Orion Innovation
formerly MERA

ExoPlayer для IPTV приложений

КУЛИКОВА НАДЕЖДА, ORION INNOVATION

24.11.2021

Об авторе

- Старший Android разработчик в компании Orion Innovation
- 7 лет в мобильной разработке
- Опыт в проектах разного рода: Android TV, Automotive, VOIP
- Почти в каждом проекте мне везет поработать с ExoPlayer.

NADEZHDA.S.KULIKOVA@GMAIL.COM

Telegram @KULIKOVANADEZHDA

Анонс

Не будет:
- Базовой информации
- Устройства ExoPlayer

Будет:
- Набор улучшений для ExoPlayer
- Полезные фишки, улучшающие производительность

Уровень подготовки:
- Для тех, кто когда либо работал с Exoplayer

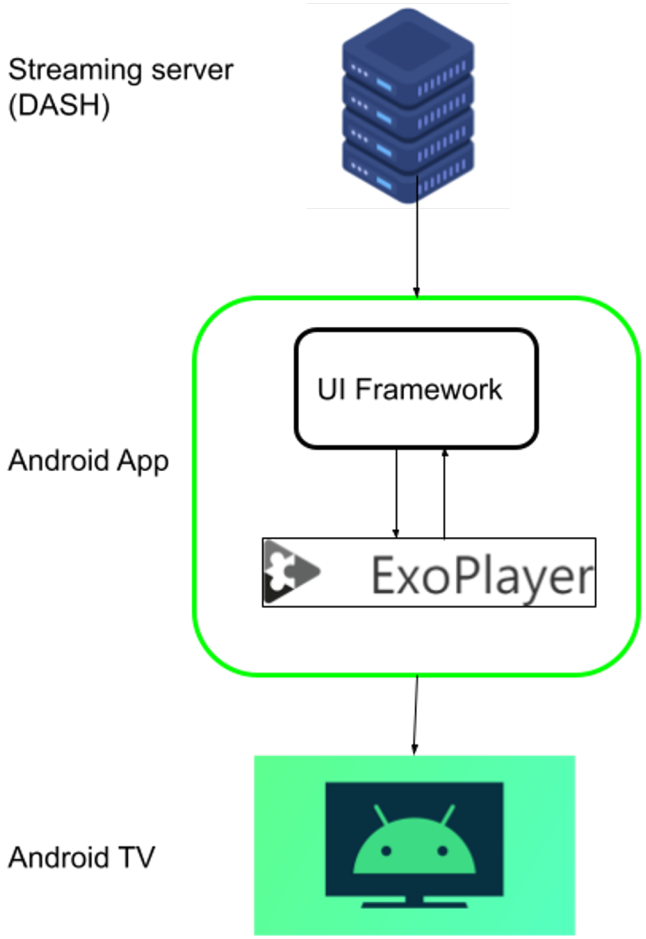
О чем поговорим

- **ТЗ и требования, протоколы**
- **Оптимизация проигрывания VOD**
  - Переключение качества аудио/видео
  - Быстрые закладки
  - Кеширование манифеста
- **Оптимизация проигрывания Live видео**
  - DefaultLoadControl
- **Работа с проблемными кодеками**
  - Как Exoplayer выбирает кодеки
  - Отключение проблемных кодеков
  - Отключение проблемных профилей кодека
- **Продвинутые функции проигрывания**
  - Перемотка

О чем поговорим

- **ТЗ и требования, протоколы**
- **Оптимизация проигрывания VOD**
  - Переключение качества аудио/видео
  - Быстрые закладки
  - Кеширование манифеста
- **Оптимизация проигрывания Live видео**
  - DefaultLoadControl
- **Работа с проблемными кодеками**
  - Как Exoplayer выбирает кодеки
  - Отключение проблемных кодеков
  - Отключение проблемных профилей кодека
- **Продвинутые функции проигрывания**
  - Перемотка

ТЗ и требования, протоколы

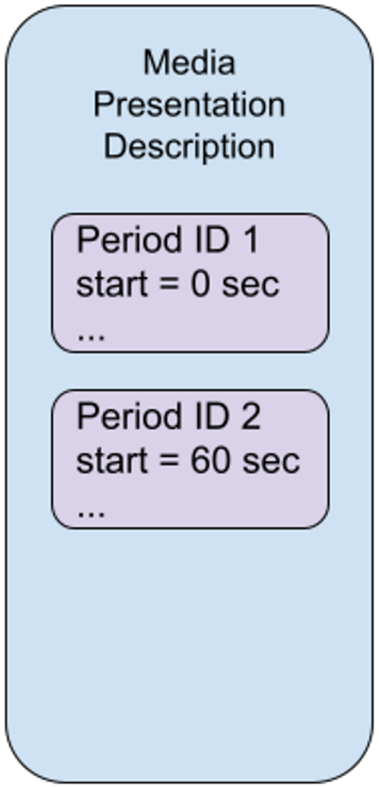


1. Отображение UI от фреймворка
2. Android плеер по запросу фреймворка
3. Управление и сбор информации с плеера
4. DASH – плейлисты
5. Live и VOD
6. DRM Widevine
7. Быстрое переключение между стримами ~ 2.3 non-encrypted, 3.2 encrypted

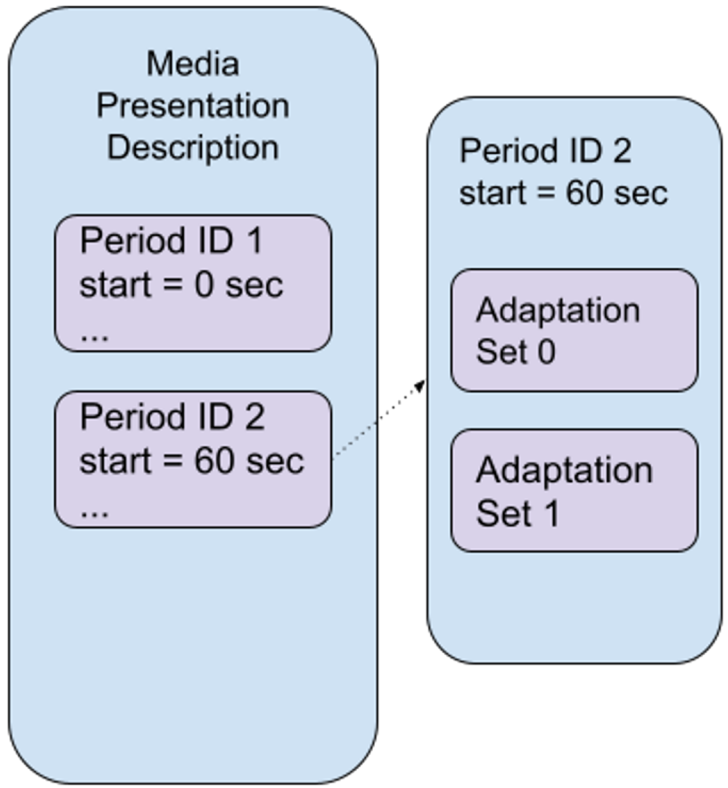
ТЗ и требования, протоколы DASH

- Преимущества:
- Работает с любыми кодеками
- Поддерживает все DRM технологии
- Поддерживает позиционирование, перемотку (быстрее чем в HLS)
- Поддержка рекламы

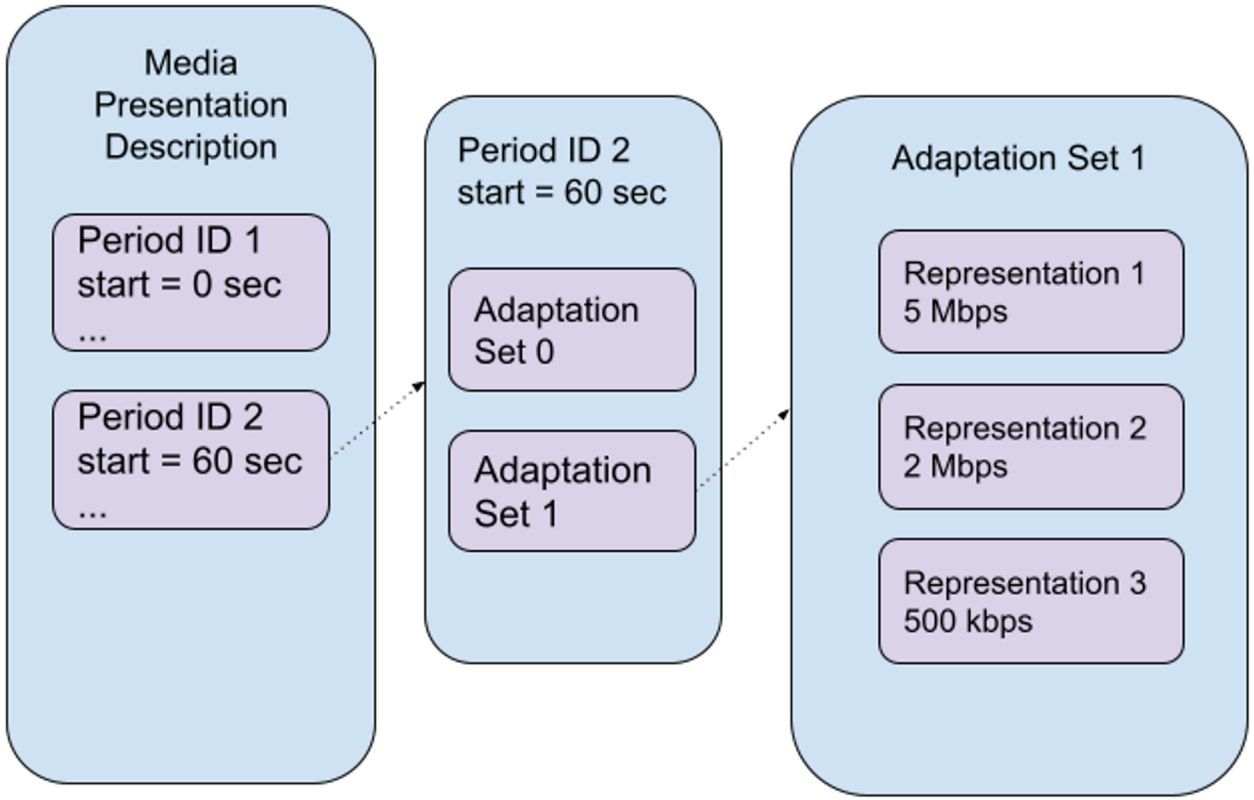
DASH
Структура манифеста



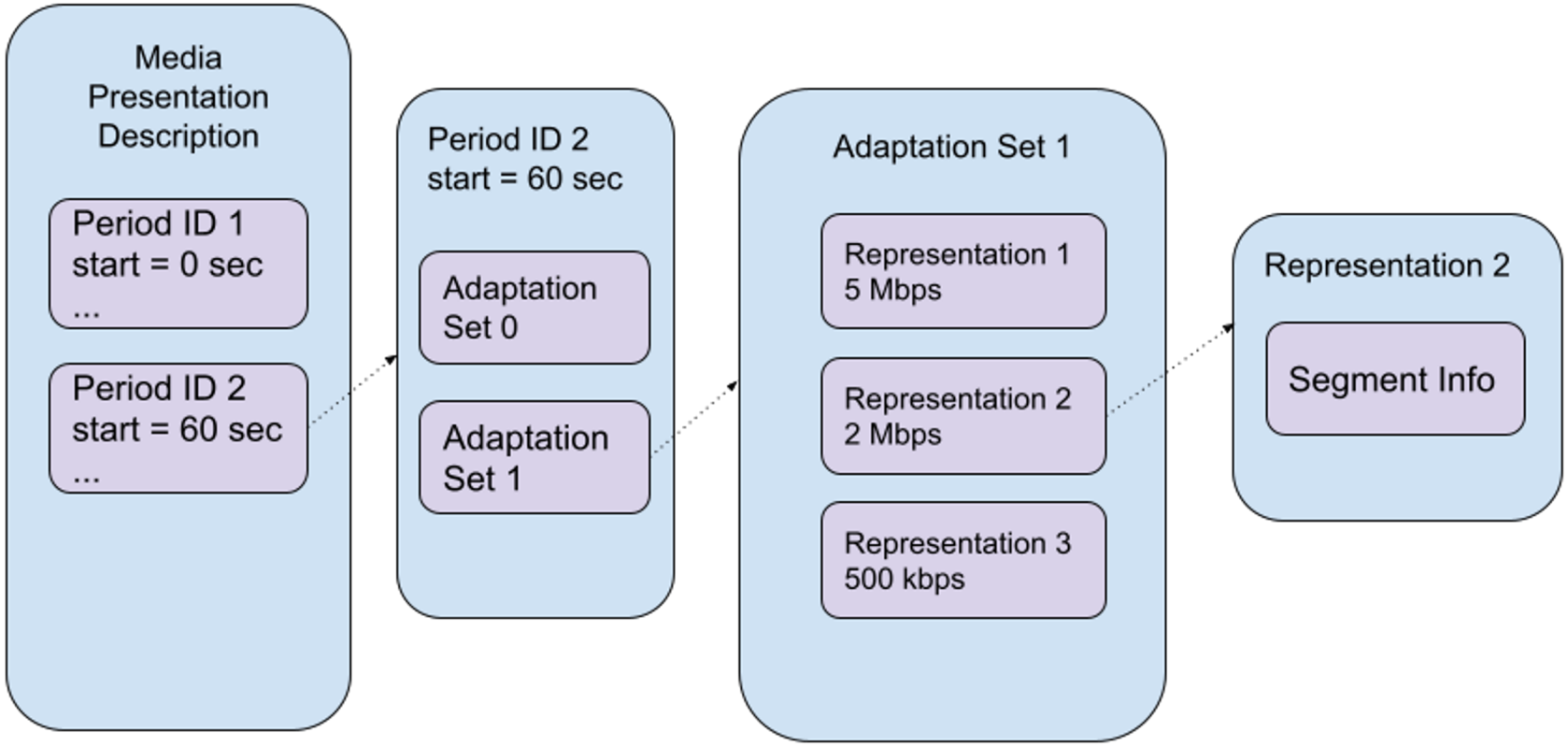
DASH
Структура манифеста



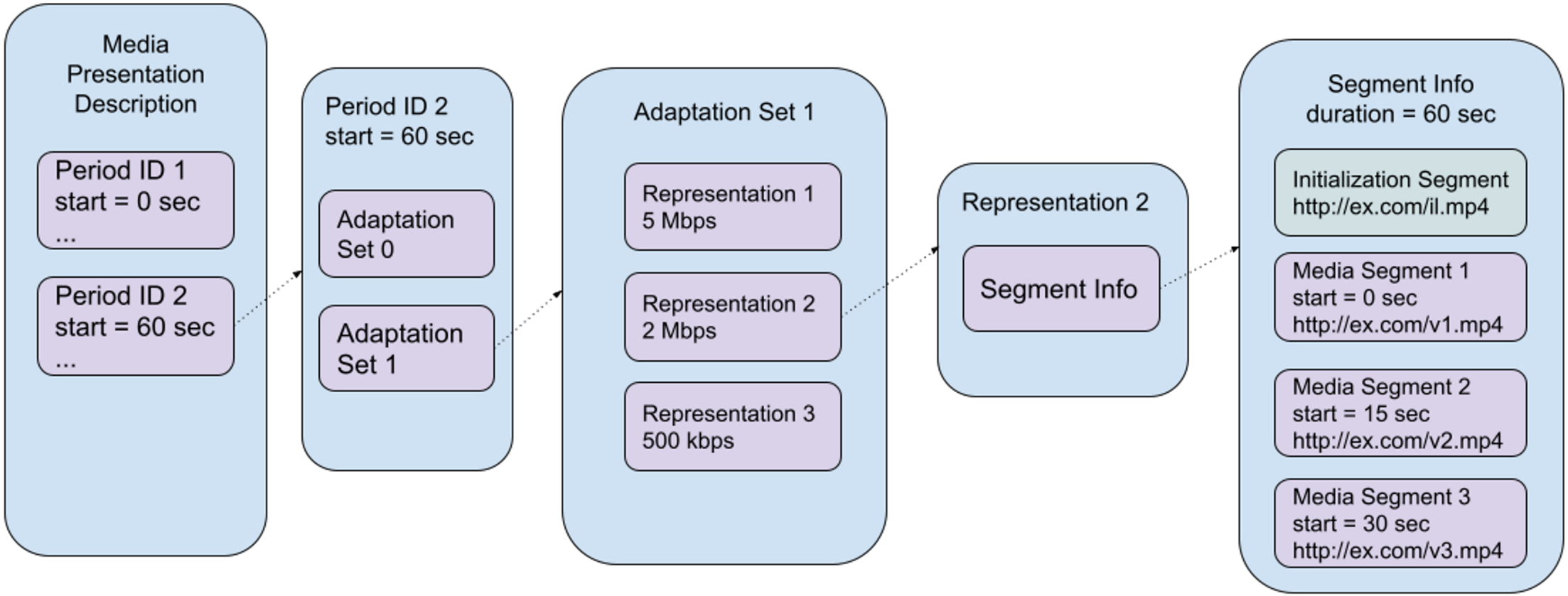
DASH
Структура манифеста



DASH
Структура манифеста



DASH
Структура манифеста



О чем поговорим

- ТЗ и требования, протоколы
- **Оптимизация проигрывания VOD**
  - **Переключение качества аудио/видео**
  - **Быстрые закладки**
  - **Кеширование манифеста**
- **Оптимизация проигрывания Live видео**
  - DefaultLoadControl
- **Работа с проблемными кодеками**
  - Как Exoplayer выбирает кодеки
  - Отключение проблемных кодеков
  - Отключение проблемных профилей кодека
- **Продвинутые функции проигрывания**
  - Перемотка

Оптимизация проигрывания VOD

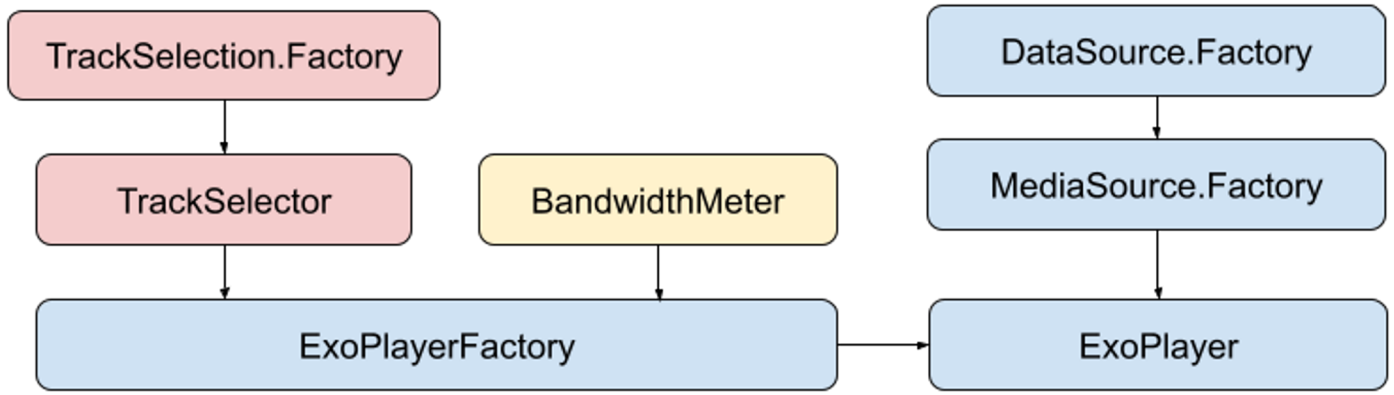
Что важно для проигрывания?

- Быстрый старт проигрывания
- Оптимальное качество с учетом сети
- Возможность сохранения текущей позиции не во вред производительности

Оптимизация проигрывания VOD

Переключение качества аудио/видео

Шаг 1. Адаптивный выбор качества.



Оптимизация проигрывания VOD

Переключение качества аудио/видео

Шаг 2. Выбор подхода

BandwidthMeter

- базируется на качестве сети
- можно выбрать качество на старте (например среднее)
- класс final

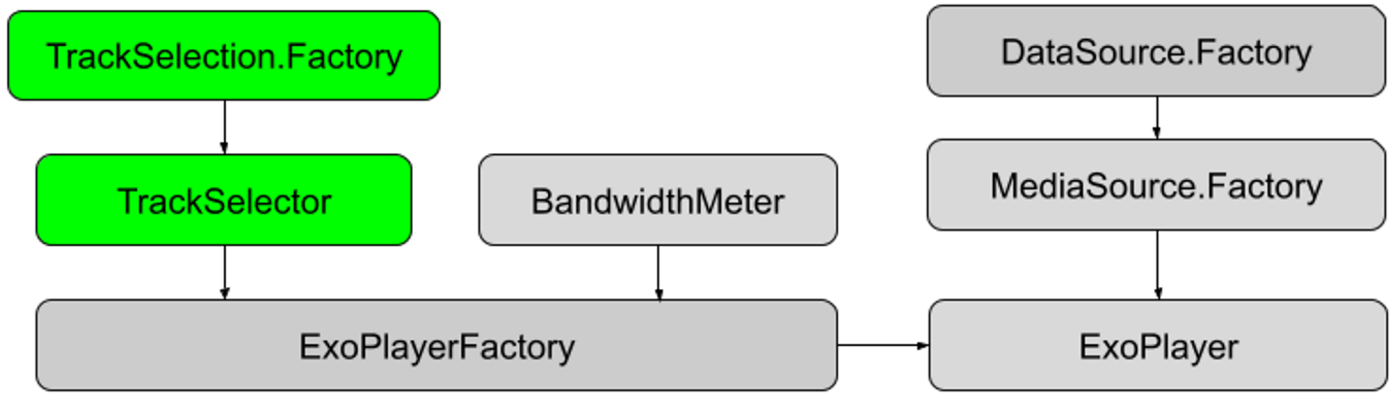
TrackSelector

- полный контроль выбора трека
- класс доступен для наследования

Оптимизация проигрывания VOD

Переключение качества аудио/видео

Шаг 2. Выбор подхода



Оптимизация проигрывания VOD

Переключение качества аудио/видео

Шаг 3. Кастомный TrackSelector

```
class ExtendedTrackSelector(
    context: Context,
    ...
    ) : DefaultTrackSelector(ExtendedAdaptiveTrackSelection.Factory(...)) {
        ...
}
```

Оптимизация проигрывания VOD

Переключение качества аудио/видео

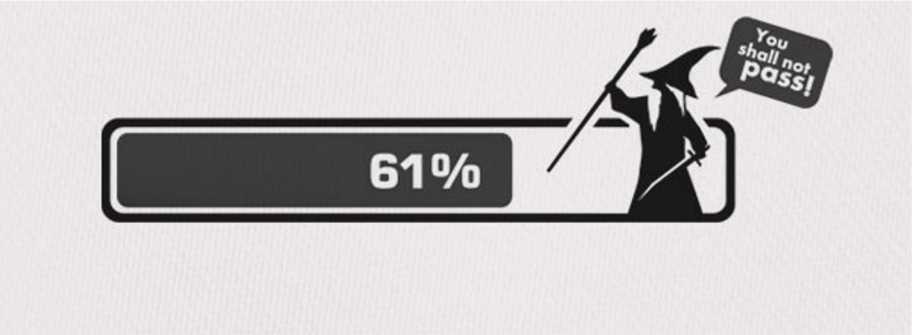
Шаг 3. Кастомный AdaptiveTrackSelection.

```
class ExtendedAdaptiveTrackSelection private constructor(...)
                    : AdaptiveTrackSelection(...) {
    var useMinBitrate: Boolean = true

    override fun getSelectedIndex(): Int =
        if (useMinBitrate) {
            (0 until length).minByOrNull { getFormat(it).bitrate }
        } else { super.getSelectedIndex() }
    ...
}
```

Оптимизация проигрывания VOD

Кеширование манифеста

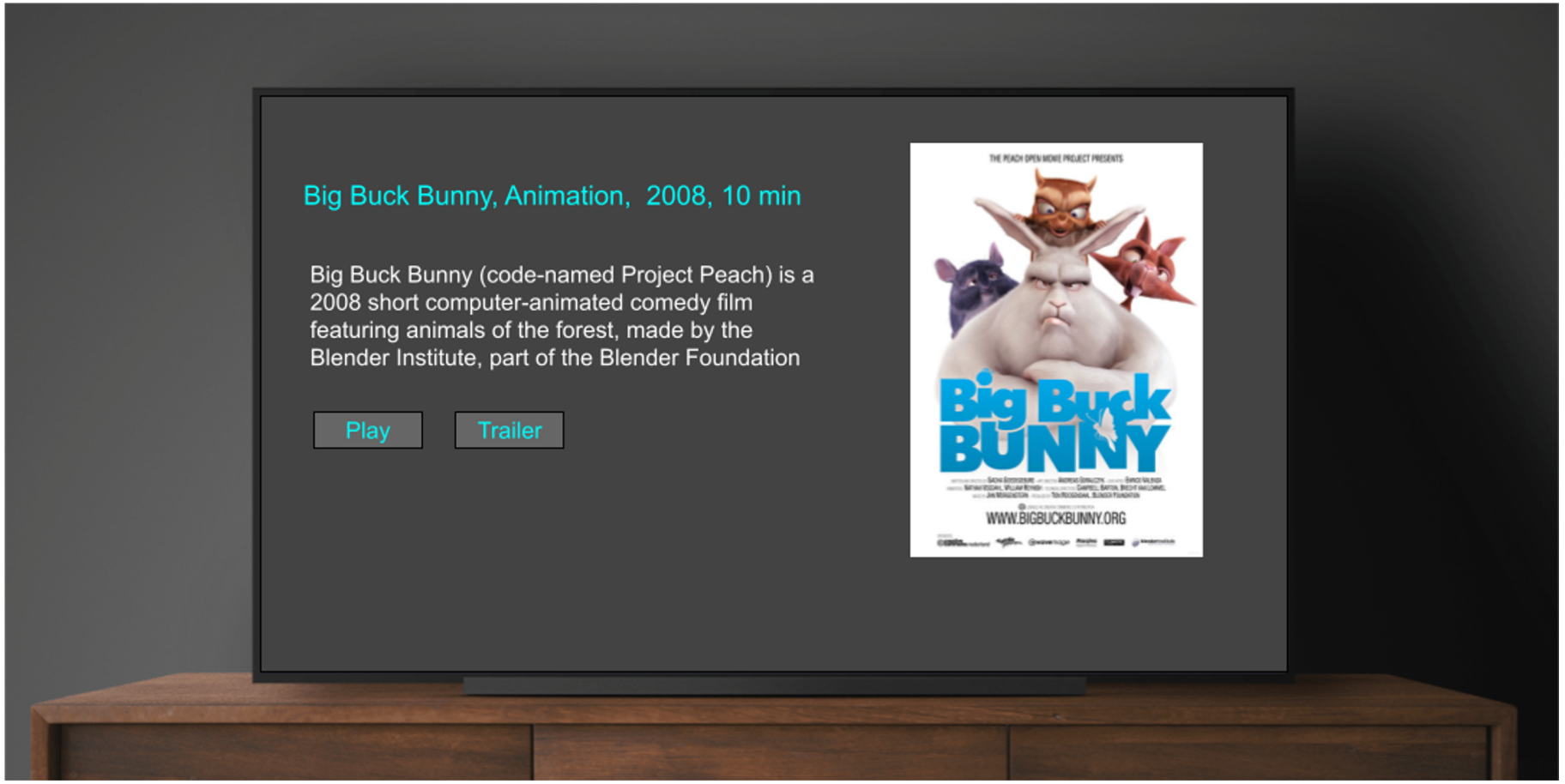
Загрузка манифеста ~ 1,5 секунд



Когда мы можем закешировать манифест?

- Он НЕ динамический
- Если плейлист DASH.

Оптимизация проигрывания VOD
Кеширование манифеста

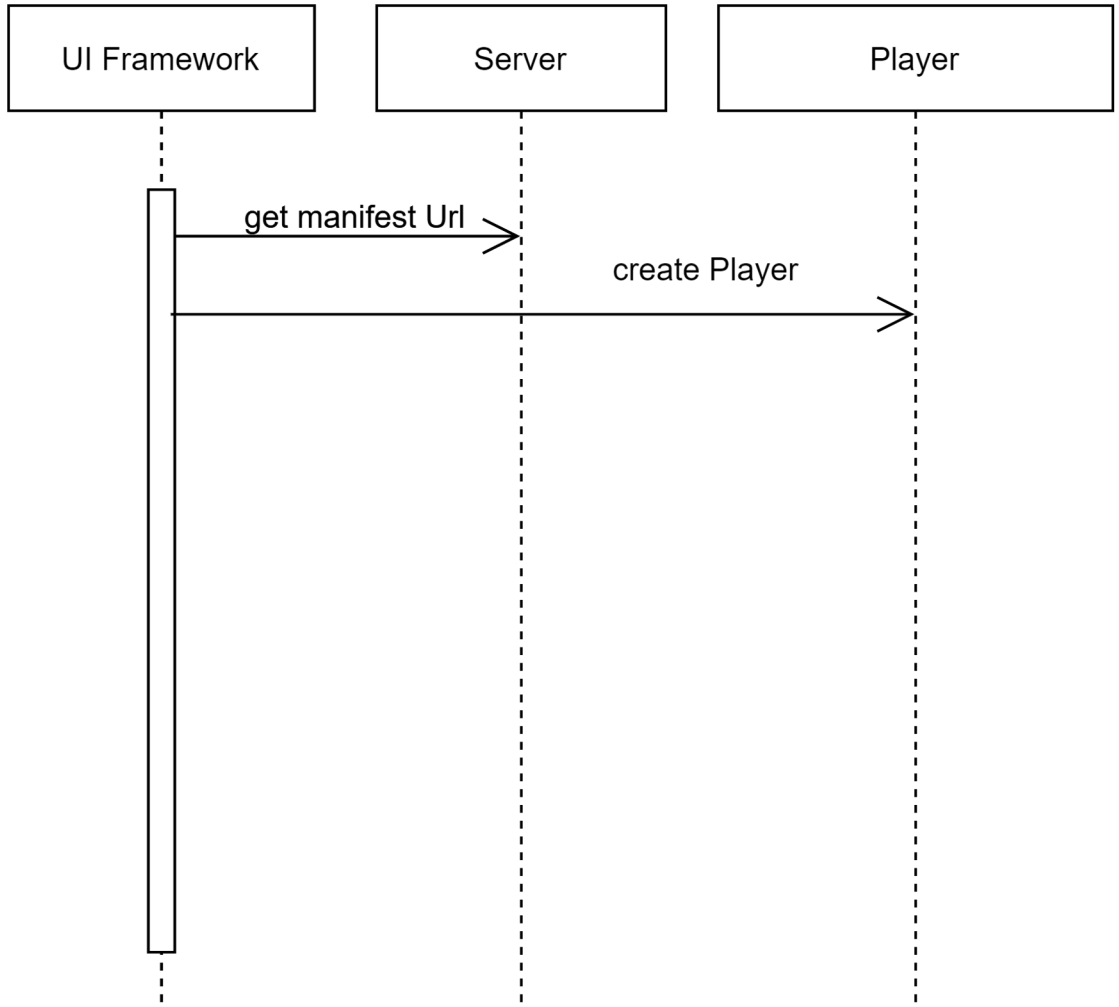


Оптимизация проигрывания VOD
Кеширование манифеста

Оптимизация проигрывания VOD

Кеширование манифеста

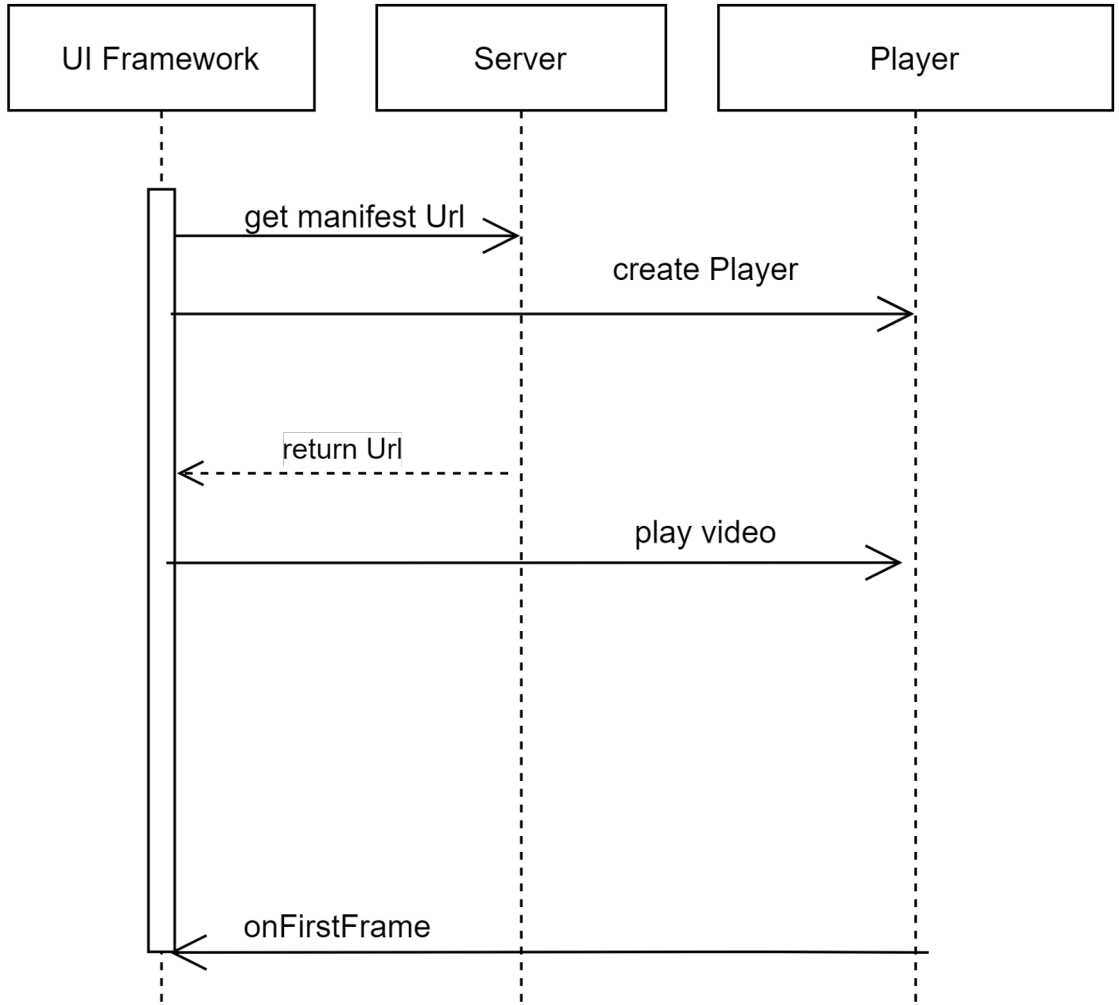
Шаг 1. Концепт



Оптимизация проигрывания VOD

Кеширование манифеста

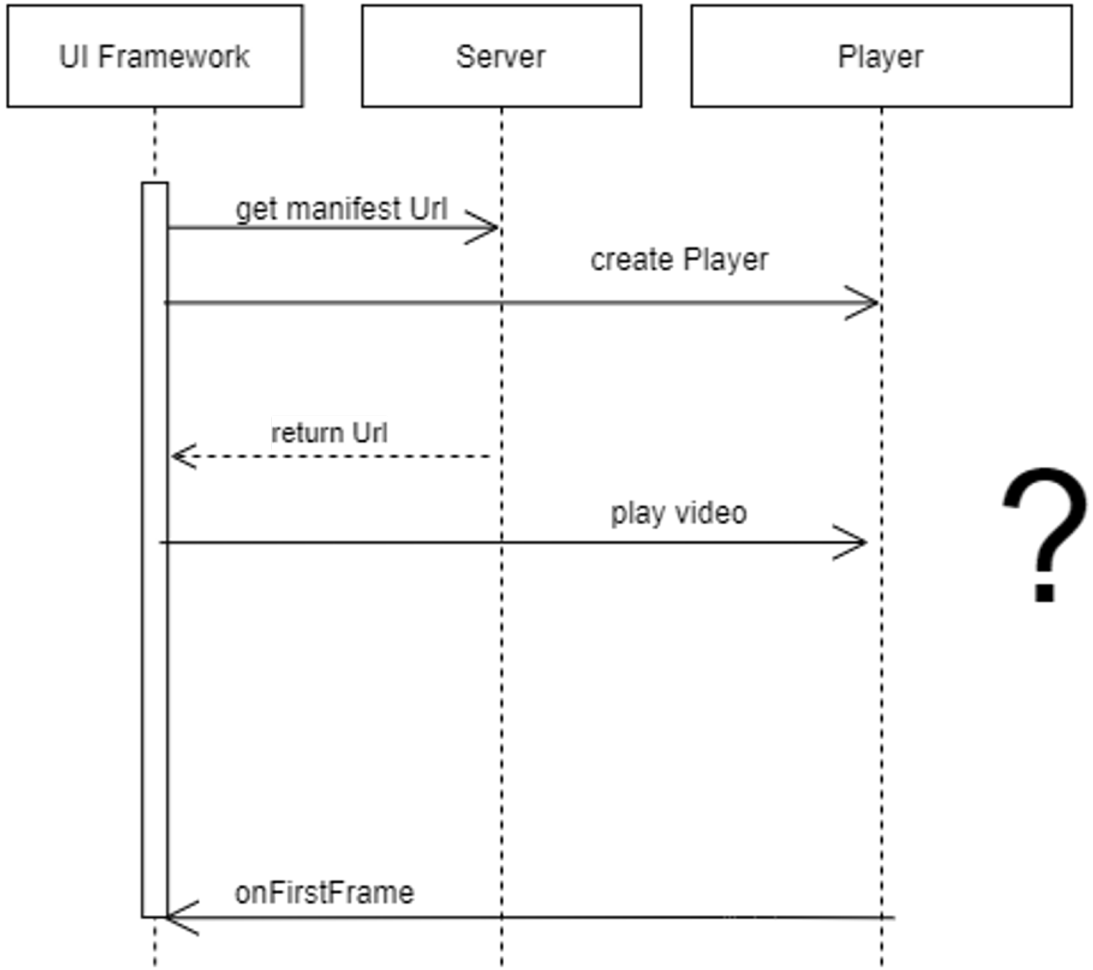
Шаг 1. Концепт



Оптимизация проигрывания VOD

Кеширование манифеста

Шаг 1. Концепт



?

Loader

```
if (no cache manifest)
        loadManifest
else
        playWithManifest
```

Оптимизация проигрывания VOD

Кеширование манифеста

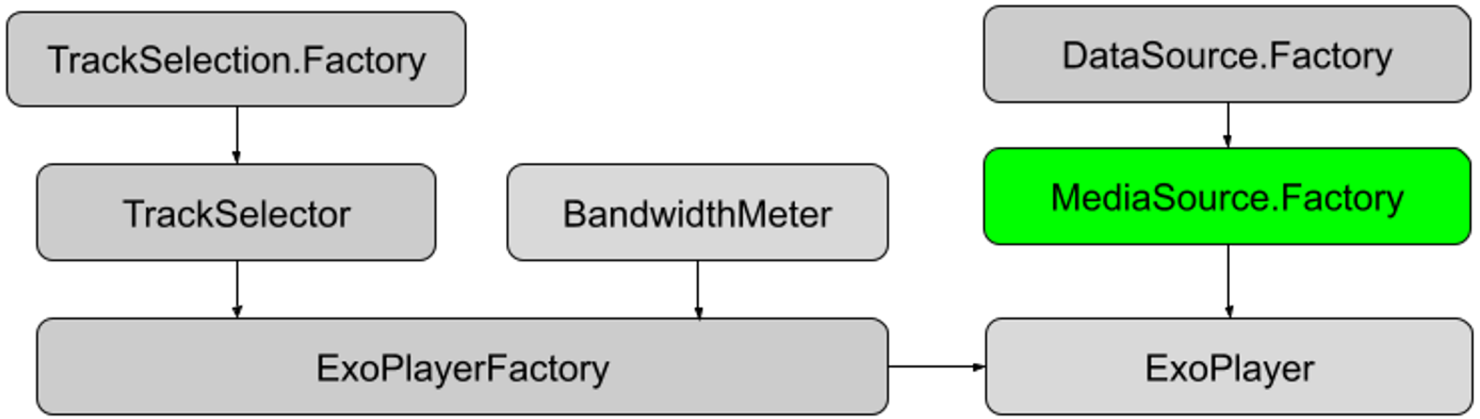
Шаг 2. Создадим Loader

```
class ManifestLoader() : MediaSourceFactorySet.ManifestCache,
ManifestCacheInterface {

    private val manifests = SparseArray<DashManifest>()

    override fun download(uri: String): Int {
        val manifest: DashManifest
        ...
        manifest = DashUtil.loadManifest(dataSourceFactory.createDataSource(),
                                                        Uri.parse(uri))
        ...

        manifests.put(nextID, manifest)
        return nextID++
    }
}
```

Оптимизация проигрывания VOD

Кеширование манифеста

Шаг 3. Создадим свою MediaSourceFactory



Оптимизация проигрывания VOD

Кеширование манифеста

Шаг 3. Создадим свою MediaSourceFactory

```
/**
 * It is created by analogy with
 * {@link com.google.android.exoplayer2.source.DefaultMediaSourceFactory}
 */
class MediaSourceFactorySet(val manifestCache: ManifestCache)
: MediaSourceFactory {

    interface ManifestCache {
        fun getCachedManifest(id: Int?): DashManifest?
    }
}
```

Оптимизация проигрывания VOD

Кеширование манифеста

Шаг 4. Построим связь между Loader и MediaSourceFactory

```
class ManifestLoader() : MediaSourceFactorySet.ManifestCache,
                                    ManifestCacheInterface {
    private val manifests = SparseArray<DashManifest>()

    override fun getCachedManifest(id: Int?): DashManifest? =
        if (id != null) manifests[id] else null
}
```

Оптимизация проигрывания VOD

Кеширование манифеста

Шаг 5. Как начать проигрывание с кэшированным манифестом?

```
package com.google.android.exoplayer2.source.dash.DashMediaSource;

/**
 * Returns a new {@link DashMediaSource} using the current parameters and the
 * specified sideloaded manifest.
 *
 * @param manifest The manifest. {@link DashManifest#dynamic} must be false.
 * @param mediaItem The {@link MediaItem} to be included in the timeline.
 * @return The new {@link DashMediaSource}.
 * @throws IllegalArgumentException If {@link DashManifest#dynamic} is true.
 */
public DashMediaSource createMediaSource(DashManifest manifest,
                                         MediaItem mediaItem)
```

Оптимизация проигрывания VOD

Кеширование манифеста

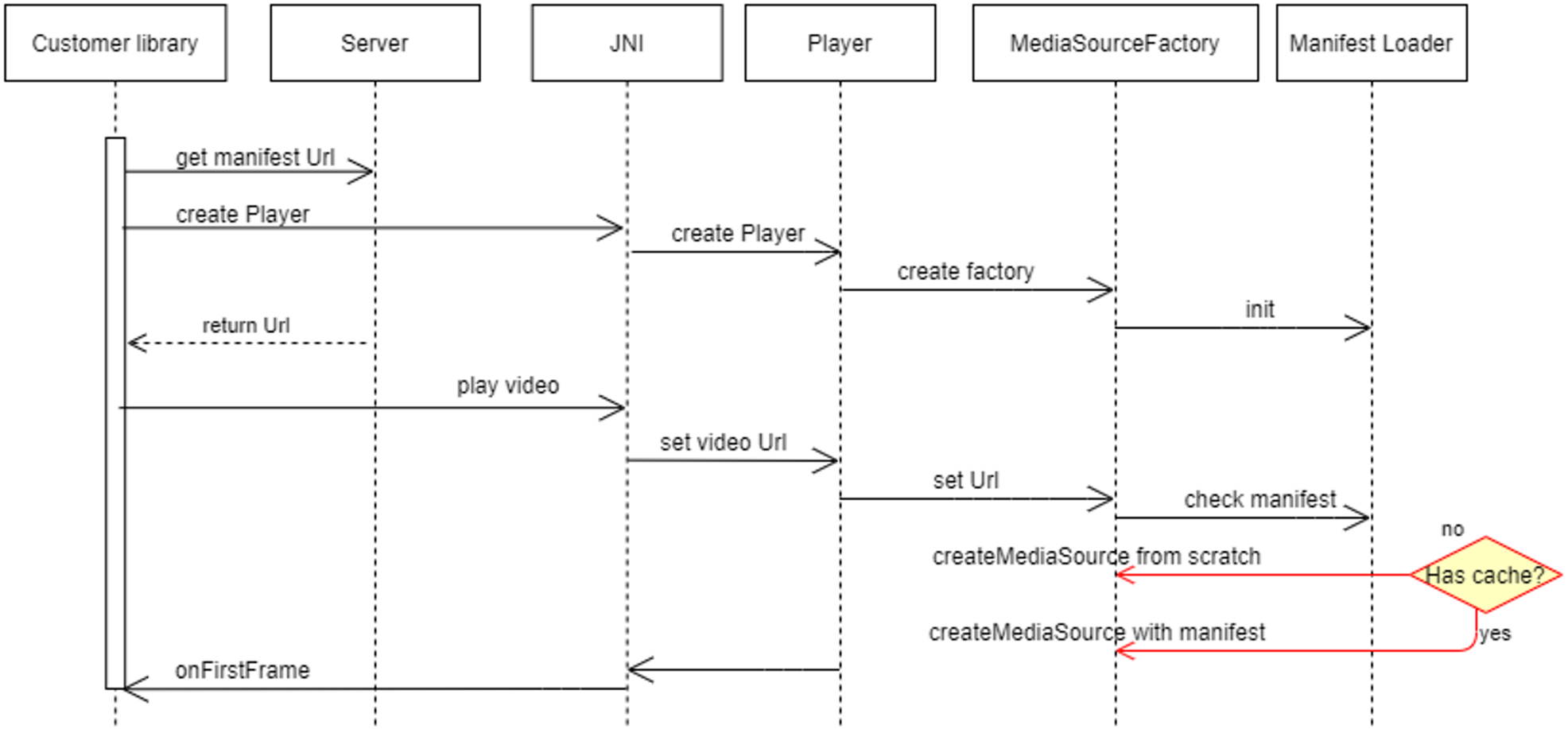
Шаг 6. Реализуем проигрывание с загруженным ранее манифестом

```
override fun createMediaSource(mediaItem: MediaItem): MediaSource {
    val properties = mediaItem.playbackProperties!!
    val uri = properties.uri
    if (uri.scheme == CACHE) {
        val factory = mediaSourceFactories.get(C.TYPE_DASH) as? DashMediaSource.Factory
        val manifest = manifestCache.getCachedManifest(uri.host?.toIntOrNull())
        if (manifest != null && factory != null) {
            return factory.createMediaSource(manifest, mediaItem)
        }
    }
    …
    return factory.createMediaSource(mediaItem)
}
```

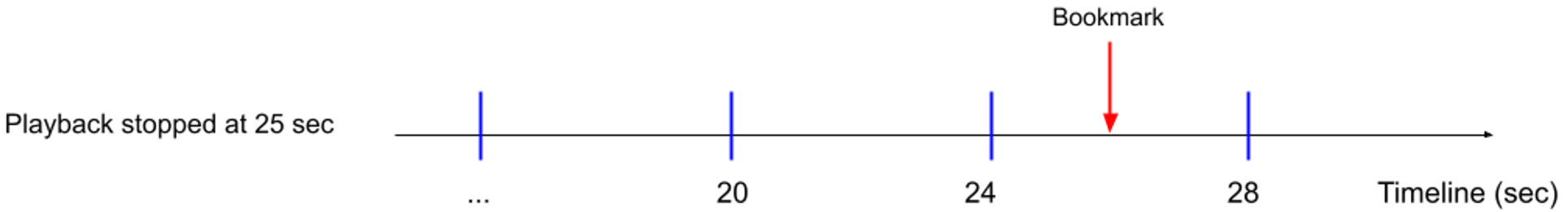
Оптимизация проигрывания VOD

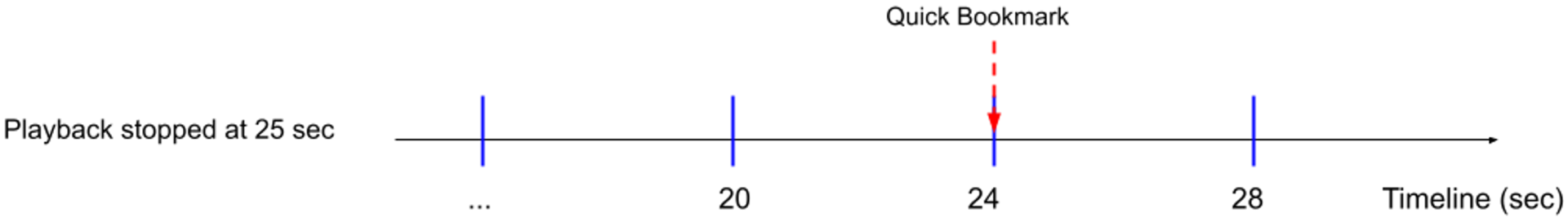
Кеширование манифеста

Итоговый Flow



Оптимизация проигрывания VOD
Быстрые закладки





Оптимизация проигрывания VOD
Быстрые закладки

Берем представление сегмента через VideoFormat

```
private fun getSegmentRepresentation(): Representation.MultiSegmentRepresentation? {
        currentVideoFormat?.also { videoFormat ->
            (currentManifest as? DashManifest)?
                        .getPeriod(currentPeriodIndex)?.also {
                for (adaptationSet in it.adaptationSets) {
                    if (adaptationSet.type == C.TRACK_TYPE_VIDEO) {
                        for (represenation in adaptationSet.representations) {
                            if (represenation.format.id == videoFormat.id) {
                                if (represenation
                                        is Representation.MultiSegmentRepresentation) {
                                        return represenation
                                }
        ...
        return null
}
```

Оптимизация проигрывания VOD
Быстрые закладки

Даем доступ к размеру сегмента

```
fun getSegmentDuration(): Long {
        ...
        val segmentRepresentation = getSegmentRepresentation()
        ...
        val periodPositionUs =
                currentTimeline.getPeriodPosition(..., C.msToUs(currentPosition))
      val num =
                segmentRepresentation.getSegmentNum(periodPositionUs.second,...)
        return C.usToMs(segmentRepresentation.getDurationUs(num, ...))
    }
}
```

Оптимизация проигрывания VOD
Быстрые закладки

Даем доступ ко времени начала сегмента

```
fun getSegmentStartTime(): Long {
        ...
        val w = currentTimeline.getWindow(...)
        val segmentRepresentation = getSegmentRepresentation()
        ...
        val periodPositionUs = currentTimeline
                                .getPeriodPosition(..., C.msToUs(currentPosition))
       val num = segmentRepresentation.getSegmentNum(periodPositionUs.second, ...)
        return C.usToMs(segmentRepresentation.getTimeUs(num))
}
```

Оптимизация проигрывания VOD

Подведем итог

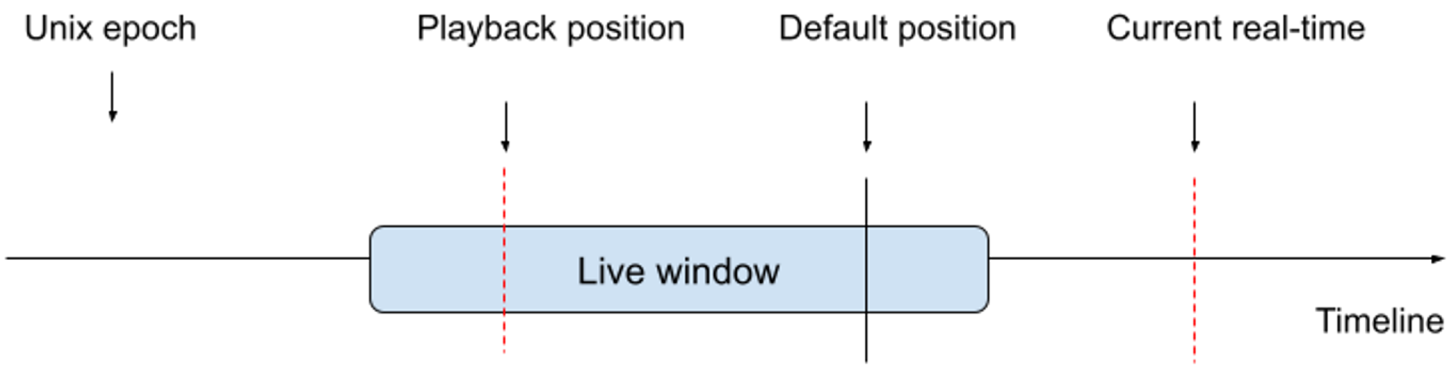
Для быстрого старта VOD есть не сложные и не трудоемкие способы:

- TrackSelector и выбор минимального качества на старте
- Кеширование манифеста
- Закладки только по началу чанка

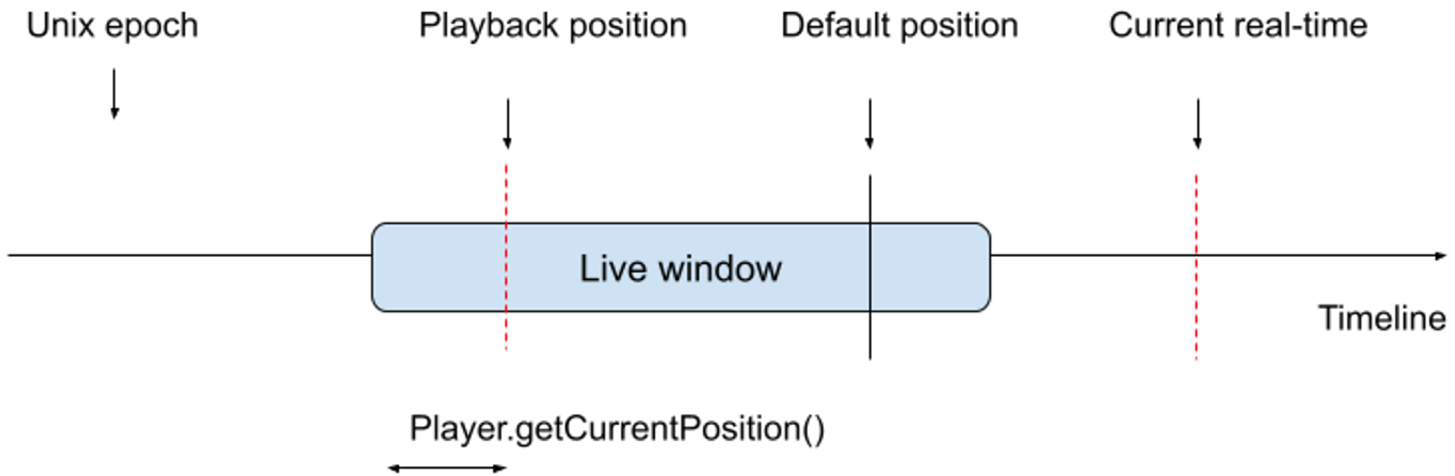
О чем поговорим

- ТЗ и требования, протоколы
- Оптимизация проигрывания VOD
  - Переключение качества аудио/видео
  - Быстрые закладки
  - Кеширование манифеста
- **Оптимизация проигрывания Live видео**
  - **DefaultLoadControl**
- **Работа с проблемными кодеками**
  - Как Exoplayer выбирает кодеки
  - Отключение проблемных кодеков
  - Отключение проблемных профилей кодека
- **Продвинутые функции проигрывания**
  - Перемотка

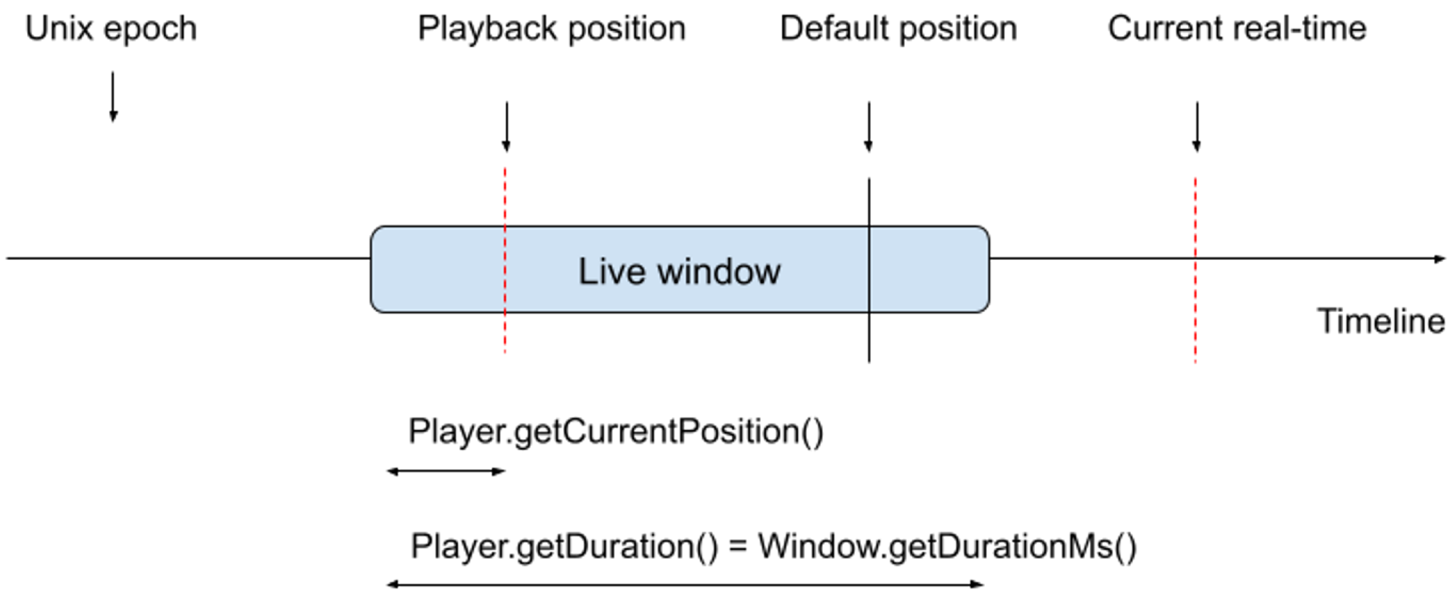
Оптимизация проигрывания Live видео
Особенности Live стрима



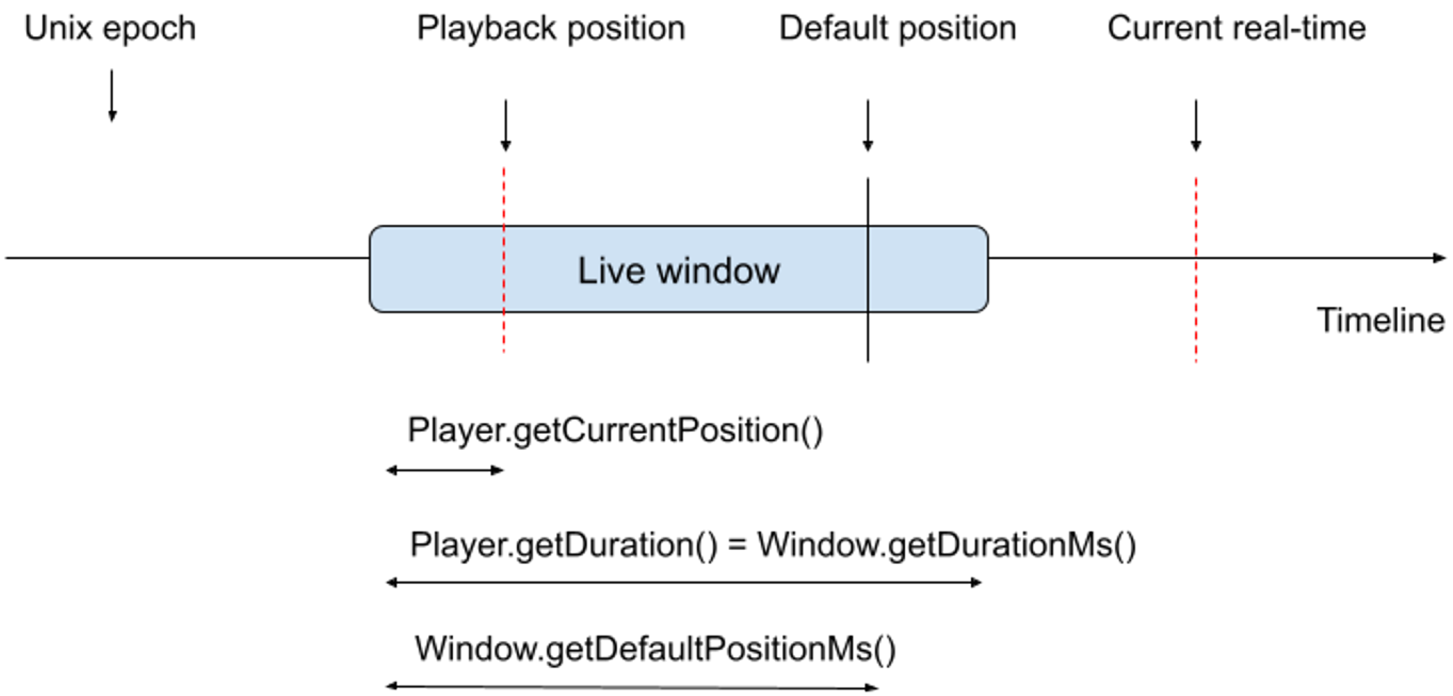
Оптимизация проигрывания Live видео
Особенности Live стрима



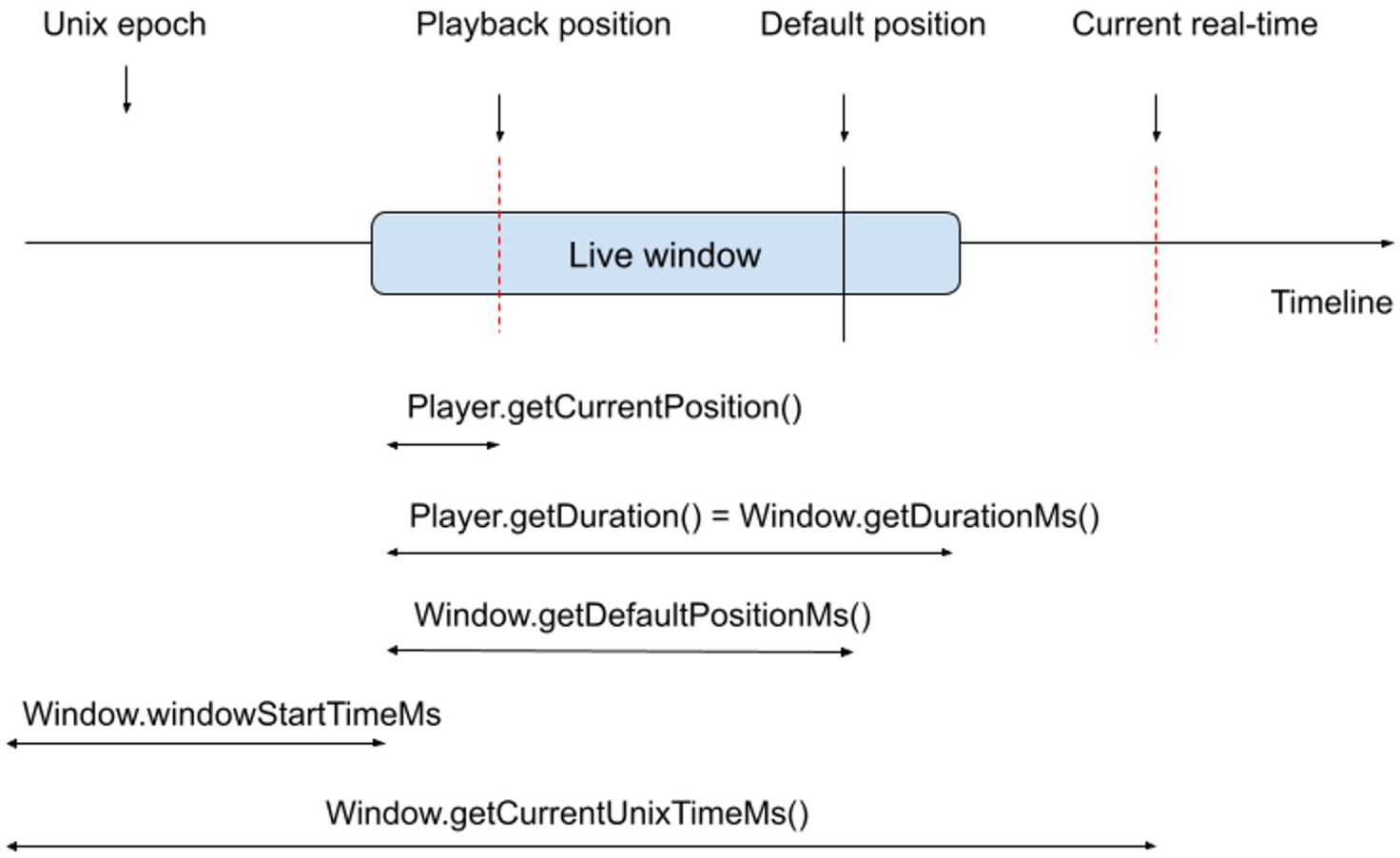
Оптимизация проигрывания Live видео
Особенности Live стрима



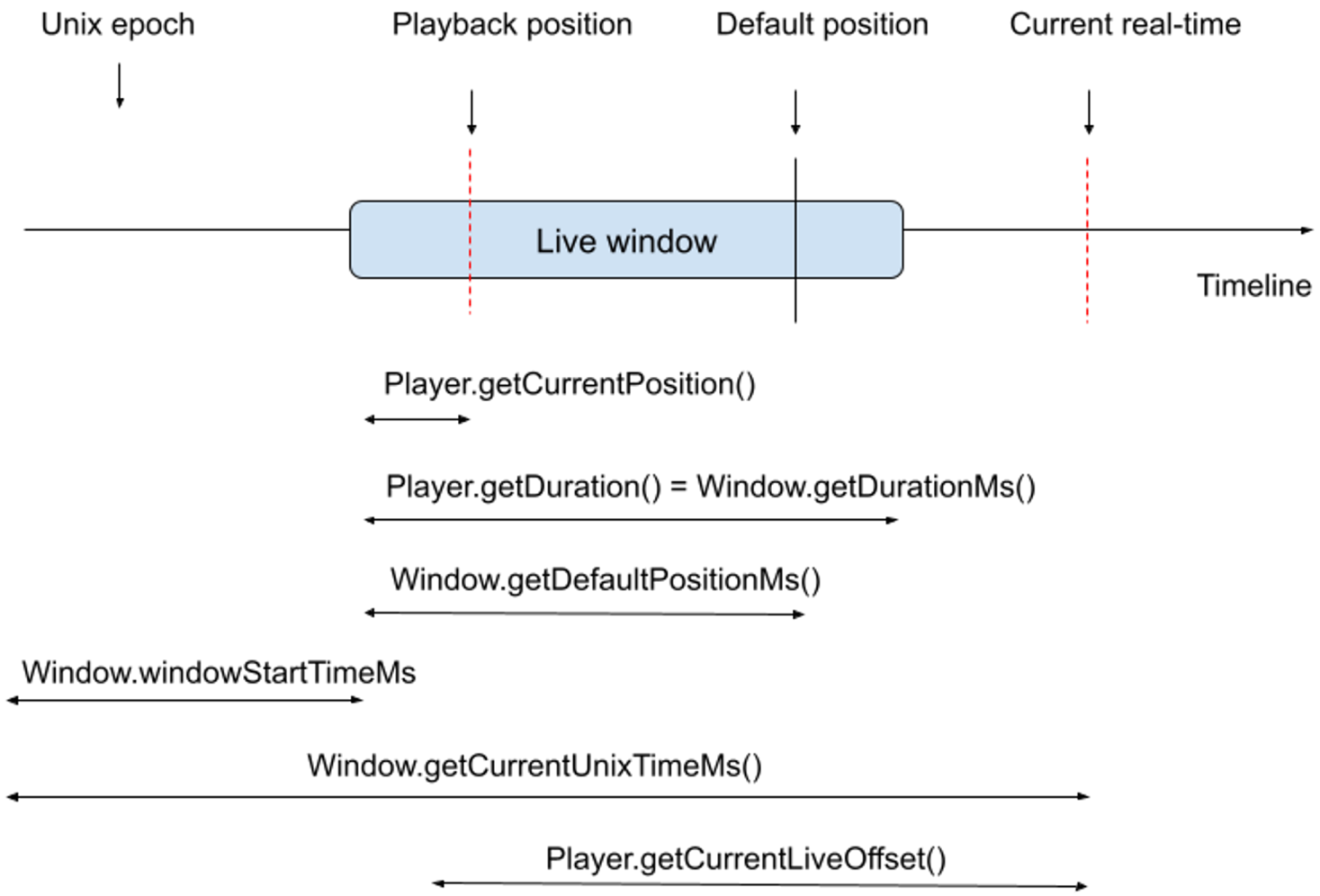
Оптимизация проигрывания Live видео
Особенности Live стрима



Оптимизация проигрывания Live видео
Особенности Live стрима



Оптимизация проигрывания Live видео
Особенности Live стрима



Оптимизация проигрывания Live видео
Особенности Live стрима

- Нет duration
- Динамический манифест
- Могут быть проблемы с буфферизацией
- Важно не отставать от лайв точки даже при изменениях сети

Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl


```
public class DefaultLoadControl implements LoadControl {
  public static final int DEFAULT_MIN_BUFFER_MS = 15_000;
  public static final int DEFAULT_MAX_BUFFER_MS = 50_000;
}
```

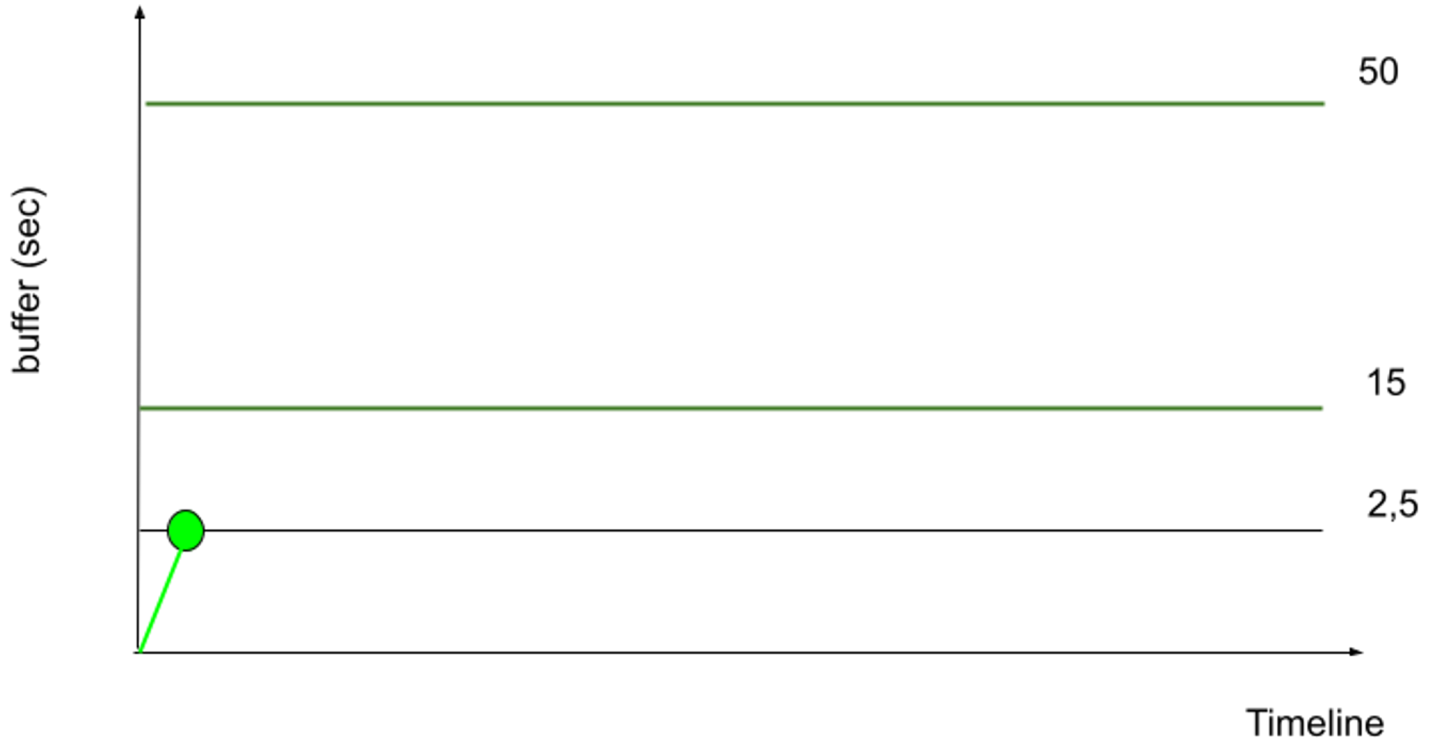
Позволяет задать мин и макс продолжительность буффера в мс,
который должен быть заполнен данными прежде чем скачивать новую порцию данных

Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl

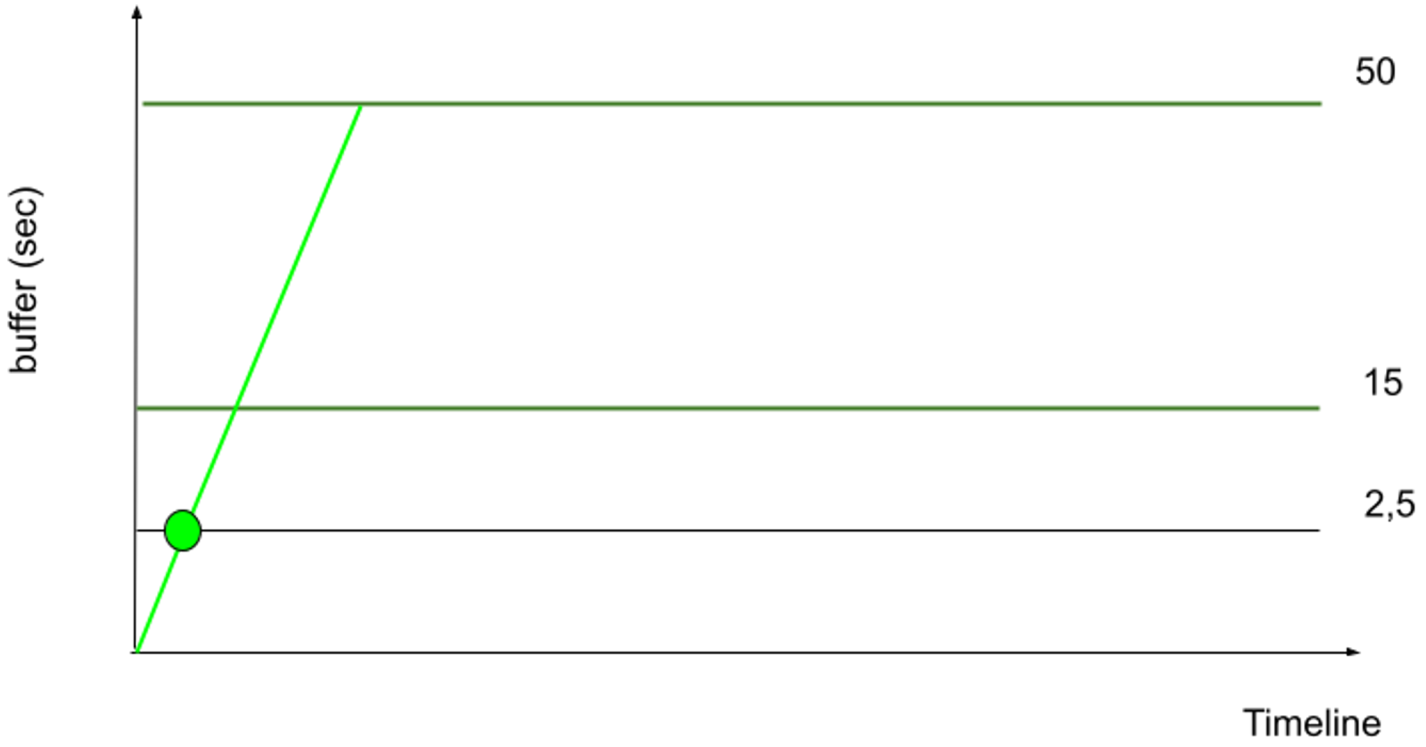

```
public class DefaultLoadControl implements LoadControl {

  public static final int DEFAULT_BUFFER_FOR_PLAYBACK_MS = 2500;
  …
}
```

Задает продолжительность видео, которая должна быть забуфферизирована перед проигрыванием в результате действий пользователя (пауза-проигрывание или seek)

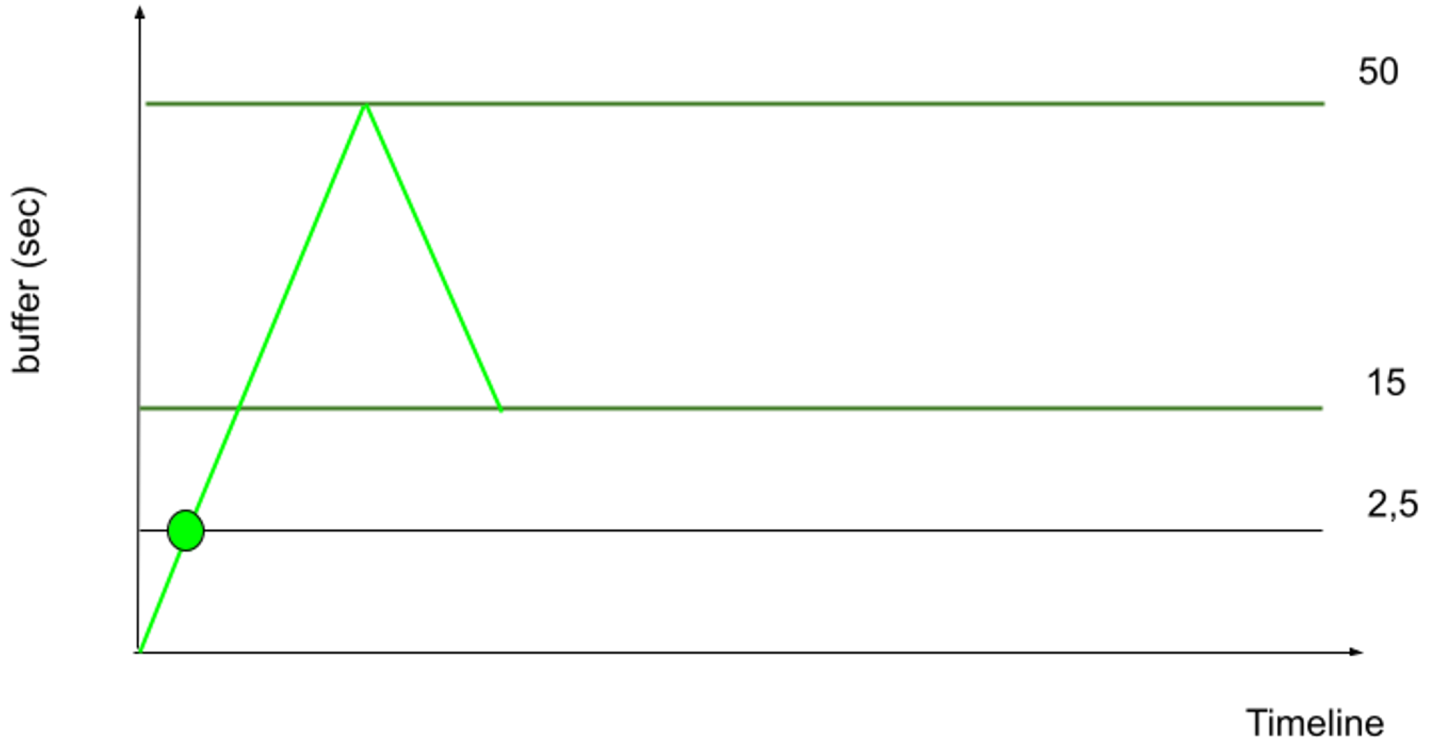
Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl



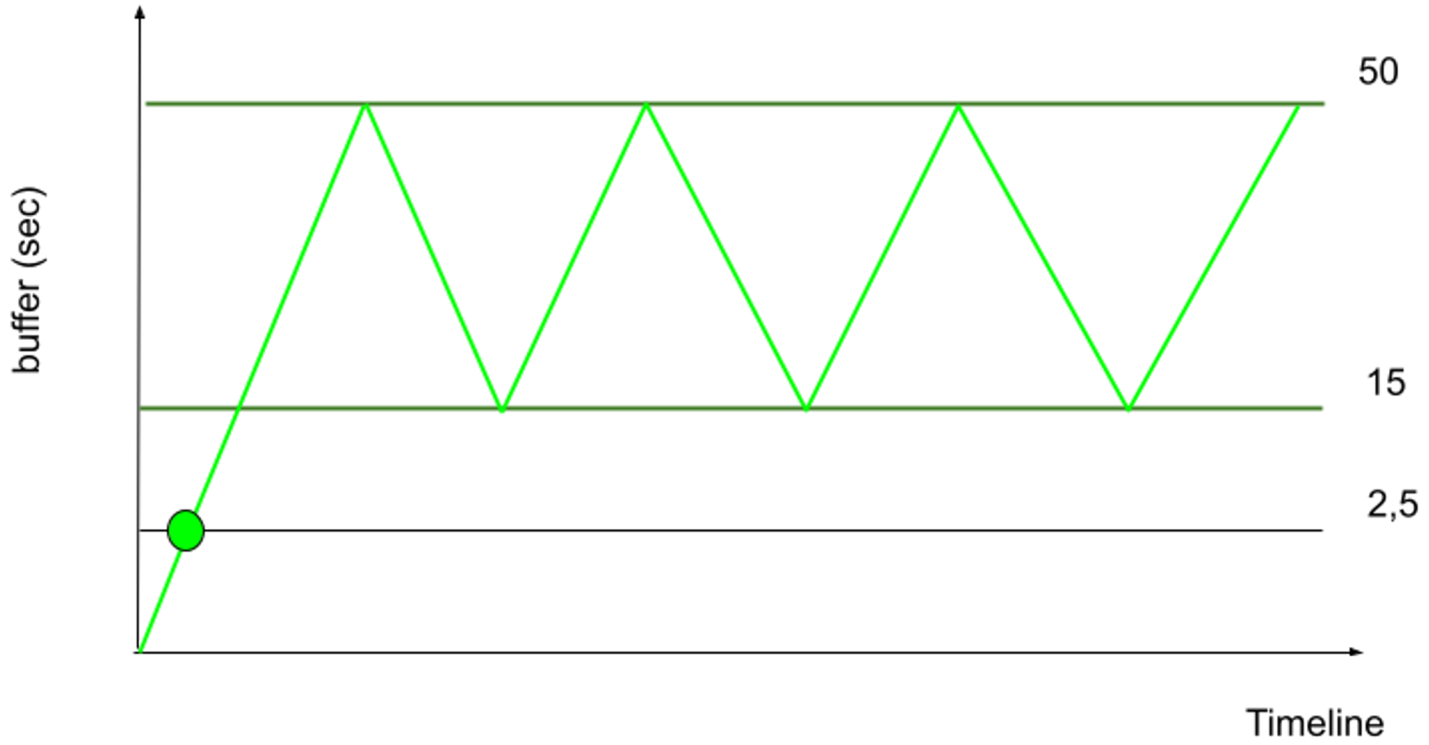
Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl



Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl



Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl



Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl


```
public class DefaultLoadControl implements LoadControl {

  public static final int DEFAULT_BUFFER_FOR_PLAYBACK_AFTER_REBUFFER_MS = 5000;
  …
}
```

Задает продолжительность видео, которая должна быть забуфферизирована перед проигрыванием в результате очистки буффера (повторной буфферизации)

Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl

```
public static final int MIN_BUFFER_MS = 15_000;

public static final int MAX_BUFFER_MS = 50_000;

public static final int BUFFER_FOR_PLAYBACK_MS = 2500;

public static final int BUFFER_FOR_PLAYBACK_AFTER_REBUFFER_MS = 5000;
```

Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl

```
public static final int MIN_BUFFER_MS = 15_00050_000;

public static final int MAX_BUFFER_MS = 50_000;

public static final int BUFFER_FOR_PLAYBACK_MS = 25001000;

public static final int BUFFER_FOR_PLAYBACK_AFTER_REBUFFER_MS = 50001000;
```

Оптимизация проигрывания Live видео
Буфферизация и DefaultLoadControl

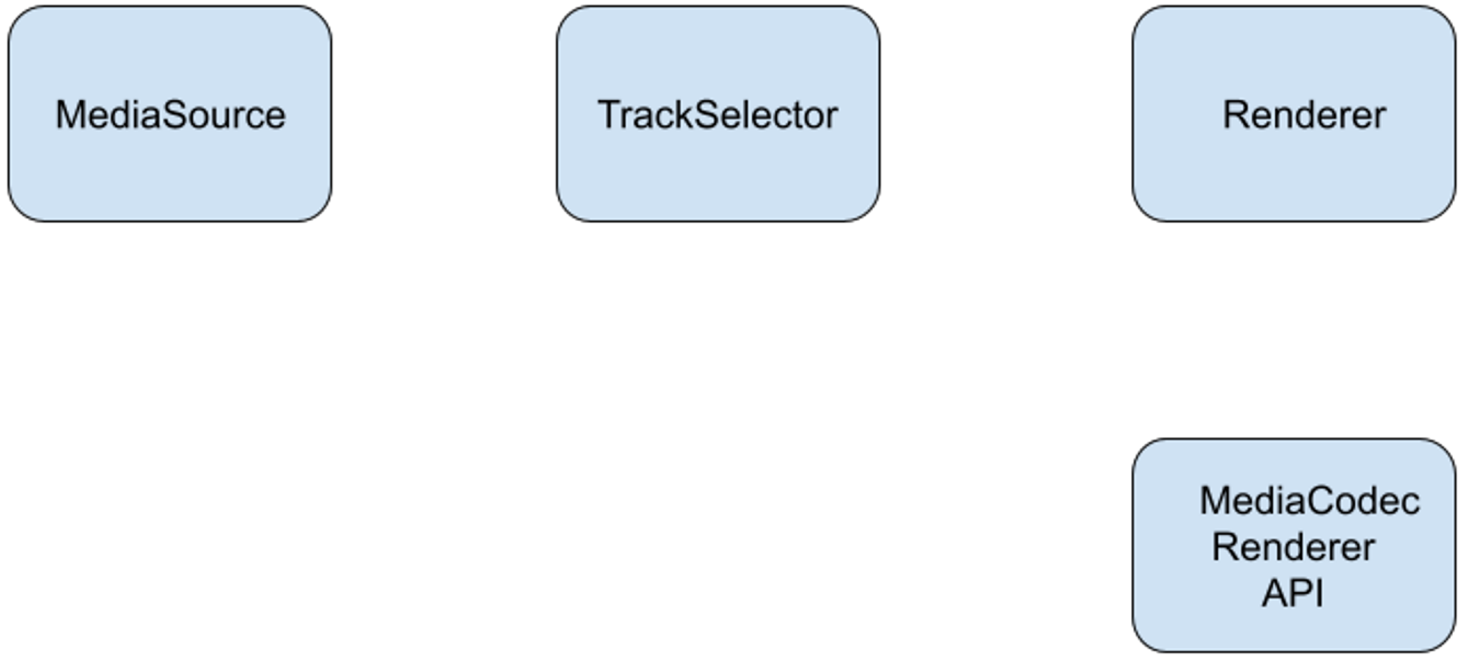
Итог анализа и подбора параметров:

- Не рекомендуется использовать разные значения minBufferMs и maxBufferMs
- Уменьшить значение bufferForPlaybackMs опытным путем. Слишком заниженное значение увеличит риск повторной буфферизации

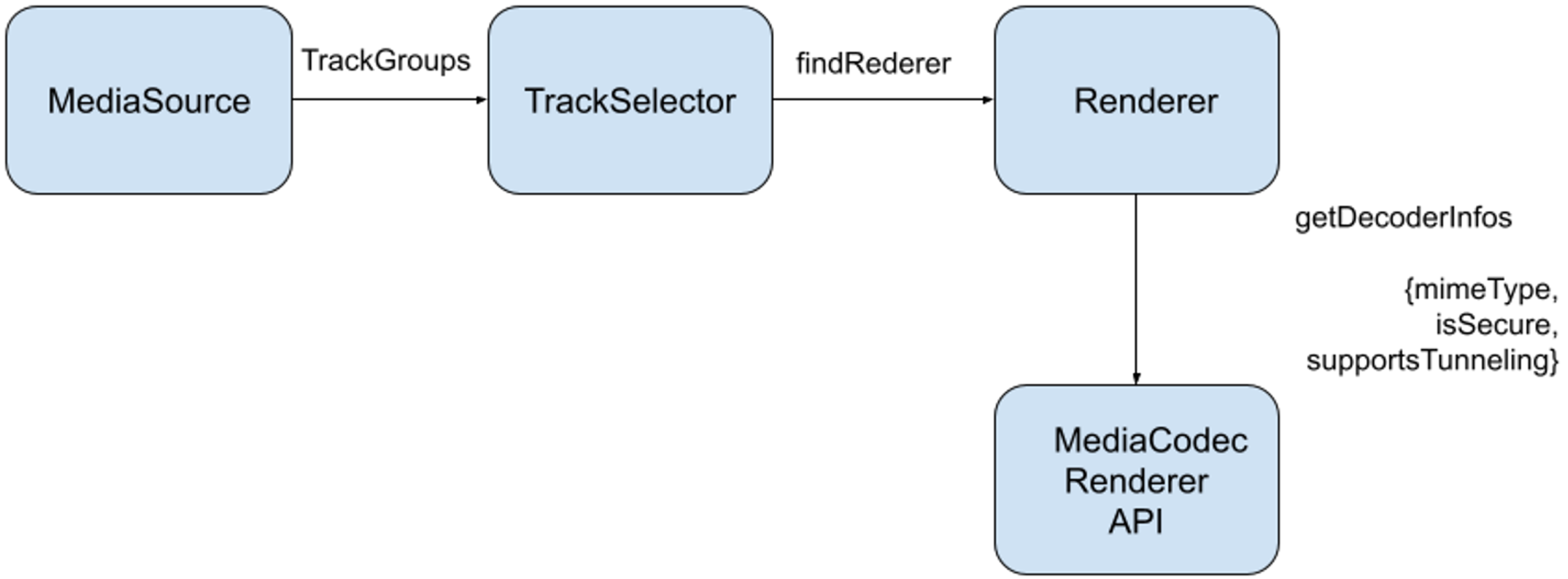
О чем поговорим

- ТЗ и требования, протоколы
- Оптимизация проигрывания VOD
  - Переключение качества аудио/видео
  - Быстрые закладки
  - Кеширование манифеста
- Оптимизация проигрывания Live видео
  - DefaultLoadControl
- **Работа с проблемными кодеками**
  - **Как Exoplayer выбирает кодеки**
  - **Отключение проблемных кодеков**
  - **Отключение проблемных профилей кодека**
- **Продвинутые функции проигрывания**
  - Перемотка

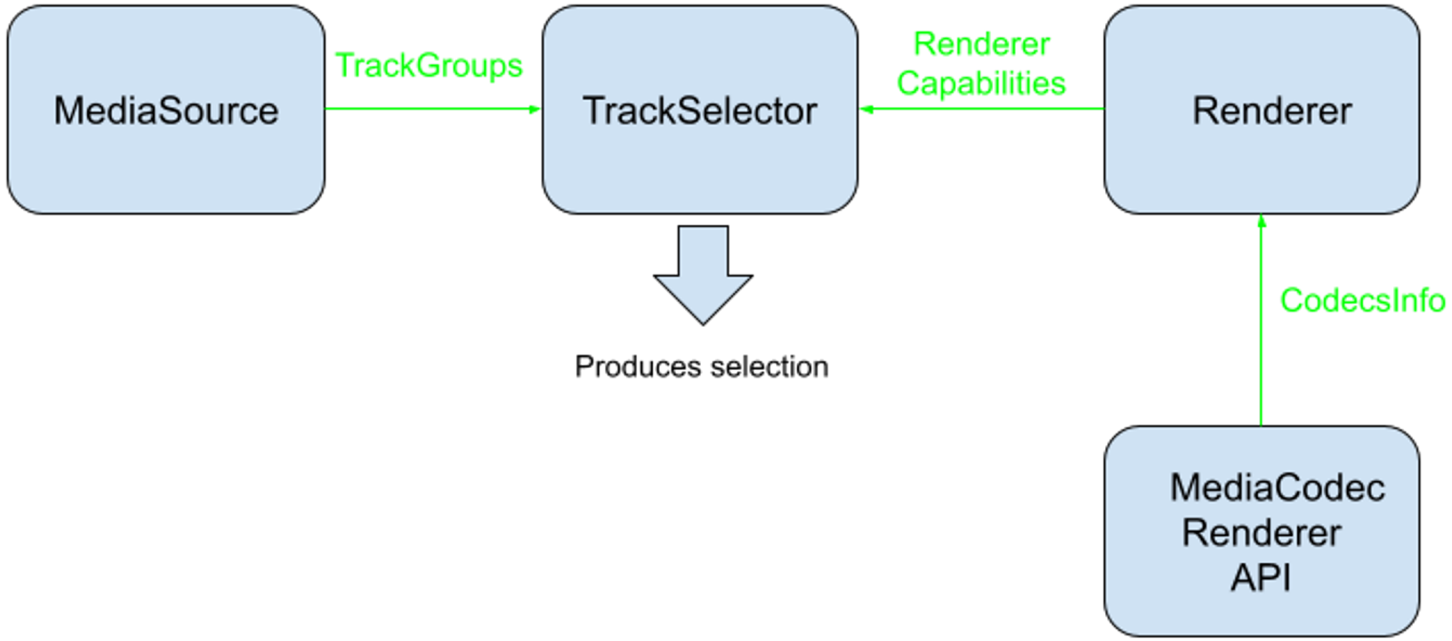
Работа с проблемными кодеками
Как Exoplayer выбирает кодеки



Работа с проблемными кодеками
Как Exoplayer выбирает кодеки



Работа с проблемными кодеками
Как Exoplayer выбирает кодеки



Работа с проблемными кодеками
Отключение проблемных кодеков

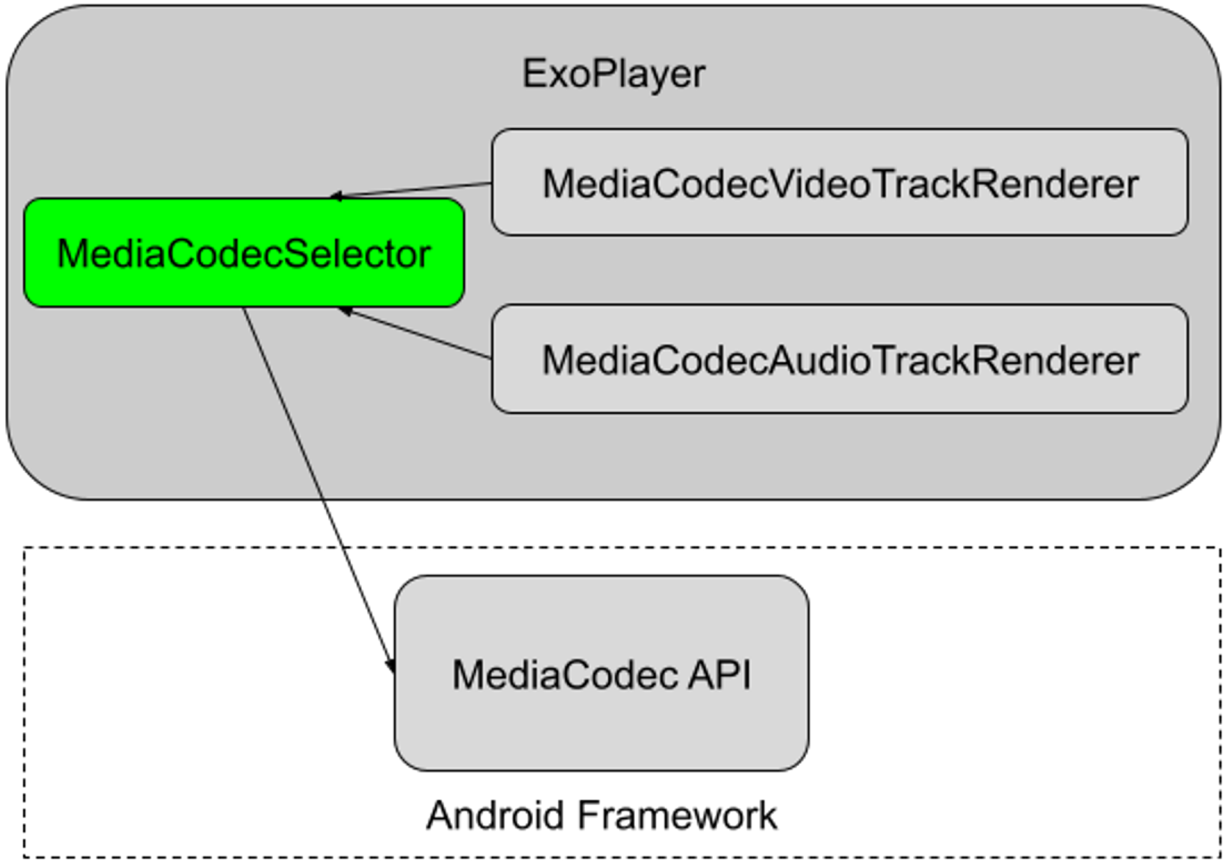
Зачем?

- Видео corruption на определенном кодеке
- Ошибка воспроизведения на определенном кодеке
- В таких случаях можно отказаться от использования HW кодека в пользу SW
- Часто такие проблемы device-specific

Работа с проблемными кодеками

Отключение проблемных кодеков

Шаг 1. Свой MediaCodecSelector



Работа с проблемными кодеками

Отключение проблемных кодеков

Шаг 1. Свой MediaCodecSelector

```
class CodecManager(): MediaCodecSelector {

    private val disabledCodecs = HashSet<String>()

    override fun getDecoderInfos(…): MutableList<MediaCodecInfo> =
        MediaCodecSelector
        .DEFAULT
        .getDecoderInfos(...)
        .filterTo(ArrayList()) { mediaCodecInfo -> !isDisabled(mediaCodecInfo.name) }
        .let { Collections.unmodifiableList(it) }

    fun isDisabled(name: String) = disabledCodecs.contains(name)
}
```

Работа с проблемными кодеками

Отключение проблемных кодеков

Шаг 2. Отдаем Selector Renderer'y

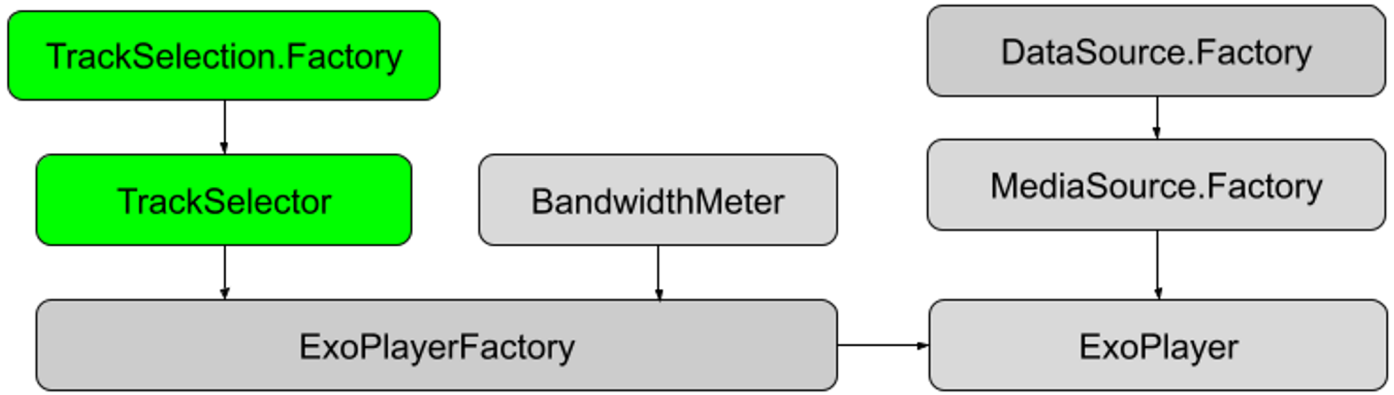
```
val defaultRenderersFactory = DefaultRenderersFactory(context)

//включим SW кодеки
defaultRenderersFactory.setExtensionRendererMode(...)
defaultRenderersFactory.setMediaCodecSelector(codecSelector)

val builder = SimpleExoPlayer.Builder(context, defaultRenderersFactory).build()
```

Работа с проблемными кодеками
Отключение проблемных профилей кодека

Зачем?

- Если у девайса всего один кодек
- Если все кодеки работают некорректно
- Первый способ не помог

Работа с проблемными кодеками
Отключение проблемных профилей кодека



Работа с проблемными кодеками

Отключение проблемных профилей кодека

Шаг 1. Наследуемся от DefaultTrackSelector

```
class ExtendedTrackSelector(val disabledCodecs: Array<String>) :
        DefaultTrackSelector() {
    override fun selectAllTracks(…): Array<ExoTrackSelection.Definition> {
        val tracks = super.selectAllTracks(…)
        for (i in tracks.indices) {
            val src: ExoTrackSelection.Definition? = tracks[i]
            if (src != null) {
                tracks[i] = ExoTrackSelection.Definition(src.group, src.tracks
                        .filter { index ->
                            src.group.getFormat(index).codecs?.let { codecs ->
                                disabledCodecs.none
                                { codecs.startsWith(it, true) }
                            } ?: true
                        }.toIntArray(), src.type)
            }
        }
        return tracks
    }
```

Работа с проблемными кодеками

Отключение проблемных профилей кодека

Шаг 2. Передаем новый Selector в Player

```
val trackSelector = ExtendedTrackSelector(...)
val builder = SimpleExoPlayer.Builder(context, defaultRenderersFactory)
builder.setMediaSourceFactory(sourceFactory)
                                    .setTrackSelector(trackSelector).build()
```

Работа с проблемными кодеками
Итог

В решении проблем с воспроизведением медиа есть ряд способов фильтрации кодеков:

- Полностью не использовать кодек
- Отключить определенные профили кодека
- Для расширения возможностей включить SW кодеки

О чем поговорим

- ТЗ и требования, протоколы
- Оптимизация проигрывания VOD
  - Переключение качества аудио/видео
  - Быстрые закладки
  - Кеширование манифеста
- Оптимизация проигрывания Live видео
  - DefaultLoadControl
- Работа с проблемными кодеками
  - Как Exoplayer выбирает кодеки
  - Отключение проблемных кодеков
  - Отключение проблемных профилей кодека
- **Продвинутые функции проигрывания**
  - **Перемотка**

Продвинутые функции проигрывания
Перемотка

Поставленные перед нами цели:

- Перемотка в стиле VHS с плавной сменой кадров
- Перемотка вперед и назад
- Смена кадров минимум раз в секунду!

С чем пришлось столкнуться:

- Exoplayer не поддерживает отрицательную скорость проигрывания

Продвинутые функции проигрывания
Перемотка

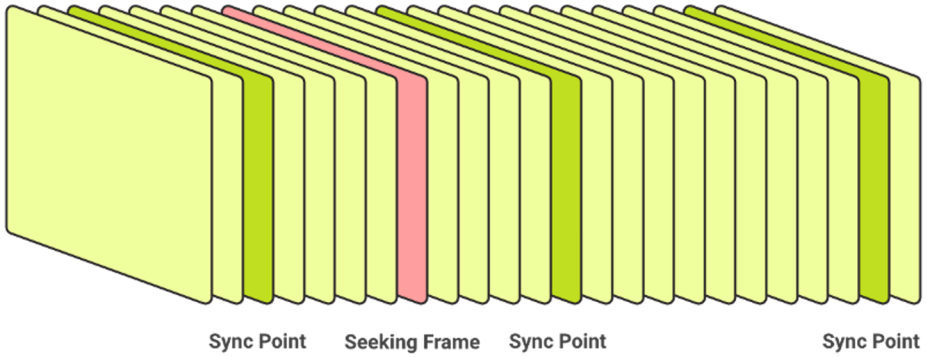
Как решаем?

- Заменяем перемотку ExoPlayer’a из коробки на seek
- Пробуем SeekParameters для увеличения частоты кадров

Продвинутые функции проигрывания

Перемотка

Шаг 1. Используем SeekParameters

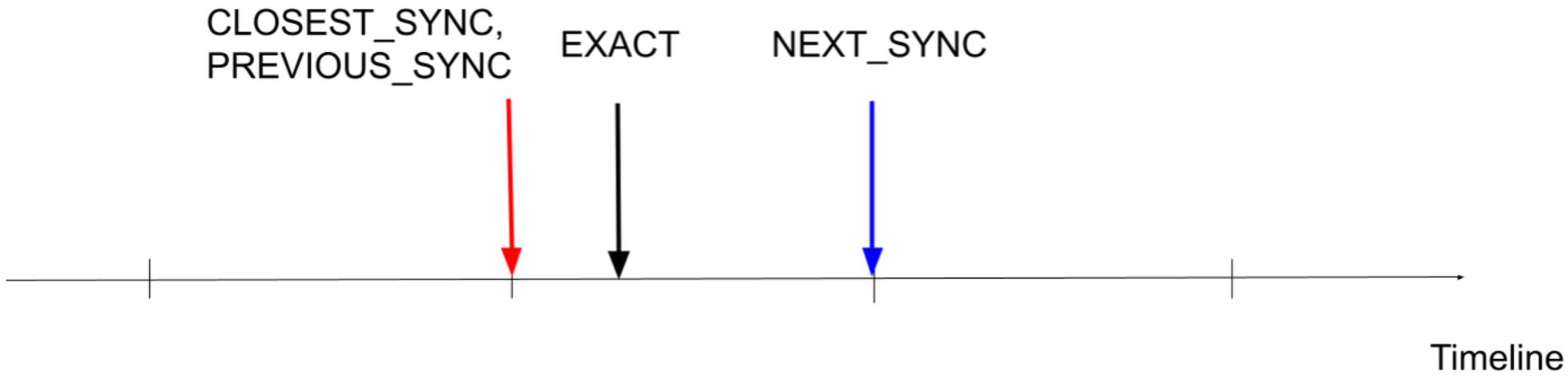


CLOSEST_SYNC
~~EXACT~~
PREVIOUS_SYNC
NEXT_SYNC

Продвинутые функции проигрывания

Перемотка

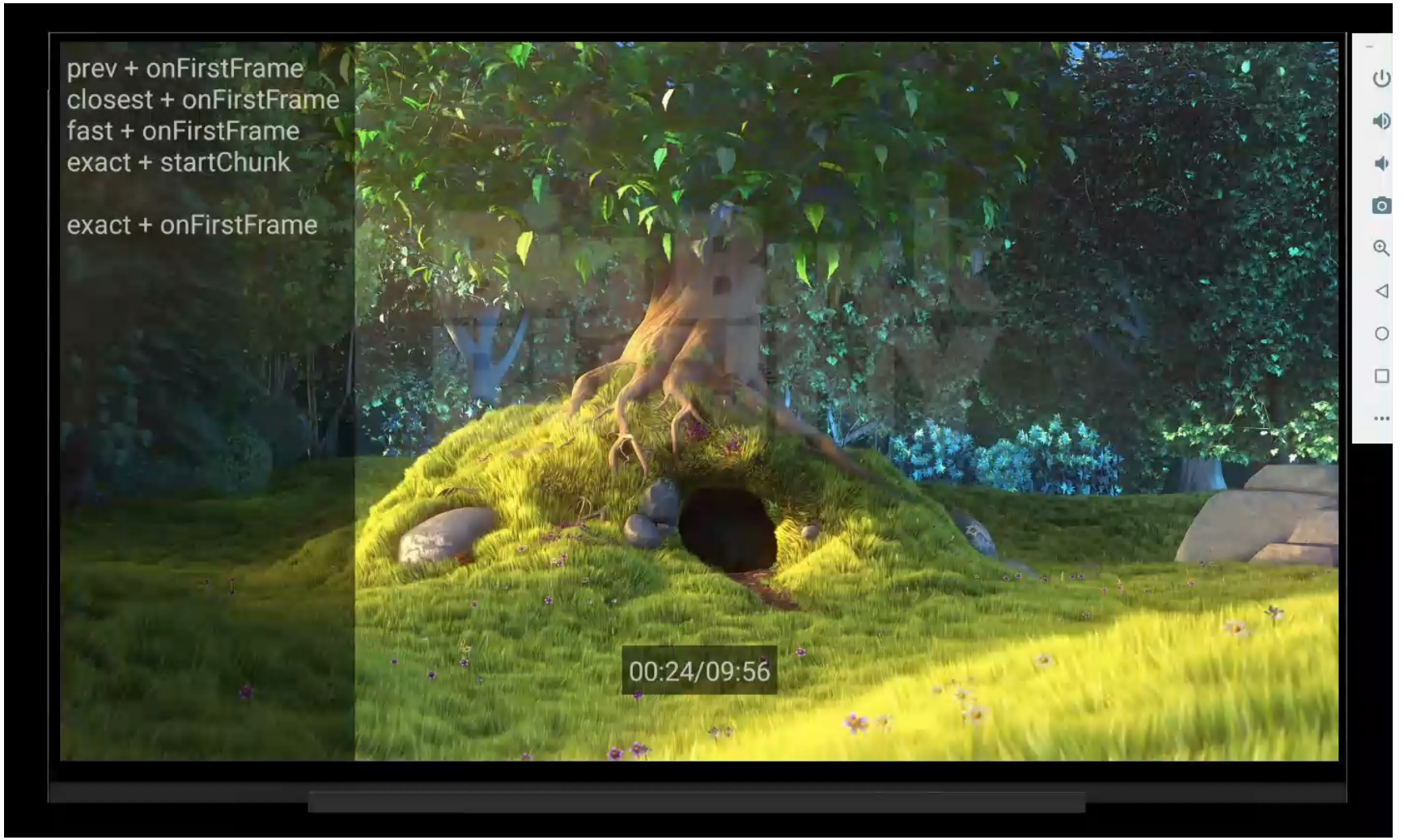
Шаг 1. Используем SeekParameters



Продвинутые функции проигрывания

Перемотка

Шаг 1. Используем SeekParameters. Exact

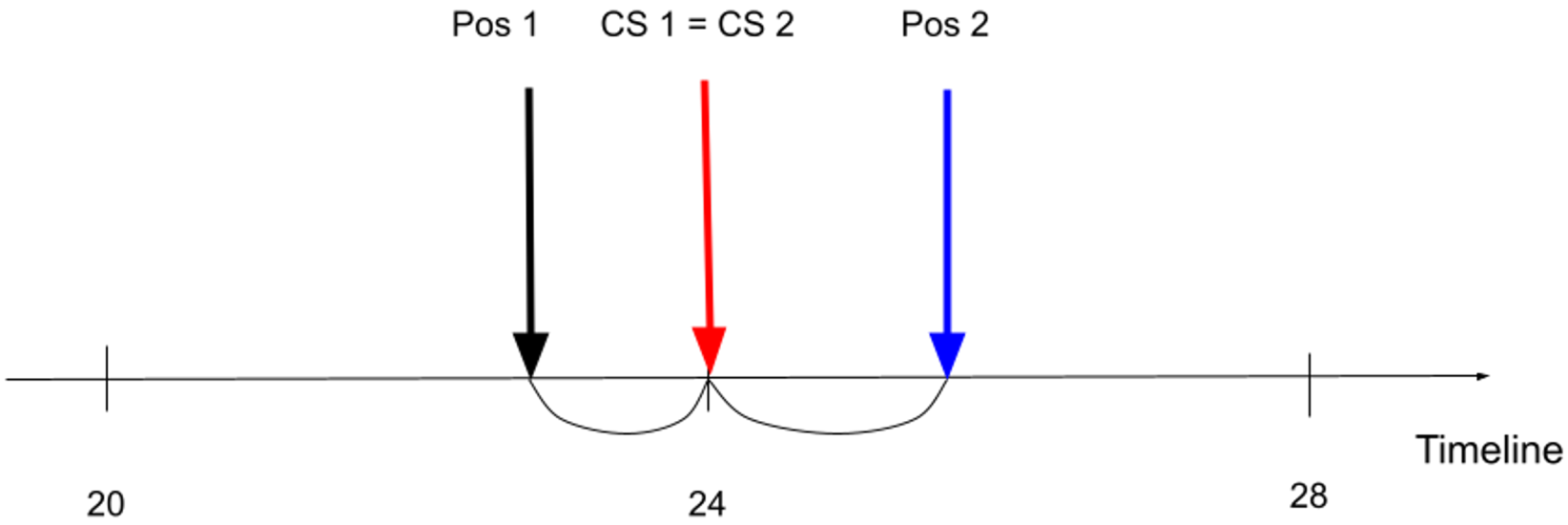


Продвинутые функции проигрывания

Перемотка

Шаг 2. Почему CLOSEST_SYNC не помогает?

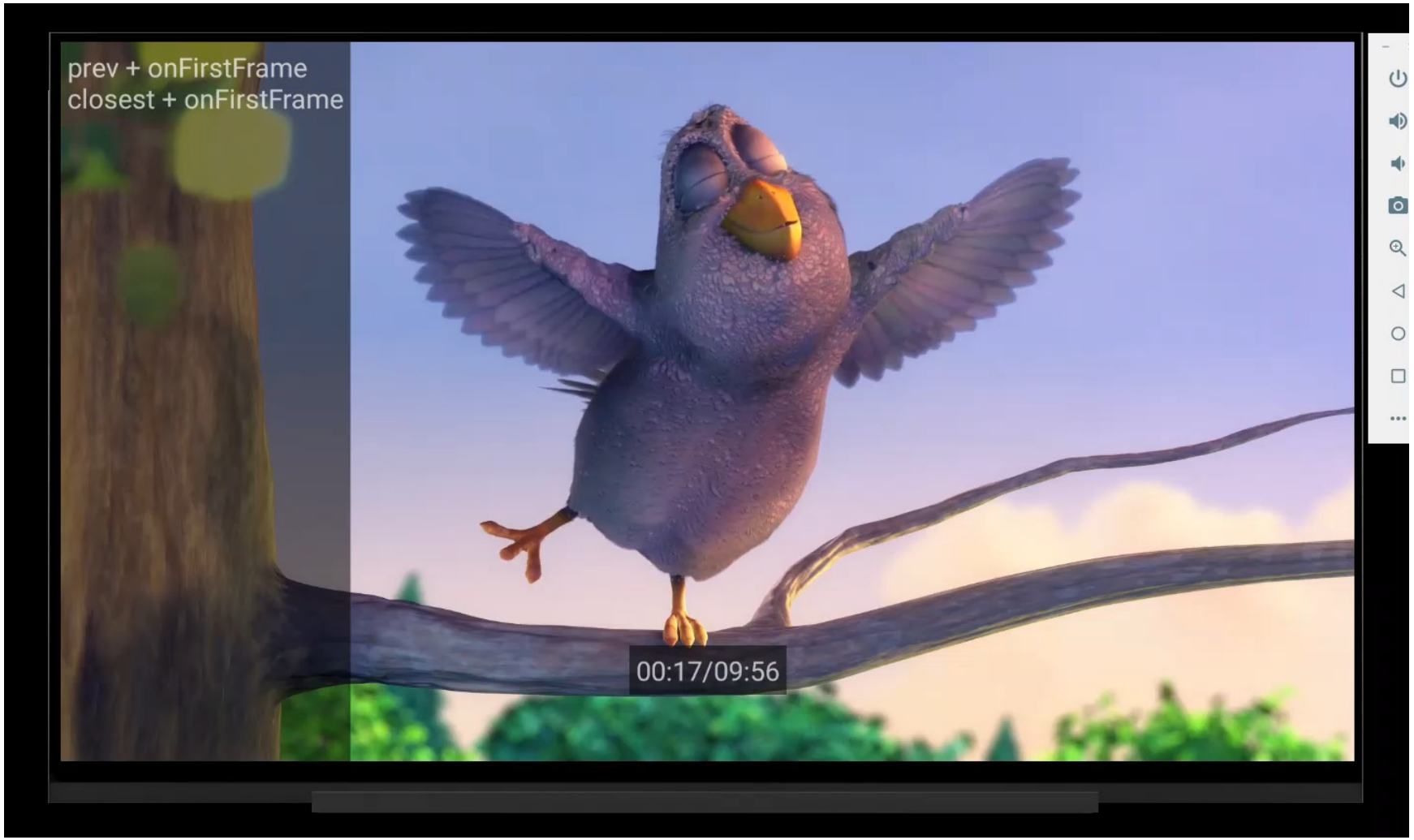
CS = CLOSEST_SYNC
Pos = position to seek



Продвинутые функции проигрывания

Перемотка

Шаг 2. Почему CLOSEST_SYNC не помогает?



Продвинутые функции проигрывания

Перемотка

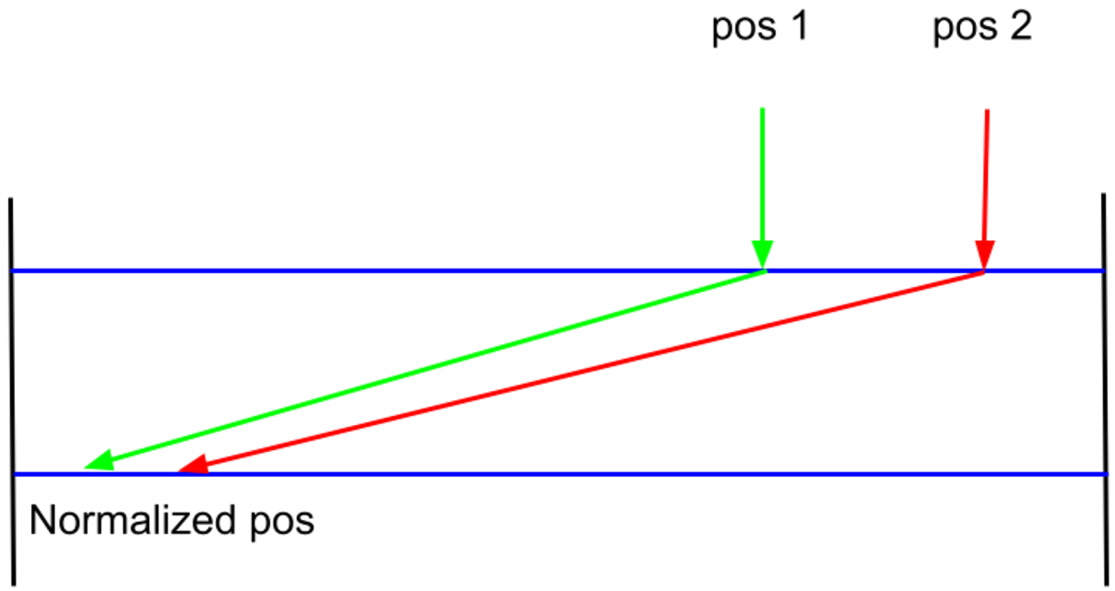
Шаг 3. Решение на базе коэффициента перемотки

```
CustomDashChunkSource(...) {
    override fun getAdjustedSeekPositionUs(positionUs: Long,
                        seekParameters: SeekParameters): Long {
        if (seekType == SeekType.FAST) {
            for (representationHolder in representationHolders) {
                if (representationHolder.segmentIndex != null) {
                    val segmentNum =
                            representationHolder.getSegmentNum(positionUs)
                     val firstSyncUs =
                            representationHolder.getSegmentStartTimeUs(segmentNum)
                     return firstSyncUs + (positionUs - firstSyncUs) / DISPLACEMENT_FACTOR
        ...
        return super.getAdjustedSeekPositionUs(positionUs, seekParameters)
    }
}
```

Продвинутые функции проигрывания

Перемотка

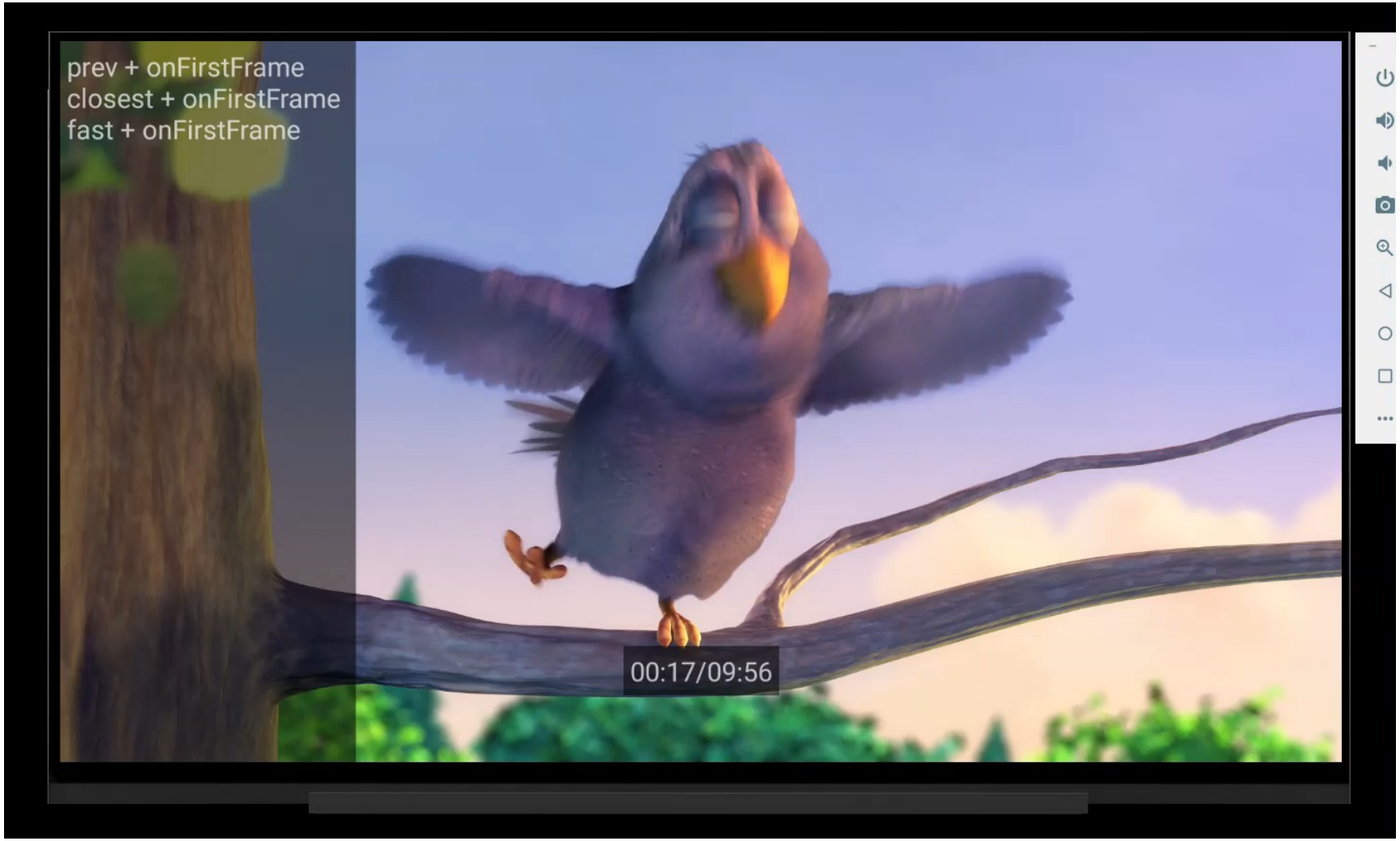
Шаг 3. Решение на базе коэффициента перемотки



Продвинутые функции проигрывания

Перемотка

Шаг 3. Решение на базе коэффициента перемотки



Продвинутые функции проигрывания
Перемотка
Итог

	Exact		Closest		Fast	
	SeekPositions	SeekTime	SeekPositions	SeekTime	SeekPositions	SeekTime
	15993	617	102291	305	201066	203
	22793	1762	105891	924	201866	203
	25193	623	107891	514	203866	511
	27993	717	113091	1339	205066	304
	34793	1733	115491	621	206266	309
	39193	1120	120291	1227	208666	627
	42793	927	121491	307	210266	408
AVG		1071		748		366

Q&A

Куликова Надежда
Nadezhda.s.Kulikova@gmail.com
https://t.me/KulikovaNadezhda