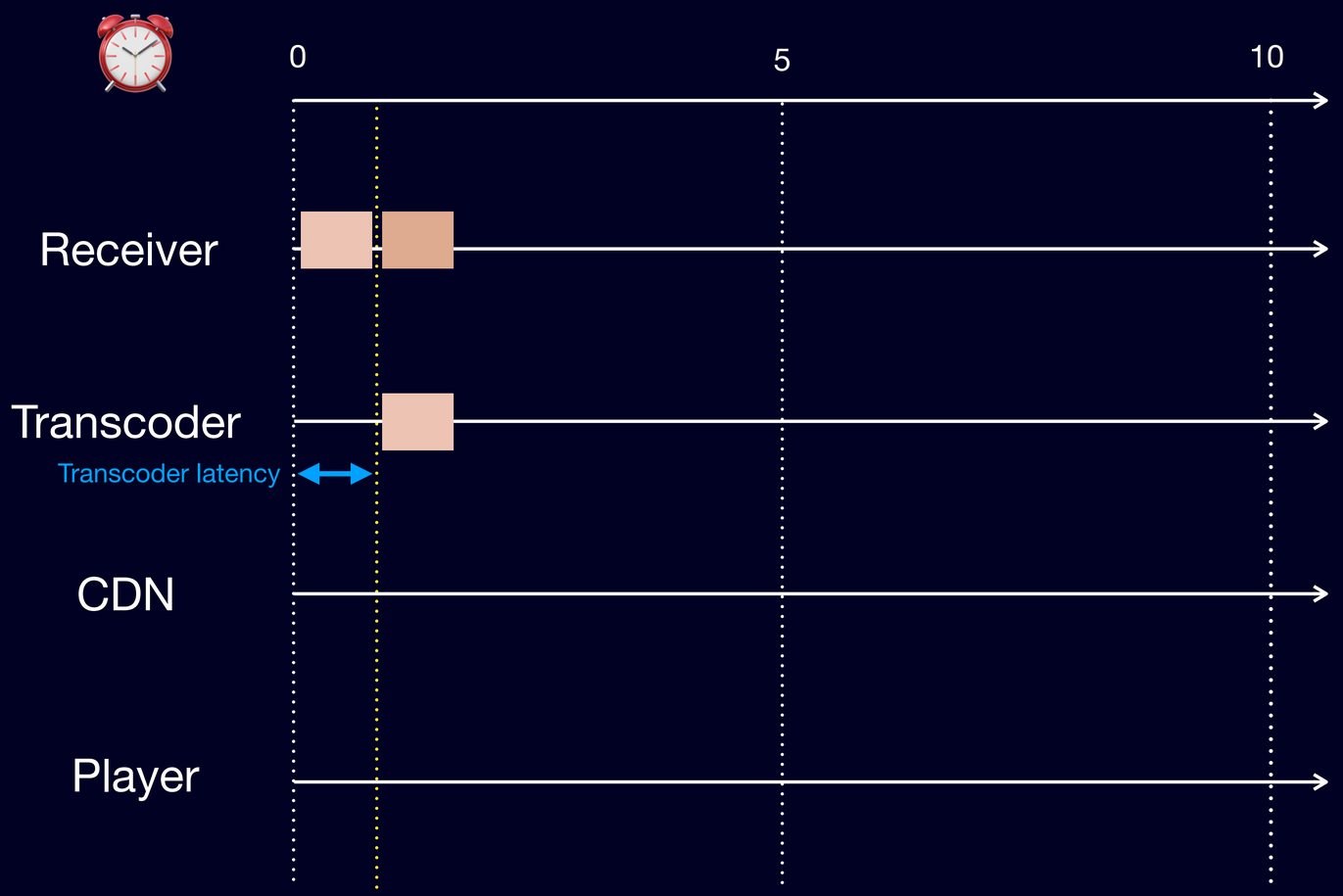
Cache follow в CDN



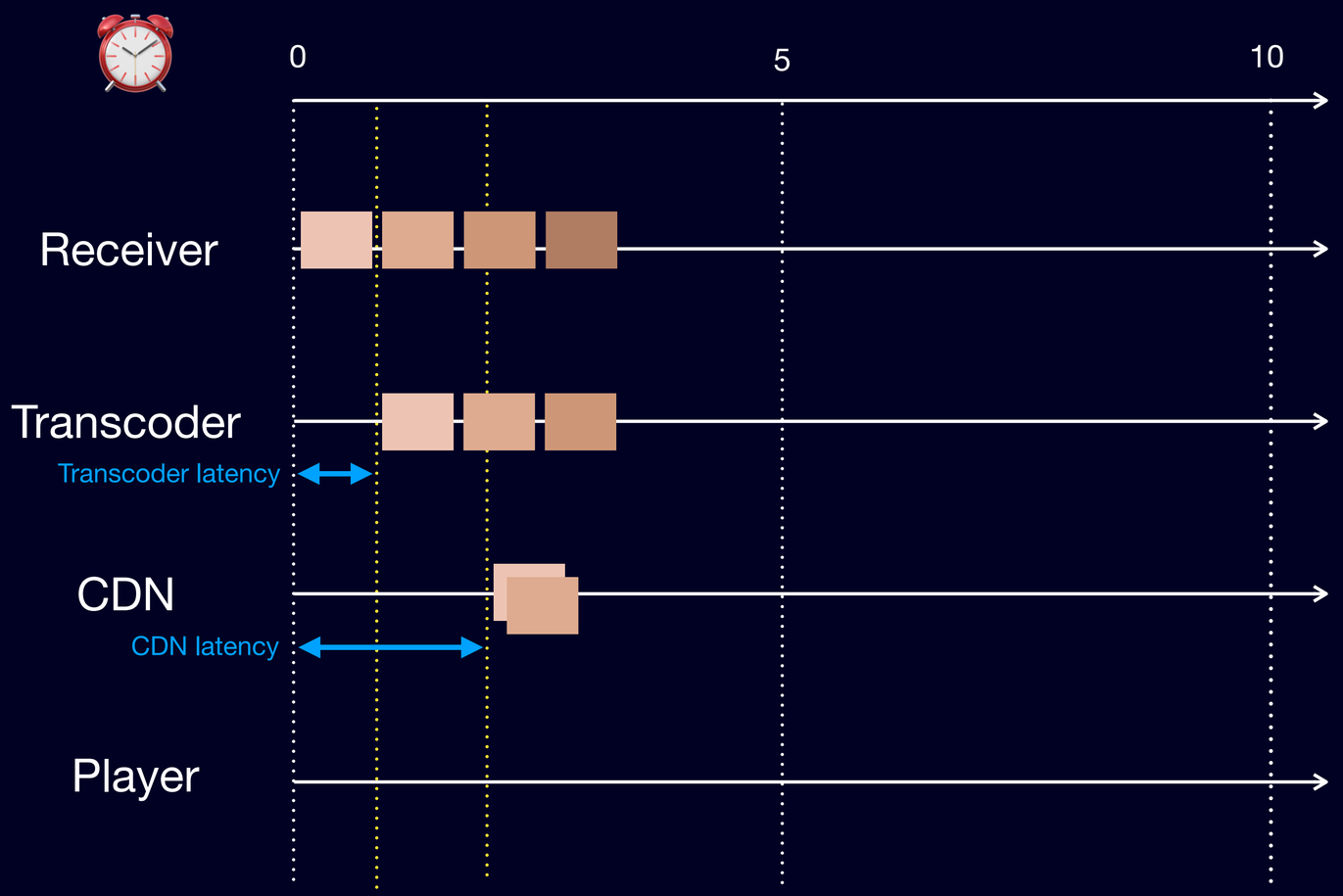
Метрики



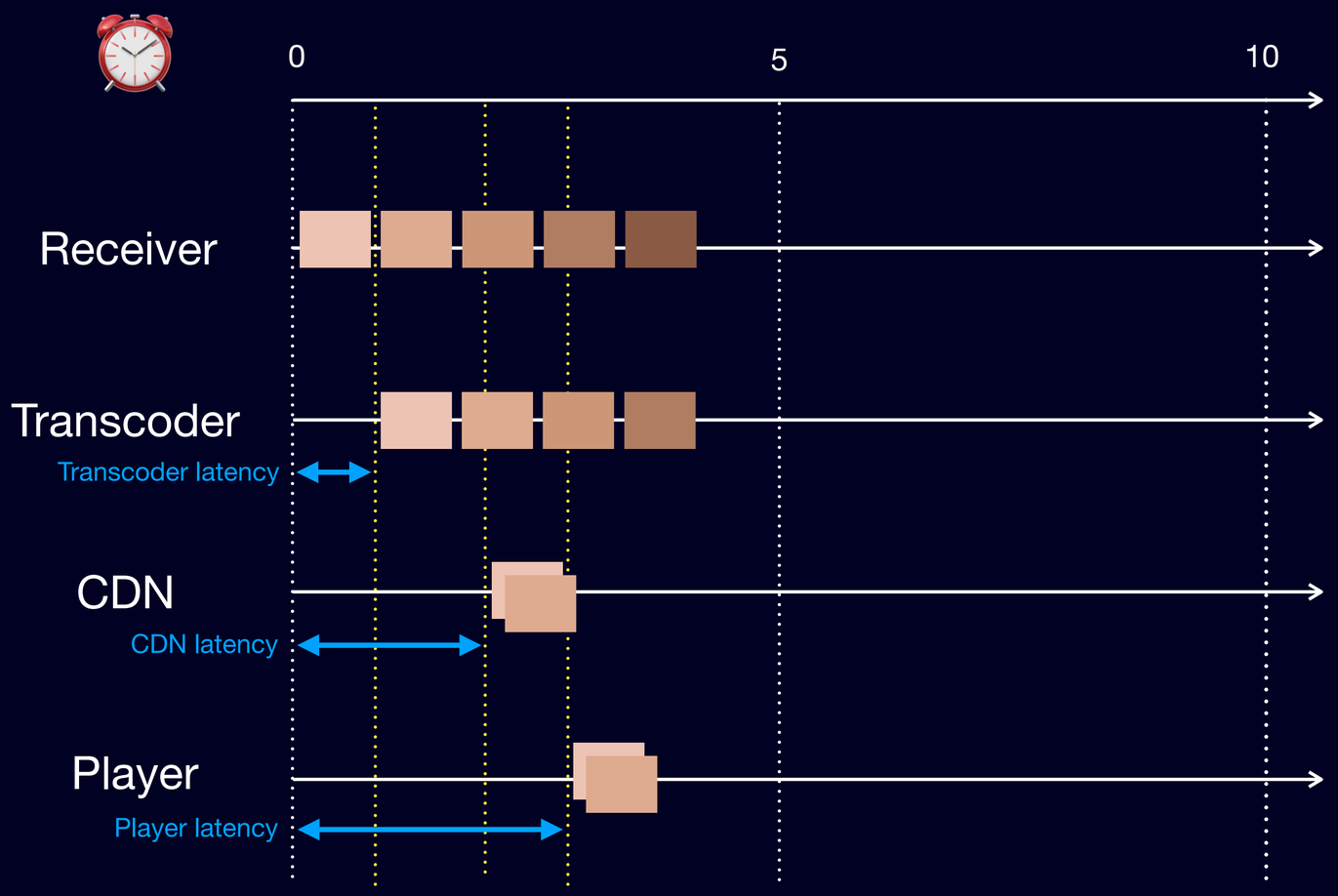
Метрики



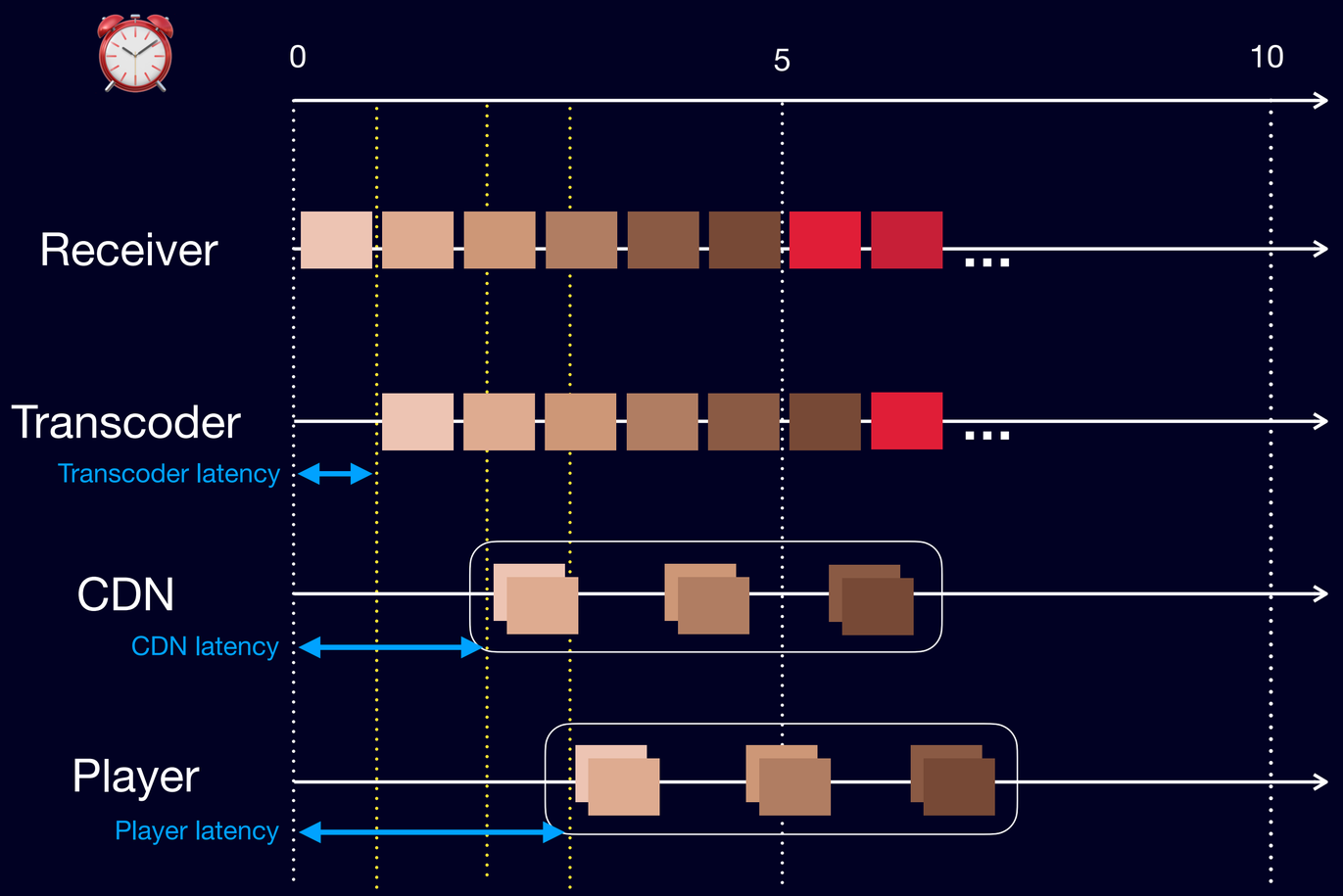
Метрики


0
5
10
Receiver
Transcoder
Transcoder latency
CDN
CDN latency
Player

Метрики


0
5
10
Receiver
Transcoder
Transcoder latency
CDN
CDN latency
Player
Player latency

Метрики

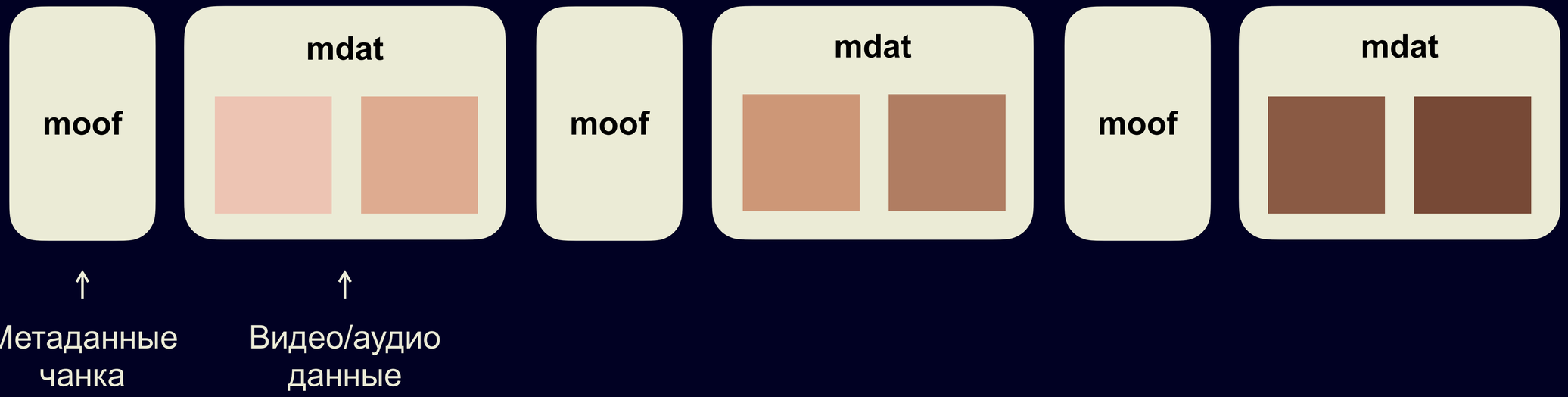

0
5
10
Receiver
Transcoder
Transcoder latency
CDN
CDN latency
Player
Player latency

Раскладываем задержку по компонентам

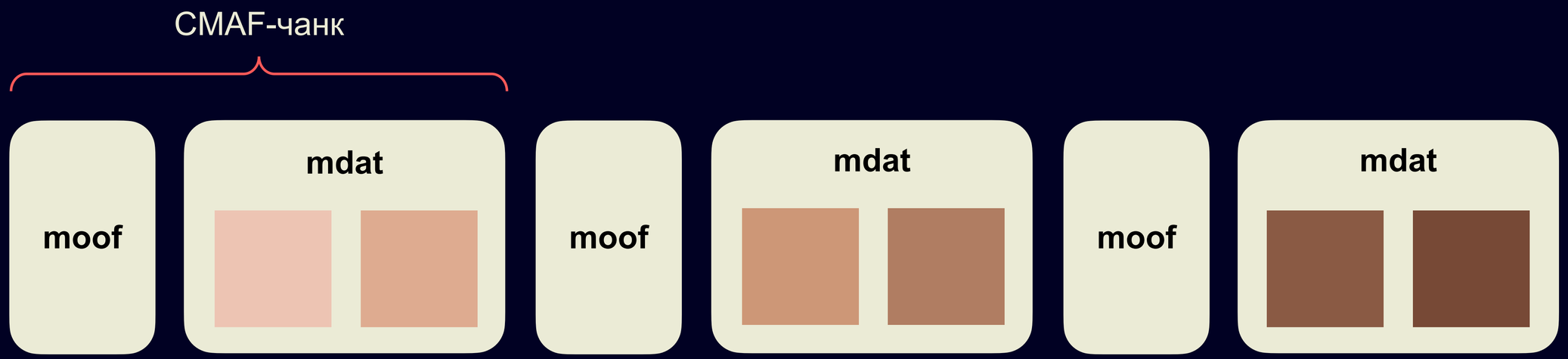
Итоговая задержка в CDN

Как показать CMAF сегменты в плеере?

Как устроен CMAF-сегмент?



Как устроен CMAF-сегмент?



Нужно уметь читать
тело ответа
в процессе загрузки

XMLHttpRequest

XMLHttpRequest 💔

Fetch API

Fetch API
+
ReadableStream API

ReadableStream API

Usage % of all users ?

Global 94.61%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *
	12-13	2-64	4-42	3.1-10	10-29	3.2-10.2		
6-10	14-93	65-92	43-94	10.1-14.1	30-79	10.3-14.8		2.1-4.4.4
11	94	93	95	15	80	15	all	94
		94-95	96-98	TP				

caniuse.com/mdn-api_readablestream

```
const response = await fetch('segment.m4s');
const reader = response.body.getReader();

async function pump() {
    const { value: chunk, done } = await reader.read();
    if (done) {
        return;
    }

    pushToSourceBuffer(chunk);
    return pump();
}

await pump();
```

```
const response = await fetch('segment.m4s');
const reader = response.body.getReader();

async function pump() {
    const { value: chunk, done } = await reader.read();
    if (done) {
        return;
    }

    pushToSourceBuffer(chunk);
    return pump();
}

await pump();
```

Получили заголовок ответа

```
const response = await fetch('segment.m4s');
const reader = response.body.getReader();

async function pump() {
    const { value: chunk, done } = await reader.read();
    if (done) {
        return;
    }

    pushToSourceBuffer(chunk);
    return pump();
}

await pump();
```

Запустили чтение
через ReadableStream

```
const response = await fetch('segment.m4s');
const reader = response.body.getReader();

async function pump() {
    const { value: chunk, done } = await reader.read();
    if (done) {
        return;
    }

    pushToSourceBuffer(chunk);
    return pump();
}

await pump();
```

Прочитали чанк данных

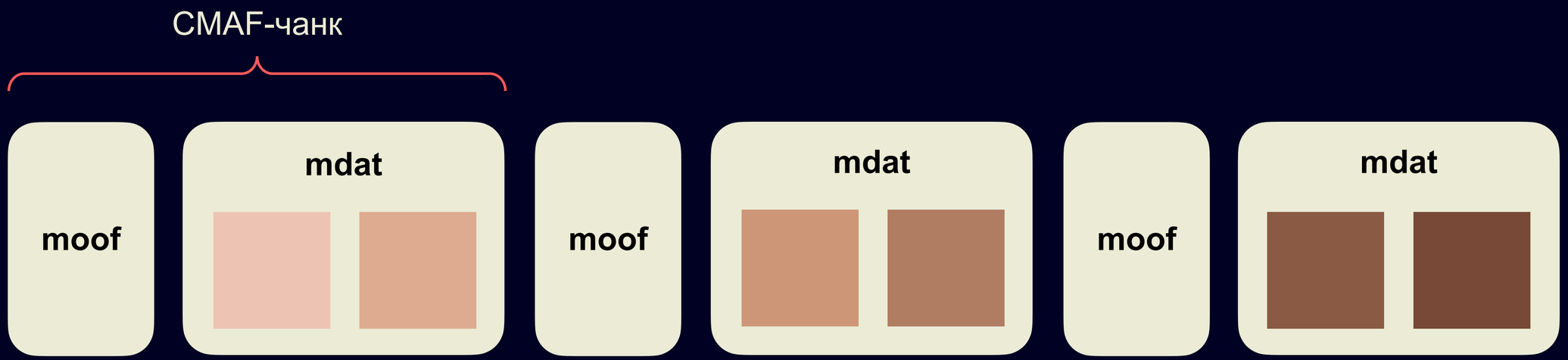
```
const response = await fetch('segment.m4s');
const reader = response.body.getReader();

async function pump() {
    const { value: chunk, done } = await reader.read();
    if (done) {
        return;
    }

    pushToSourceBuffer(chunk);
    return pump();
}

await pump();
```

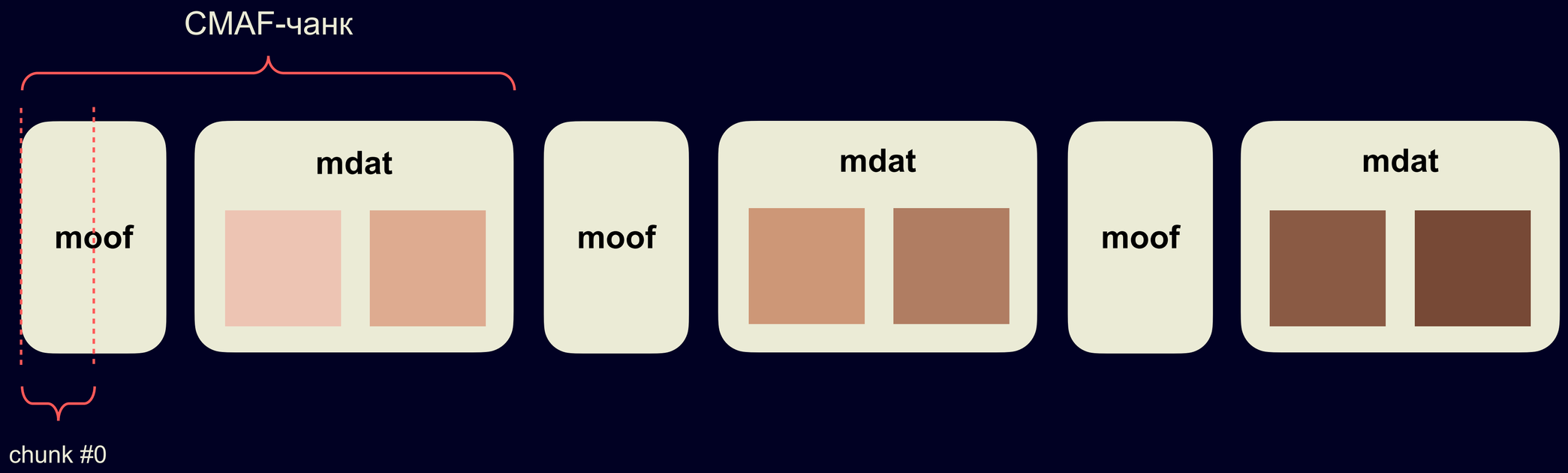
Тело ответа загружено полностью
(сервер отдал чанк нулевой длины)

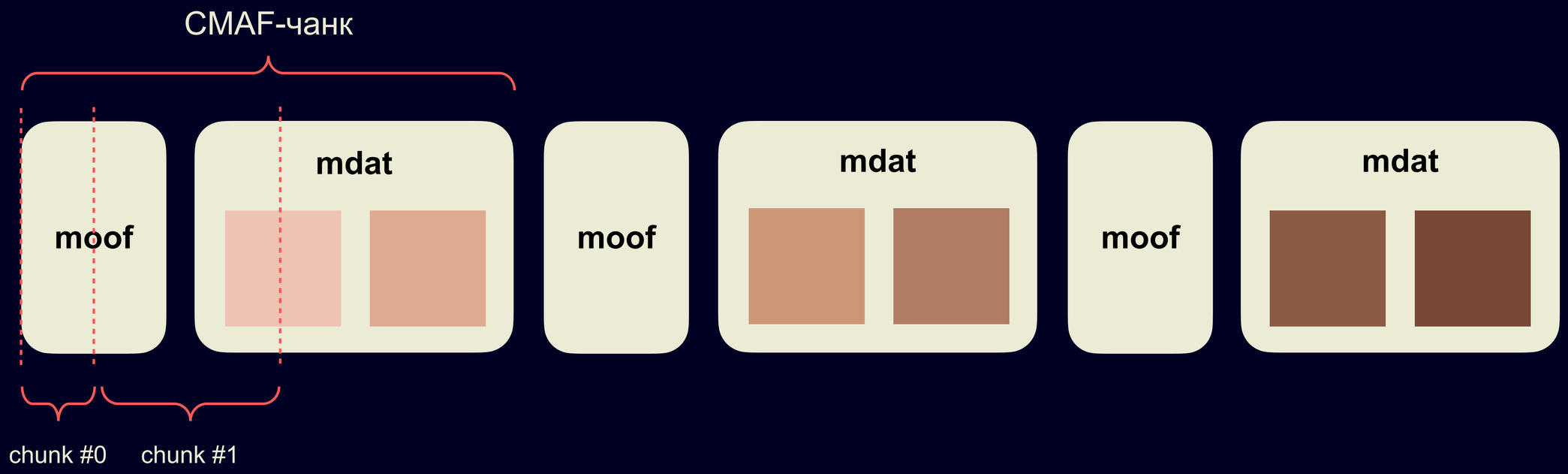
```
const response = await fetch('segment.m4s');
const reader = response.body.getReader();

async function pump() {
    const { value: chunk, done } = await reader.read();
    if (done) {
        return;
    }

    pushToSourceBuffer(chunk);
    return pump();
}

await pump();
```

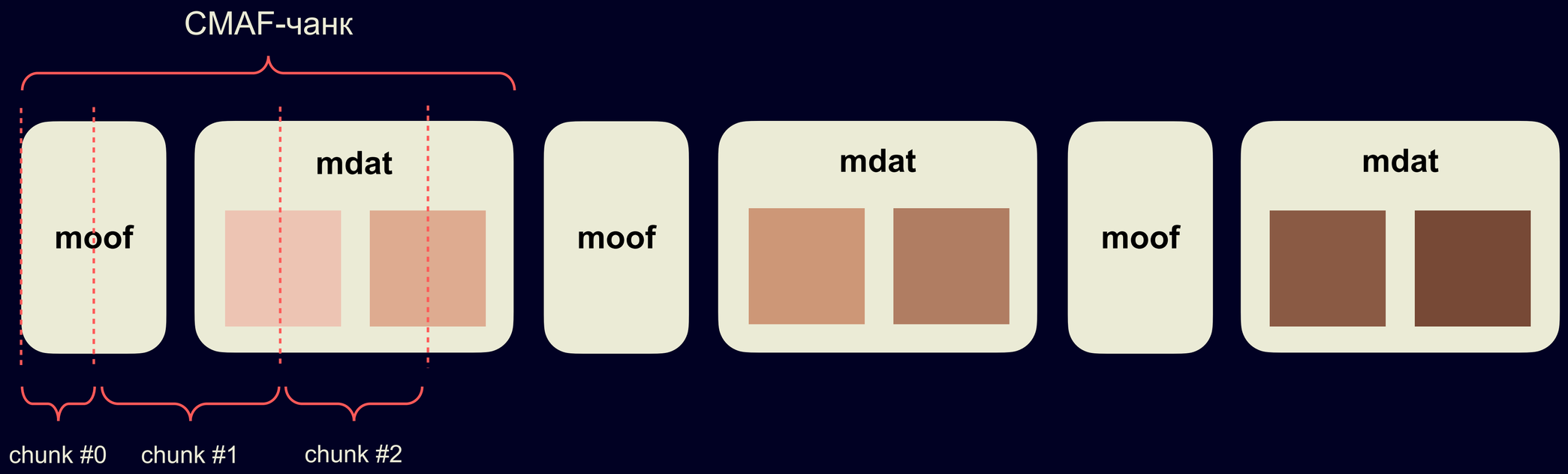
Обрабатываем
чанк сегмента

⚠️ MEDIA_ERR_DECODE


CMAF-чанк
moof
mdat
moof
mdat
moof
mdat

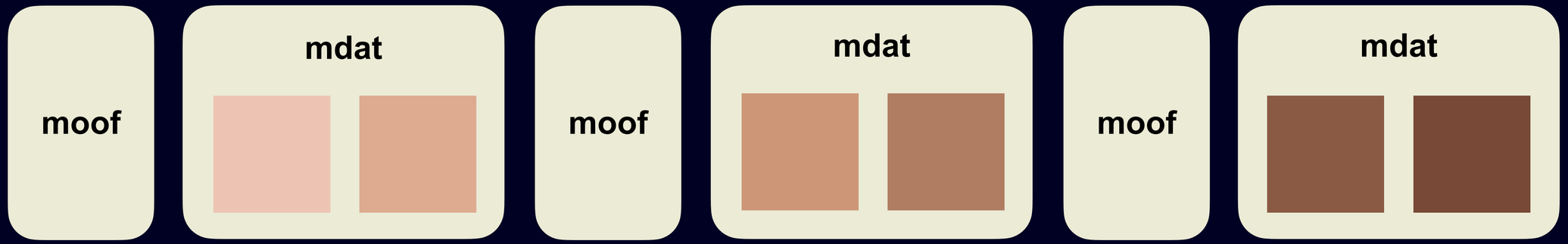

CMAF-чанк
moof
mdat
moof
mdat
moof
mdat
chunk #0


CMAF-чанк
moof
mdat
moof
mdat
moof
mdat
chunk #0
chunk #1

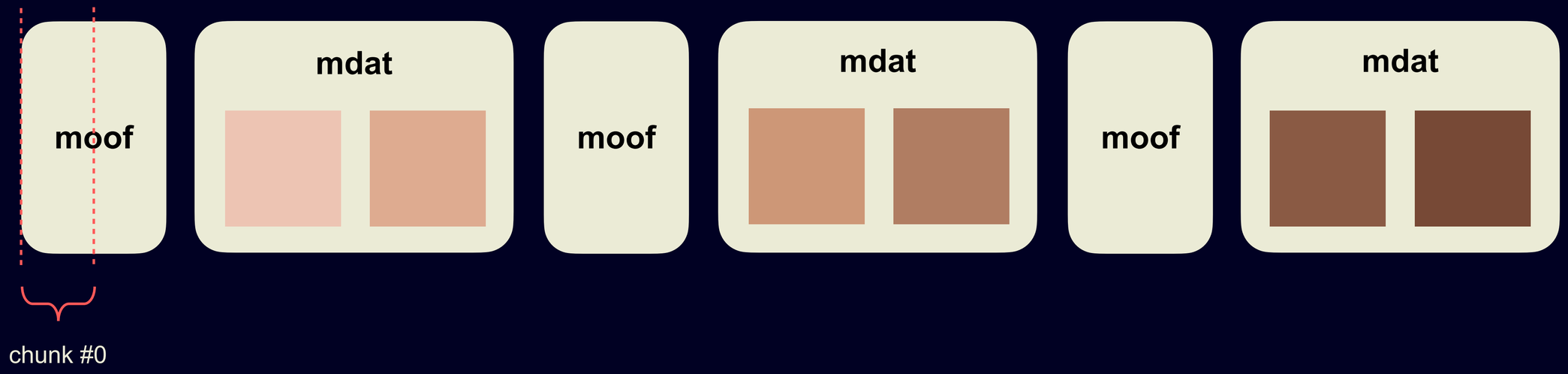

CMAF-чанк
moof
mdat
moof
mdat
moof
mdat
chunk #0
chunk #1
chunk #2

Декодер может обрабатывать
только полные
mp4-атомы!

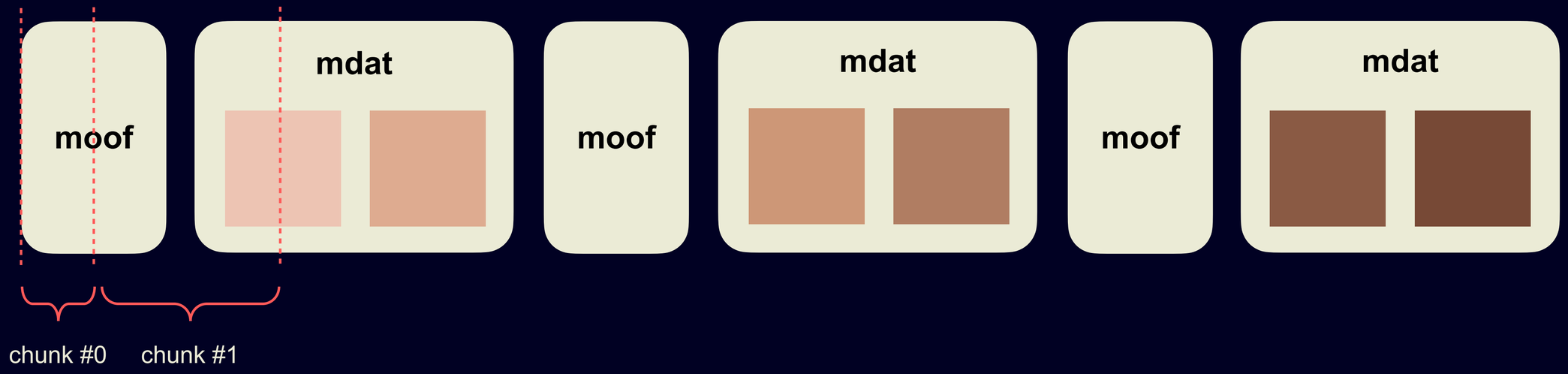
Парсим mp4 на клиенте



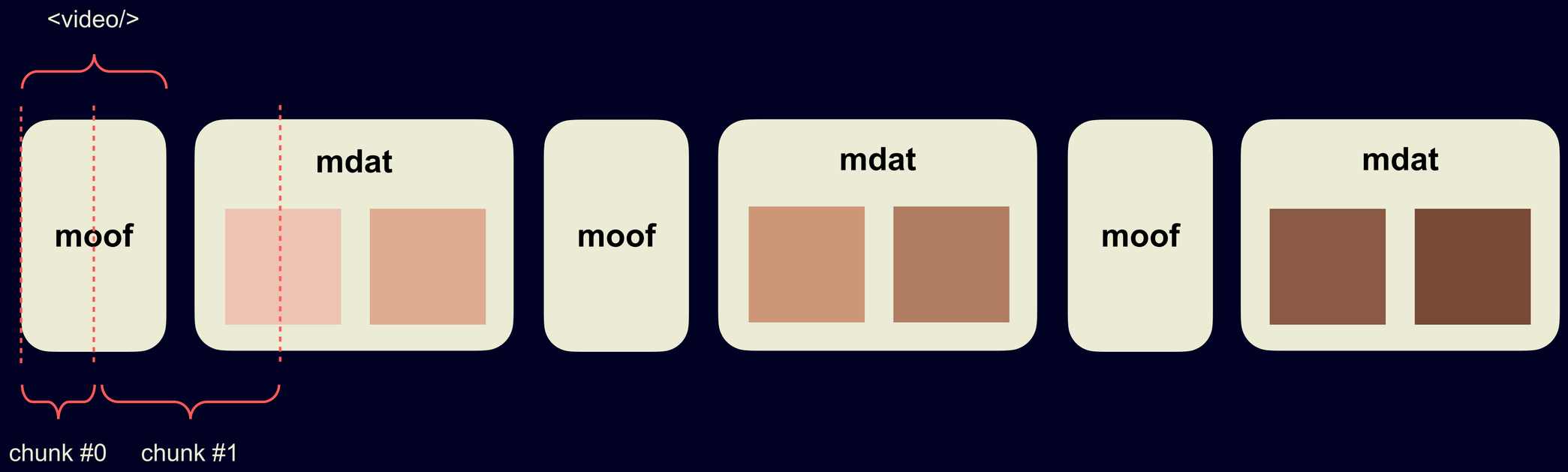
Парсим mp4 на клиенте



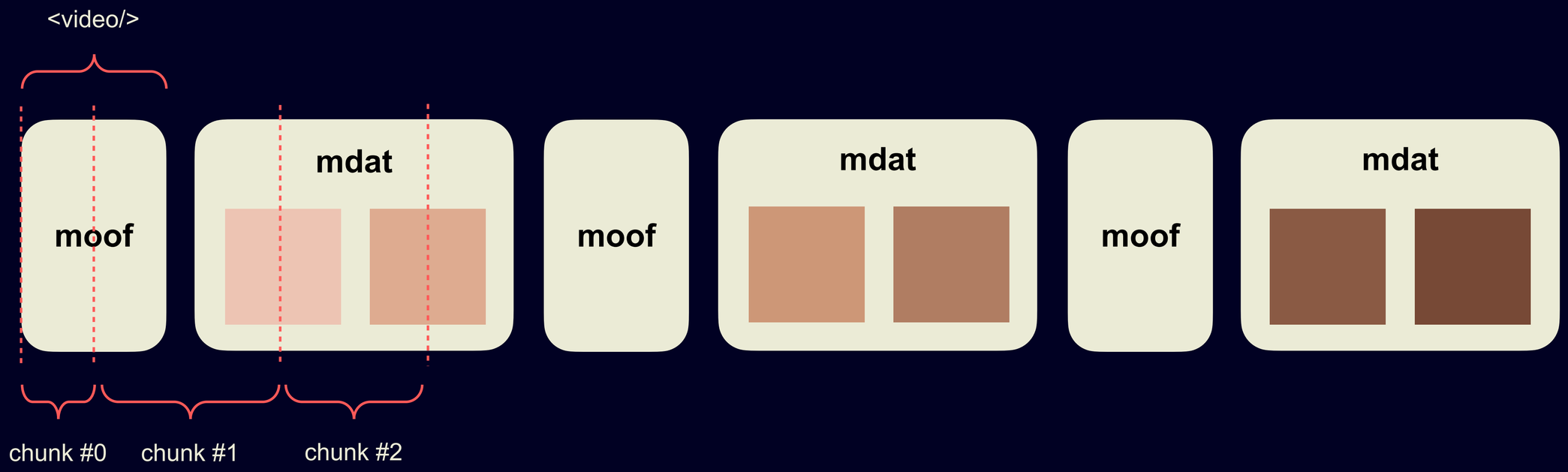
Парсим mp4 на клиенте



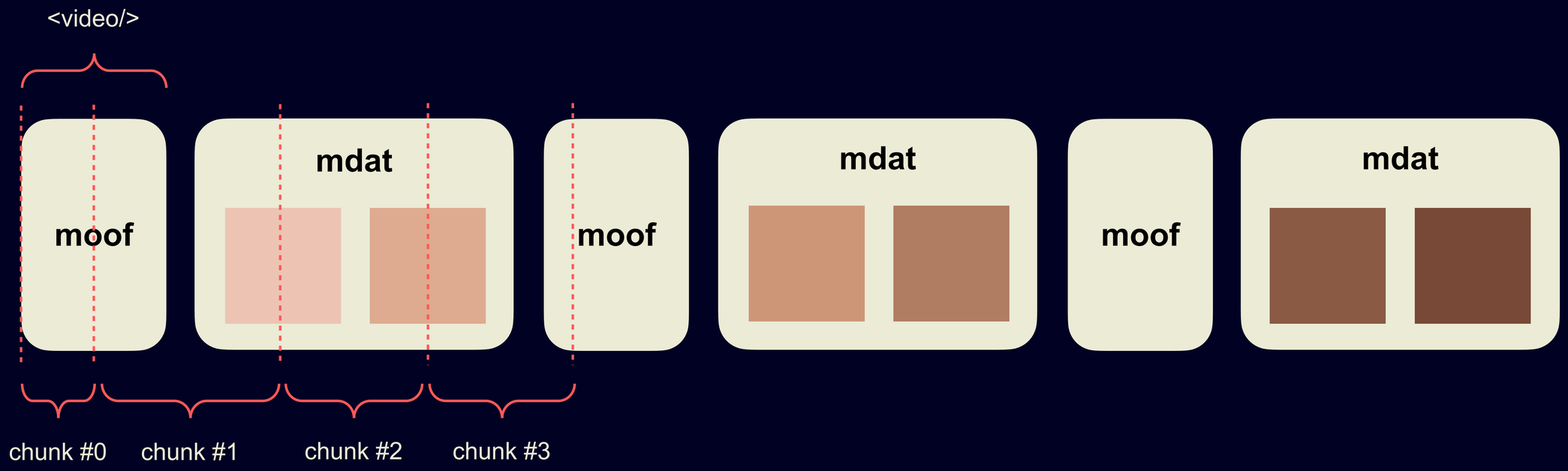
Парсим mp4 на клиенте



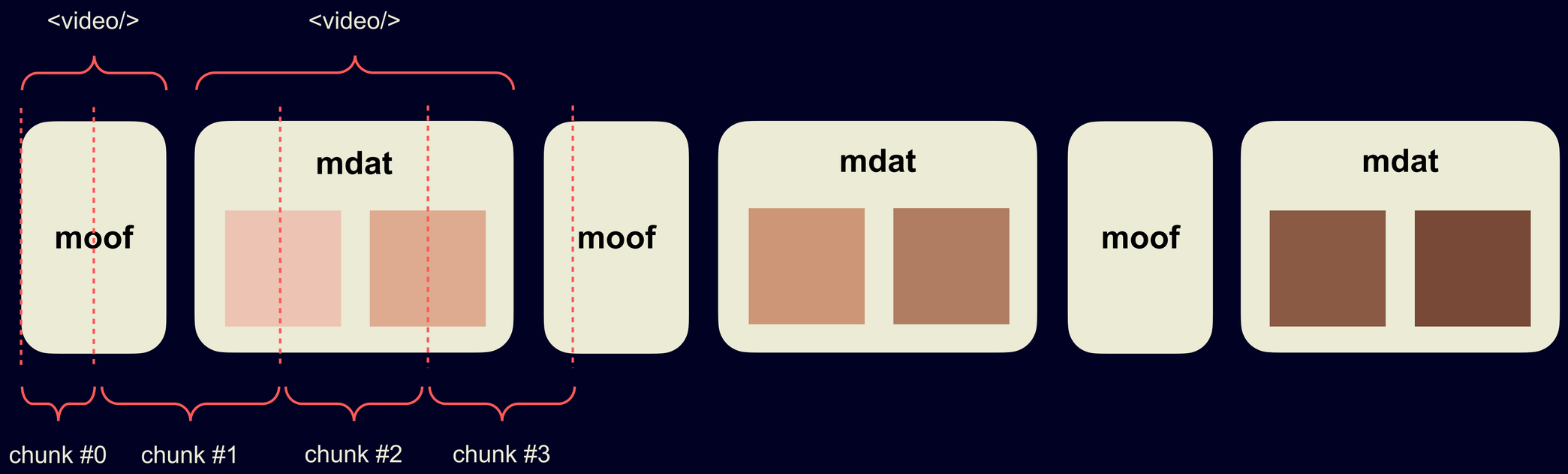
Парсим mp4 на клиенте



Парсим mp4 на клиенте



Парсим mp4 на клиенте



MP4Box.js

github.com/gpac/mp4box.js/

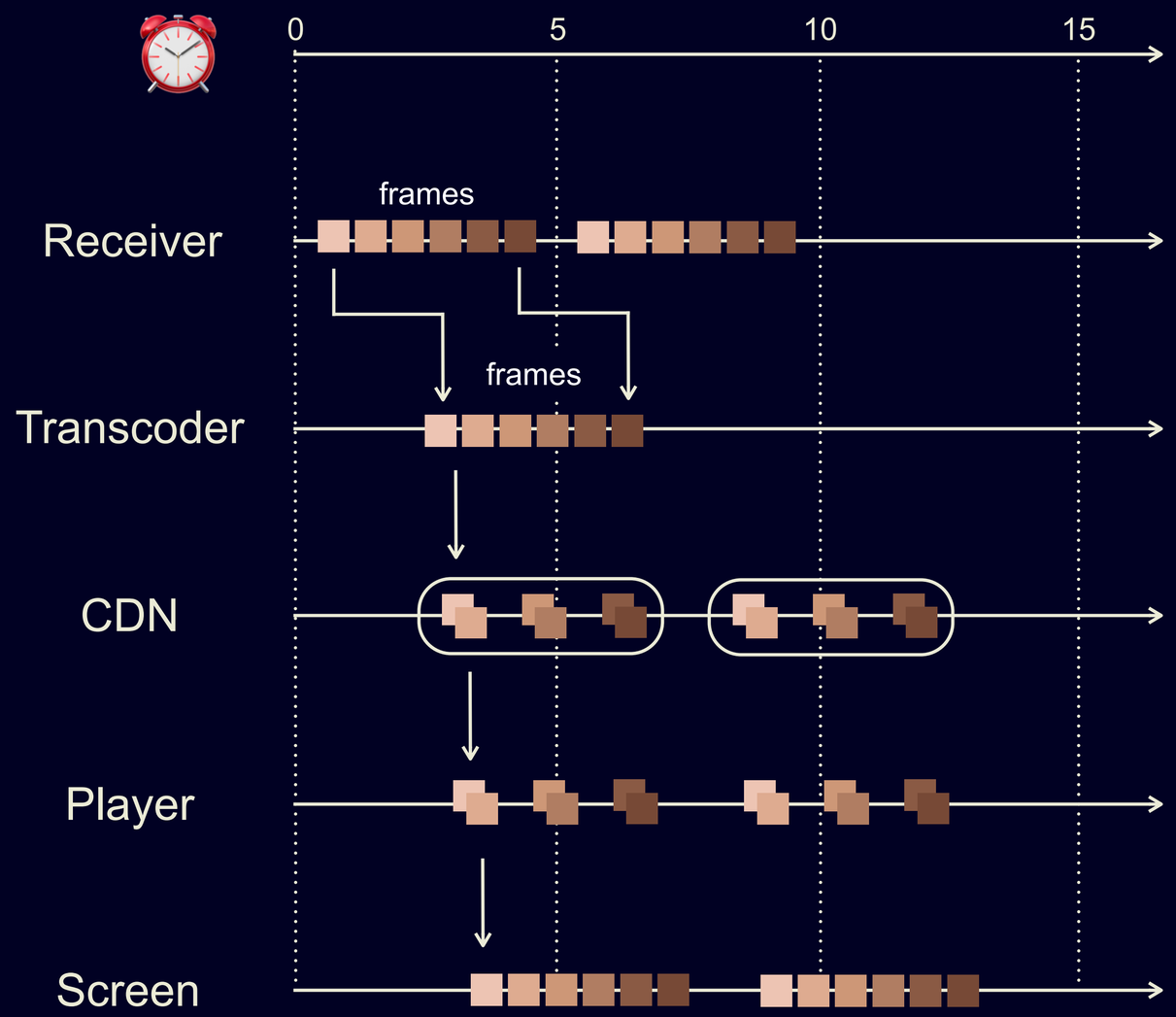
или самописный парсер

Теперь можно делать
low latency!

Target latency — целевое время отставания

Стартовая позиция на таймлайне вычисляется как отступ на target latency от правой границы таймлайна

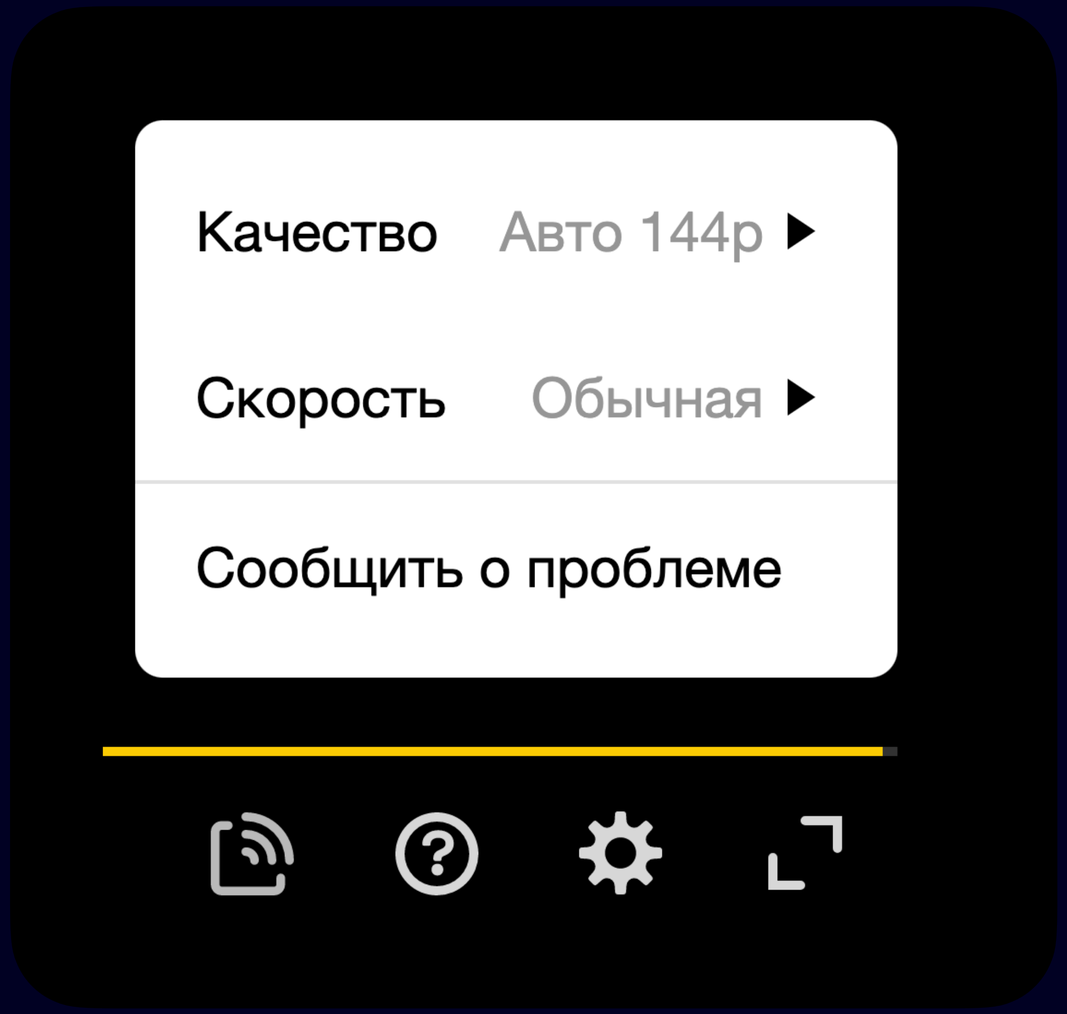
Target latency ≲
бюджет задержки (10 сек)



Буфер больше не накапливаем

Растет вероятность буферизации!

Бюджет задержки сошелся!
Катим?



10 из 10

Кто виноват,
и что делать?

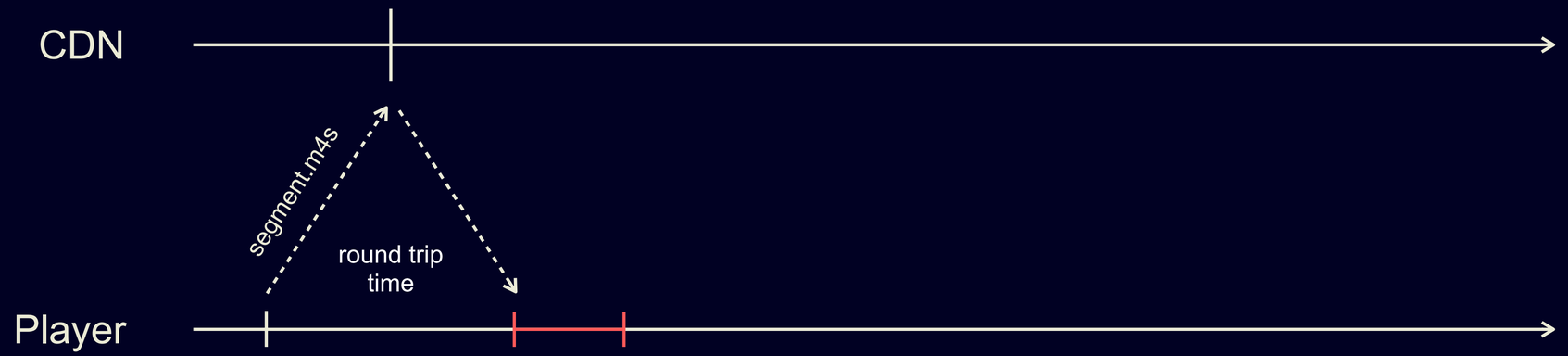
ABR – Adaptive BitRate

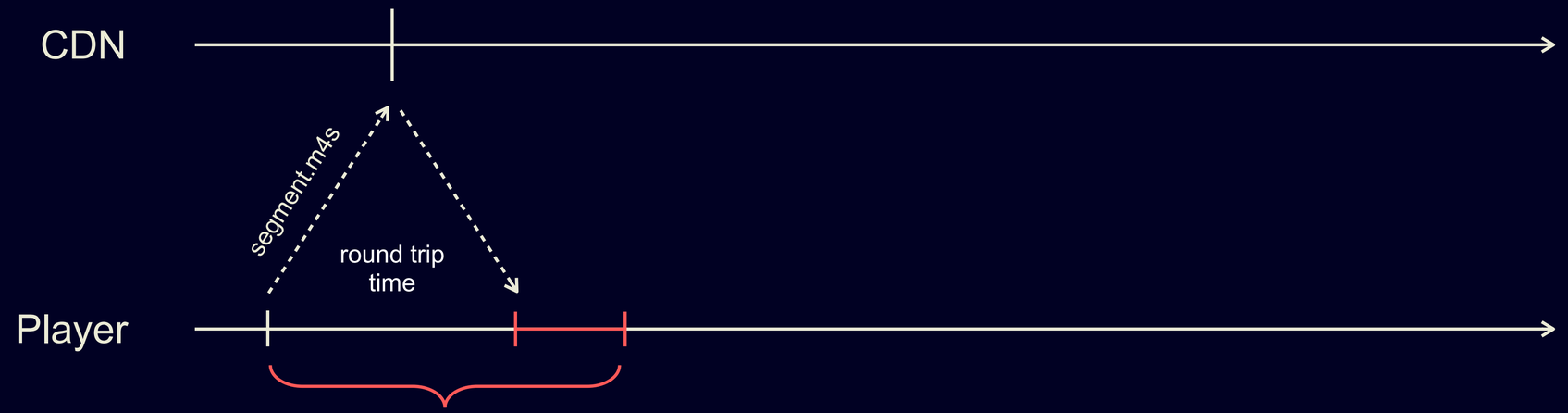
Компонент и набор алгоритмов, отвечающий
за выбор видео/аудио дорожки в адаптивном стриминге

Как устроен ABR-менеджер?

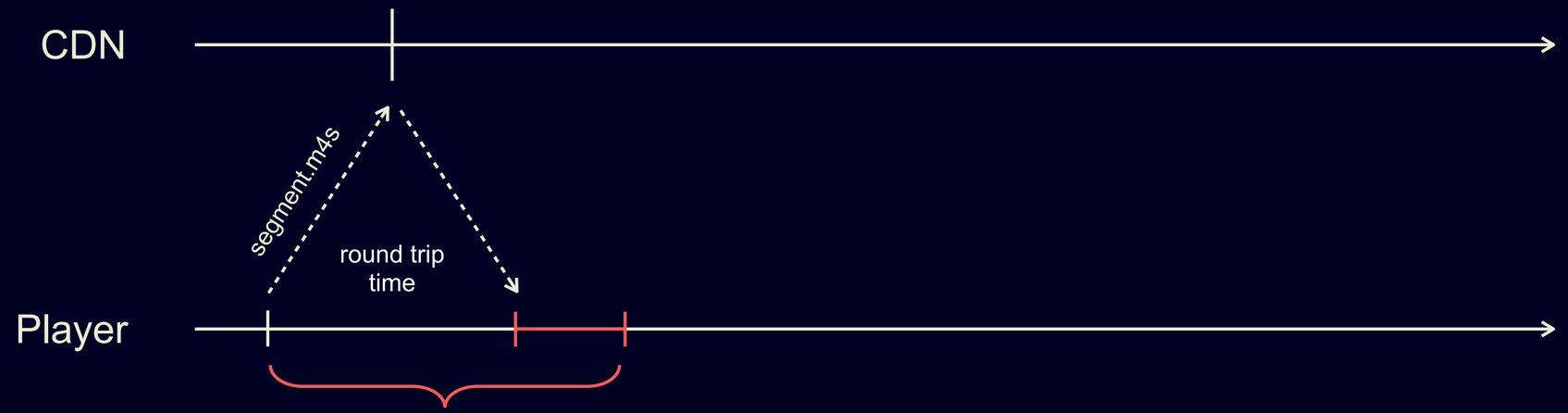

CDN
segment.m4s
Player


CDN
segment.m4s
round trip
time
Player


CDN
segment.m4s
round trip
time
Player



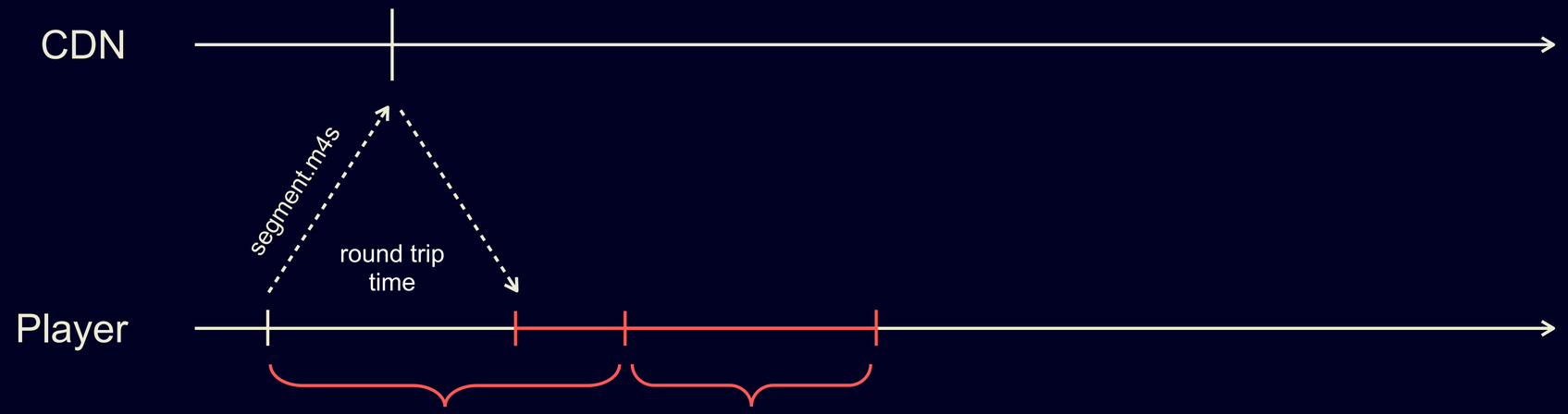
Загружено байт	b_0
Время загрузки	t_0



Загружено байт b_0

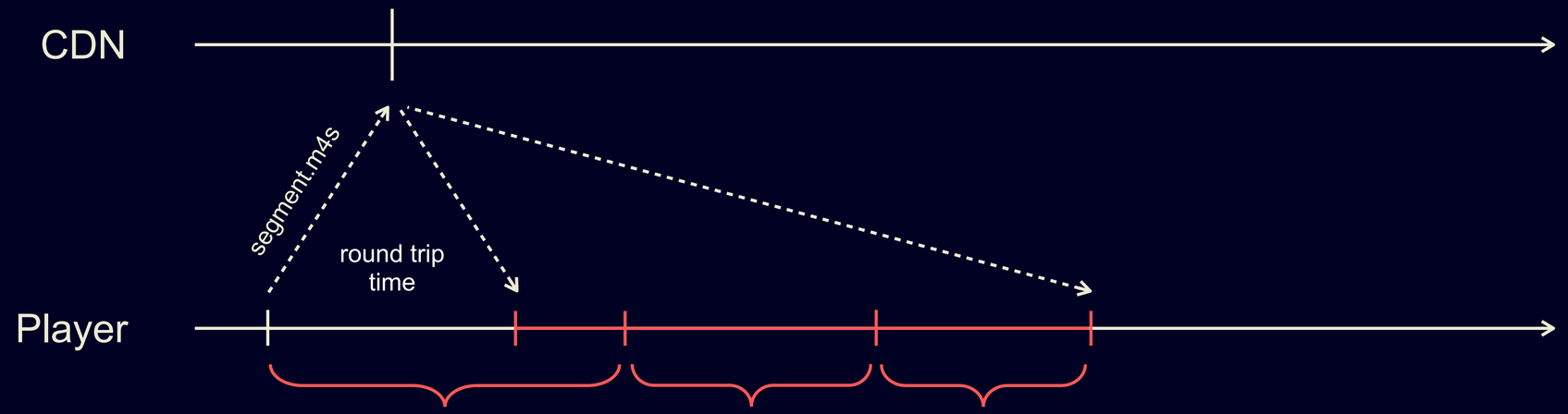
Время загрузки t_0

$rtt \ll t_0$



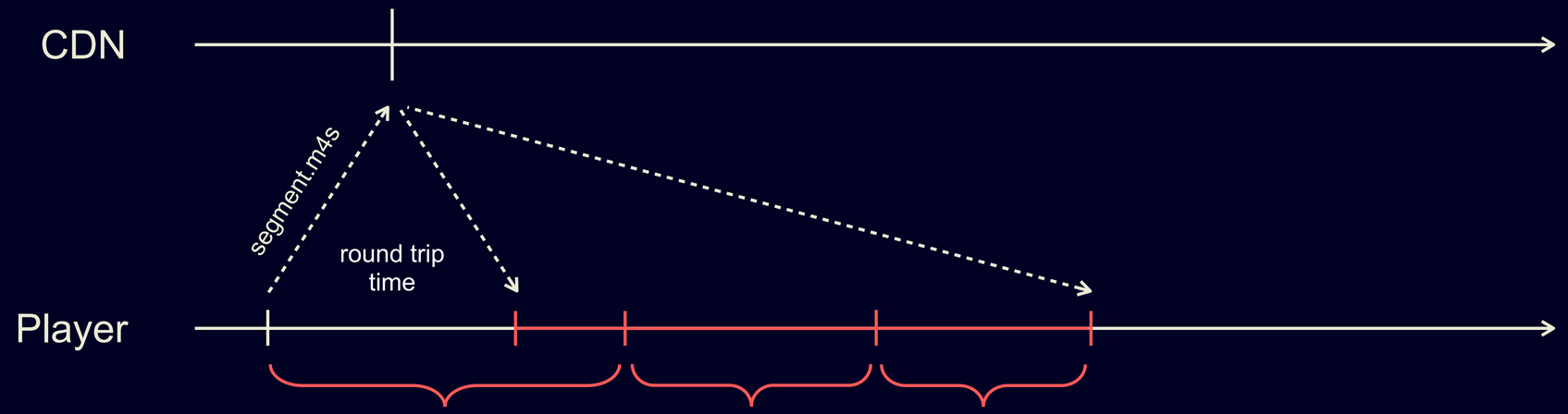
Загружено байт	b_0	b_1
Время загрузки	t_0	t_1

$rtt \ll t_0$



Загружено байт	b_0	b_1	b_2
Время загрузки	t_0	t_1	t_2

$rtt \ll t_0$



Загружено байт

Время загрузки

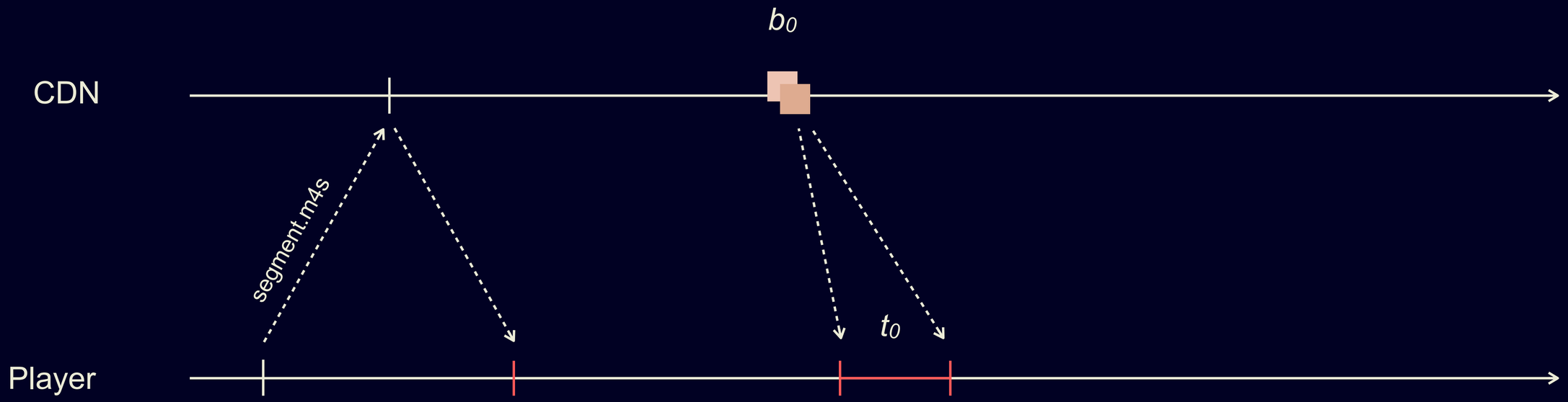
$$\frac{b_0}{t_0} = bw_0 \qquad \frac{b_1}{t_1} = bw_1 \qquad \frac{b_2}{t_2} = bw_2$$

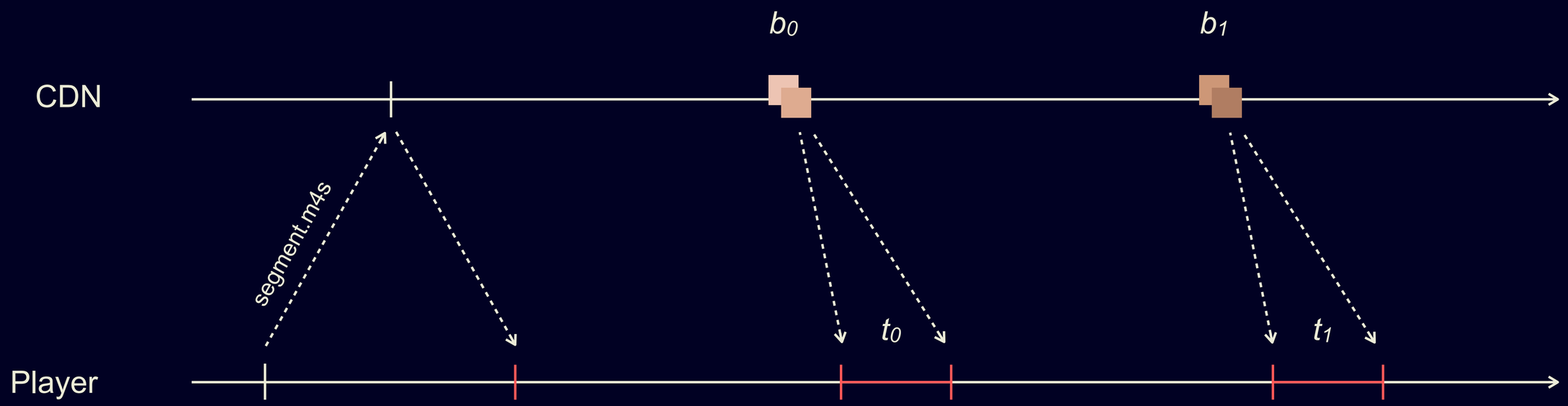
$$rtt \ll t_0$$

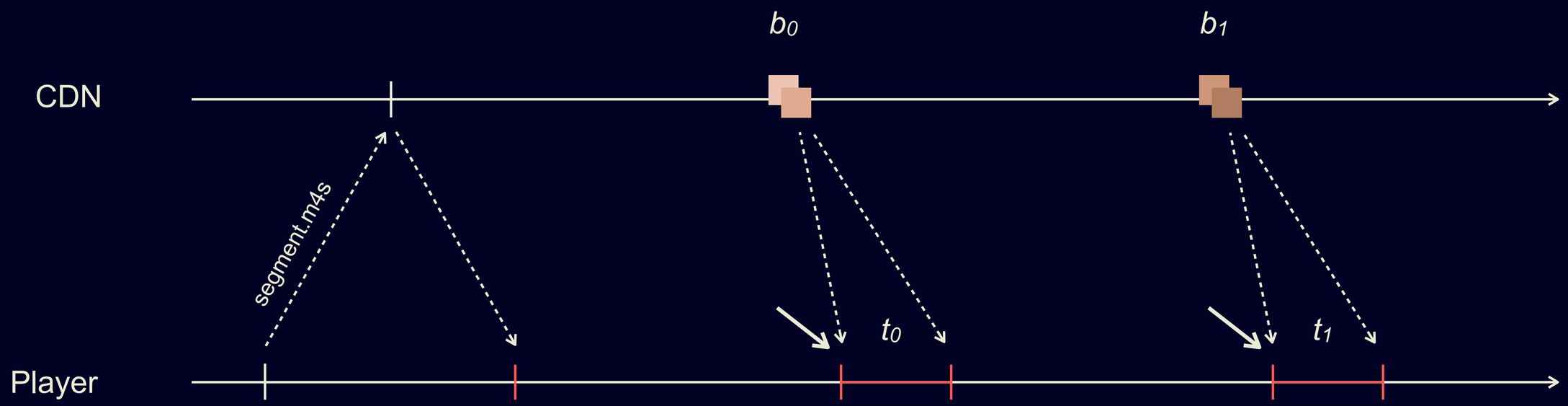
Что происходит со CMAF?

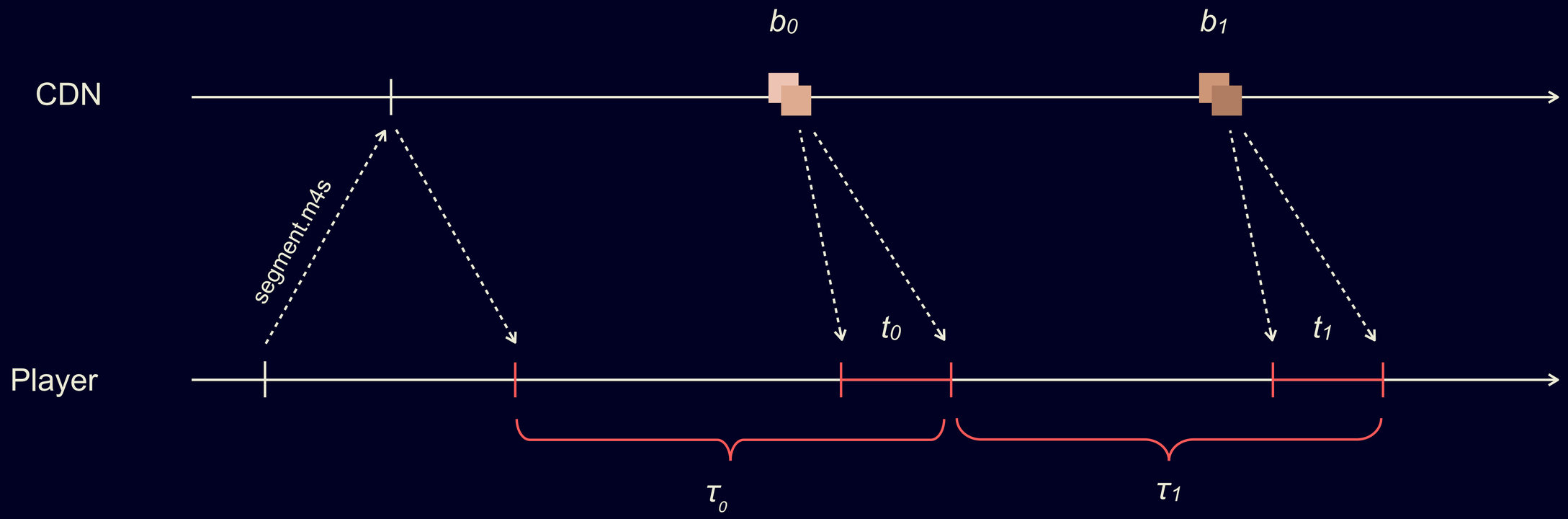

CDN
segment.m4s
Player

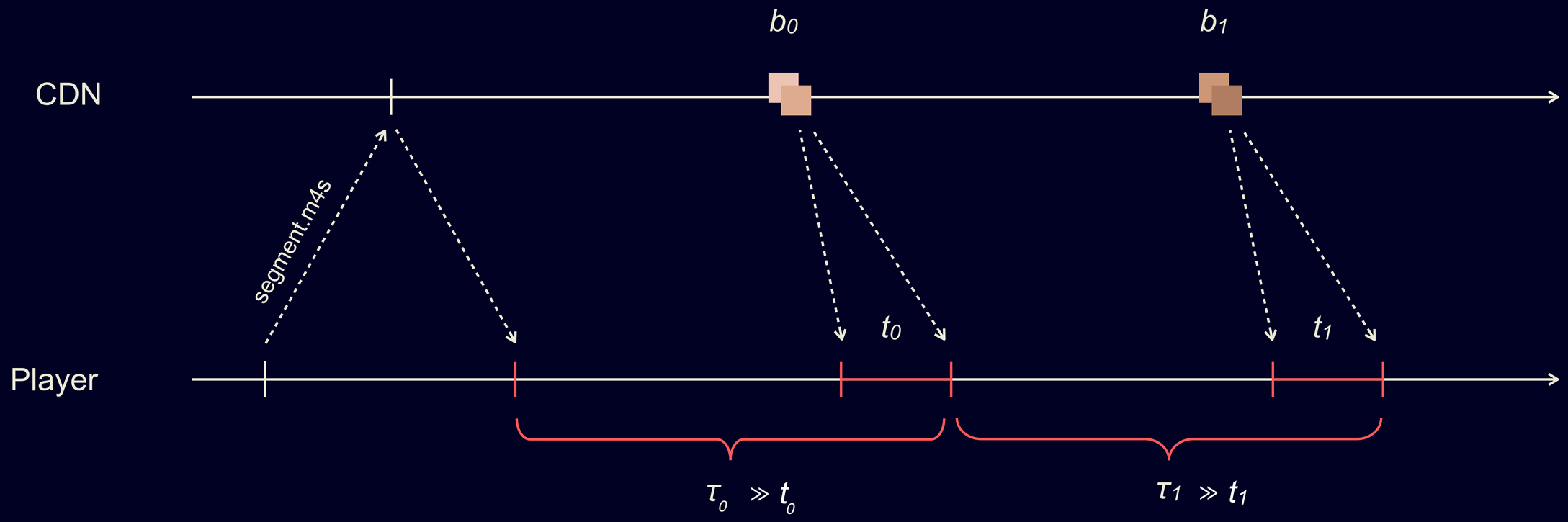

CDN
segment.m4s
Player


b_0
CDN
segment.m4s
t_0
Player


b_0
b_1
CDN
segment.m4s
t_0
t_1
Player


b_0
b_1
CDN
segment.m4s
t_0
t_1
Player


CDN
Player
segment.m4s
b_0
b_1
t_0
t_1
τ_0
τ_1


CDN
Player
segment.m4s
b_0
b_1
t_0
t_1
$\tau_0 \gg t_0$
$\tau_1 \gg t_1$

Реальная скорость сети: $\frac{b}{t}$

Вычисляемая скорость сети: $\frac{b}{\tau}$

Реальная скорость сети: $\frac{b}{t}$

Вычисляемая скорость сети: $\frac{b}{\tau}$

$$\frac{b}{\tau} \ll \frac{b}{t}$$

Реальная скорость сети: $\frac{b}{t}$

Вычисляемая скорость сети: $\frac{b}{\tau}$

$\frac{b}{\tau} \ll \frac{b}{t} \longrightarrow$

144p

ABR-менеджер нужно адаптировать для low latency

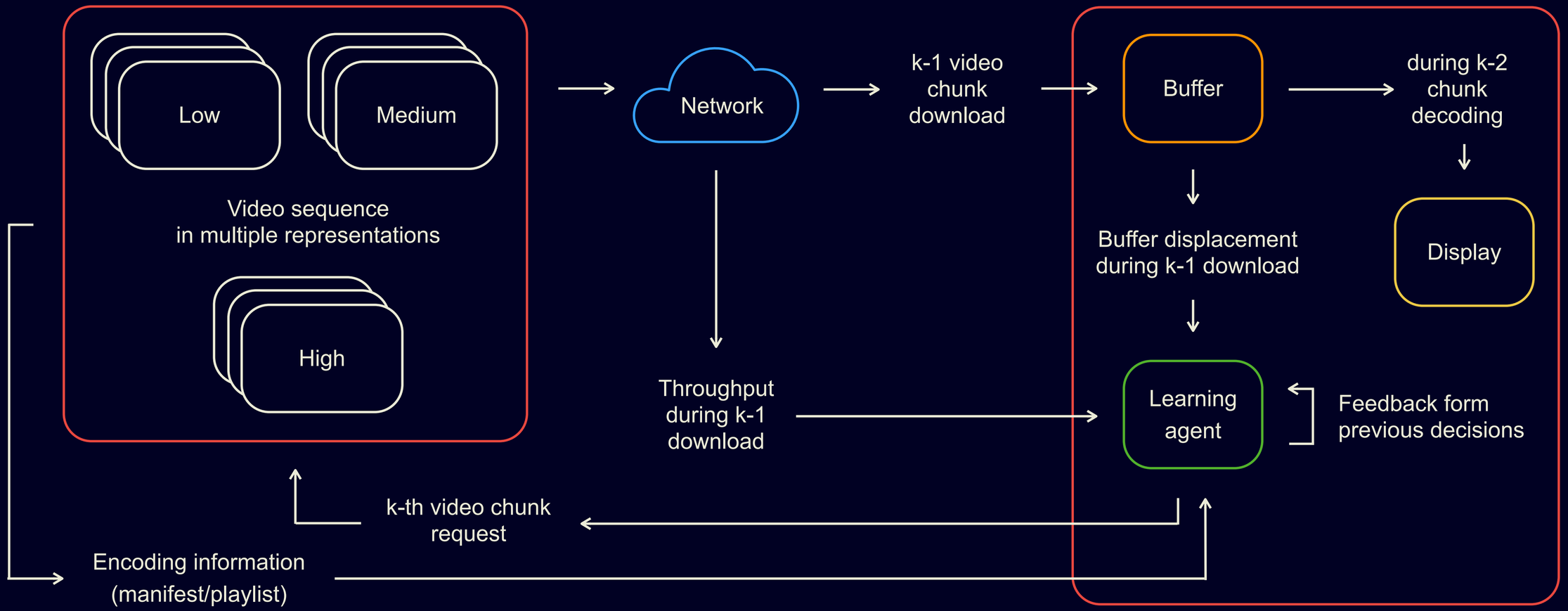
А как решают проблему другие сервисы?

ACM MMSys 2020 Grand Challenge on

Adaptation Algorithms for Near-Second Latency

Organized and sponsored by

Learn2Adapt-LowLatency (L2A-LL)



unified-streaming.com/blog/learn2adapt-low-latency-adaptive-streaming

dash.js

Имплементация L2A-LL

github.com/Dash-Industry-Forum/dash.js/pull/3424

Dash JavaScript Player

Not Secure | reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html

Update

Stream | https://cmafref.akamaized.net/cmaf/live-ull/2006350/akambr/out.mpd | Show Options | Stop | Load

Additional samples can be found in the Sample Section.

Video | Audio

- [x] Buffer Length : 2.993
- [x] Bitrate Downloading : 6000 kbps
- [ ] Index Downloading : 7 / 7
- [ ] Index playing : 7 / 7
- [ ] Dropped Frames : 5
- [ ] Latency (min|avg|max) : 0.01 | 0.10 | 0.19
- [ ] Download (min|avg|max) : 0.00 | 1.28 | 1.82
- [ ] Ratio (min|avg|max) : 1.10 | 1.56 | 667.33
- [ ] Live Latency: 3.006

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse | Requestly

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

20000 ms | 40000 ms | 60000 ms | 80000 ms | 100000 ms | 120000 ms | 140000 ms | 160000 ms | 180000 ms | 200000 ms | 220000 ms | 240000 ms | 260000 ms

Dash JavaScript Player

Not Secure | reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html

Update

Stream | https://cmafref.akamaized.net/cmaf/live-ull/2006350/akambr/out.mpd | Show Options | Stop | Load

Additional samples can be found in the Sample Section.

Video | Audio

- [x] Buffer Length : 2.993
- [x] Bitrate Downloading : 6000 kbps
- [ ] Index Downloading : 7 / 7
- [ ] Index playing : 7 / 7
- [ ] Dropped Frames : 5
- [ ] Latency (min|avg|max) : 0.01 | 0.10 | 0.19
- [ ] Download (min|avg|max) : 0.00 | 1.28 | 1.82
- [ ] Ratio (min|avg|max) : 1.10 | 1.56 | 667.33
- [ ] Live Latency: 3.006

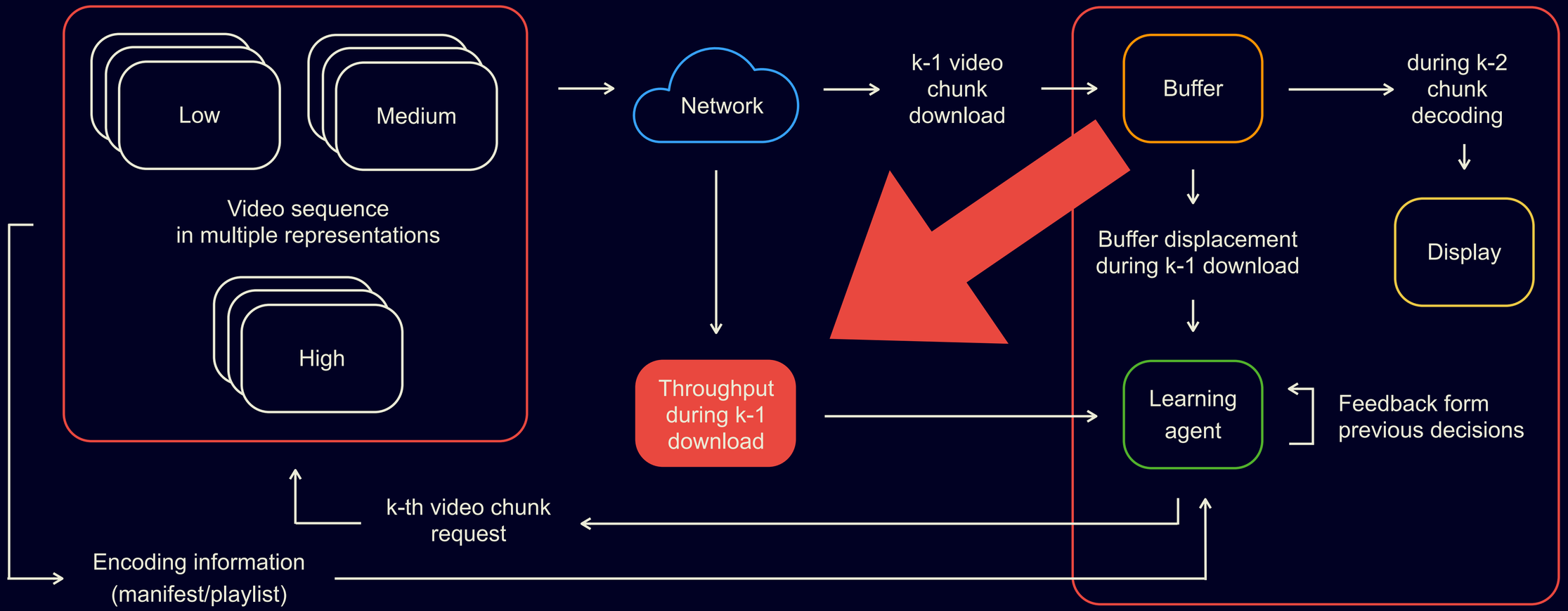
Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse | Requestly

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

20000 ms | 40000 ms | 60000 ms | 80000 ms | 100000 ms | 120000 ms | 140000 ms | 160000 ms | 180000 ms | 200000 ms | 220000 ms | 240000 ms | 260000 ms

Learn2Adapt-LowLatency (L2A-LL)

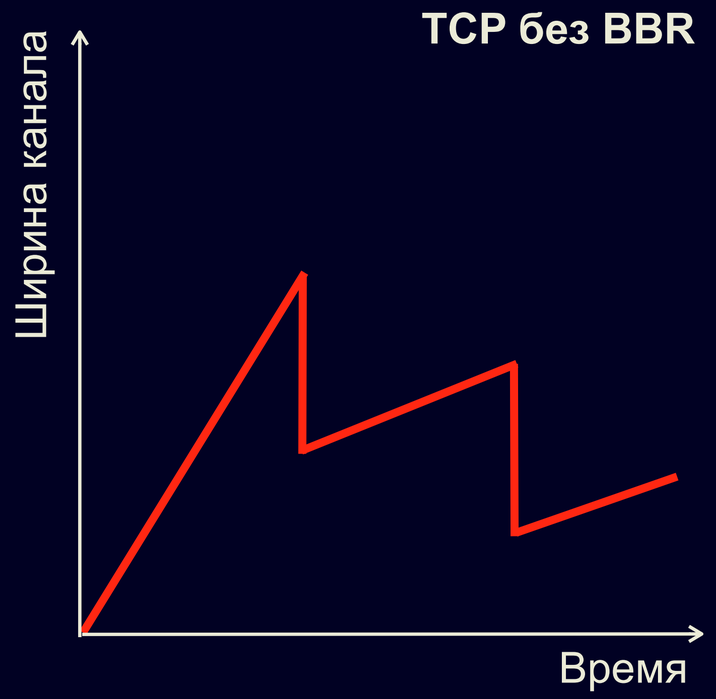


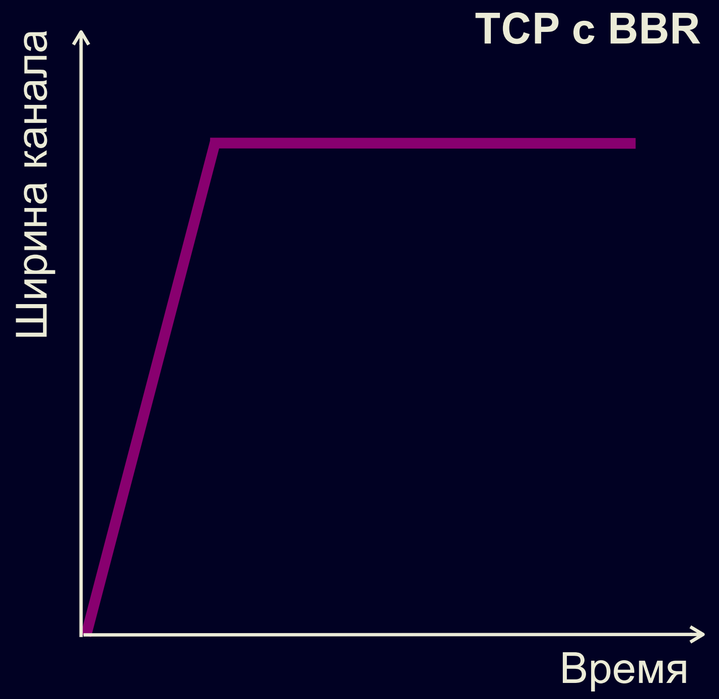
clck.ru/YLqcq

Проблема оценки скорости сети никуда не девается 🤔

Если мы не можем делать что-то
на клиенте, давайте делать
это на сервере?

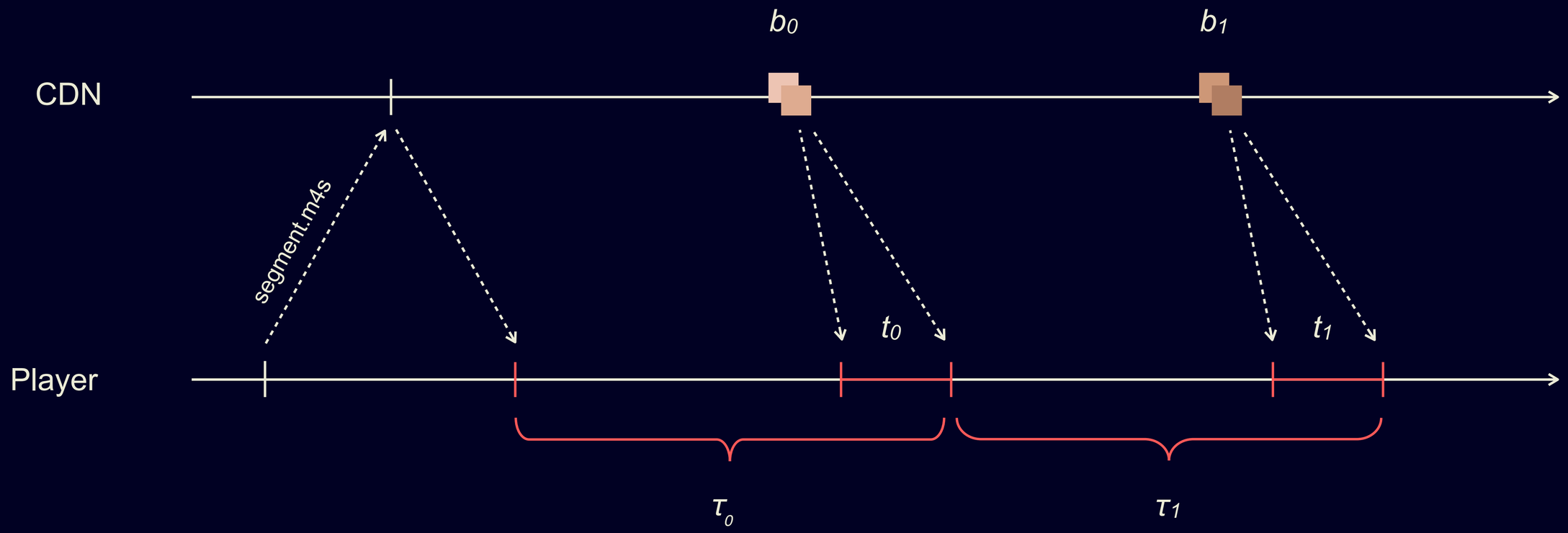
TCP BBR: быстрый и простой способ ускорения загрузки страниц

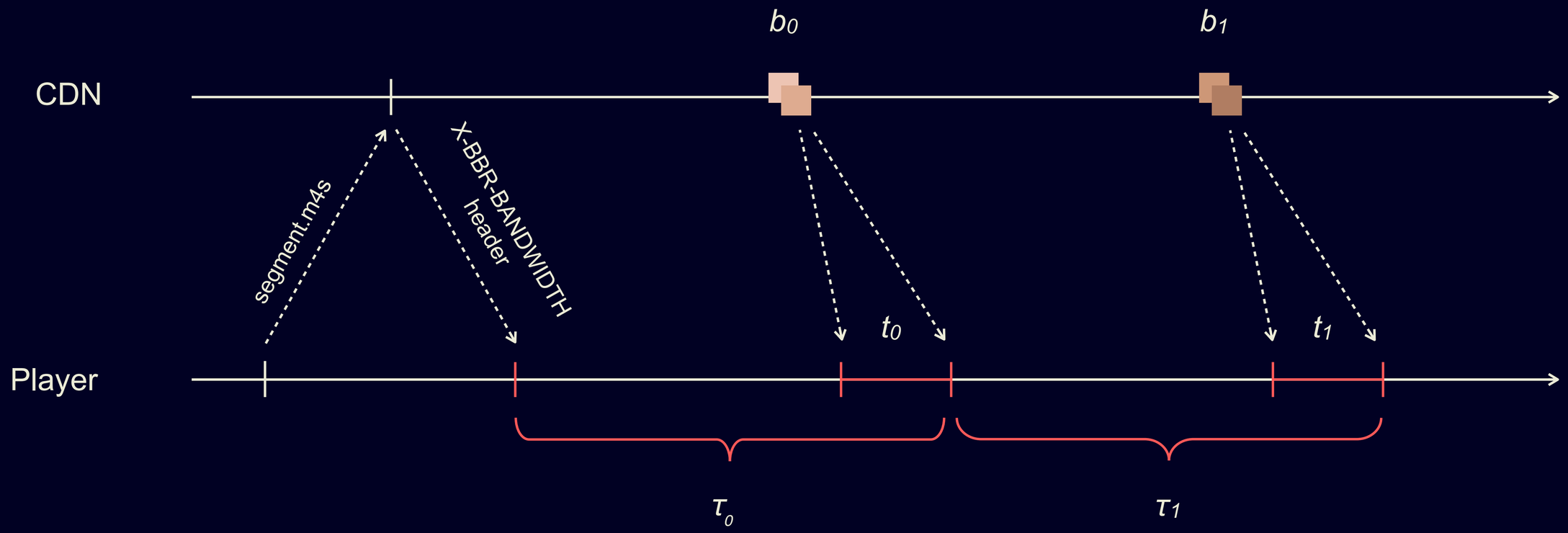


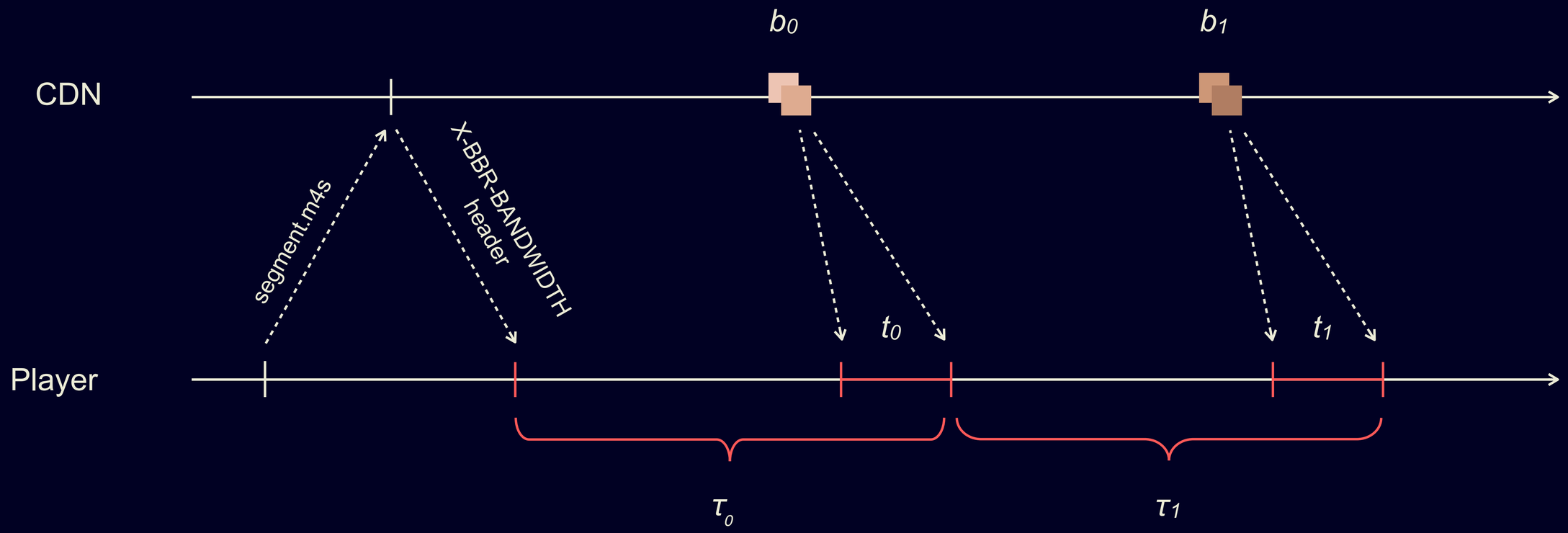


youtube.com/watch?v=hOr9GP_czFs
habr.com/ru/company/yandex/blog/533530/

Попробуем?


CDN
Player
segment.m4s
b_0
b_1
t_0
t_1
τ_0
τ_1


CDN
Player
segment.m4s
X-BBR-BANDWIDTH
header
b_0
b_1
t_0
t_1
τ_0
τ_1



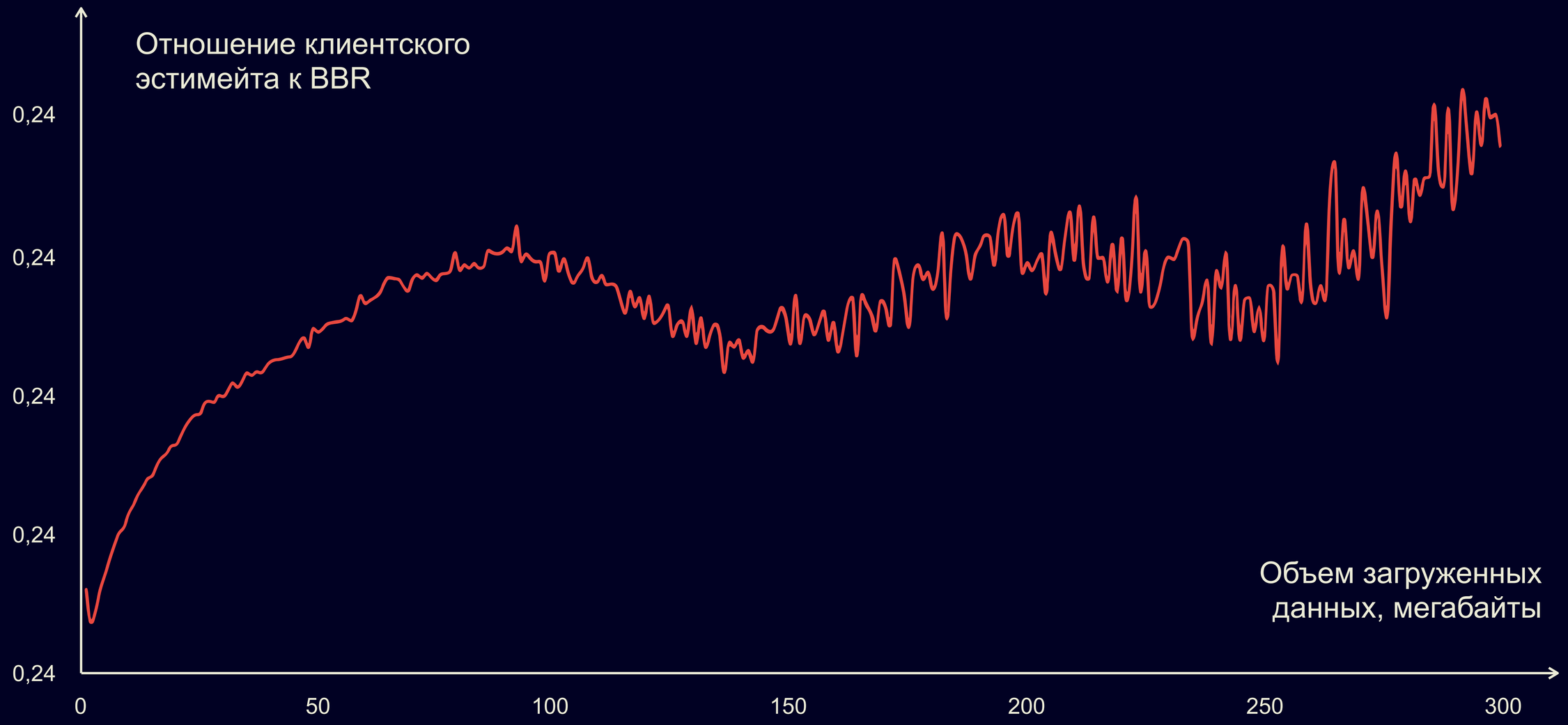
$$\frac{b}{\tau} \longrightarrow \text{X-BBR-BANDWIDTH}$$

Если на VOD ок —
то и на LIVE ок

A/B-эксперимент

Сравниваем метрики буферизаций

+30% времени буферизации


Отношение клиентского
эстимейта к BBR
0,24
0,24
0,24
0,24
0
50
100
150
200
250
300
Объем загруженных
данных, мегабайты

Выводы

Выводы

1. BBR завышает оценку скорости

Выводы

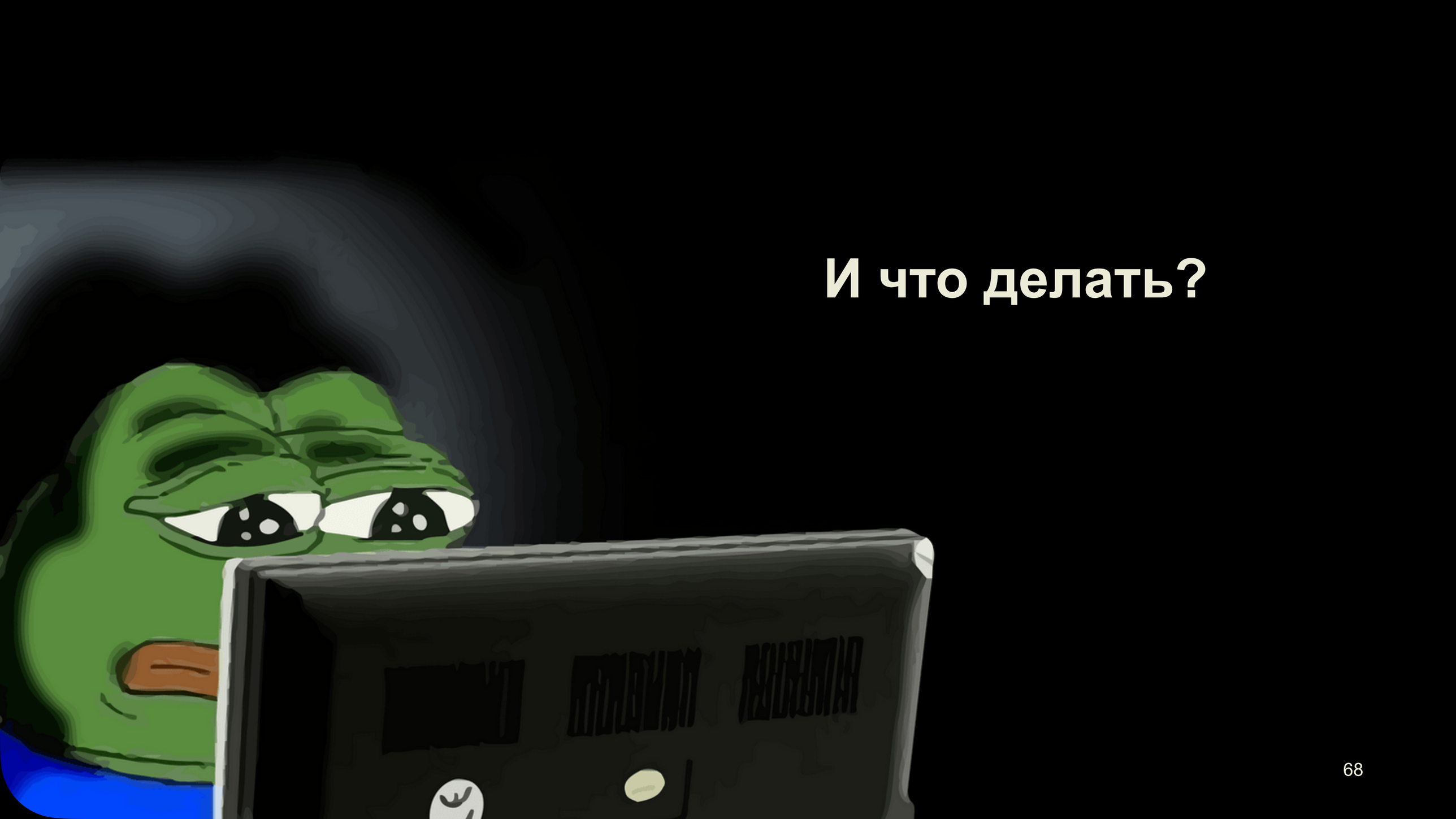
1. BBR завышает оценку скорости
2. BBR медленно сходится

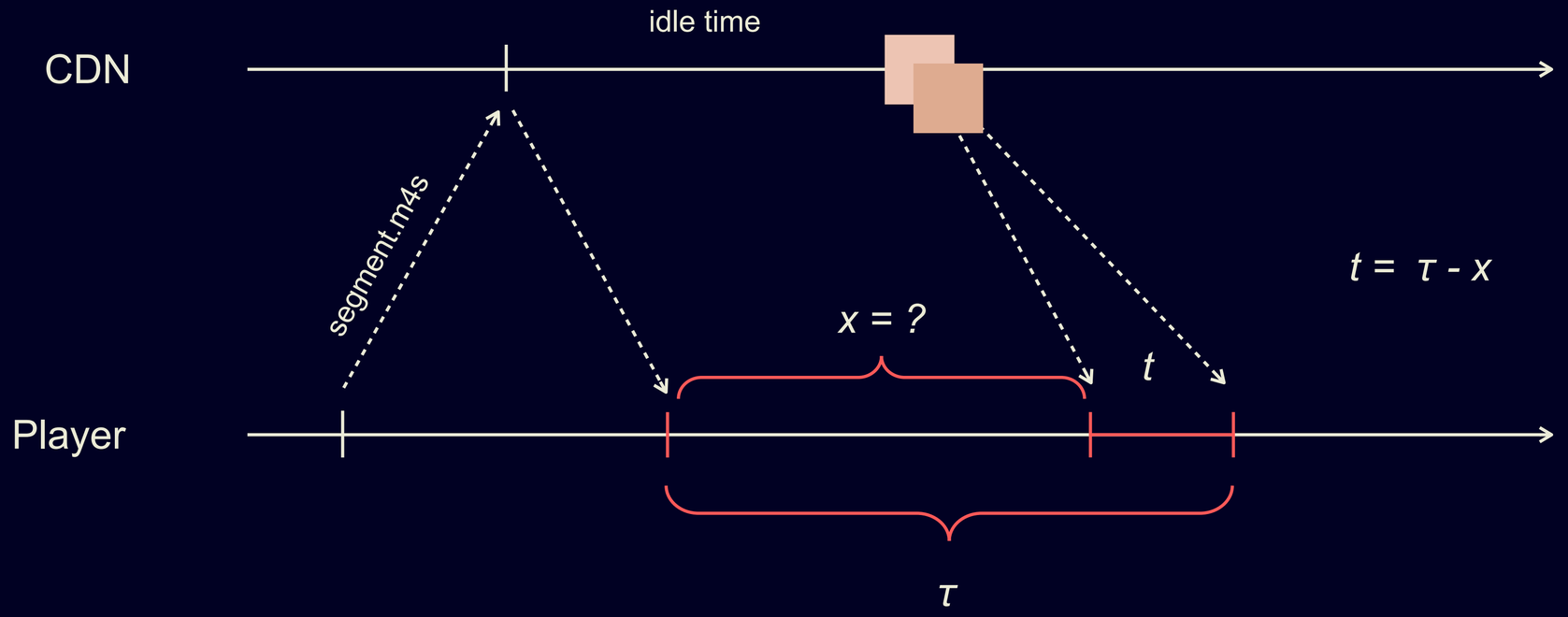
Выводы

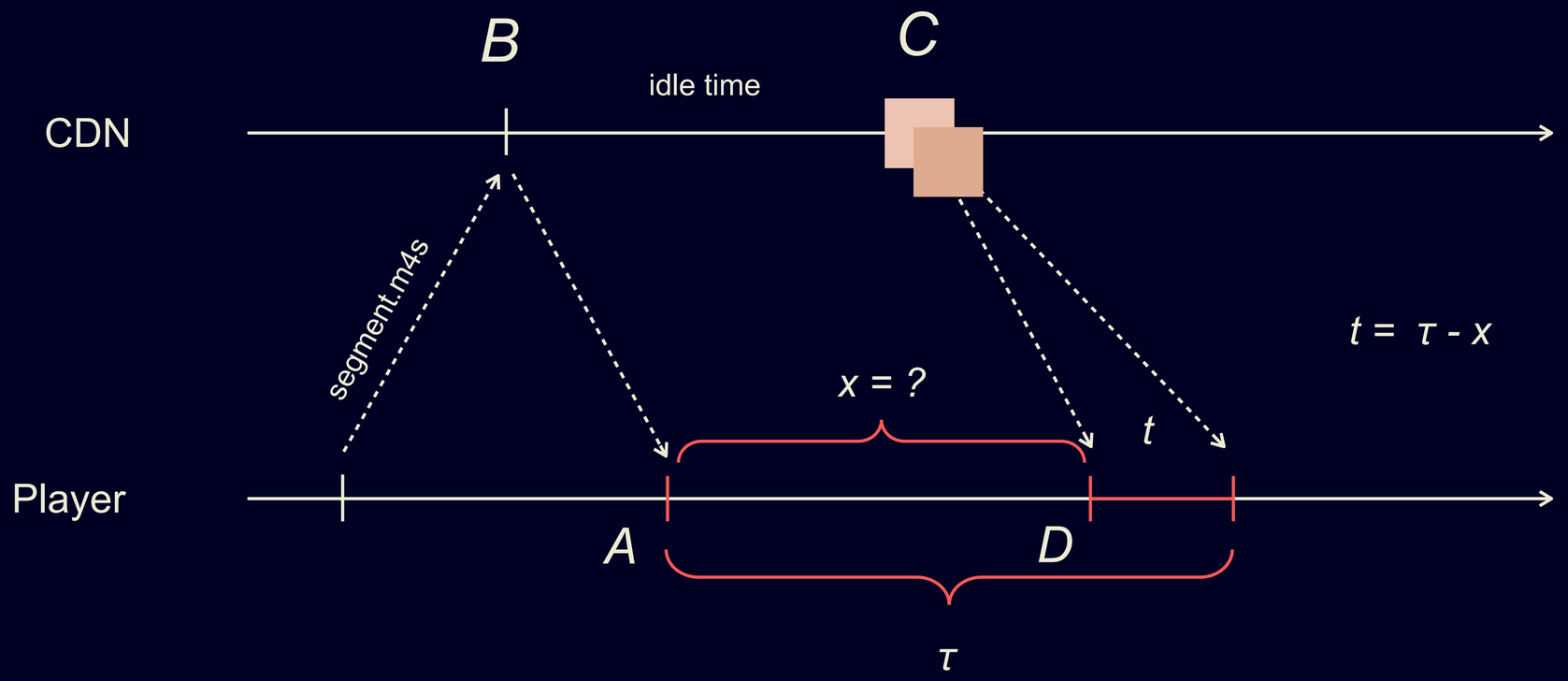
1. BBR завышает оценку скорости
2. BBR медленно сходится

Вывод: это решения нам не подходит

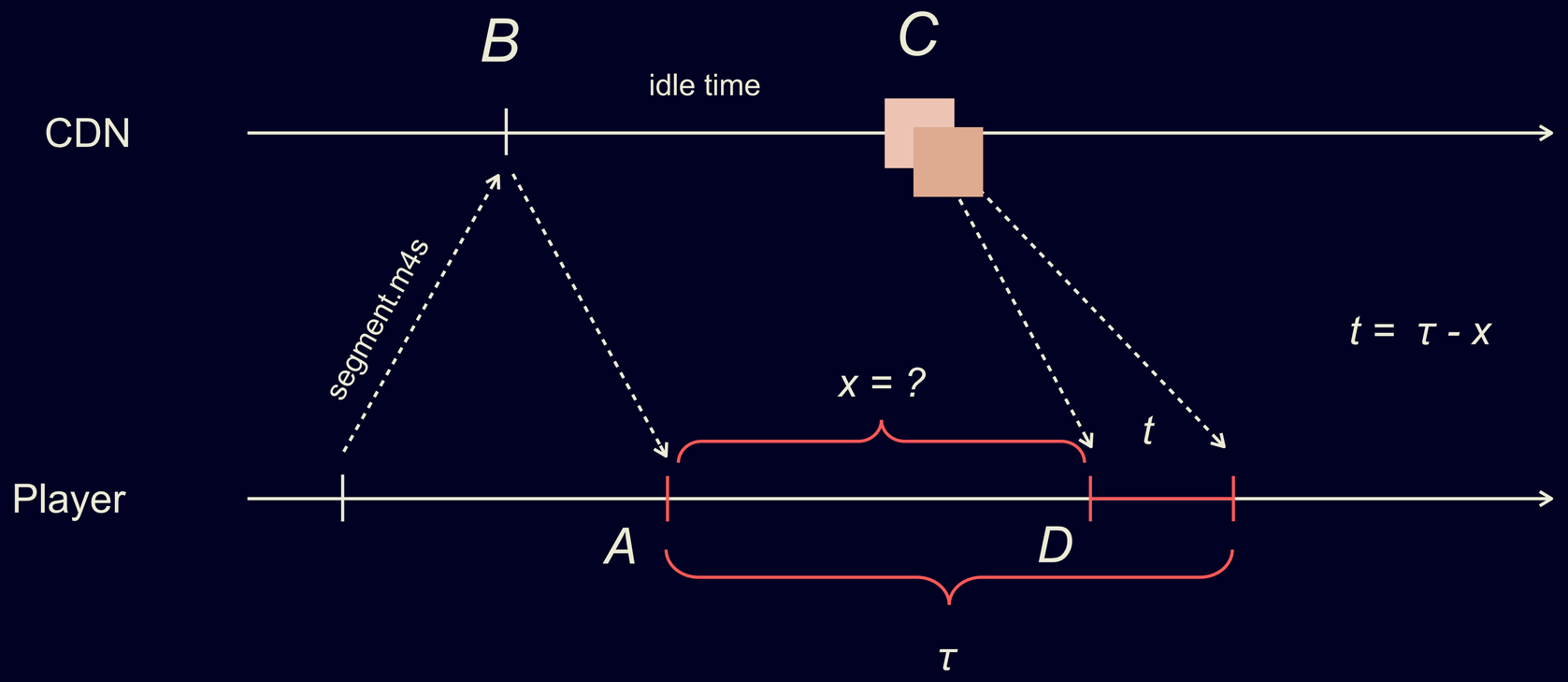
И что делать?




idle time
CDN
segment.m4s
t = τ - x
x = ?
t
Player
τ

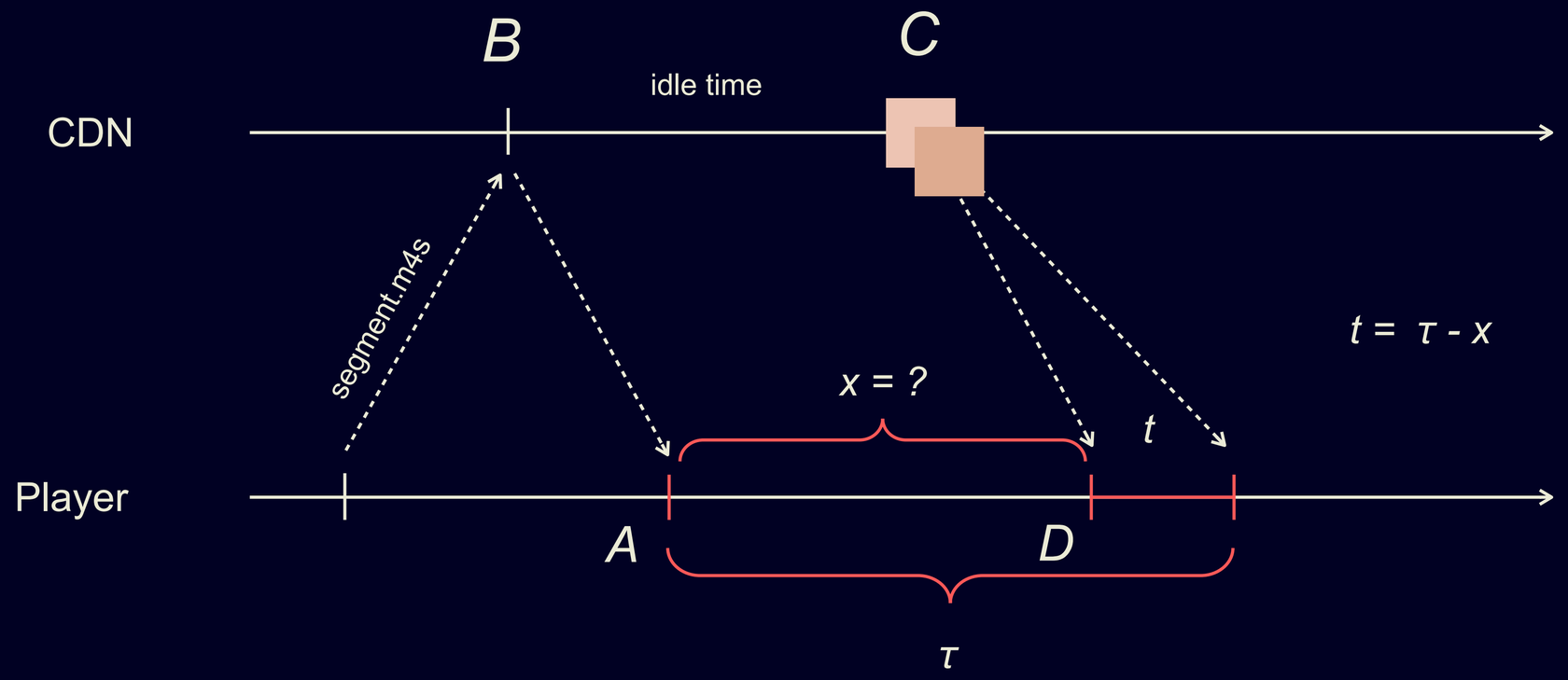

B
C
idle time
CDN
segment.m4s
t = τ - x
x = ?
t
Player
A
D
τ

AB || CD

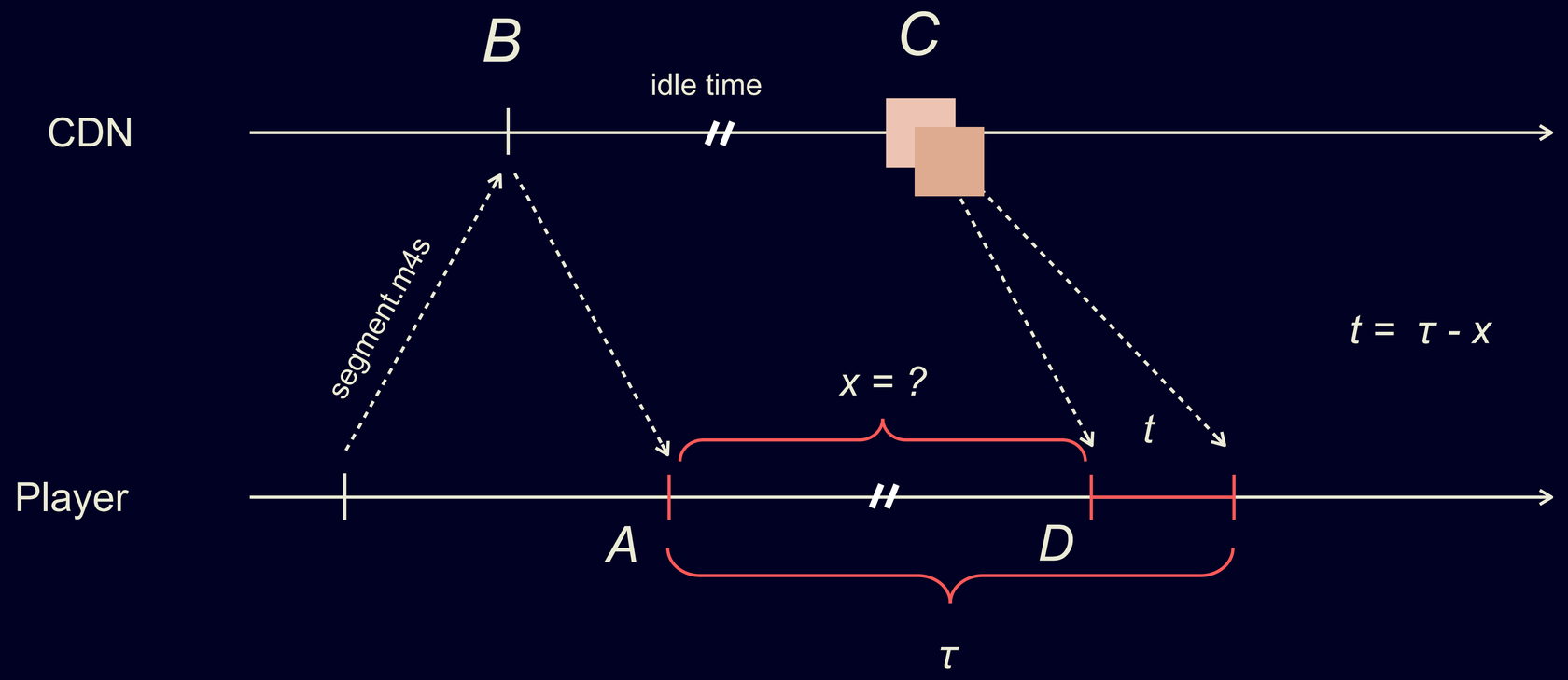


AB || CD

ABCD — параллелограмм



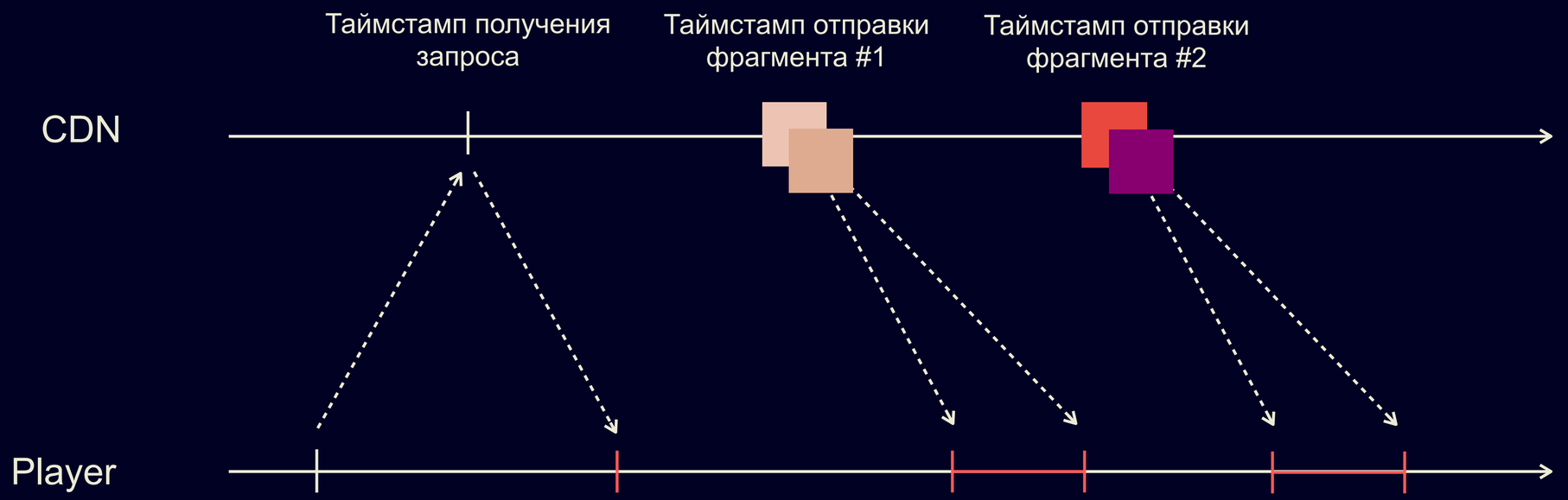
AB || CD ABCD — параллелограмм BC = AD



$t \approx \tau$ – *idle time*

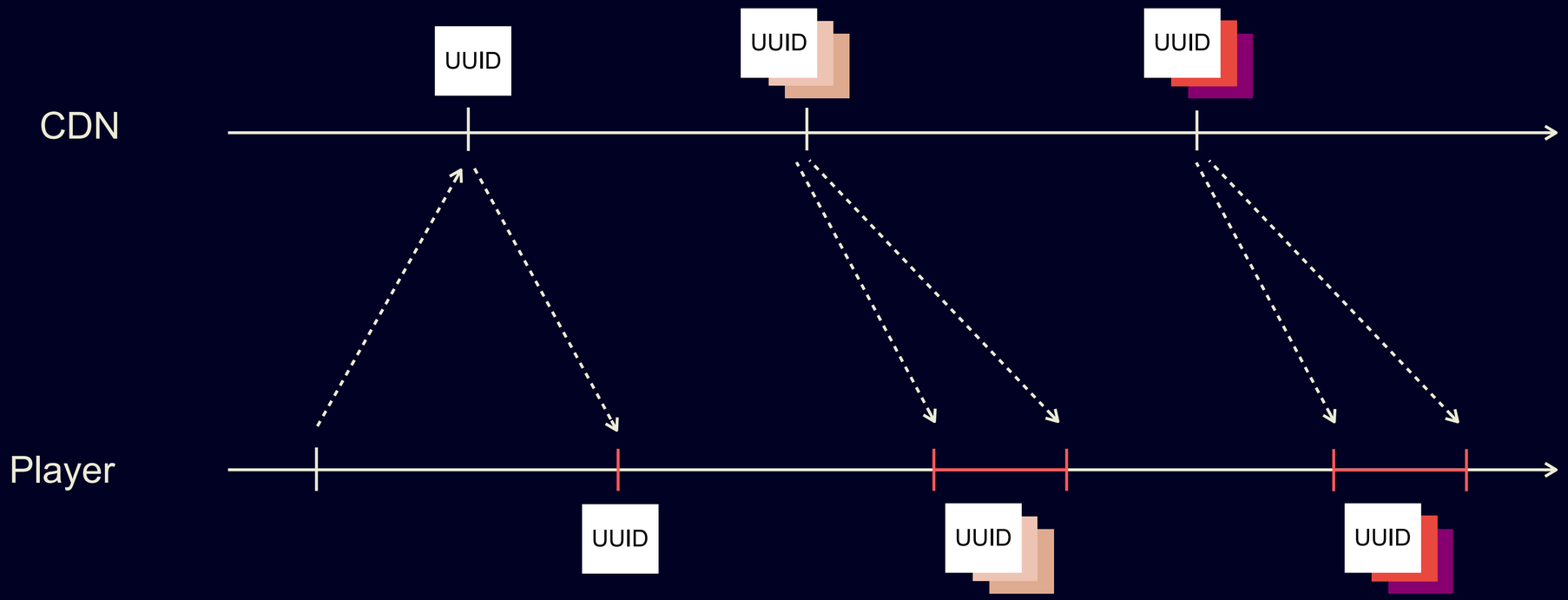
длительность загрузки фрагмента ≈
длительность получения фрагмента — время «простоя»

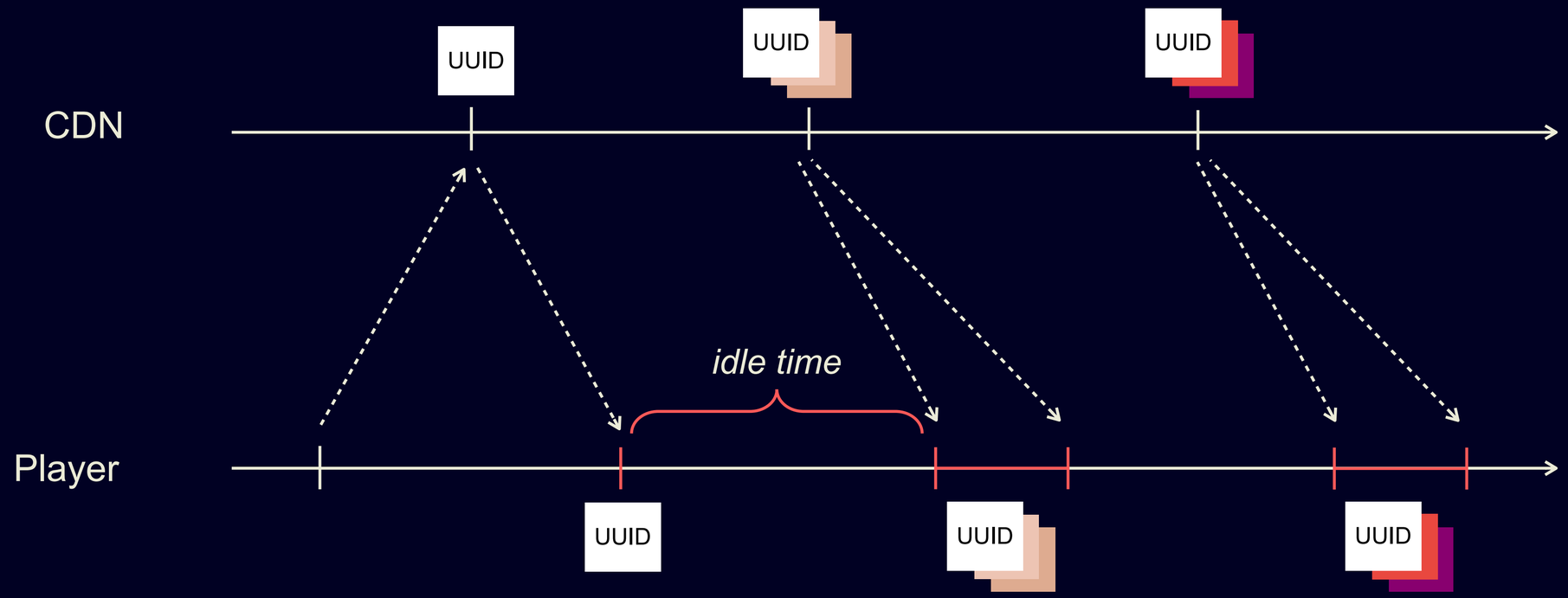
Как получить
idle time на клиенте?


Таймстамп получения запроса
Таймстамп отправки фрагмента #1
Таймстамп отправки фрагмента #2
CDN
Player

UUID-box

- 16 байт uuid-идентификатор + payload
- В качестве метаданных в payload запишем серверный таймстамп!


UUID
UUID
UUID
CDN
Player
UUID
UUID
UUID



idle time = UUID#2 − UUID#1

Оценка скорости канала

$$bw \approx \frac{b}{\tau - (UUID\#2 - UUID\#1)}$$

Итоги

1. Медиана задержки — 6 секунд
(при изначальном бюджете — 10 секунд)
с потенциалом дальнейшего улучшения

2. Нет значимых изменений остальных
продуктовых и технических метрик

Что мы планируем делать дальше

1. Уменьшать задержку за счет дальнейших оптимизаций и "подкрутки" скорости воспроизведния, чтобы нагнать отставание после буферизации

2. Улучшить алгоритм оценки скорости сети - сейчас он игнорирует параллельно загружаемые аудио-данные

3. Отключать low-latency режим после нескольких прерываний просмотра

Мы нанимаем!

Фронтенд

- Разработчик в веб-плеер
- Стажер в веб-плеер

Android

- Разработчик в android-плеер

clck.ru/Z6gJV

videotech

Спасибо

Дмитрий Кравцов,
Яндекс.Видеостриминг

Алексей Гусев,
Яндекс.Дзен