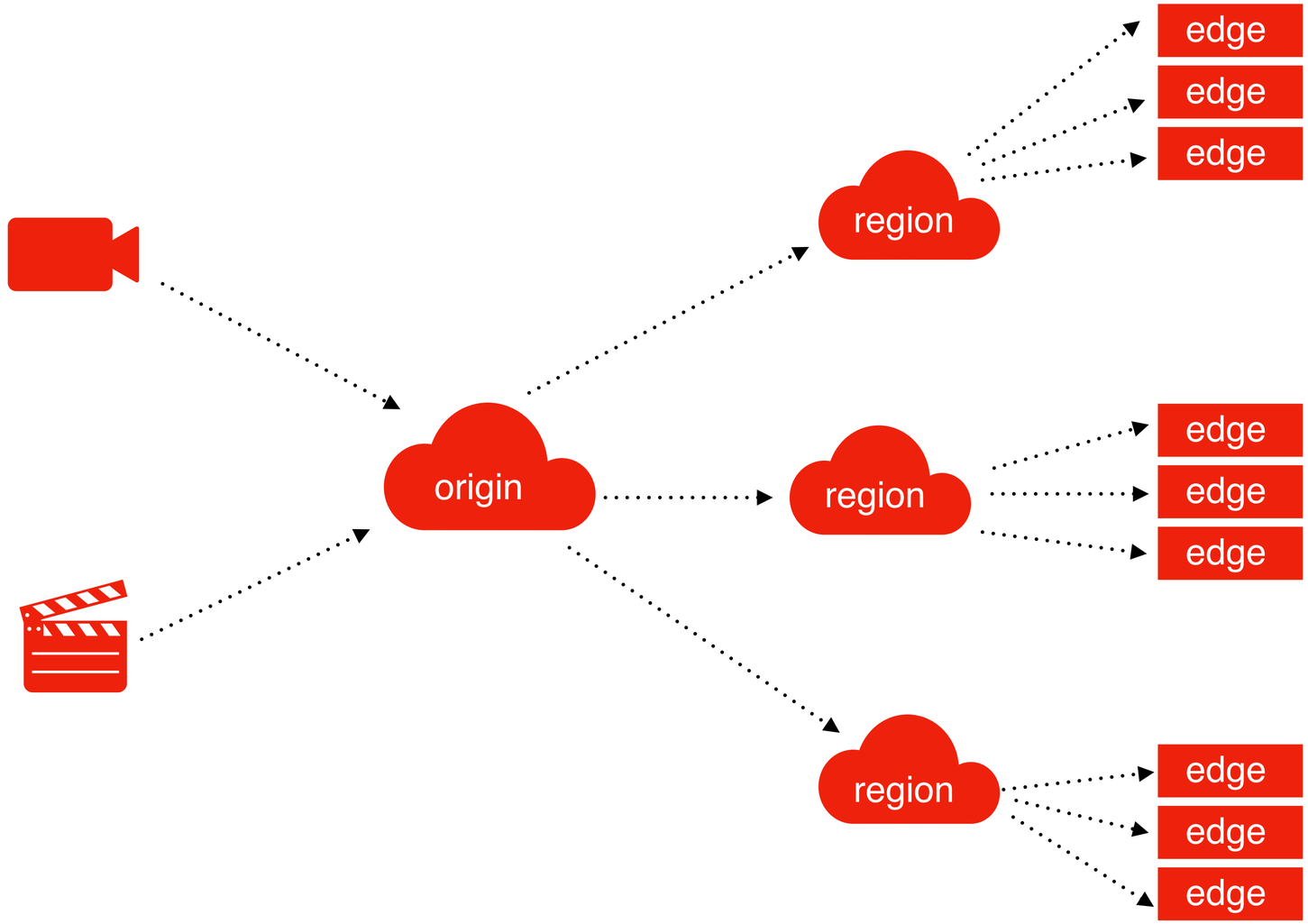
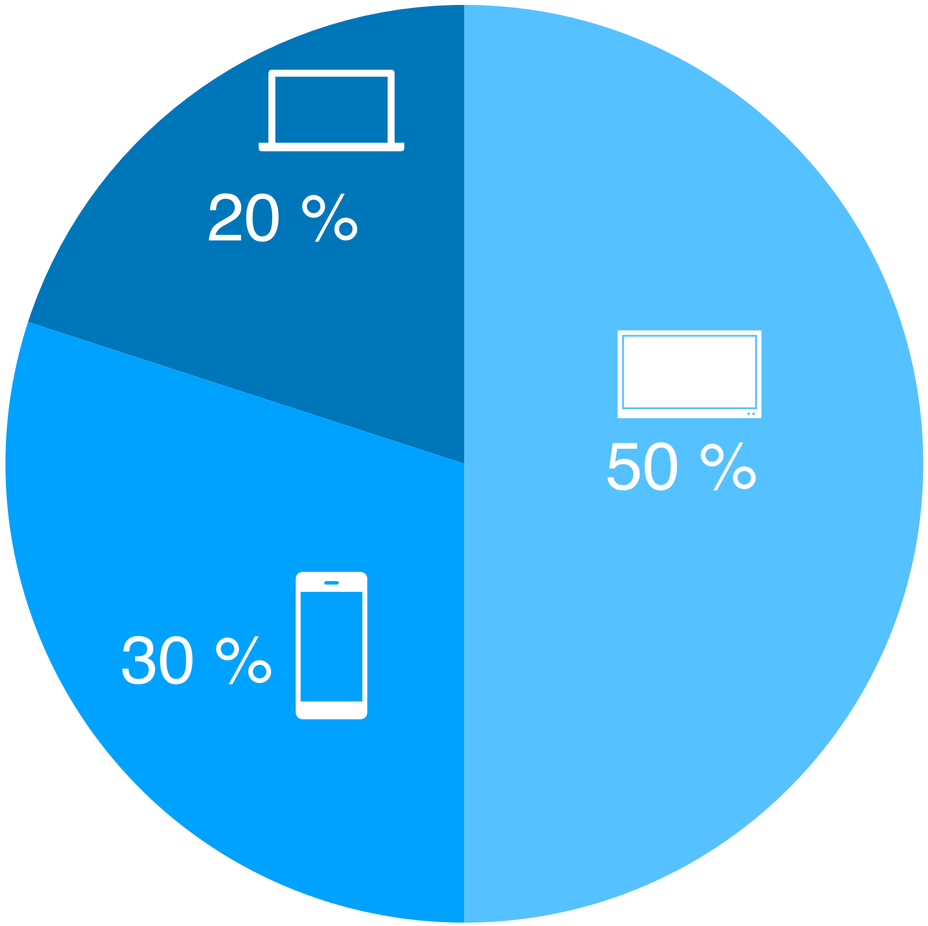
Peer-to-Peer доставка видео на базе WebRTC

Традиционная схема доставки видео



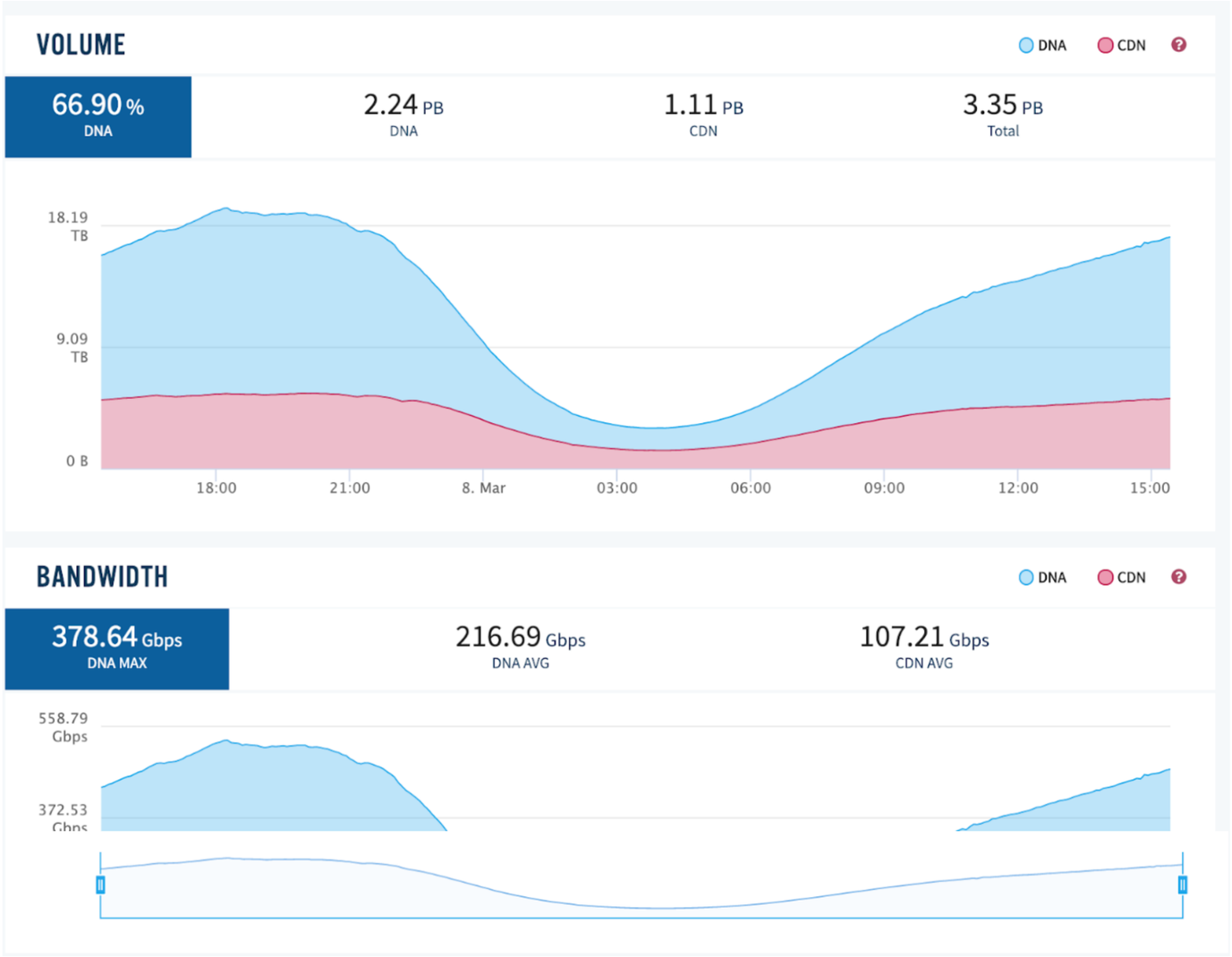
В случае VOD при отсутствии контента на edge'ах, пользователь направляется на вышестоящий узел

Доля трафика по платформам



1. Доля Smart TV растет
2. Десктопный трафик перетекает в мобильный и Smart TV

Трафик

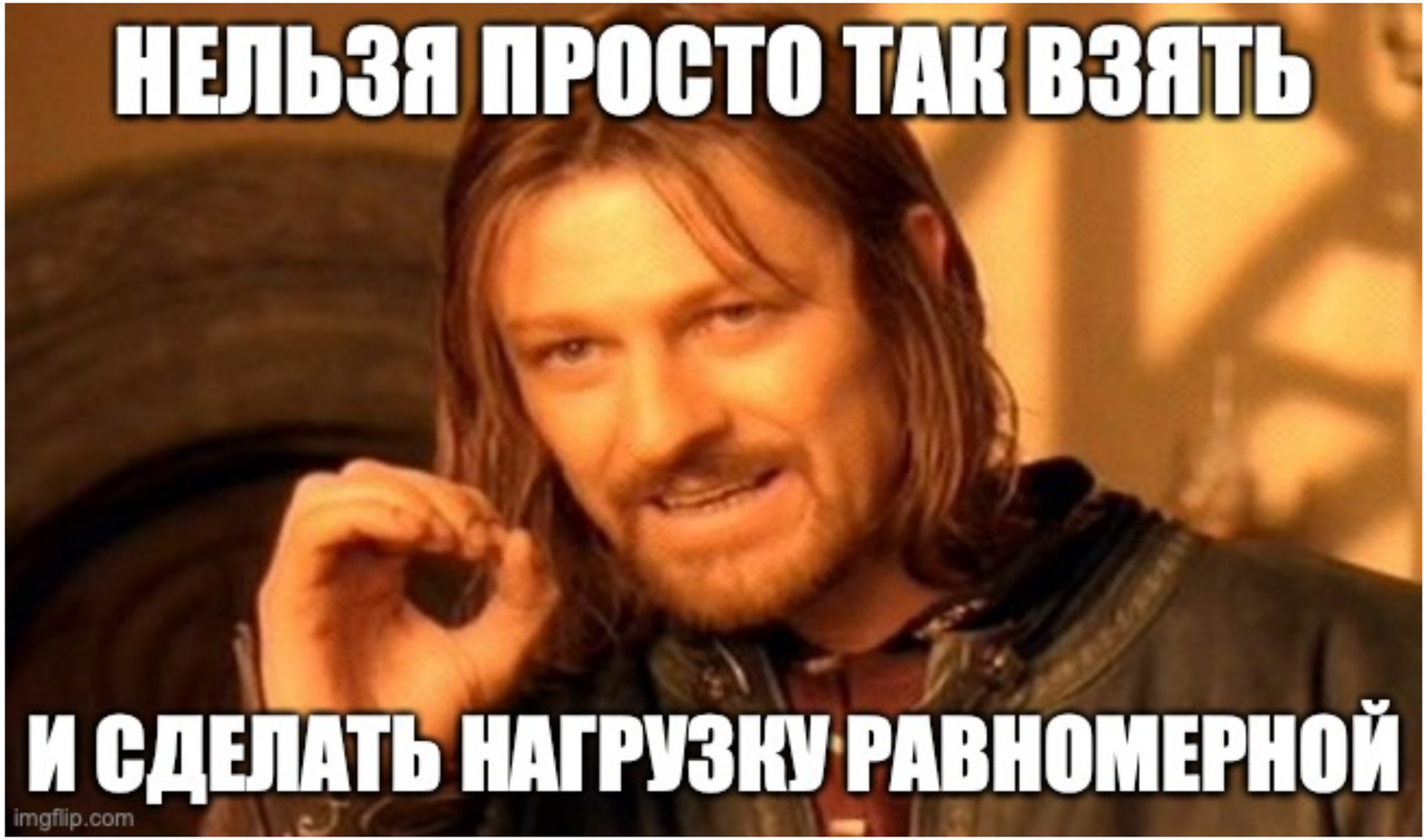


— никогда не бывает линейным

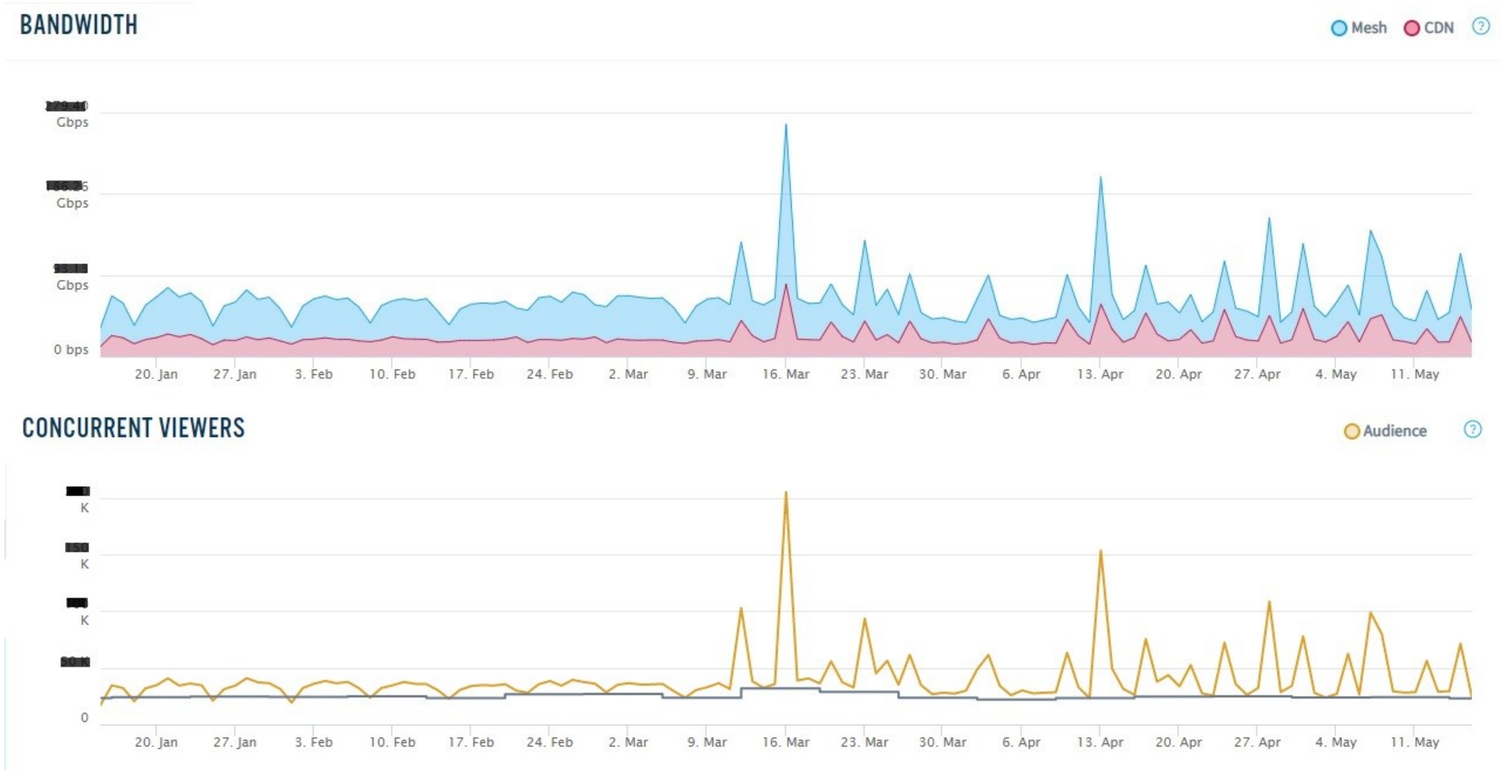
— ярко выраженный prime time

— сезонность -- зимой холодно, гоу домой смотреть ламповое кино на ivi

Проблема №1 любого CDN и OVP(online video platform)



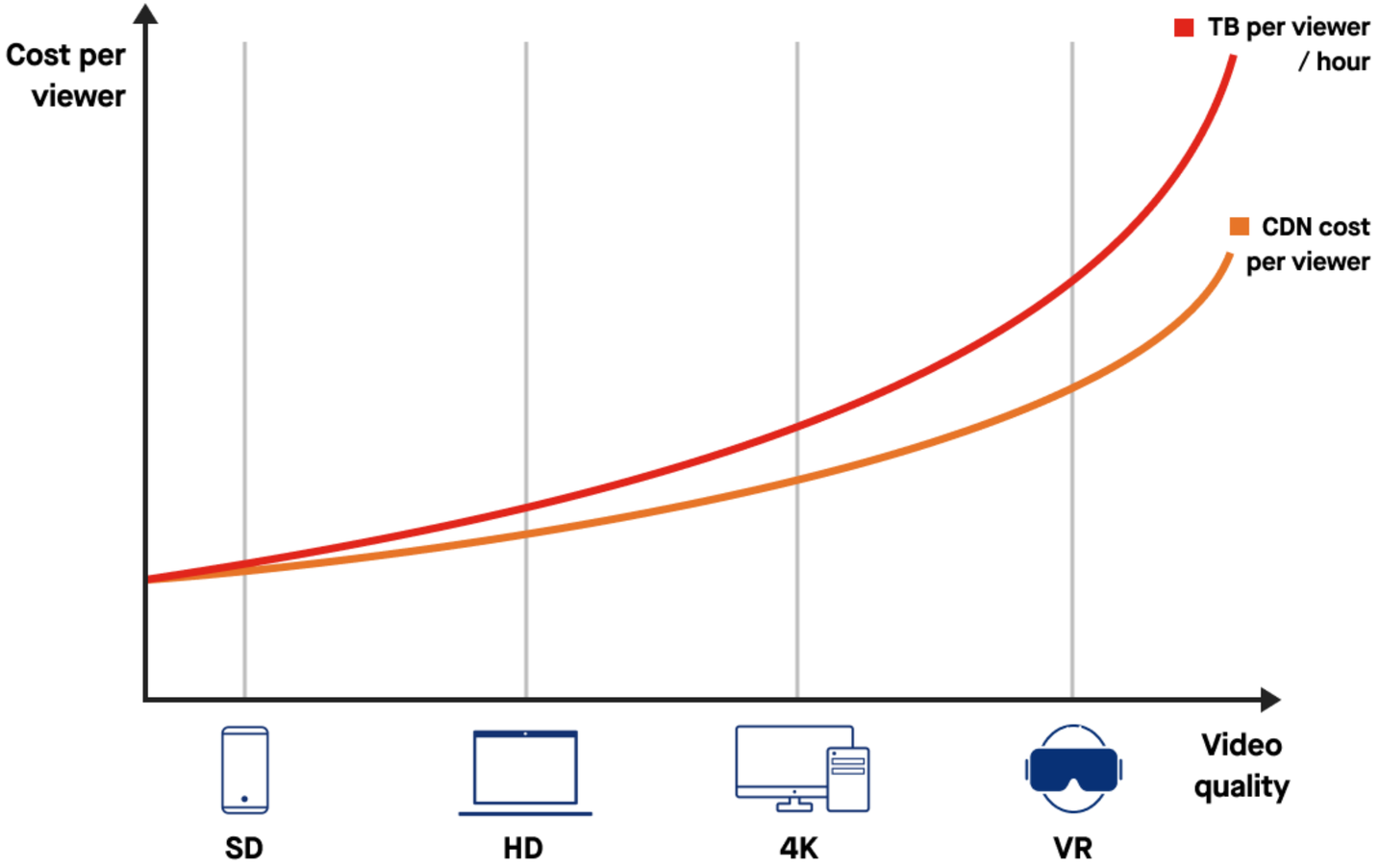
Последствия неравномерности трафика для OVP



Непостоянность нагрузки на CDN приводит:

- сложность бюджетирования (flat fee vs пики трафика, traffic quota overuse)
- Просадке среднего битрейта на клиентах
- Увеличению буфферизаций на клиентах
- Отказу CDN -- "кина не будет, CDN кончился"
- Потенциальному отказу OVP платформы

Сложность бюджетирования

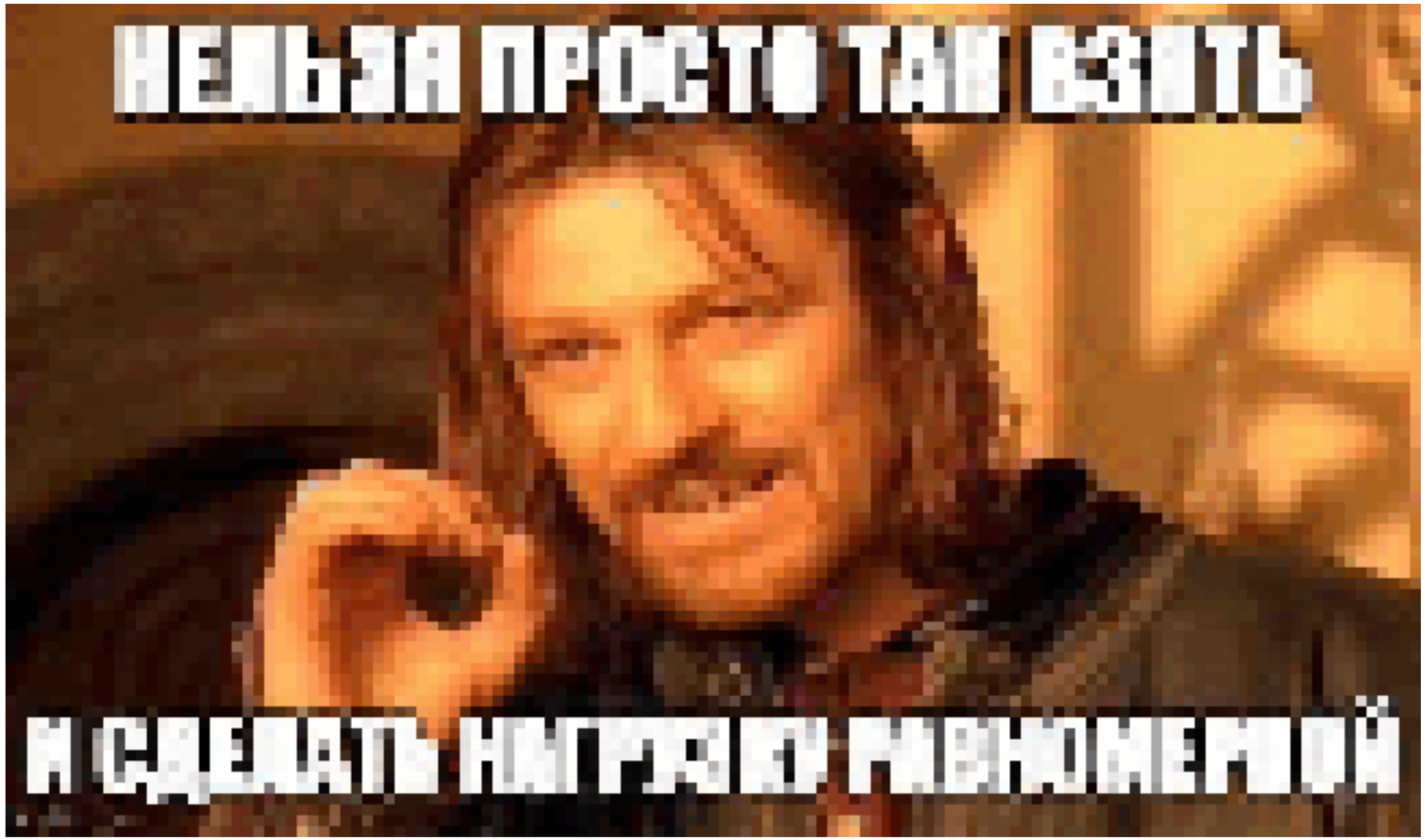


Парадокс OVP юнит-экономики*:

- хотим привлекать больше подписчиков за меньшие деньги
- чтобы привлекать больше подписчиков, нужно повышать качество трансляций
- повышение качества трансляций стоит денег =(

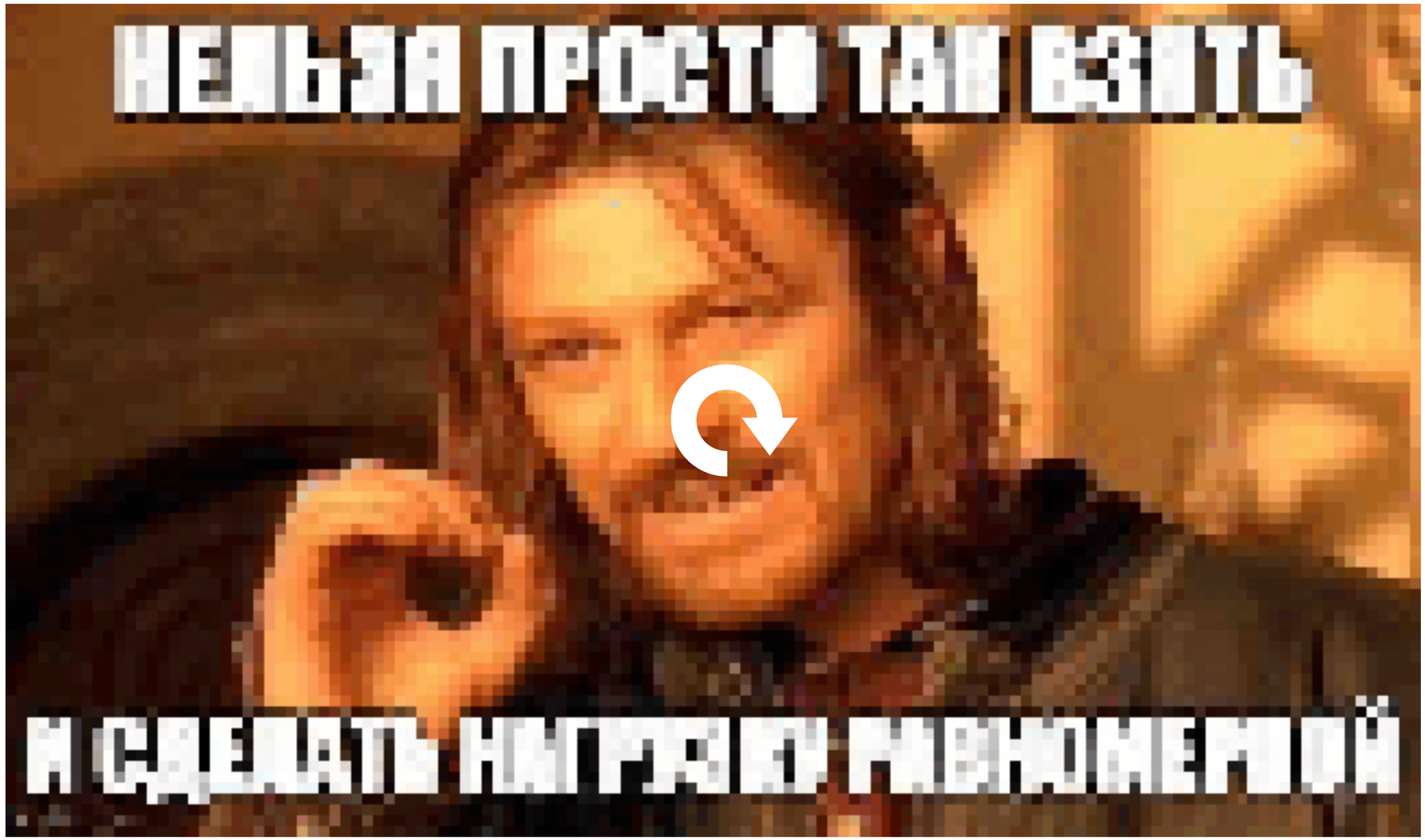
*актуально, если вы монопродукт на конкурентом рынке и у вас нет таксопарка

Просадка среднего битрейта



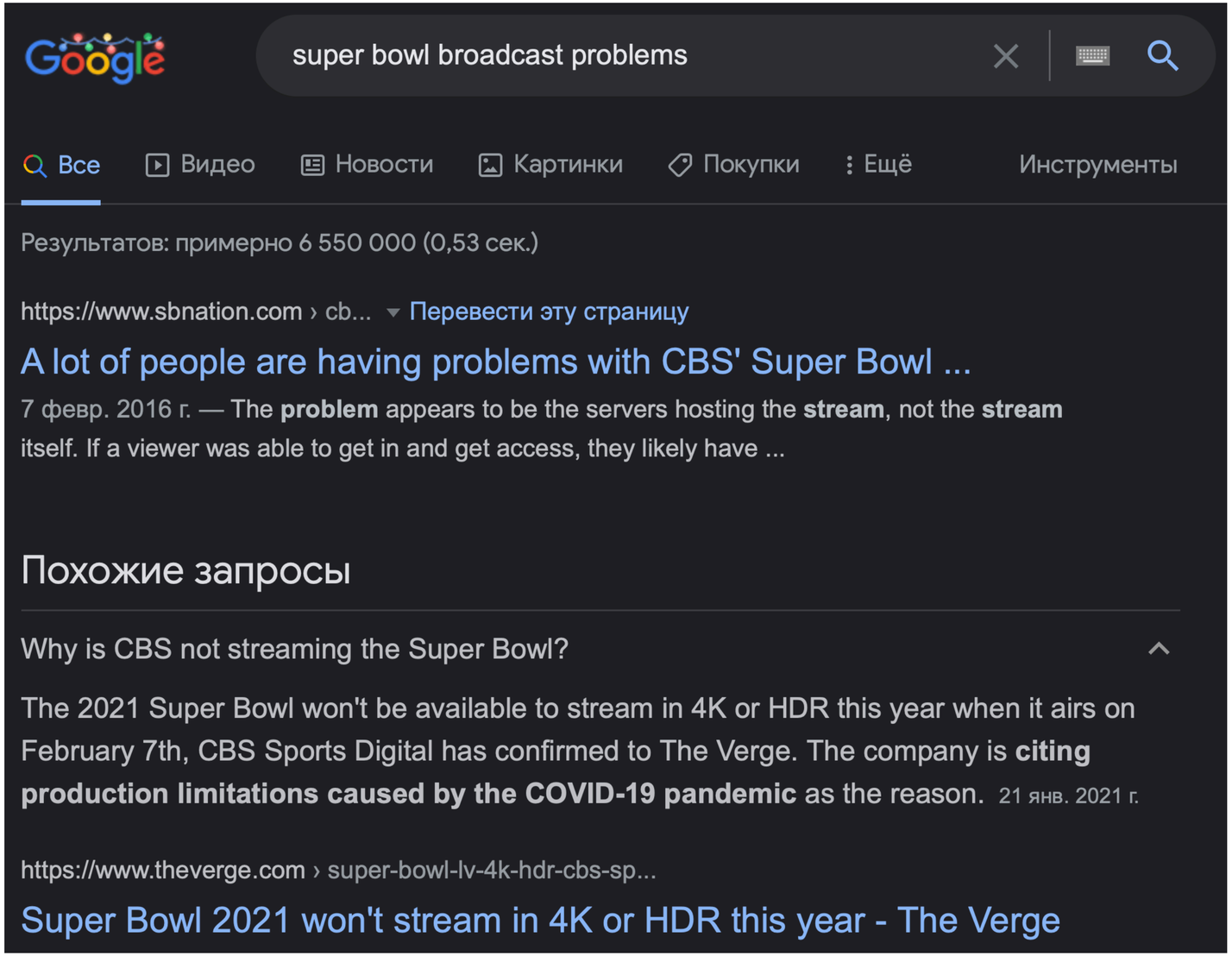
А скажут, скажут, что это ⬆️ 4K!

Рост количества сессий с буфферизациями



И что почистите кэш и куки и заработает!

Смерть трансляции



5.7 миллиона пользователей смотрели Супер Кубок в 2021 году через онлайн платформы

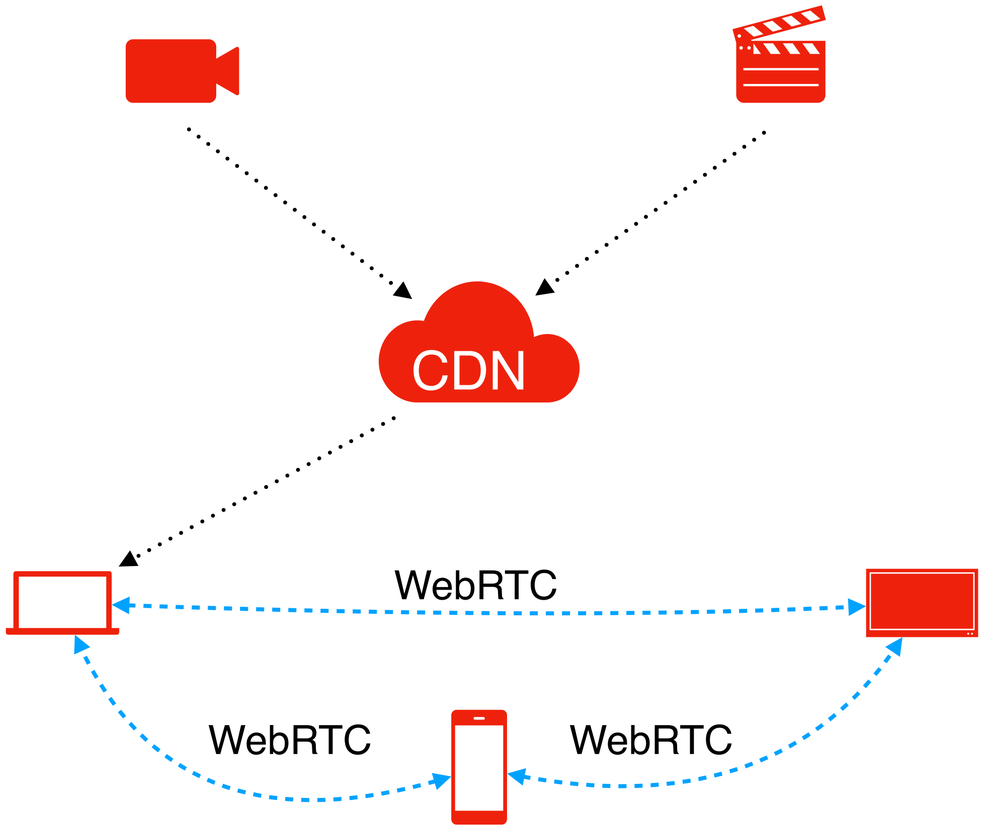
Раздавать видео без CDN*

*ну почти без

CDN + P2P(WebRTC)

WebRTC

- передача видео, аудио и **произвольных бинарных данных(Data Channels)**
- передача данных между клиентами **по прямому соединению**
- RTP и SCTP
- потребность в CDN снижается
- кросс-платформенный



Демонстрация в браузере

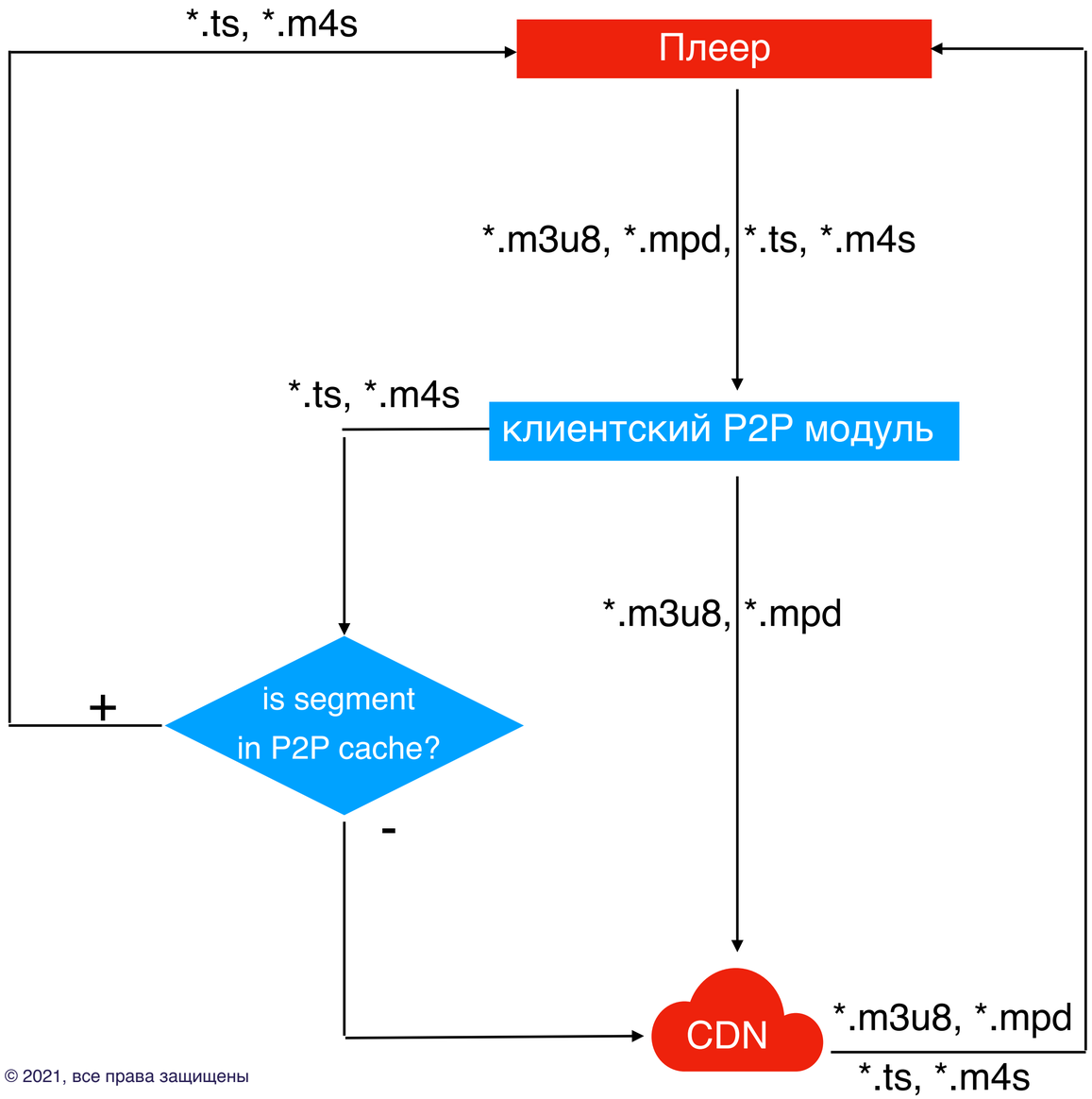
https://streamroot.io/demo/

(можно и с мобилок, и с утюга, но с целью "поковыряться" лучше из Chrome или FF)

(chrome://webrtc-internals, FF -- about:webrtc)

Как это работает

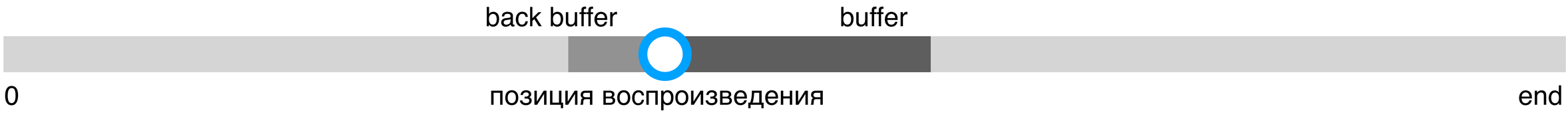
На пальцах



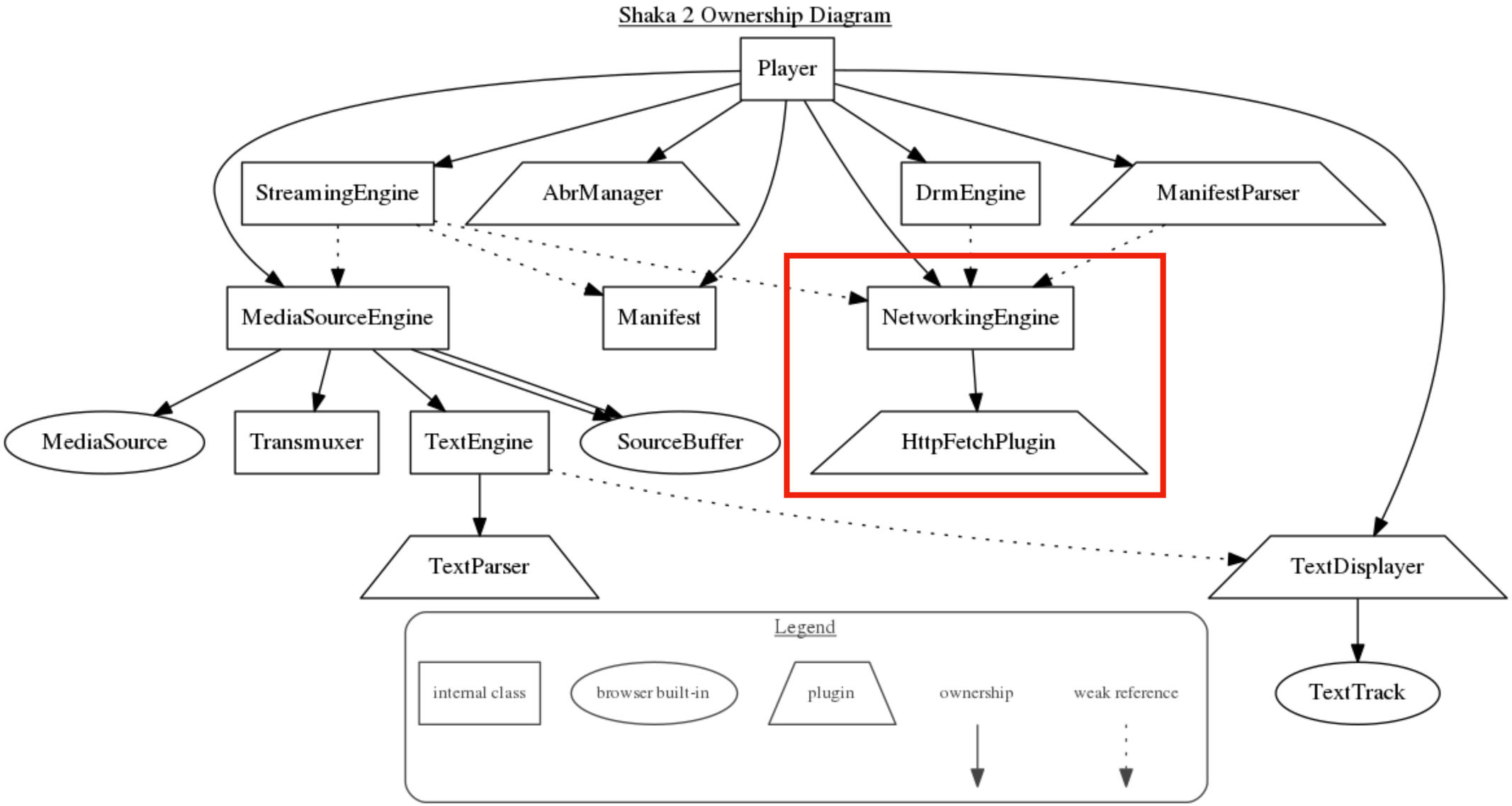
1. Перехватываем запрос манифеста от плеера, чтобы узнать URLs сегментов.
2. Заранее скачиваем сегменты через WebRTC у тех, у кого они уже есть.
3. Перехватываем запросы сегментов от плеера, и если сегменты были скачаны заранее, отдаем их в плеер, минуя CDN

Видеоплееры адаптивного стриминга

1. Адаптивный стриминг -- DASH, HLS, MSS, Adobe HDS(good bye, sweet prince)
2. Scheduler(Streaming engine) -- функция с аргументами "timestamp, quality", результат - URL сегмента из манифеста
3. Сегмент запрашивается с CDN и после скачивания попадает в буффер плеера
4. Размер буффера задается в конфиге плеера



Устройство Shaka Player



Перехват запросов

Два варианта для Web:

1. Переопределить XMLHttpRequest и Fetch API своей реализацией (Peer 5)

2. Интеграция с транспортным слоем видеоплееров (Streamroot).

Интеграция P2P модуля с WEB плеерами

1. Мониторинг позиции воспроизведения и буффера можно осуществлять либо через API плеера, либо напрямую с медиаэлемента страницы.

2. Некоторые плееры позволяют через DI пробросить свою реализацию транспорта(hls.js, shakaplayer).

3. А некоторые нет. Но если они с открытым исходным кодом(dash.js) -- засучили рукава и написали "плагин".

4. Плееры с закрытым исходным кодом (JWPlayer, THEOplayer, Bitmovin, CatsLabs, Brightcove) как правило являются частью соответствующей коммерческой платформы и предлагают API для написания плагинов.

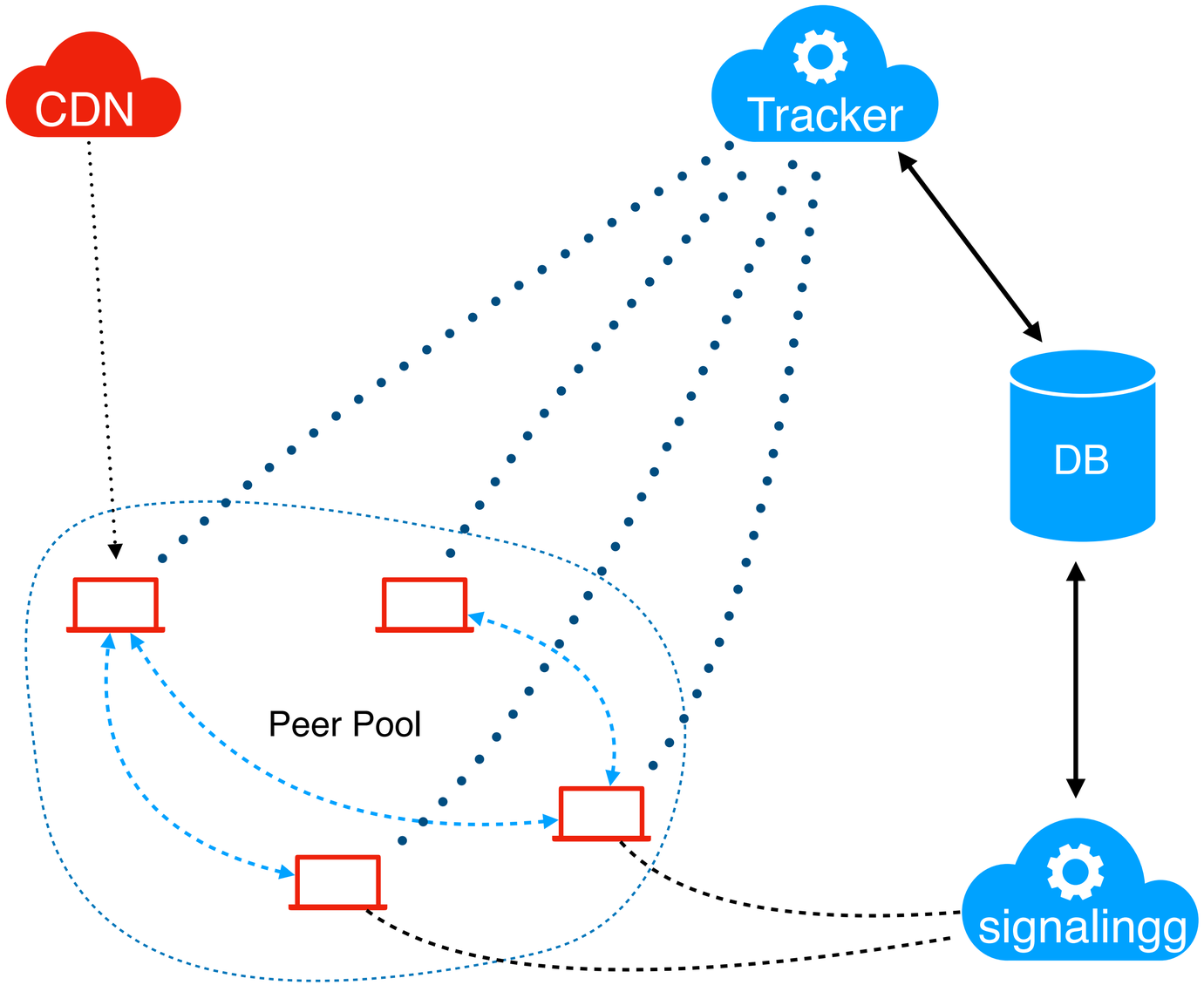
Интеграция P2P SDK с мобильными приложениями

1. Android. Это NexPlayer или ExoPlayer, оба с открытым исходным кодом, интегрируемся аналогично Web.
2. iOS. Это AVPlayer, альтернатив ему нет и он закрытый -- транспорт не переопределить. Что делать? Проявить смекалочку:
    1. Т.к. приложение, показывающее видео, контролируете вы, то URL манифеста вам известен.
    2. Скачиваем его и парсим.
    3. Прячем p2p модуль за локальным прокси сервером внутри приложения. Урлы сегментов манифеста переписываем так, чтобы они смотрели в прокси, на прокси мапинг подмененных URL в оригинальные.

Документация

https://support.streamroot.io/hc/en-us/sections/360000449494-Web-Players

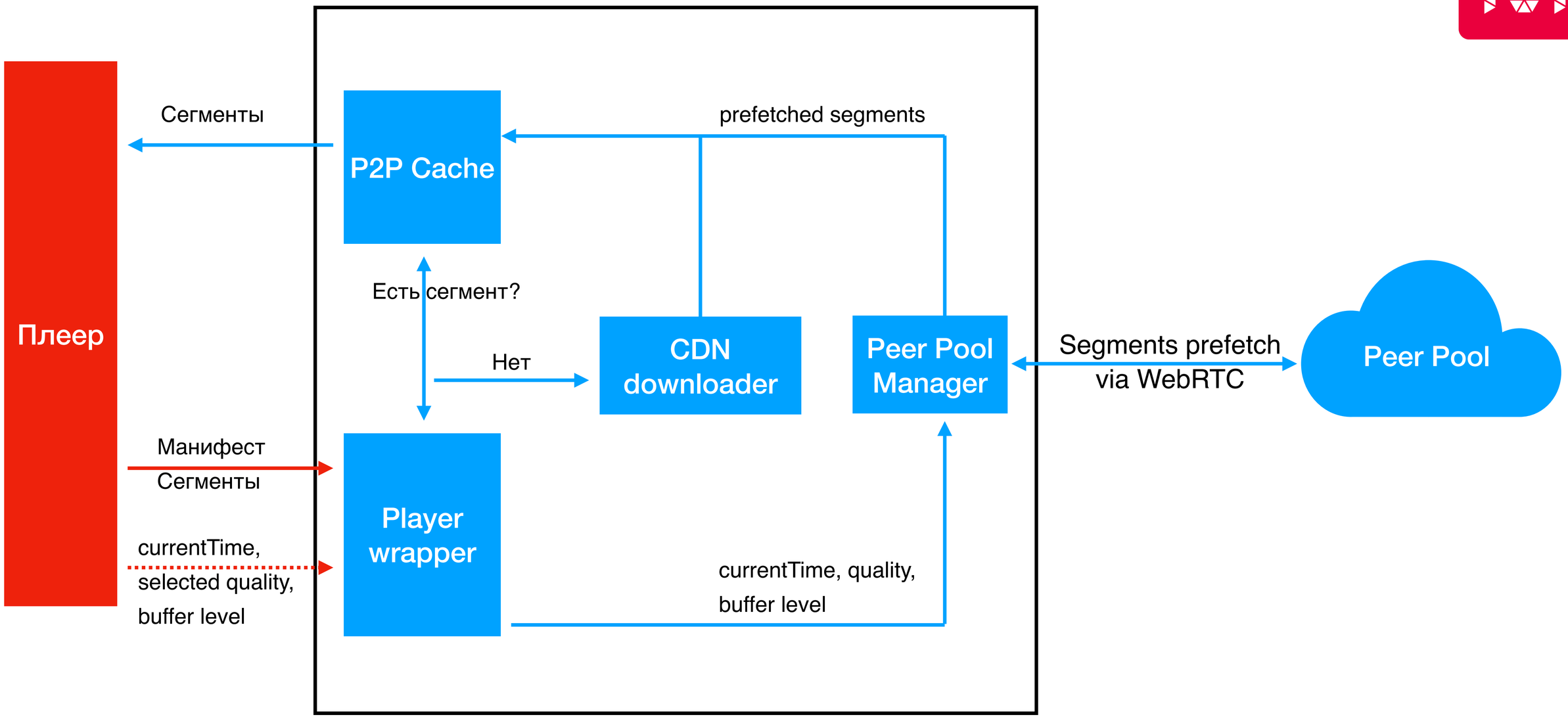
P2P



— Трэкер для авторизации клиентского P2P модуля и формирования потенциальных кандидатов в клиентский PeerPool.

— Signaling сервер для установления соединения с пирами, предложенными трэкером. Общение с Signaling через WebSocket, SDP сжимаются, все сообщения в protobuf

Клиентский P2P модуль



Клиентский P2P модуль

1. Наблюдает за позицией воспроизведения, выбранным качеством и наполненностью буффера плеера.
2. Перехватывает запросы сегментов, если они есть в P2P Cache, то отдает их плееру, минуя CDN. Иначе запрос отправляется по назначению -- в CDN.
3. Закачиваем через WebRTC DataChannels те сегменты, которые плеер скоро запросит для наполнения буфера. Скачиваем у тех, кто их скачал раньше нас.
4. Все полученные сегменты храним в P2P Cache.

Трэкер

Привет! Я:

- запущен у клиента <ключ_клиента>
- смотрю <manifest_url>
- у меня <позиция_воспроизведения>
- у меня <выбранное_качество>
- не хочу дружить с <список_плохих_пиров>

Привет! Молодец.

Попробуй соединиться с <кандидаты_в_пиры>

Трекер:

1. Аутентификация клиентских p2p модулей
2. Подбор для них лучших пиров для максимальной эффективности обмена сегментами через p2p

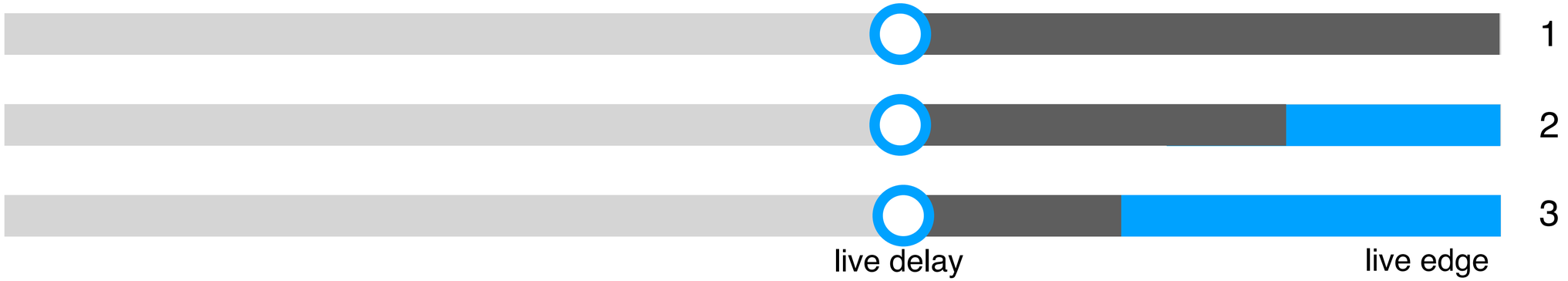
P2P модуль:

1. При смене качества или позиции воспроизведения запрашивает новых пиров у трэкера

Подбор пиров

VOD

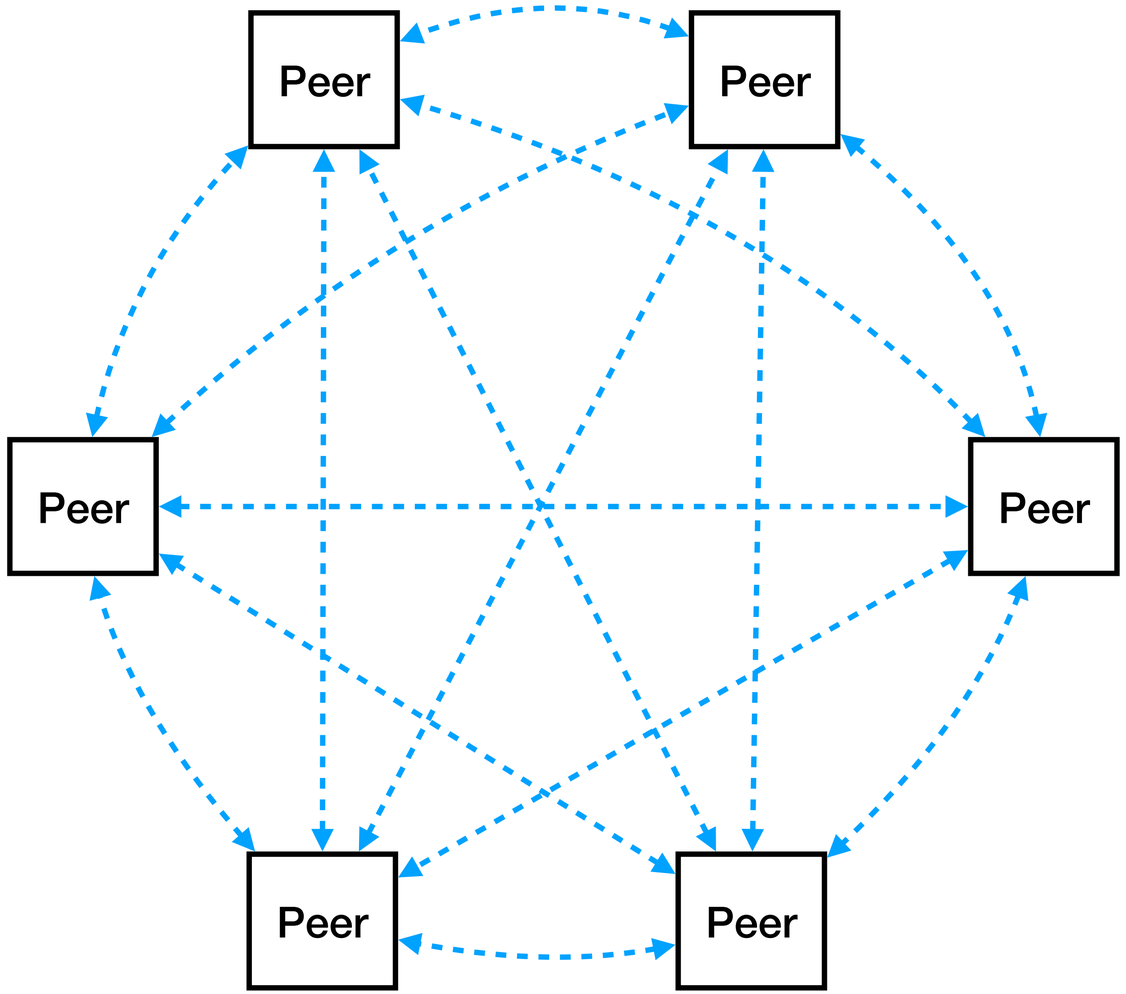
LIVE



Signaling

1. Решает классическую задачу WebRTC -- обмен SDP между клиентами
2. WebSocket
3. Protobuf
4. SDP содержит много мусора, и легко сжимается, позволяя сэкономить трафик
5. Никакого TURN, только STUN. Отбираем только UDP srflx кандидатов
6. Проверка соответствия поддерживаемых версий p2p протокола в процессе обмена SDP
7. Если все проверки пройдены и удалось установить соединение, peer попадает в Peer Pool

Peer Pool



Это:

1. Ячеистая топология
2. $2(n-1)$ соединений на клиента
3. $n^2 - n$ соединений max

Peer Pool

Спокойно делаем следующее:

1. P2P модуль непрерывно оценивает качество пиров в Peer Pool
2. Если качество обмена данными с пиром плохое -- разрываем соединение и в exclude list его. Трэкер нам его больше не пришлет.
3. При малейшей угрозе для QoS, QoE -- сокращаем размер Peer Pool вплоть до полной самодеактивации P2P модуля
4. Мониторим bandwidth, таймауты передачи, dropped_frames, cpu load там, где это возможно
5. Бережно относимся к каналу пользователя
6. Алгоритм текущего ведра

Peer Pool

Протокол

Сообщение	Назначение
ping	фильтрация отвалившихся пиров
has segment	понять, есть ли у пира нужный нам сегмент
has segment answer	ответить пиру, есть ли у нас нужный ему сегмент
chunk request	инициировать передачу чанков сегмента
info	обмен мета-информацией об окружении пиров; для статистики и мониторинга

Передача сегментов

1. У сегмента есть координаты: timestamp на presentation timeline, качество
2. Сегмент разбивается на чанки по 15Кб(может меняться)
3. У чанка есть координаты: индекс в сегменте, id сегмента
4. Сегмент может быть собран из чанков от разных пиров, а иногда и из чанков от CDN(range requests)
5. Жесткие таймауты на передачу чанков. Медленные пиры выбрасываются

P2P Cache

1. In memory хранилище бинарных данных: Map(id сегмента => Uint8Array)

2. Начальный размер хранилища задается в конфиге P2P модуля, но может динамически изменяться в процессе(Live vs VOD, super seeder, QoS, QoE).

3. ABR алгоритмы плееров. Нельзя просто так взять и мгновенно отдать сегмент из P2P Cache в плеер -- это приведет к тому, что ABR алгоритм плеера завысит оценку пропускной способности канала и от этого может пострадать QoS/QoE. Поэтому сегмент отдается плееру с задержкой, рассчитанной так, чтобы не исказить bandwidth estimate плеера.

4. GC съедает старые сегменты(находящиеся далеко от currentTime).

Разработка, тестирование, деплой, мониторинг

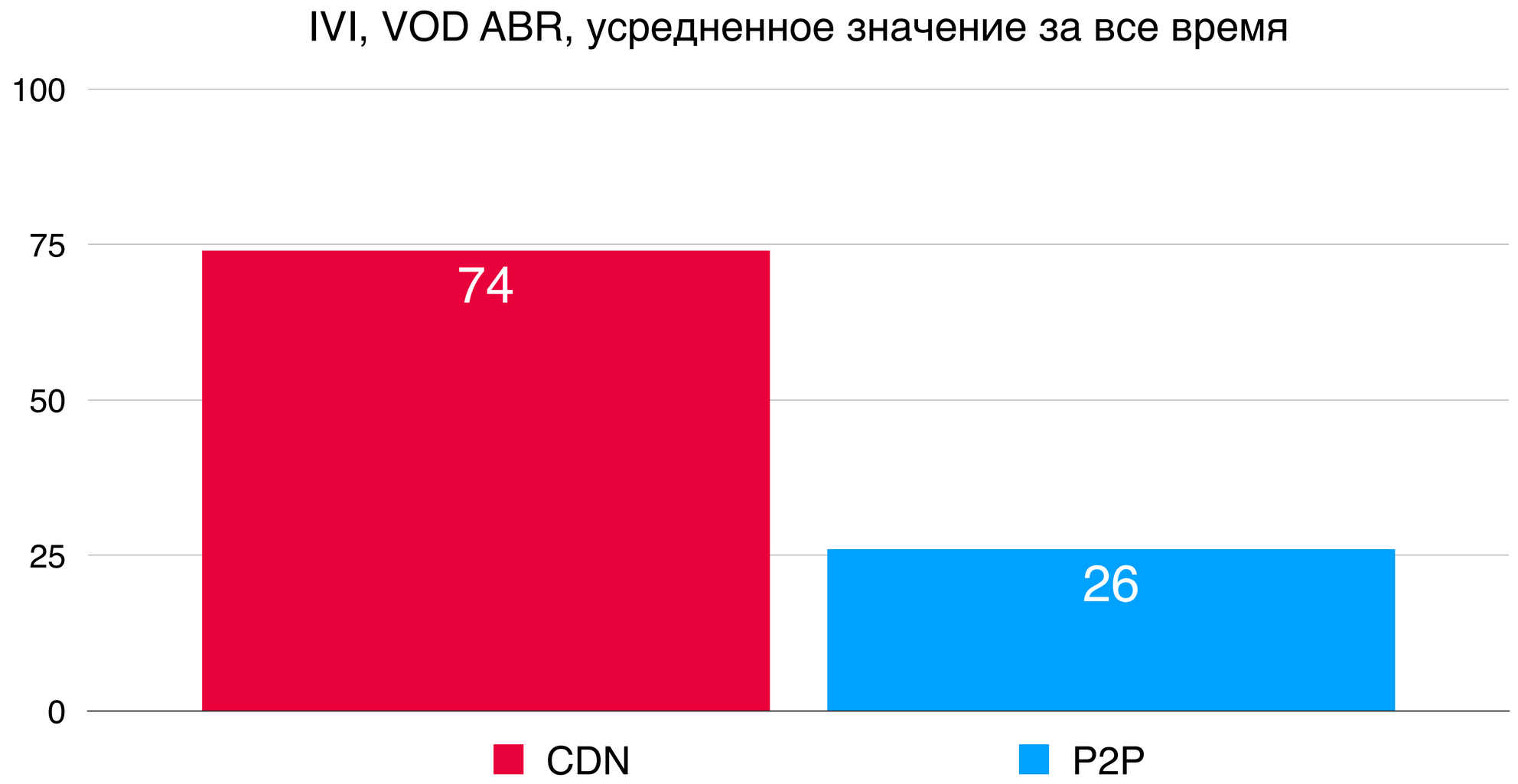
- Порядка 350 параметров в конфиге P2P модуля, настраиваемых под каждого клиента индивидуально(постоянный мониторинг и АБ тесты), часть из них может меняться на лету на стороне клиенте для максимальной эффективности
- Feature flags
- Canary release
- Первый этап мониторинга метрик - убеждаемся, что релиз ISO с выключенными флагами
- Включаем флаги, мониторим
- Откатываемся крайне редко, вместо отката - выключаем feature flag, выкатываем потенциальный фикс как новый релиз и повторяем цикл оценки

Недостатки, ограничения

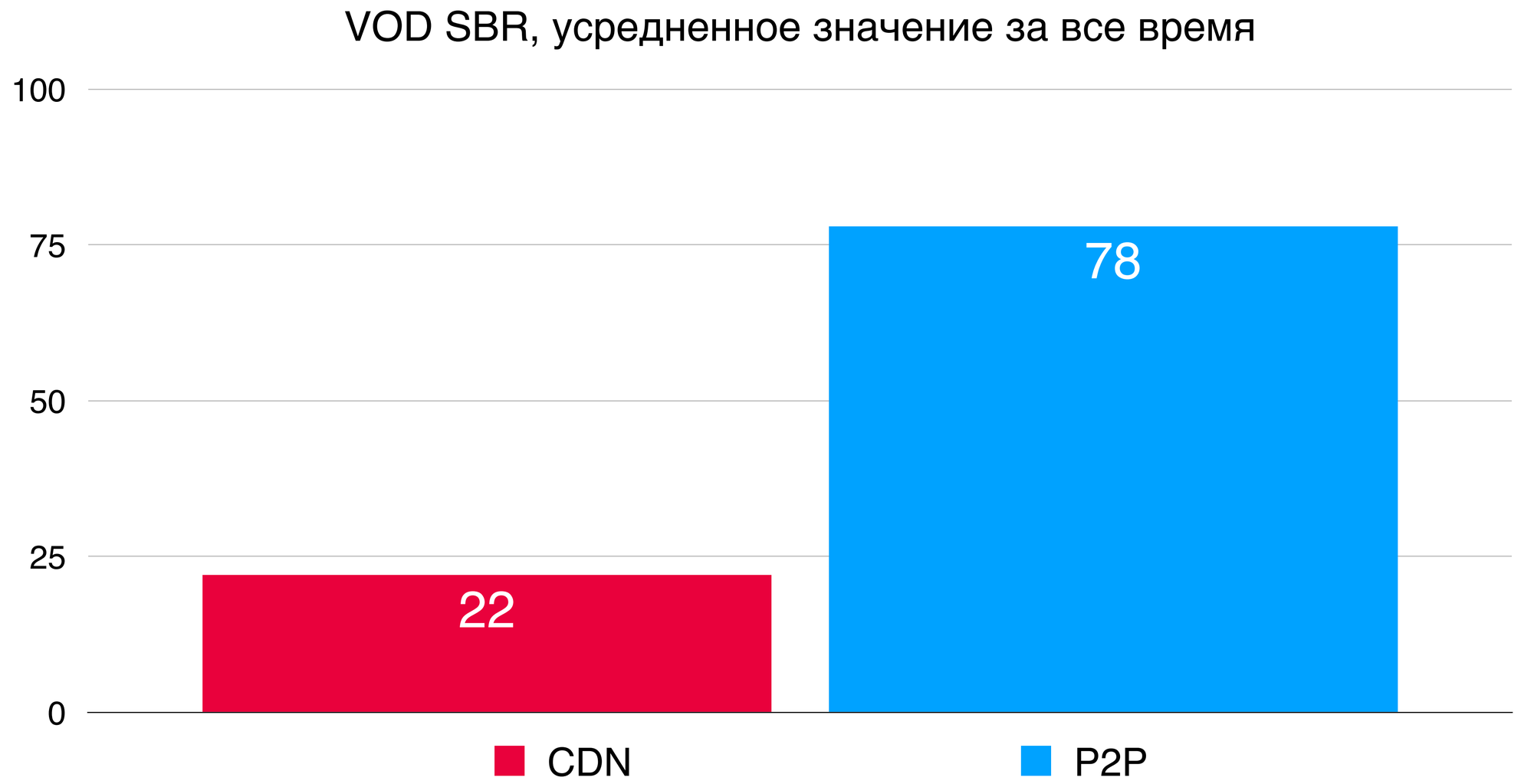
- Не работает с low latency в Live -- чем ближе мы к live edge, тем меньше размер буффера
- Для крупных OVP с десятками тысяч тайтлов в каталоге может быть сложность с построением Peer Pool для непопулярных тайтлов
- Персонифицированные сегменты в манифесте/плейлисте(SSAI)
- ADSL

А это хорошо работает?

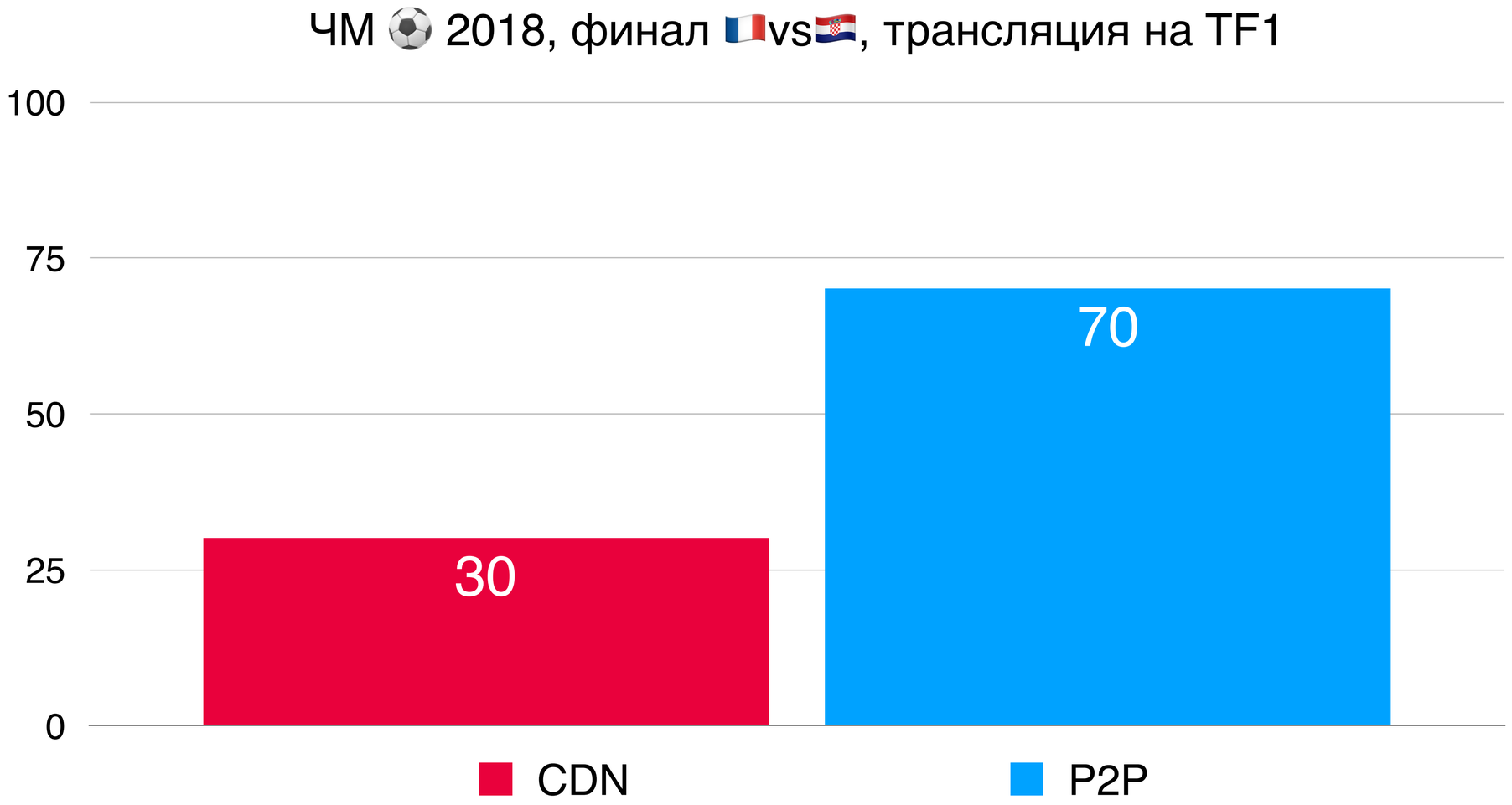
Эффективность. VOD



Эффективность. VOD



Эффективность. Live



https://blog.streamroot.io/world-cup-ceo-note/

Эффективность. Live

"Streamroot delivered an average of 70% of partner broadcasters’ total World Cup traffic"

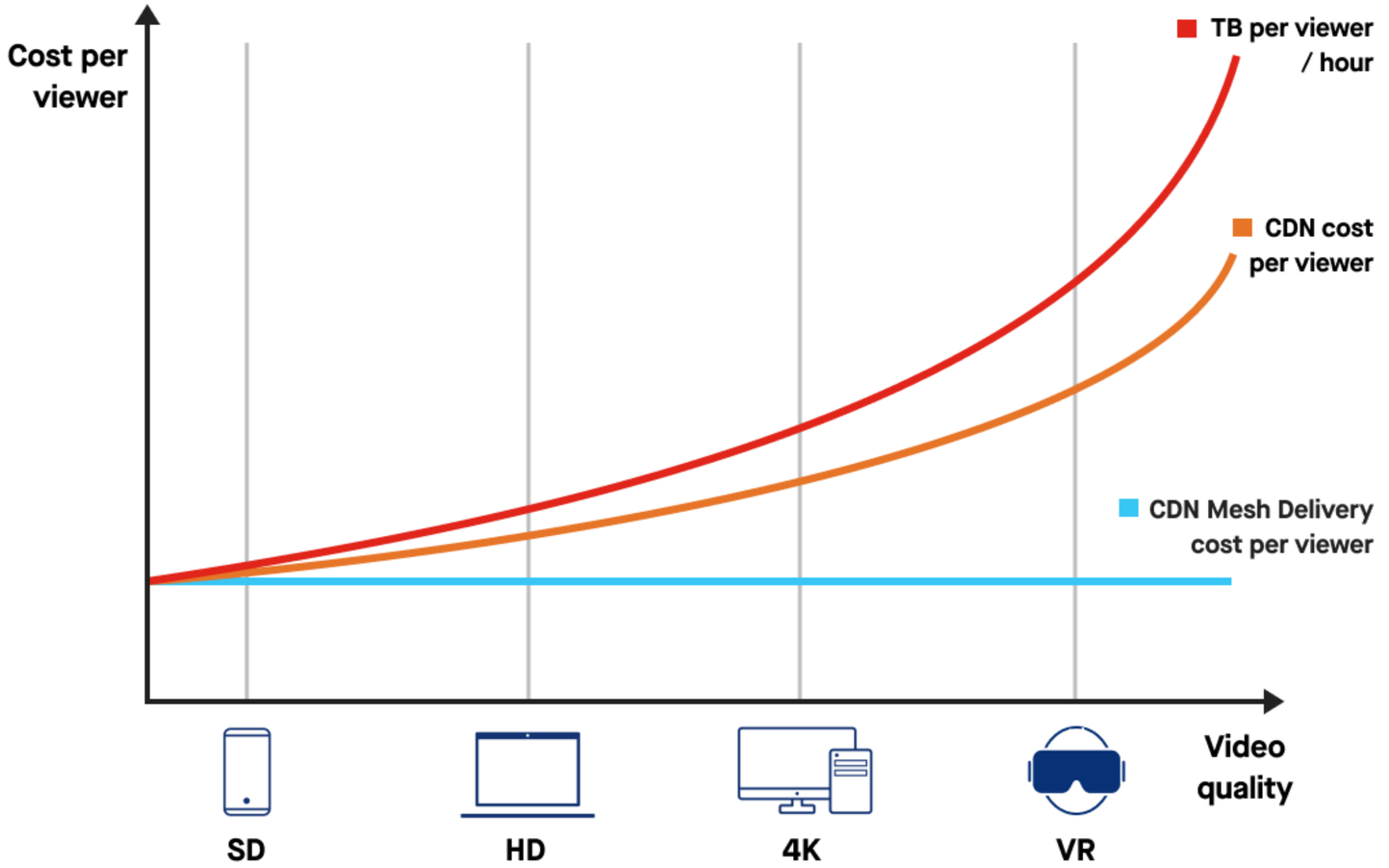
"Rebuffering rates on platforms using Streamroot fell 11% in Europe, and up to 33% in Latin America."

https://blog.streamroot.io/streamroot-powers-largest-ever-deployment-of-distributed-network-delivery-video-streaming-world-cup-2018/

Выводы

1. Традиционные CDN как продукт становятся commodity(как, например, энергоносители).
2. Клиенты на CDN рынке это клиенты на рынке энергоносителей -- между газом или электричеством от вендора А, Б или В особой разницы нет, пока лампочка горит, клиент выбирает самый дешевый.
3. CDN как технология масштабируется экстенсивно, и это масштабирование не решает проблем, а лишь оттягивает(откладывает?) их наступление.
4. Децентрализация CDN через WebRTC дает ярковыраженное конкурентное преимущество, позволяя обойти ограничения CDN без существенных вложений в инфраструктуру.

Выводы



Благодарю за внимание

ruslandinov@gmail.com
https://www.linkedin.com/in/ruslandinov/