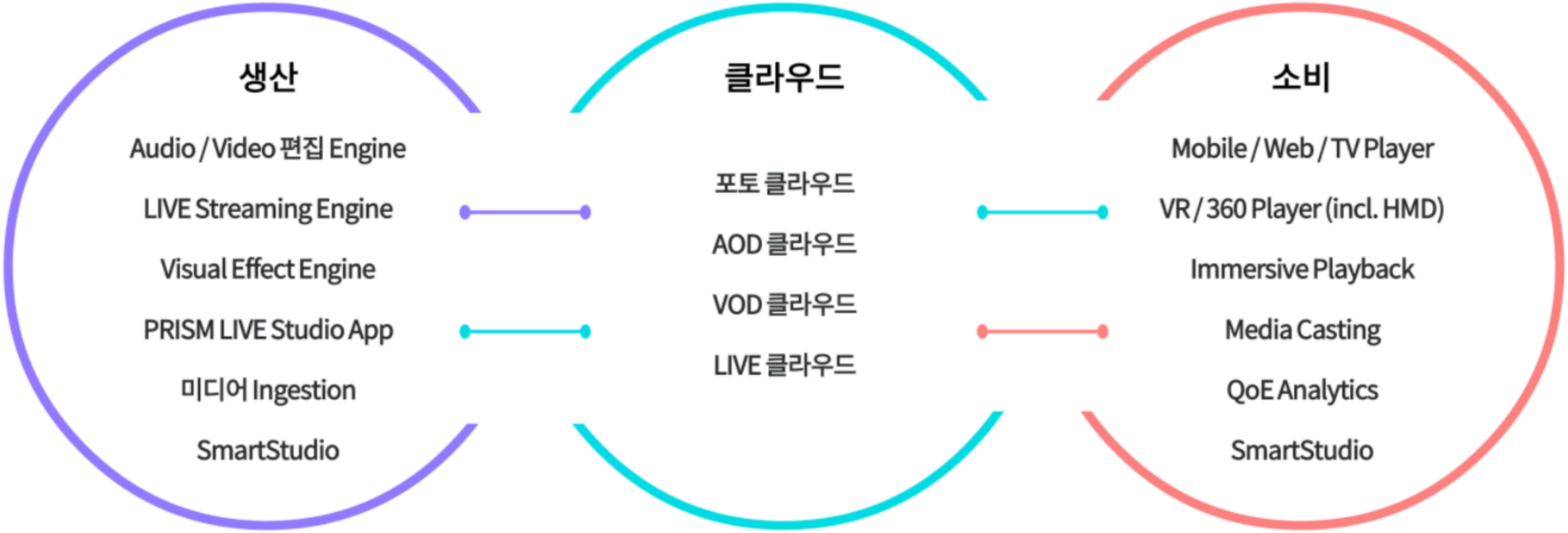
AVPlayer Custom Resource Loader 노하우 공유

강성현 NAVER ETECH

NAVER
Emerging
TECHnology

NAVER ETECH.

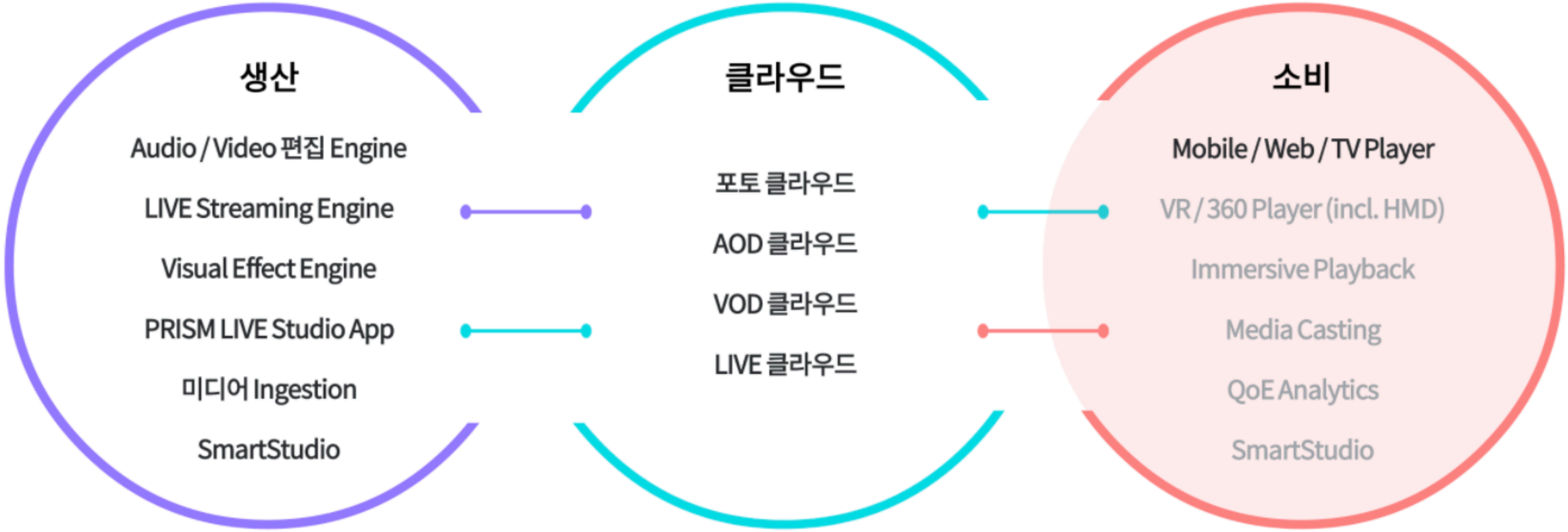
포토 / 오디오 / 비디오의 <생산 – 클라우드 – 소비> 워크플로의 전구간 기술 연구와 개발을 담당합니다.
글로벌 환경에서 시간 / 공간 / 용량의 제약 사항을 극복하고 생생한 현장 느낌과 안정적인 지원을 위해
이머징 기술 연구와 개발을 통한 원격의 시대를 준비하고 있습니다.



NAVER Emerging TECHnology

NAVER ETECH.

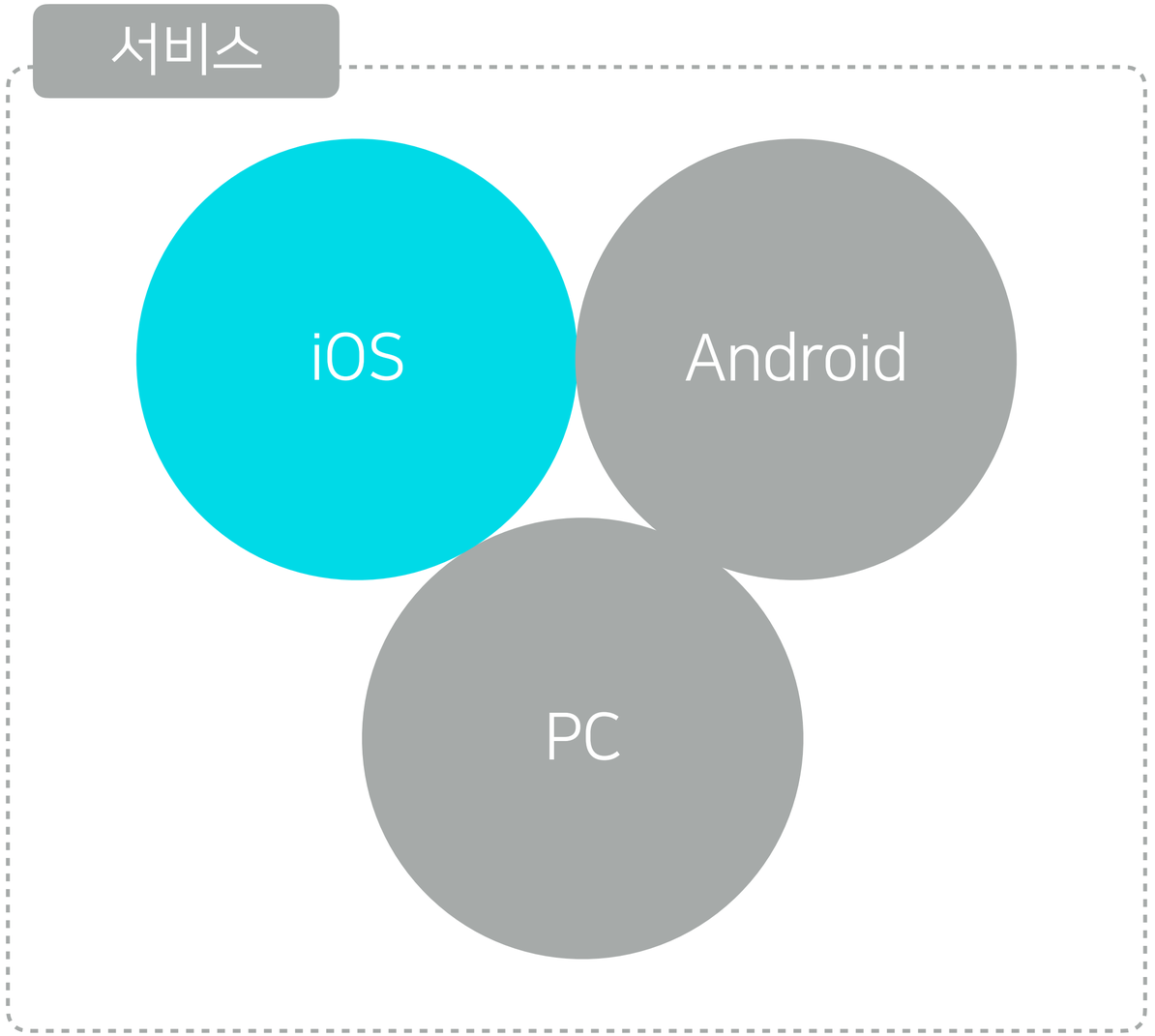
포토 / 오디오 / 비디오의 <생산 – 클라우드 – 소비> 워크플로의 전구간 기술 연구와 개발을 담당합니다.
글로벌 환경에서 시간 / 공간 / 용량의 제약 사항을 극복하고 생생한 현장 느낌과 안정적인 지원을 위해
이머징 기술 연구와 개발을 통한 원격의 시대를 준비하고 있습니다.



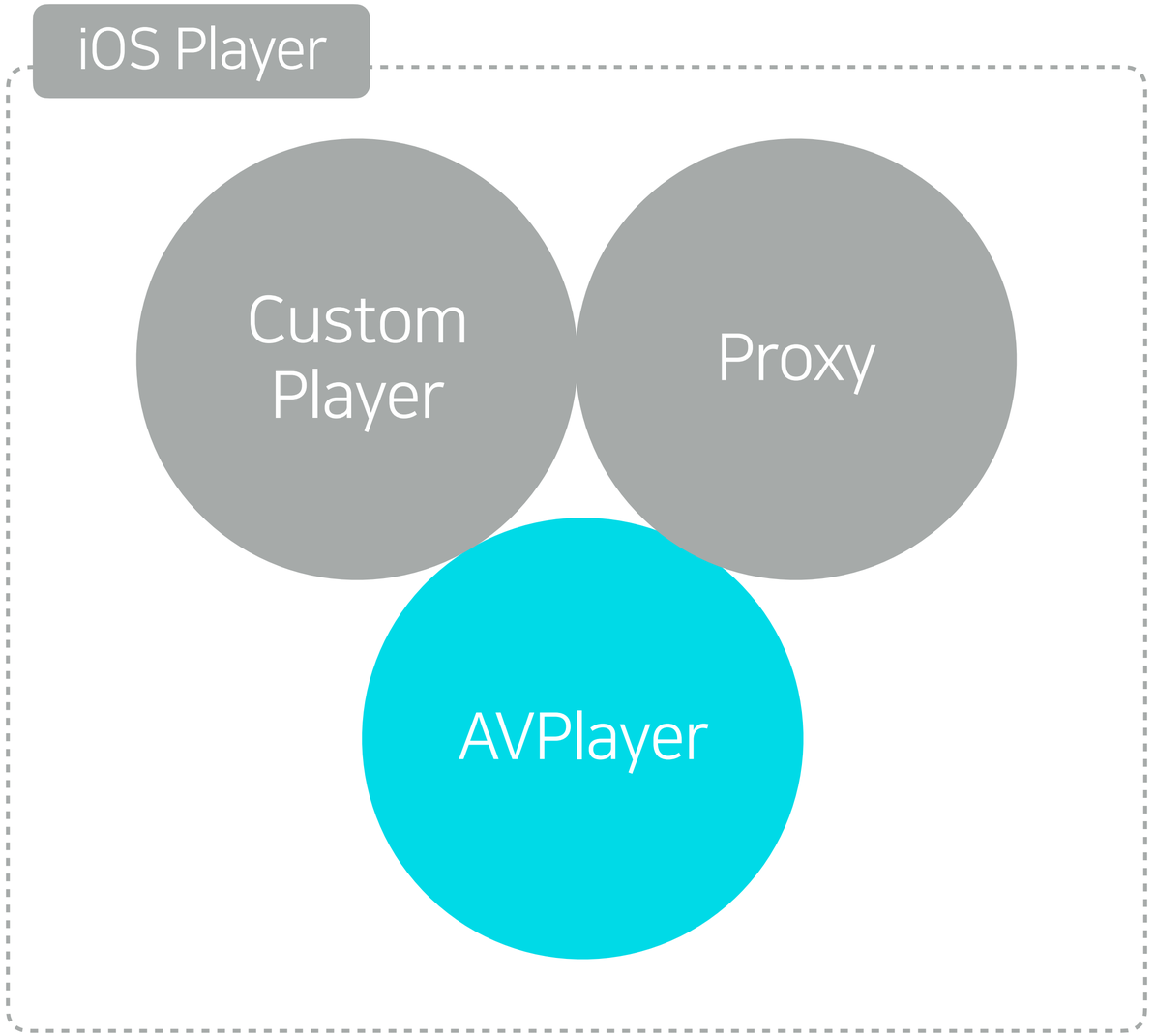
DEVIEW 2021

Customizing

Intro. Customizing



Intro. Customizing

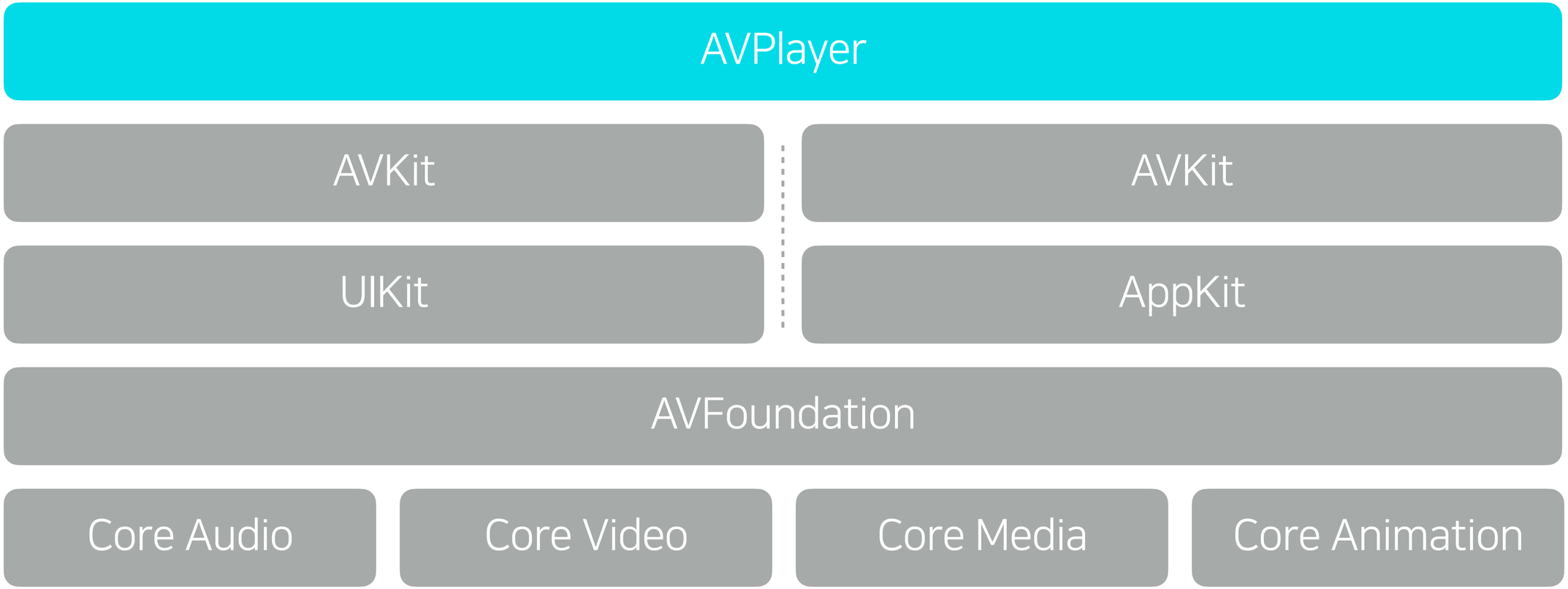


Intro. Customizing

DEVIEW 2021

AVPlayer?

Apple에서 제공하는 미디어 재생을 제어하기 위한 도구

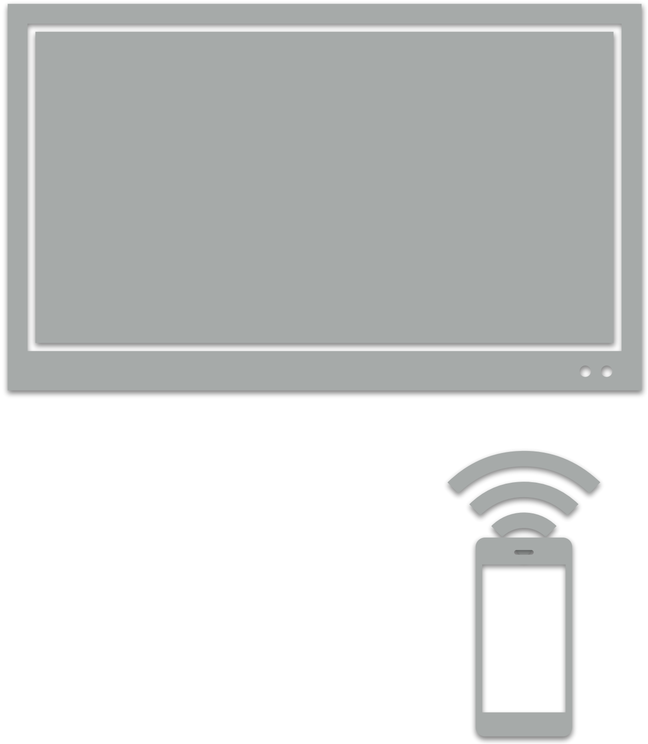


Intro. Customizing

DEVIEW 2021

매년 추가되는 강력한 추가 기능

- AirPlay
- PIP (Picture In Picture)
- SharePlay (Group Activities)



공개되지 않은 내부 구조

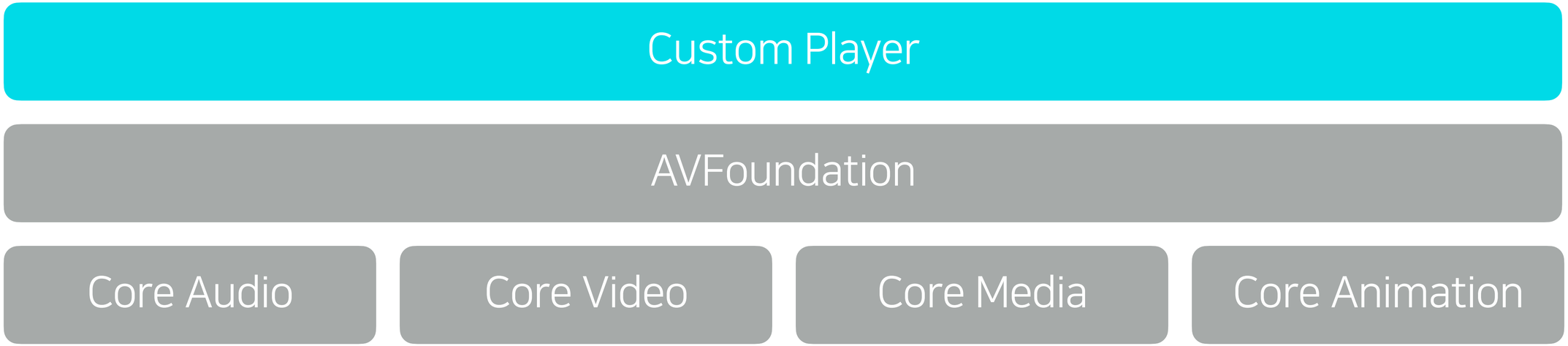
- 제한된 스펙
- 어려운 확장성

Intro. Customizing

DEVIEW 2021

CustomPlayer

- 제한적인 용도
- 범용적인 용도 개발로는 많은 리소스 필요성
- 최신 미디어 스펙 대응 및 iOS 업데이트 시 기능점검

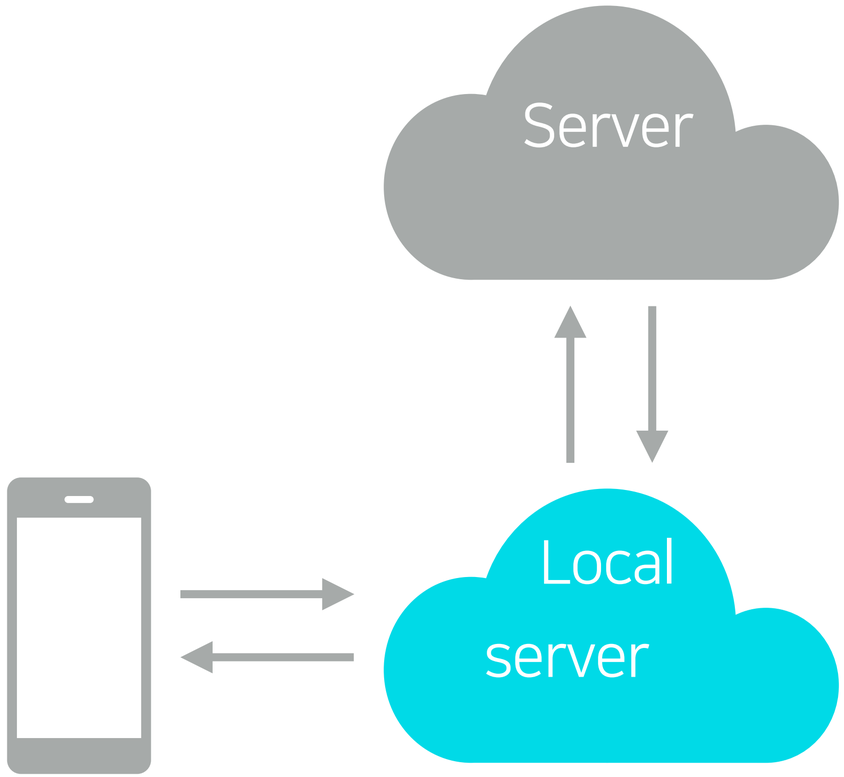


Intro. Customizing

DEVIEW 2021

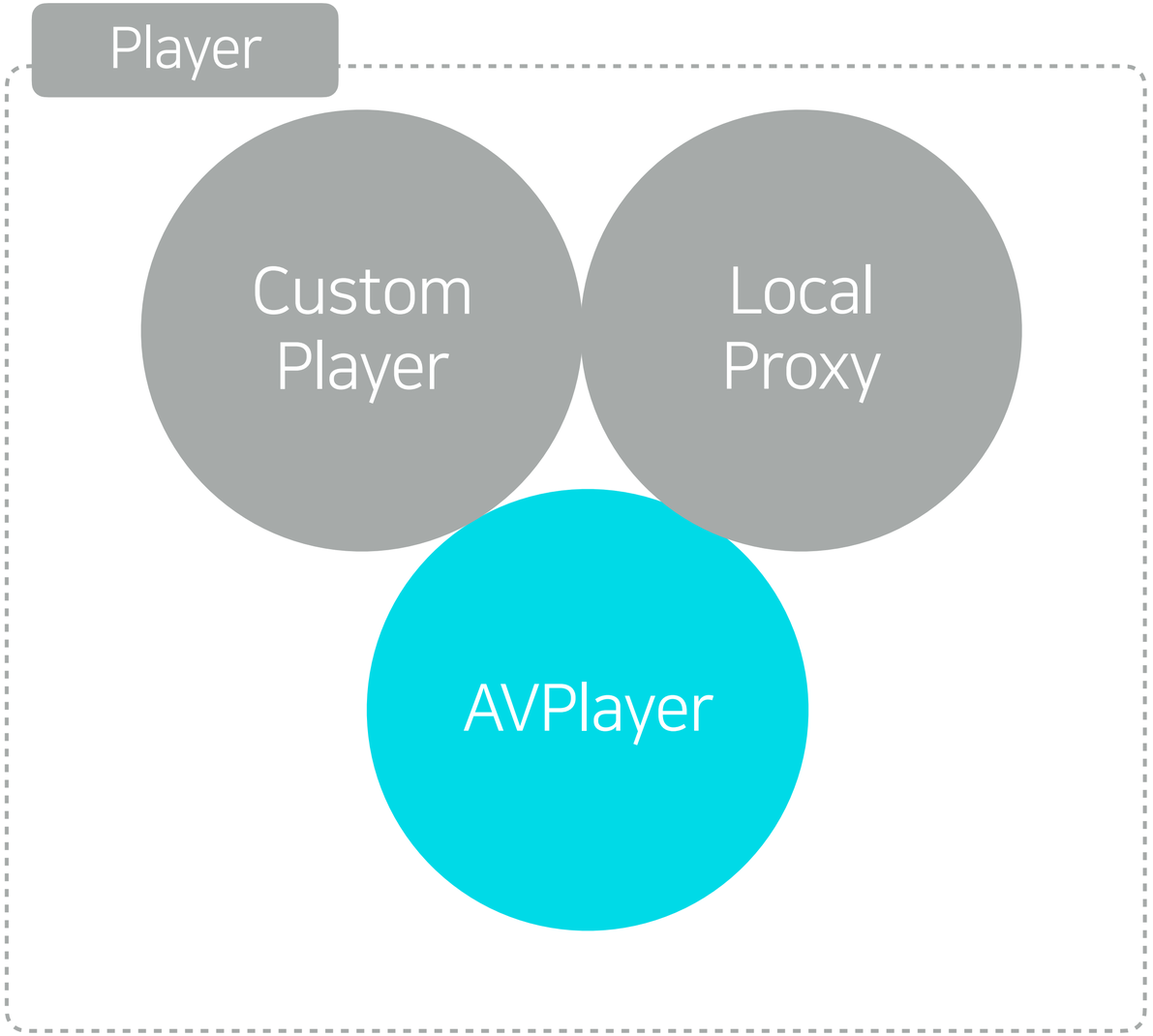
중간단계 커스터마이징

- 플레이어 또는 동영상에 로컬프록시 기능을 포함하기 부담스러운 점
- 생명주기를 적절히 맞추지 않으면 hanging에 관련한 리스크



Intro. Customizing

AVPlayer
ResourceLoader
Customizing



CONTENTS

1. 커스텀 리소스 로더 구현하기

1.1 ABR 이란?

DEVIEW 2021

ABR (Adaptive bitrate streaming)

- 사용자의 대여폭과 디바이스 성능에 따라 비디오의 품질을 조절
- 적응형 스트리밍

1.1 ABR 이란?

DEVIEW 2021

ABR (Adaptive bitrate streaming)

1.1 ABR 이란?

ABR (Adaptive bitrate streaming)

Apple - HLS

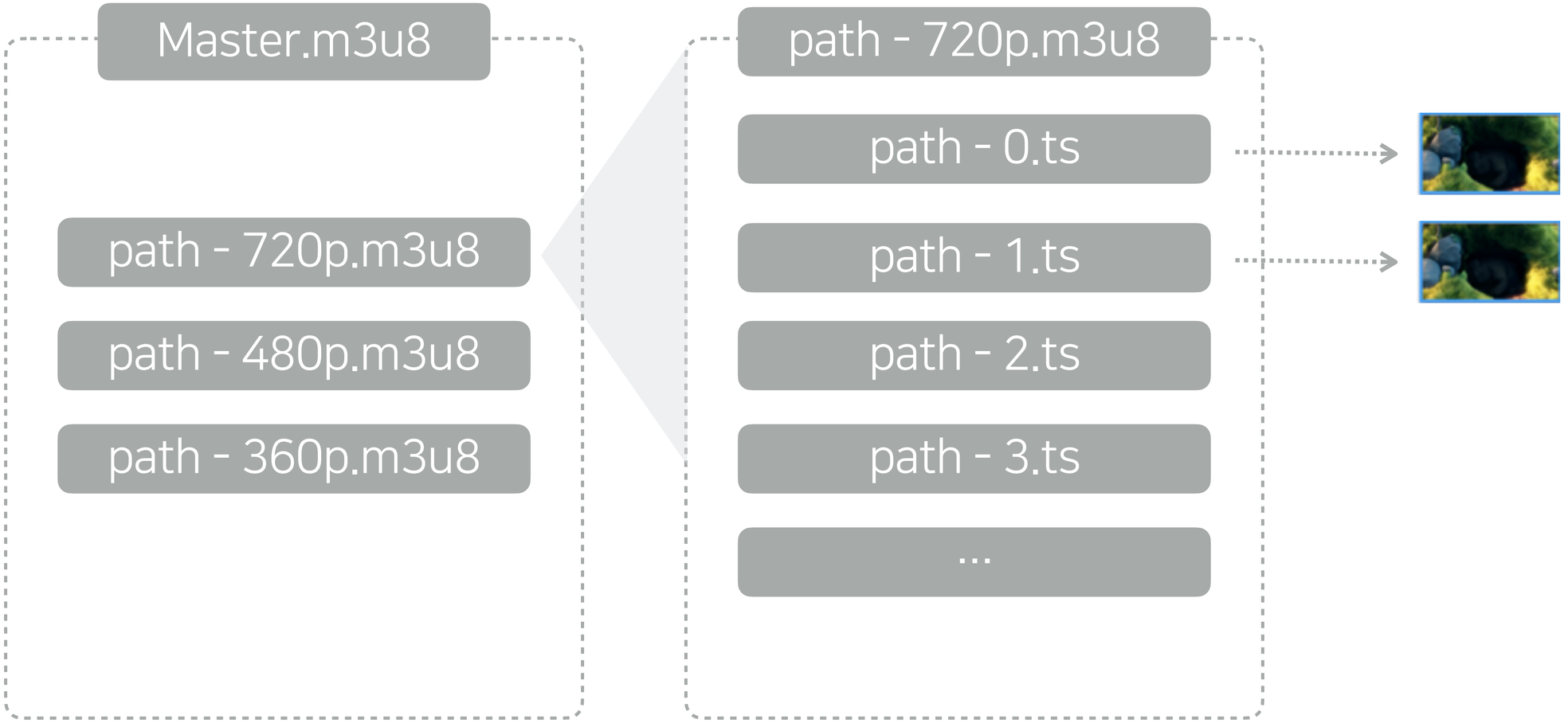
HTTP Live Streaming

MPEG - DASH

Dynamic Adaptive Streaming over HTTP

1.1 ABR 이란?

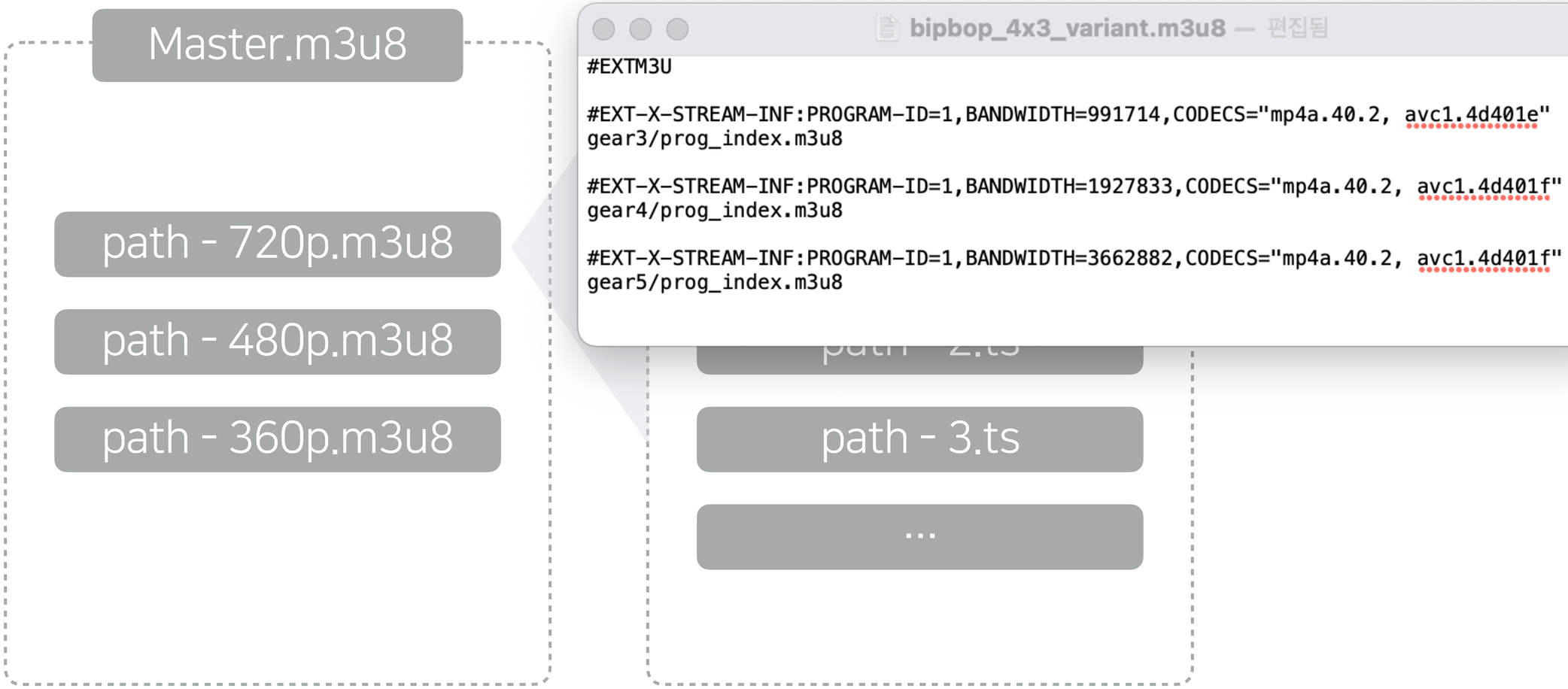
HLS 구조



1.1 ABR 이란?

DEVIEW 2021

HLS 구조

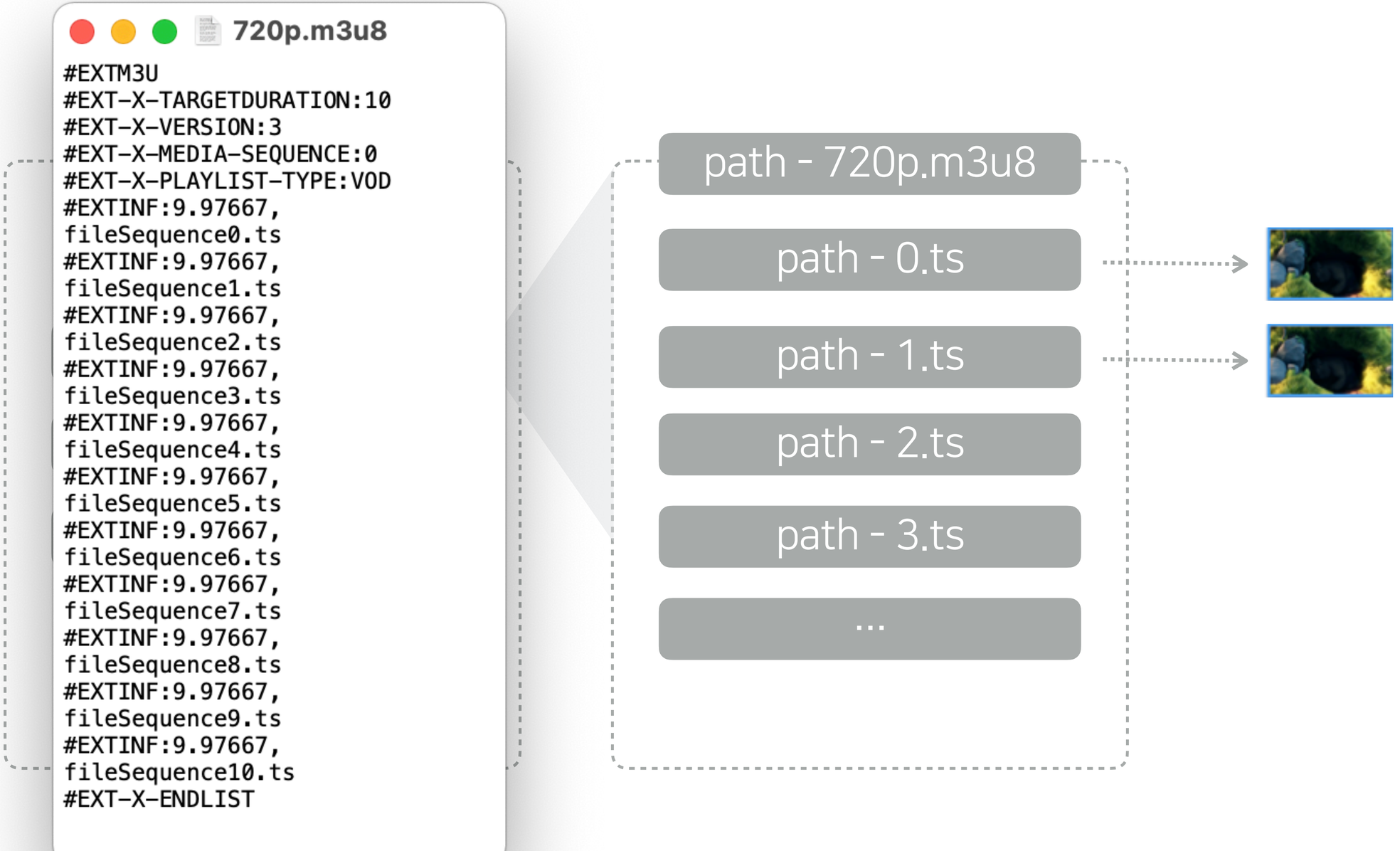


1.1 ABR 이란?

HLS 구조

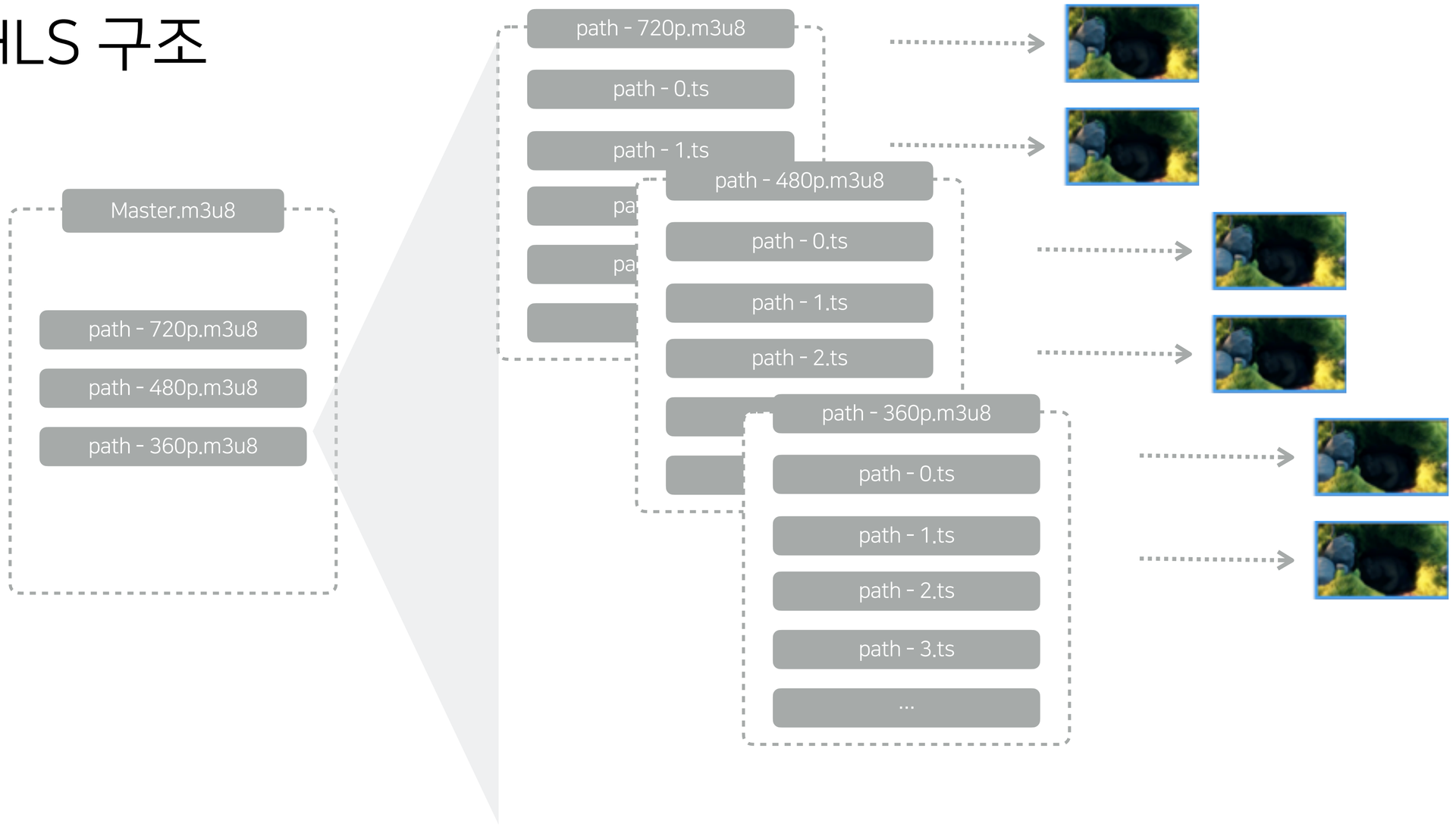
720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXTINF:9.97667,
fileSequence3.ts
#EXTINF:9.97667,
fileSequence4.ts
#EXTINF:9.97667,
fileSequence5.ts
#EXTINF:9.97667,
fileSequence6.ts
#EXTINF:9.97667,
fileSequence7.ts
#EXTINF:9.97667,
fileSequence8.ts
#EXTINF:9.97667,
fileSequence9.ts
#EXTINF:9.97667,
fileSequence10.ts
#EXT-X-ENDLIST
```



1.1 ABR 이란?

HLS 구조



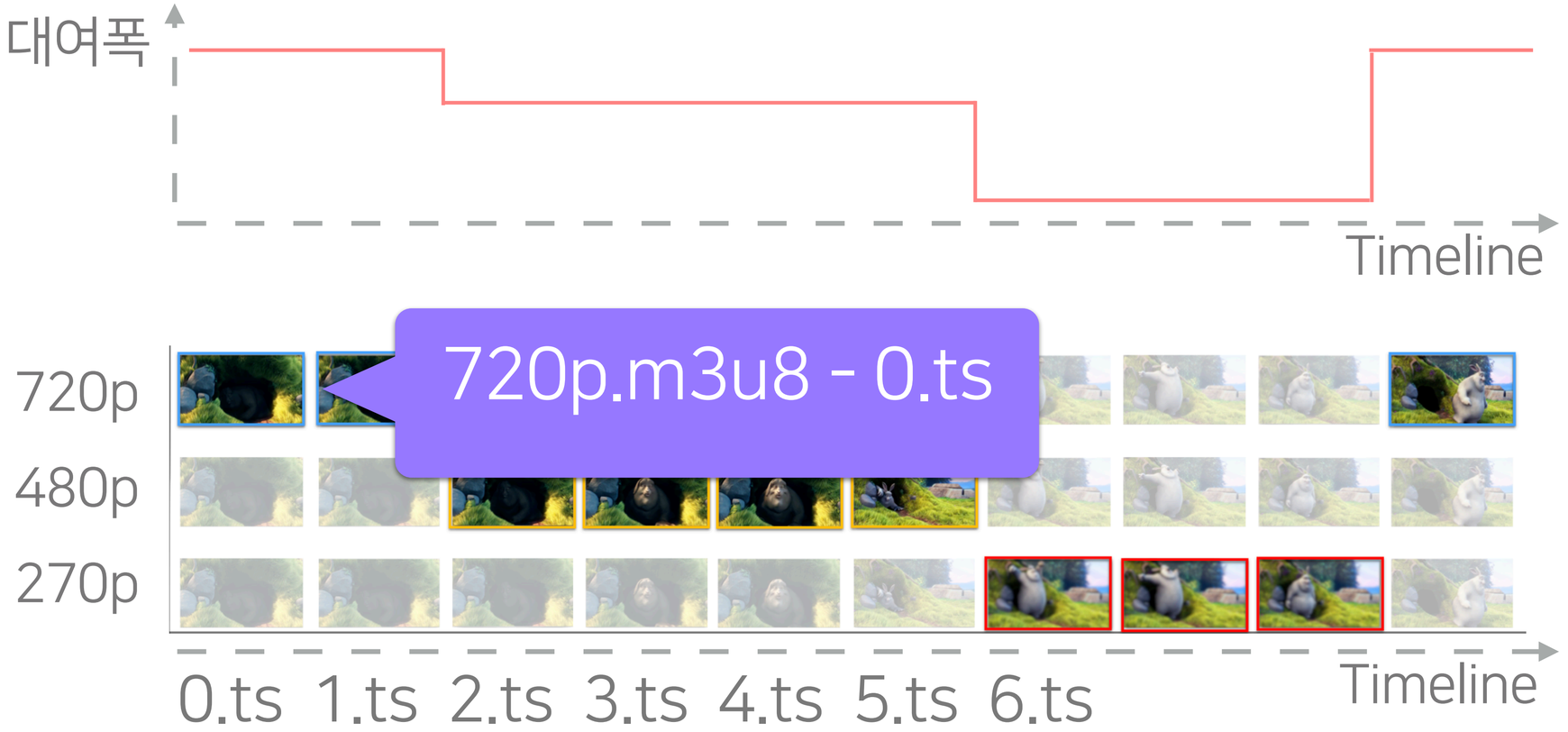
1.1 ABR 이란?

ABR (Adaptive bitrate streaming)

1.1 ABR 이란?

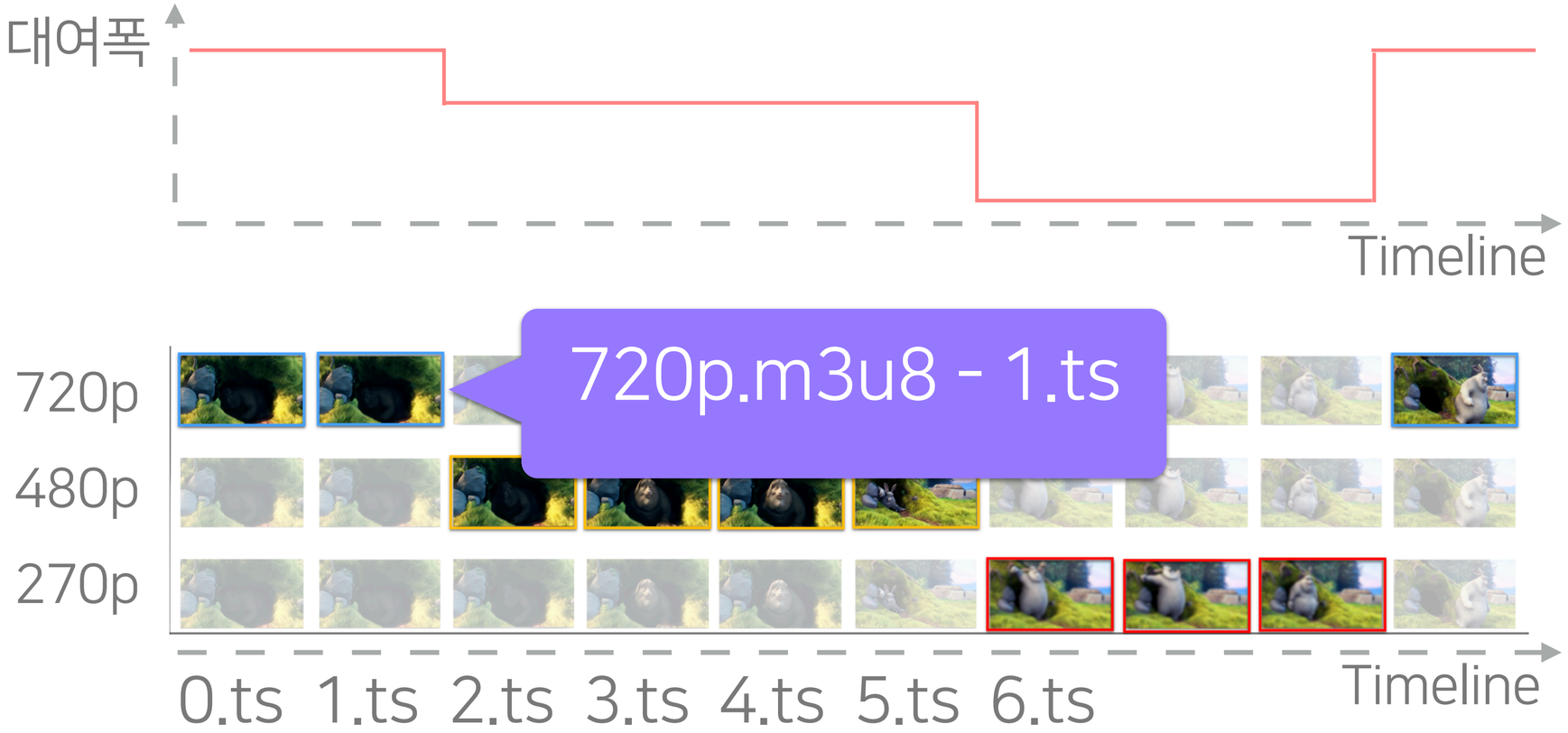
DEVIEW 2021

ABR (Adaptive bitrate streaming)



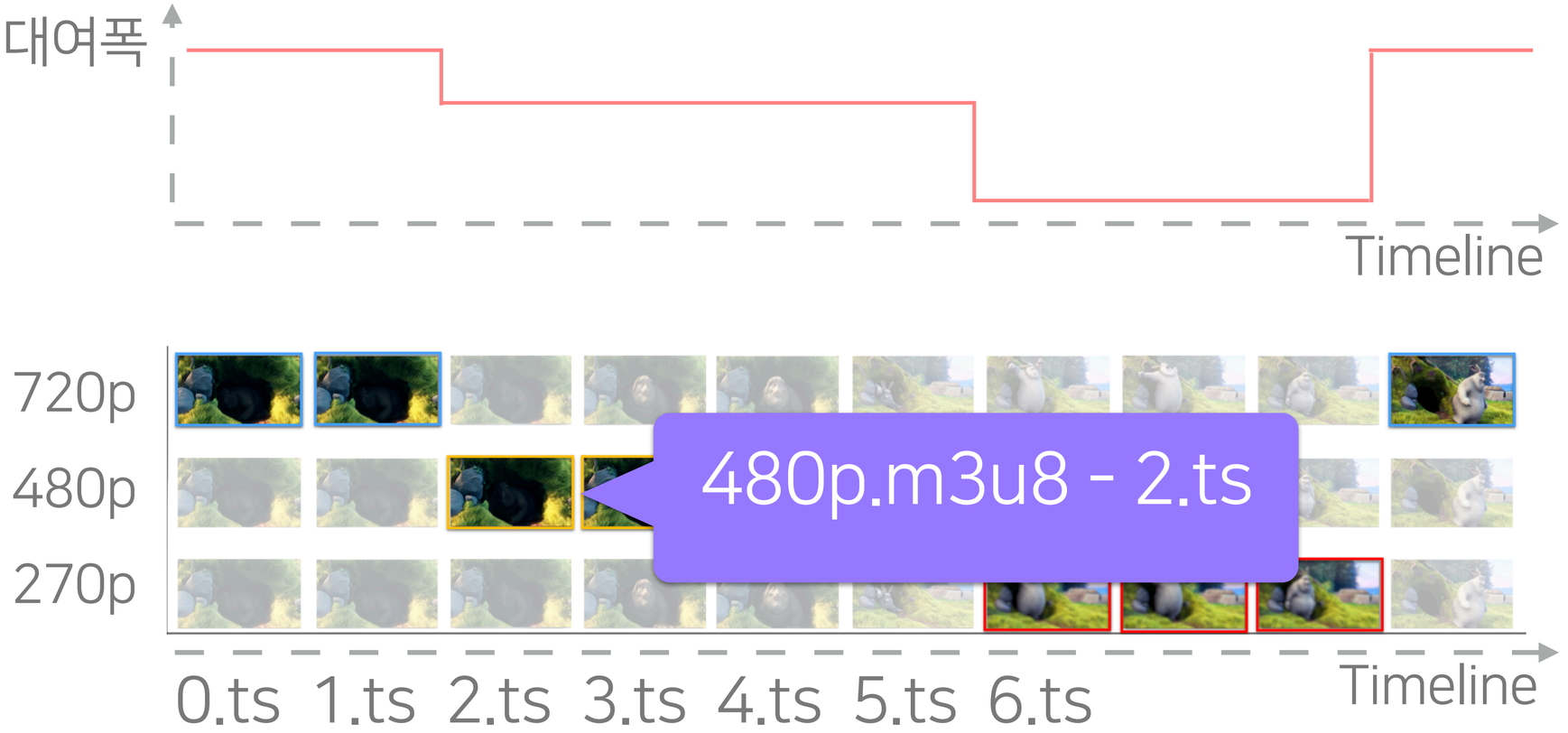
1.1 ABR 이란?

ABR (Adaptive bitrate streaming)



1.1 ABR 이란?

ABR (Adaptive bitrate streaming)



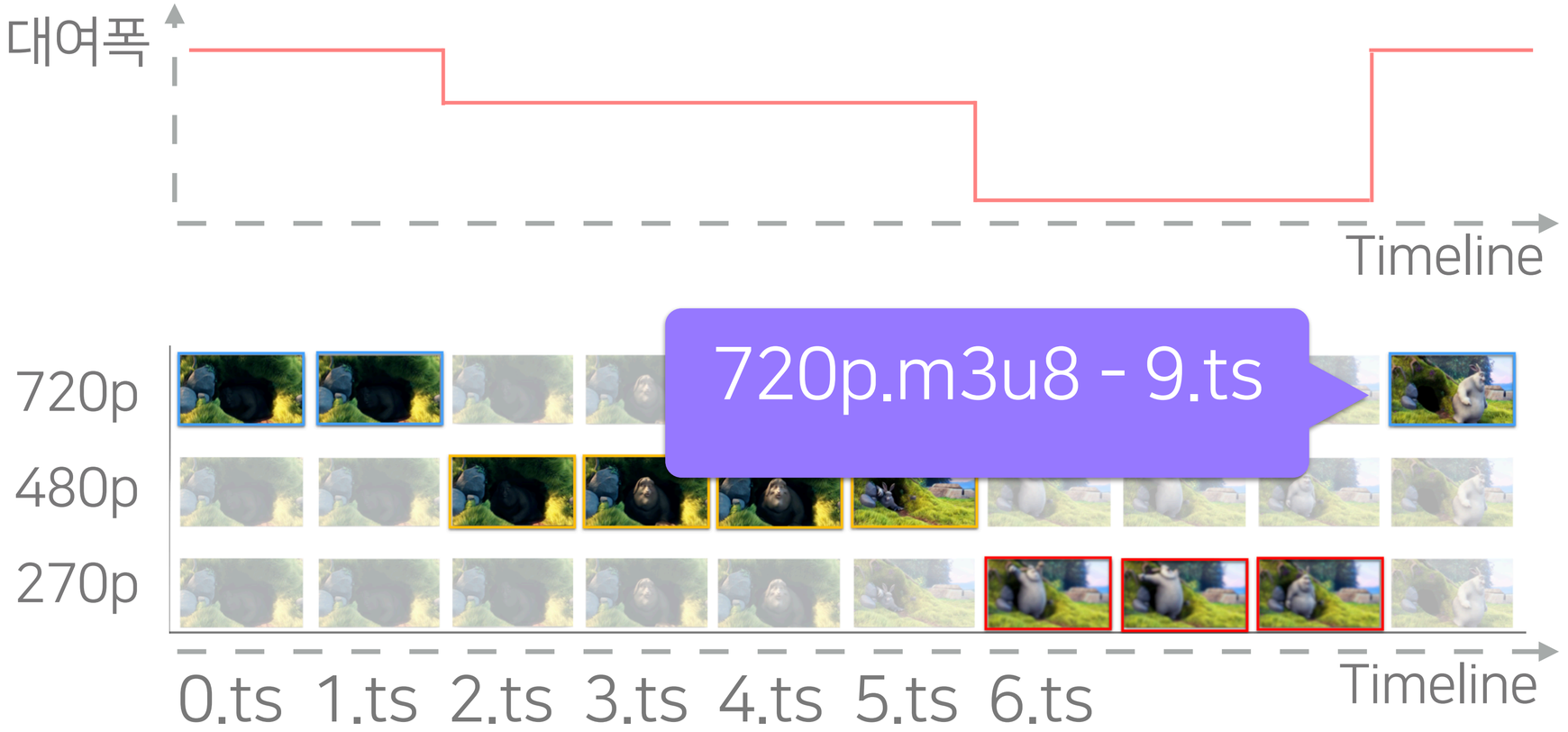
1.1 ABR 이란?

DEVIEW 2021

ABR (Adaptive bitrate streaming)

1.1 ABR 이란?

ABR (Adaptive bitrate streaming)



1.1 ABR 이란?

ABR (Adaptive bitrate streaming)

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

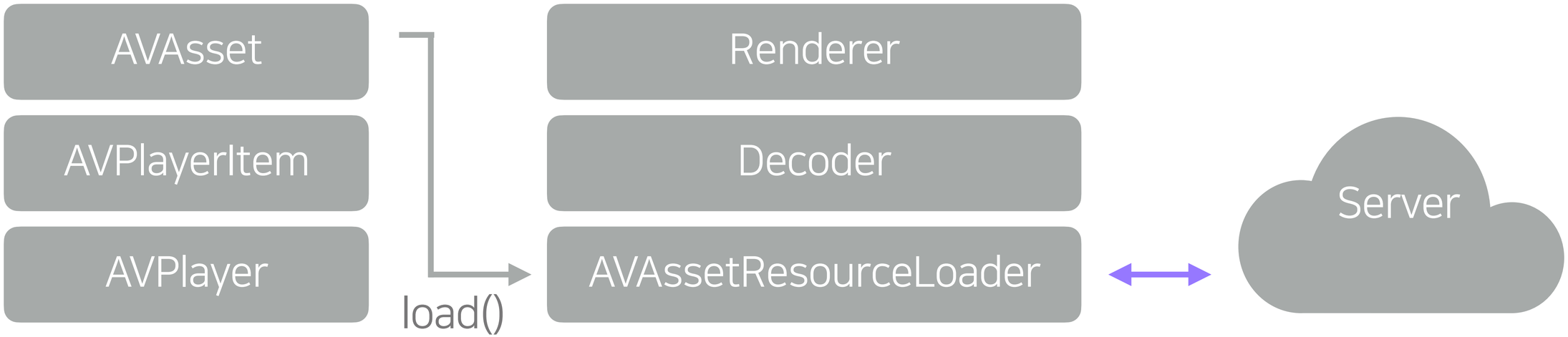
AVPlayer재생

```
func playBasic() {
    let hlsURL = URL(string: "https://deview.kr/2021/sample.m3u8")!
    let urlAsset = AVURLAsset(url: hlsURL)
    let playItem = AVPlayerItem(asset: urlAsset)

    player.replaceCurrentItem(with: playItem)
    player.play()
}
```

1.2 커스텀 리소스 로더 설정

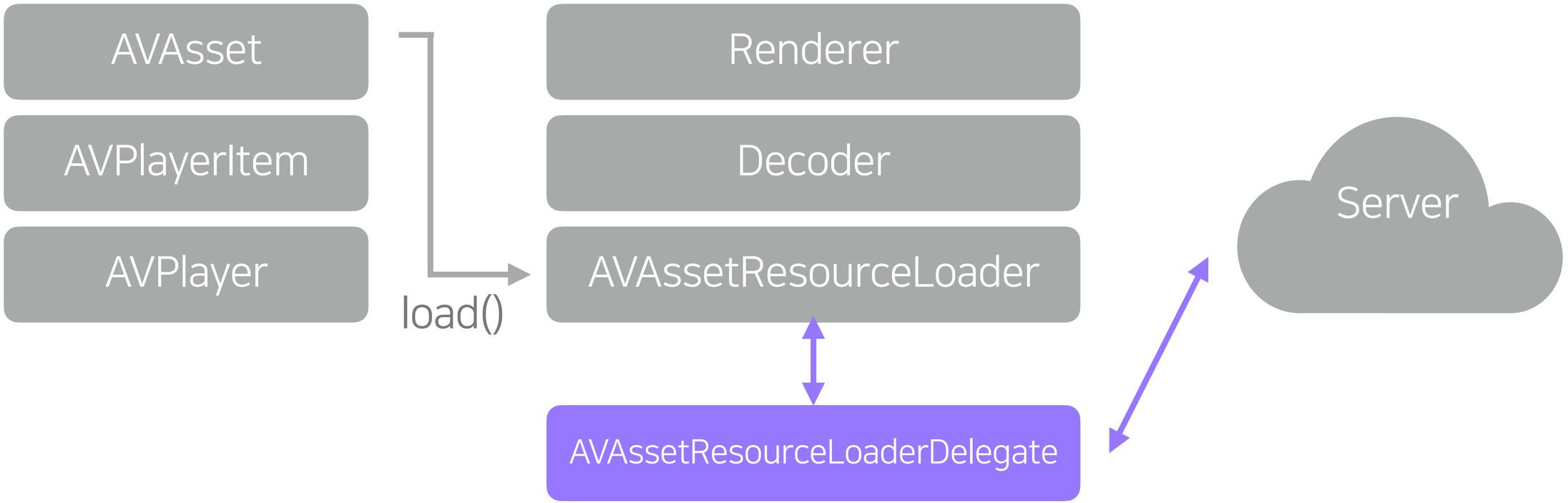
DEVIEW 2021

일반적인 AVPlayer재생



1.2 커스텀 리소스 로더 설정

커스텀 로더를 이용한 AVPlayer재생



1.2 커스텀 리소스 로더 설정

DEVIEW 2021

커스텀 로더를 이용한 AVPlayer 재생

1. URL의 Scheme를 조정합니다.
2. 위임받기 위한 AVAssetResourceLoaderDelegate의 구현체를 생성합니다.
3. AVURLAsset.resouceLoader에 위임자를 연결합니다.

1

```
        let delegate = CustomAssetDel
                                        "chls://c

        func play() {
            let hlsURL = URL(string: "https://deview.kr/2021/sample.m3u8")!
            let redirectURL = URL(string: "chls://deview.kr/2021/sample.m3u8")!
            let urlAsset = AVURLAsset(url: redirectURL)
            urlAsset.resourceLoader.setDelegate(delegate,
                                                queue: DispatchQueue.global(qos: .background))

            let playItem = AVPlayerItem(asset: urlAsset)
            player.replaceCurrentItem(with: playItem)
            player.play()
        }
    }

    class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {}
```

1.2 커스텀 리소스 로더 설정

커스텀 로더를 이용한 AVPlayer 재생

1. URL의 Scheme를 조정합니다.
2. 위임받기 위한 AVAssetResourceLoaderDelegate의 구현체를 생성합니다.
3. AVURLAsset.resouceLoader에 위임자를 연결합니다.

```
let delegate = CustomAssetDelegate()

func play() {
    let hlsURL = URL(string: "https://deview.kr/2021/sample.m3u8")!
    let redirectURL = URL(string: "chls://deview.kr/2021/sample.m3u8")!
    let urlAsset = AVURLAsset(url: redirectURL)
    urlAsset.resourceLoader.setDelegate(delegate,
                                        queue: DispatchQueue.global(qos: .background))

    let playItem = AVPlayerItem(asset: urlAsset)
    player.replaceCurrentItem(with: playItem)
    player.play()
```

1

2

```
class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {}
```

class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {}

1.2 커스텀 리소스 로더 설정

커스텀 로더를 이용한 AVPlayer 재생

1. URL의 Scheme를 조정합니다.
2. 위임받기 위한 AVAssetResourceLoaderDelegate의 구현체를 생성합니다.
3. AVURLAsset.resouceLoader에 위임자를 연결합니다.

```
        let delegate = CustomAssetDelegate()

        func play() {
            let hlsURL = URL(string: "https://deview.kr/2021/sample.m3u8")!
            let redirectURL = URL(string: "chls://deview.kr/2021/sample.m3u8")!
            let urlAsset = AVURLAsset(url: redirectURL)
            urlAsset.resourceLoader.setDelegate(delegate,
                                                queue: DispatchQueue.global(qos: .background))

            let playItem = AVPlayerItem(
            player.replaceCurrentItem(w
            player.play()
        }
    }

    class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {}
```

1

2

3

```
setDelegate(delegate,
            queue: Dis
```

1.2 커스텀 리소스 로더 설정

커스텀 로더를 이용한 AVPlayer 재생

1. URL의 Scheme를 조정합니다.
2. 위임받기 위한 AVAssetResourceLoaderDelegate의 구현체를 생성합니다.
3. AVURLAsset.resouceLoader에 위임자를 연결합니다.

```
        let delegate = CustomAssetDelegate()

        func play() {
            let hlsURL = URL(string: "https://deview.kr/2021/sample.m3u8")!
            let redirectURL = URL(string: "chls://deview.kr/2021/sample.m3u8")!
            let urlAsset = AVURLAsset(url: redirectURL)
            urlAsset.resourceLoader.setDelegate(delegate,
                                                queue: DispatchQueue.global(qos: .background))

            let playItem = AVPlayerItem(asset: urlAsset)
            player.replaceCurrentItem(with: playItem)
            player.play()
        }
    }

    class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {}
```

1
2
3

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

자산요청 처리하기

- 자산요청을 대신 수행하기 위해 아래 요청을 대신 수행합니다.

```
class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {

    func resourceLoader(_ resourceLoader: AVAssetResourceLoader,
                        shouldWaitForLoadingOfRequestedResource loadingRequest:
                            AVAssetResourceLoadingRequest) -> Bool {
        loadRequestedResource(loadingRequest)
    }

    func resourceLoader(_ resourceLoader: AVAssetResourceLoader,
                        shouldWaitForRenewalOfRequestedResource renewalRequest:
                            AVAssetResourceRenewalRequest) -> Bool {
        loadRequestedResource(renewalRequest)
    }
```

공통으로 사용할 수 있는 함수
loadRequestedResource(_:)를 설정합니다.

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

자산요청 처리하기

- loadRequestedResource를 통해 API request를 대신 수행합니다.

```
class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {

    func loadRequestedResource(_ loadingRequest: AVAssetResourceLoadingRequest) -> Bool {
        guard let url = loadingRequest.request.url else { return false }

        Call.get(url: url) {
            switch $0 {
            case .success(let data):
                loadingRequest.dataRequest?.respond(with: data)
                loadingRequest.finishLoading()
            case .failure(let error):
                error.printDescription()
                loadingRequest.finishLoading(with: error)
            }
        }
        return true
    }
```

EDIT 가능한 시점,

AVPlayer로 완료응답 전달

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

자산요청 처리하기

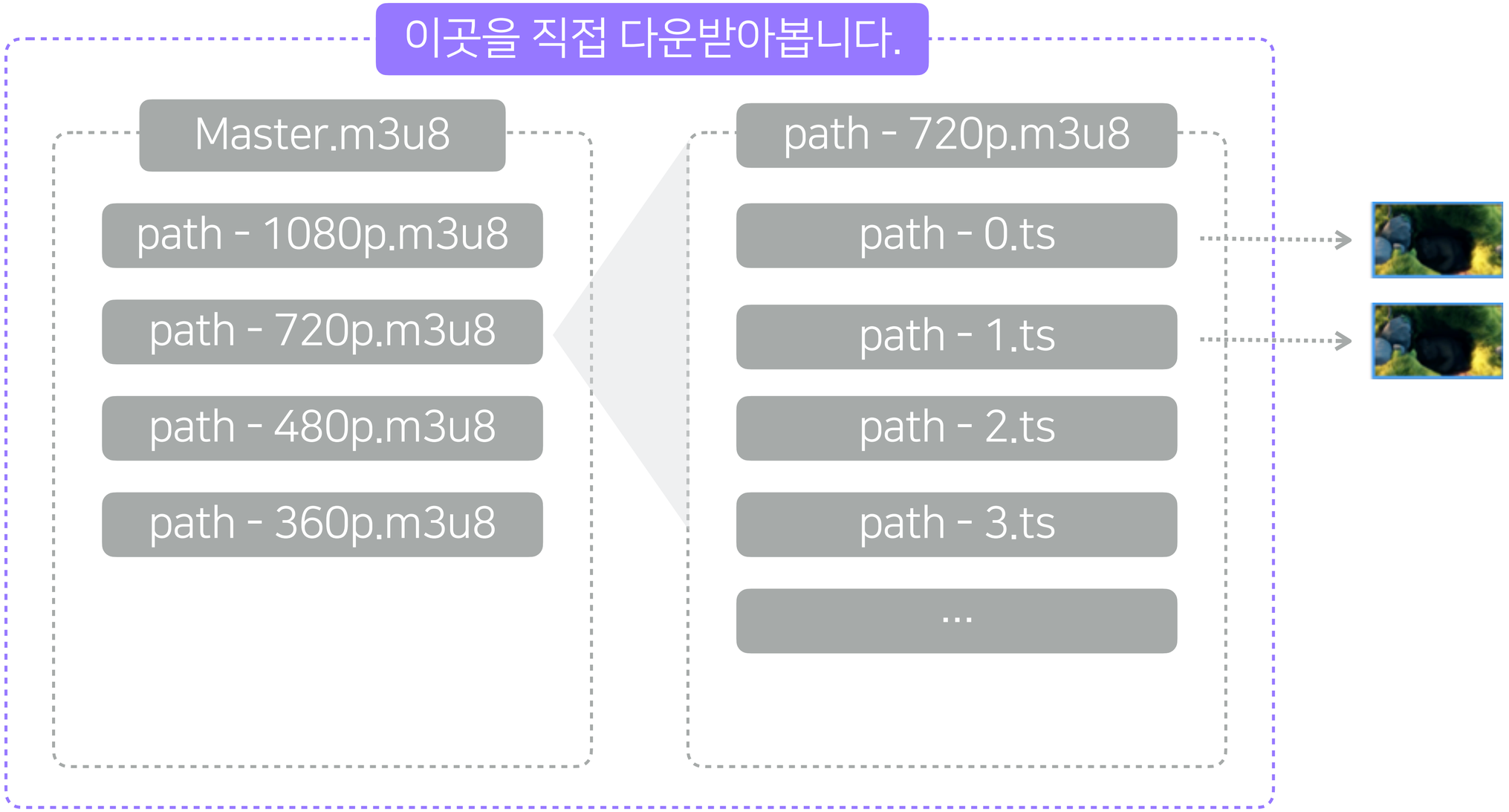
- loadRequestedResource를 통해 API request를 대신 수행합니다.

```
class CustomAssetDelegate: NSObject, AVAssetResourceLoaderDelegate {

    func loadRequestedResource(_ loadingRequest: AVAssetResourceLoadingRequest) -> Bool {
        guard let url = loadingRequest.request.url else { return false }

        Call.get(url: url) {
            switch $0 {
            case .success(let data):
                loadingRequest.dataRequest?.r
                loadingRequest.finishLoading()
            case .failure(let error):
                error.printDescription()
                loadingRequest.finishLoading(w
            }
        }
        return true
    }
```

URL = chls://~~~~
URL = https://~~~

요청을 보내기 전,
origin scheme으로 복원

1.2 커스텀 리소스 로더 설정



1.2 커스텀 리소스 로더 설정

DEVIEW 2021

선택적 로드 수행

1. m3u8 요청만 대신수행
2. TS 요청은 AVPlayer가 처리할 수 있도록 redirect

1

```
guard let url = loadingRequest.request.url e

if url.absoluteString.contains(".m3u8") {
```

```
func loadRequestedResource(_ loadingRequest: AVAssetResourceLoadingRequest) -> Bool {
    guard let url = loadingRequest.request.url else { return false }

    if url.absoluteString.contains(".m3u8") {
        Call.get(url: url) {
            switch $0 {
            case .success(let data):
                loadingRequest.dataRequest?.respond(with: data)
                loadingRequest.finishLoading()
            case .failure(let error):
                error.printDescription()
                loadingRequest.finishLoading(with: error)
            }

                                    = url.change(scheme: "https")
                                t.redirect = URLRequest(url: originURL)
                                t.response = HTTPURLResponse(url: originURL,
                                                             statusCode: 302,
                                                             httpVersion: nil,
                                                             headerFields: nil)
            loadingRequest.finishLoading()
        }

        return true
    }
```

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

선택적 로드 수행

1. m3u8 요청만 대신수행
2. TS 요청은 AVPlayer가 처리할 수 있도록 redirect

```
func loadRequestedResource(_ loadingRequest: AVAssetResourceLoadingRequest) -> Bool {
    guard let url = loadingRequest.request.url else { return false }

1   if url.absoluteString.contains(".m3u8") {
        Call.get(url: url) {
            switch $0 {
            case .success(let data):
                loadingRequest.dataRequest?.respond(with: data)
                loadingRequest.finishLoading()
            case .failure(let error):
2               let originURL = url.change(scheme: "https")
                loadingRequest.redirect = URLRequest(url: originURL)
                loadingRequest.response = HTTPURLResponse(url: originURL,
                                                          statusCode: 302,
                                                          httpVersion: nil,
                                                          headerFields: nil)

                loadingRequest.finishLoading()

        return true
    }
```

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

선택적 로드 수행

1. m3u8 요청만 대신수행
2. TS 요청은 AVPlayer가 처리할 수 있도록 redirect

```
func loadRequestedResource(_ loadingRequest: AVAssetResourceLoadingRequest) -> Bool {
    guard let url = loadingRequest.request.url else { return false }

1   if url.absoluteString.contains(".m3u8") {
        Call.get(url: url) {
            switch $0 {
            case .success(let data):
                loadingRequest.dataRequest?.respond(with: data)
                loadingRequest.finishLoading()
            case .failure(let error):
                error.printDescription()
                loadingRequest.finishLoading(with: error)
            }
        }
    } else {
2       let originURL = url.change(scheme: "https")
        loadingRequest.redirect = URLRequest(url: originURL)
        loadingRequest.response = HTTPURLResponse(url: originURL,
                                                  statusCode: 302,
                                                  httpVersion: nil,
                                                  headerFields: nil)
        loadingRequest.finishLoading()
    }

    return true
}
```

1.2 커스텀 리소스 로더 설정

DEVIEW 2021

주의점

1. ts요청을 대신수행하게 되면 에러 발생
2. .m3u8, .ts 외에도 인증, 자막 관련 요청을 포함합니다.

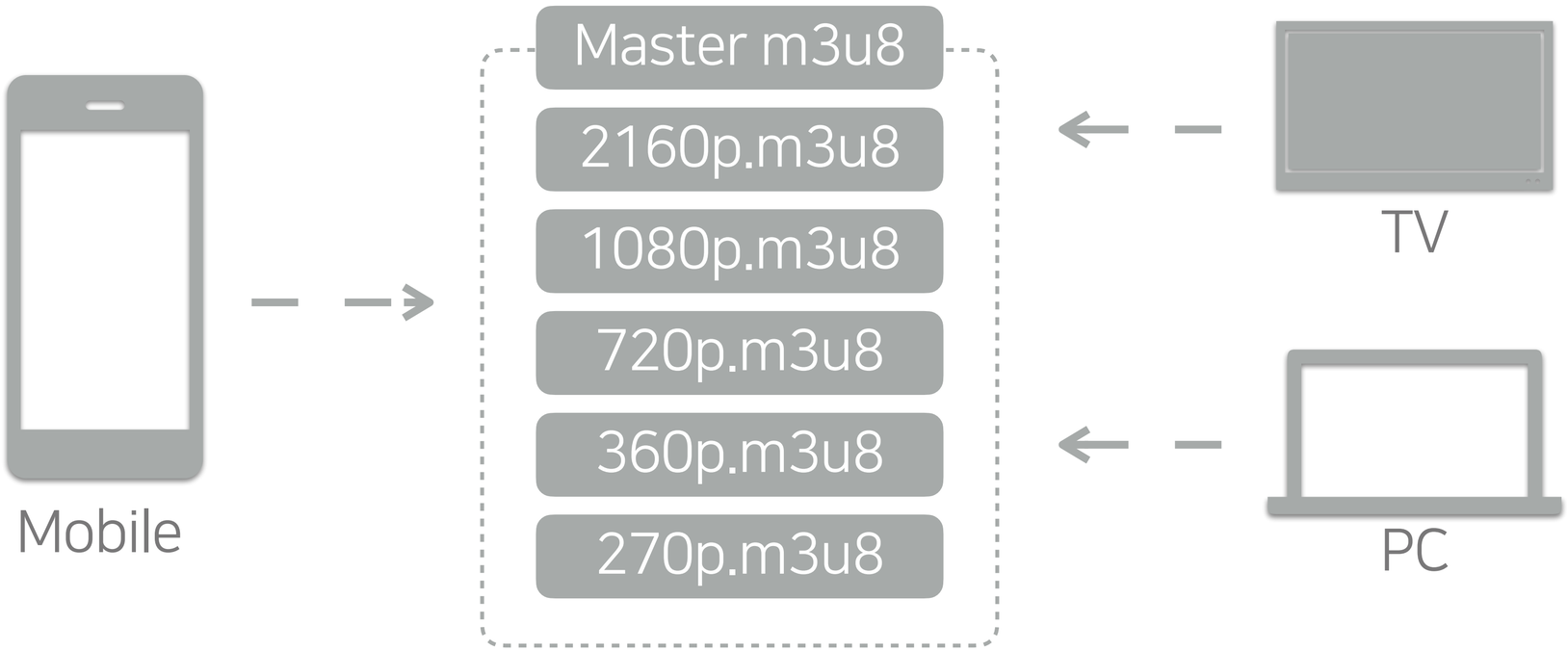
```
func loadRequestedResource(_ loadingRequest: AVAssetResourceLoadingRequest) -> Bool {
    guard let url = loadingRequest.request.url else { return false }

1   if url.absoluteString.contains(".m3u8") {
        Call.get(url: url) {
            switch $0 {
            case .success(let data):
                loadingRequest.dataRequest?.respond(with: data)
                loadingRequest.finishLoading()
            case .failure(let error):
                error.printDescription()
                loadingRequest.finishLoading(with: error)
            }
        }
    } else {
2       let originURL = url.change(scheme: "https")
        loadingRequest.redirect = URLRequest(url: originURL)
        loadingRequest.response = HTTPURLResponse(url: originURL,
                                                  statusCode: 302,
                                                  httpVersion: nil,
                                                  headerFields: nil)
        loadingRequest.finishLoading()
    }

    return true
}
```

That's all 🎉

2. HLS 편집하기

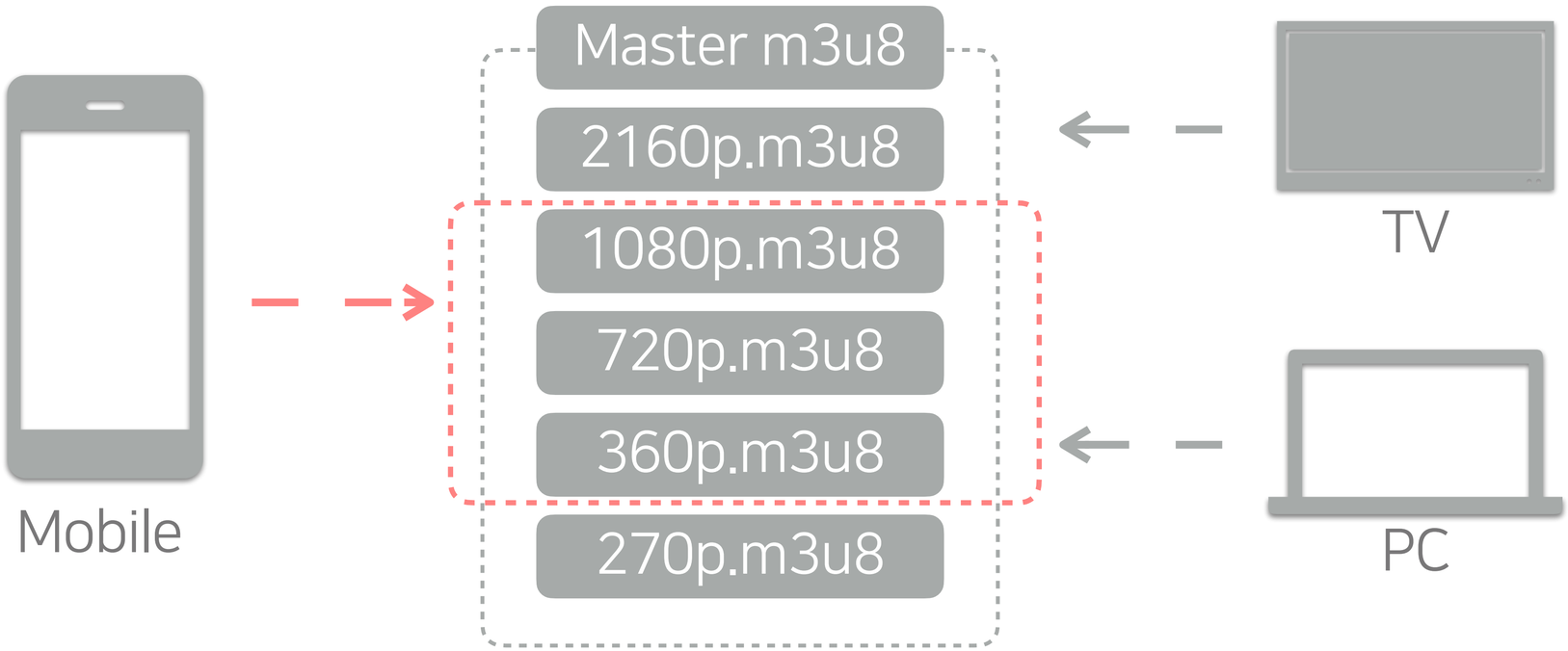
2.1 ABR 화질 제한

ABR (Adaptive bitrate streaming)



2.1 ABR 화질 제한

ABR (Adaptive bitrate streaming)



2.1 ABR 화질 제한

DEVIEW 2021

ABR (Adaptive bitrate streaming)

- 너무 낮은 품질 또는 환경에 맞지 않는 품질을 제거

AS - IS

bipbop_4x3_variant.m3u8 — 편집됨

```
#EXTM3U


#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=232370,CODECS="mp4a.40.2, avc1.4d4015"
gear1/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=649879,CODECS="mp4a.40.2, avc1.4d401e"
gear2/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=41457,CODECS="mp4a.40.2"
gear0/prog_index.m3u8
```

TO - BE

bipbop_4x3_variant.m3u8 — 편집됨

```
#EXTM3U

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8
```

2.1 ABR 화질 제한

MediaComponent

bandwidth, codec

m3u8 path

bipbop_4x3_variant.m3u8 — 편집됨

```
#EXTM3U

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=232370,CODECS="mp4a.40.2, avc1.4d4015"
gear1/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=649879,CODECS="mp4a.40.2, avc1.4d401e"
gear2/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=41457,CODECS="mp4a.40.2"
gear0/prog_index.m3u8
```

2.1 ABR 화질 제한

DEVIEW 2021

```
let bandWidths = 900_000…4_000_000
return components
    .filters { bandWidths .contains($0.bandwidth) }
    .appendHeaderBottoms
    .joined(separator: "₩n")
```

```
1.4d4015"

1.4d401e"

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=41457,CODECS="mp4a.40.2"
gear0/prog_index.m3u8
```

2.1 ABR 화질 제한

bipbop_4x3_variant.m3u8 — 편집됨

```
#EXTM3U

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8
```

2.1 ABR 화질 제한

DEVIEW 2021

ABR (Adaptive bitrate streaming)

- 너무 낮은 품질 또는 환경에 맞지 않는 품질을 제거

AS - IS

bipbop_4x3_variant.m3u8 — 편집됨

```
#EXTM3U


#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=232370,CODECS="mp4a.40.2, avc1.4d4015"
gear1/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=649879,CODECS="mp4a.40.2, avc1.4d401e"
gear2/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=41457,CODECS="mp4a.40.2"
gear0/prog_index.m3u8
```

TO - BE

bipbop_4x3_variant.m3u8 — 편집됨

```
#EXTM3U

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8
```

2.2 미디어 길이 제한

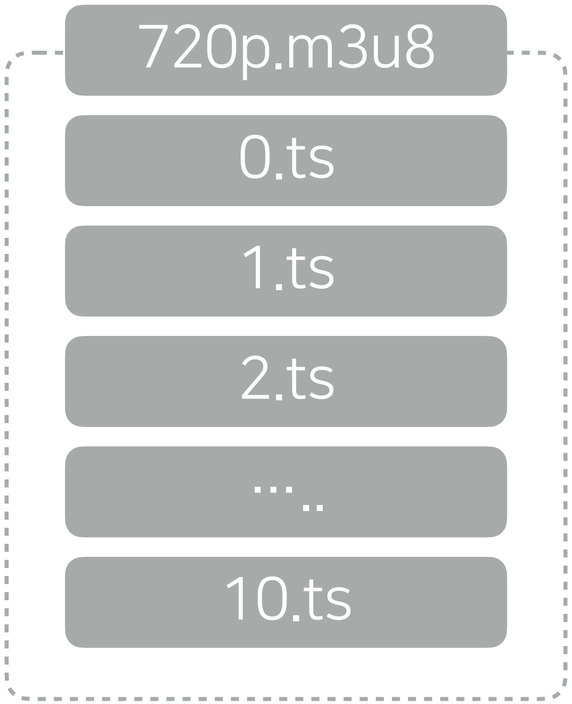
DEVIEW 2021

미디어의 일정부분만 재생

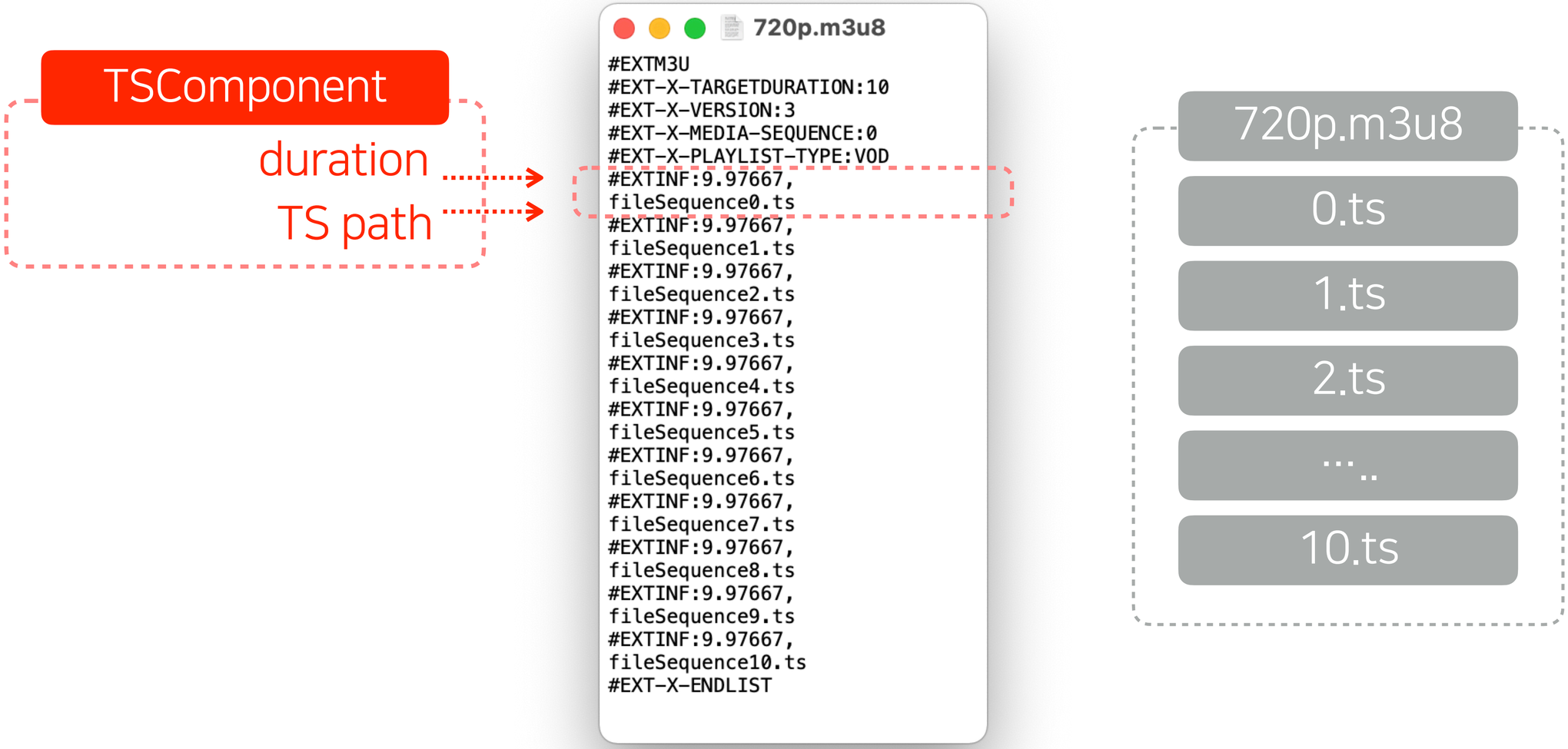
- 미리보기 제한된 시청
- 미디어 길이 109.74s -> 30s

720p.m3u8

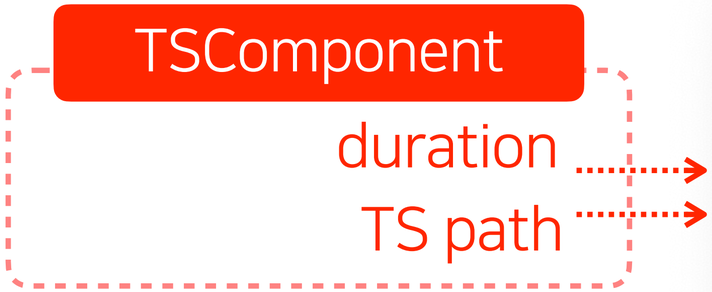
```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXTINF:9.97667,
fileSequence3.ts
#EXTINF:9.97667,
fileSequence4.ts
#EXTINF:9.97667,
fileSequence5.ts
#EXTINF:9.97667,
fileSequence6.ts
#EXTINF:9.97667,
fileSequence7.ts
#EXTINF:9.97667,
fileSequence8.ts
#EXTINF:9.97667,
fileSequence9.ts
#EXTINF:9.97667,
fileSequence10.ts
#EXT-X-ENDLIST
```



2.2 미디어 길이 제한


TSComponent
duration
TS path
720p.m3u8
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXTINF:9.97667,
fileSequence3.ts
#EXTINF:9.97667,
fileSequence4.ts
#EXTINF:9.97667,
fileSequence5.ts
#EXTINF:9.97667,
fileSequence6.ts
#EXTINF:9.97667,
fileSequence7.ts
#EXTINF:9.97667,
fileSequence8.ts
#EXTINF:9.97667,
fileSequence9.ts
#EXTINF:9.97667,
fileSequence10.ts
#EXT-X-ENDLIST
720p.m3u8
0.ts
1.ts
2.ts
….
10.ts

2.2 미디어 길이 제한



720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXTINF:9.97667,
fileSequence3.ts
#EXTINF:9.97667,
fileSequence4.ts
#EXTINF:9.97667,
fileSequence5.ts
#EXTINF:9.97667,
fileSequence6.ts
#EXTINF:9.97667,
fileSequence7.ts
#EXTINF:9.97667,
fileSequence8.ts
#EXTINF:9.97667,
fileSequence9.ts
#EXTINF:9.97667,
fileSequence10.ts
#EXT-X-ENDLIST
```

```
let maxDuration = 30
return components
    .filters {
        $0.duration < maxDuration
        maxDuration - $0.duration
    }
    .appendHeaderBottoms
    .joined(separator: "₩n")
```

2.2 미디어 길이 제한

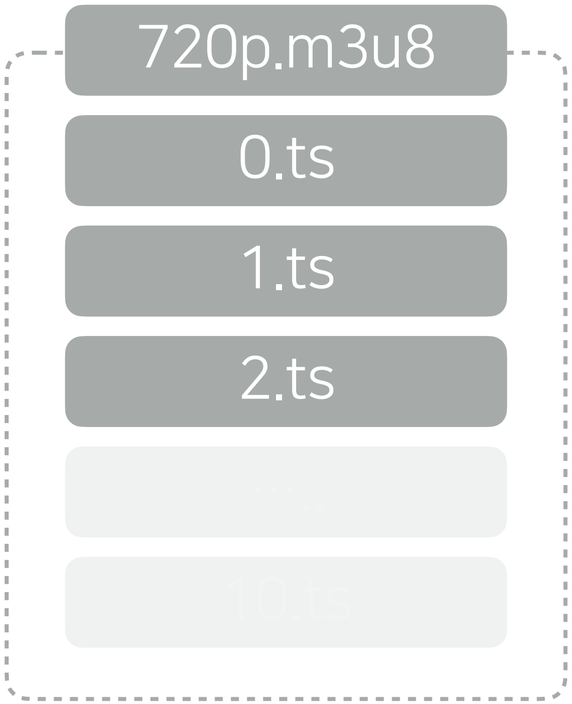
DEVIEW 2021

미디어의 일정부분만 재생

- 30초 미리보기
- 미디어 길이 29.9s

720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXT-X-ENDLIST
```



2.3 미디어 합성

DEVIEW 2021

서로다른 비디오 합성

- 광고와 컨텐츠를 합성
- 다른 컨텐츠를 하나의 타임라인에

720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXT-X-ENDLIST
```

720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:4.0,
index0.ts
#EXTINF:4.0,
index1.ts
#EXTINF:4.0,
index2.ts
#EXTINF:4.0,
index3.ts
#EXTINF:4.0,
index4.ts
#EXT-X-ENDLIST
```

2.3 미디어 합성

DEVIEW 2021

#EXT-X-TARGETDURATION

- 미디어 중 큰 미디어 값으로

720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
equence1.ts
NF:9.97667,
equence2.ts
X-ENDLIST
```

720p.m3u8

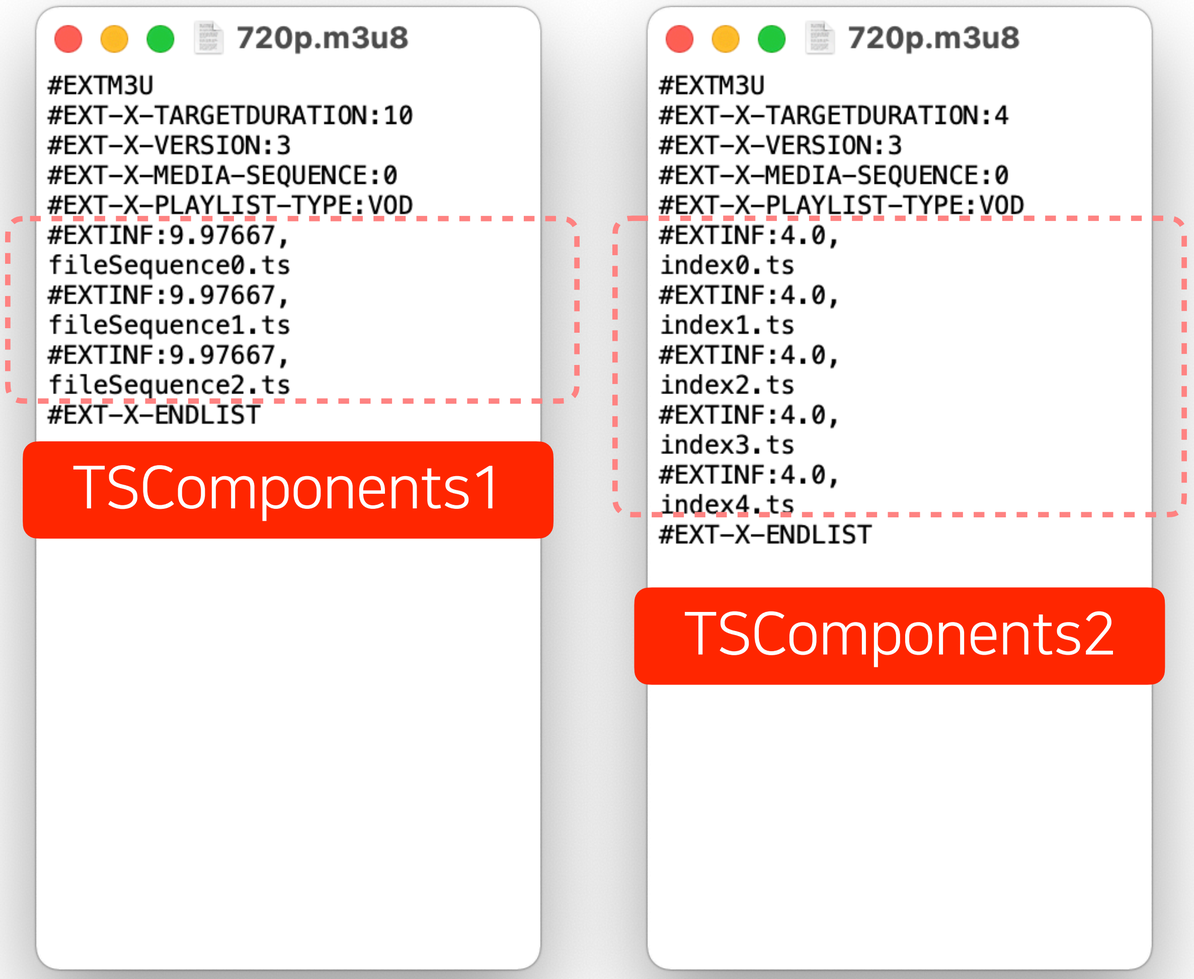
```
#EXTM3U
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:4.0,
index0.ts
#EXTINF:4.0,
index1.ts
#EXTINF:4.0,
index2.ts
#EXTINF:4.0,
index3.ts
#EXTINF:4.0,
index4.ts
#EXT-X-ENDLIST
```

2.3 미디어 합성

DEVIEW 2021

#EXT-X-TARGETDURATION

- 미디어 중 큰 미디어 값으로



2.3 미디어 합성

DEVIEW 2021

#EXT-X-DISCONTINUITY

- 미디어의 구분

```
let mediaComponents1
let mediaComponents2

return mediaComponents1
    .append("EXT-X-DISCONTINUITY")
    .append(contentsOf: mediaComponents2)
    .appendHeaderBottoms
    .joined(separator: "₩n")
```

720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXT-X-ENDLIST
```

720p.m3u8

```
#EXTM3U
#EXT-X-TARGETDURATION:4
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:4.0,
index0.ts
#EXTINF:4.0,
index1.ts
#EXTINF:4.0,
index2.ts
#EXTINF:4.0,
index3.ts
#EXTINF:4.0,
index4.ts
#EXT-X-ENDLIST
```

2.3 미디어 합성

DEVIEW 2021

서로다른 비디오 합성

- 미디어 길이 49.9s

참고사항

- TARGETDURATION은 더 큰 값에
- 컨텐츠 변경시점엔 DISCONTINUITY 태그를 추가

720p.m3u8

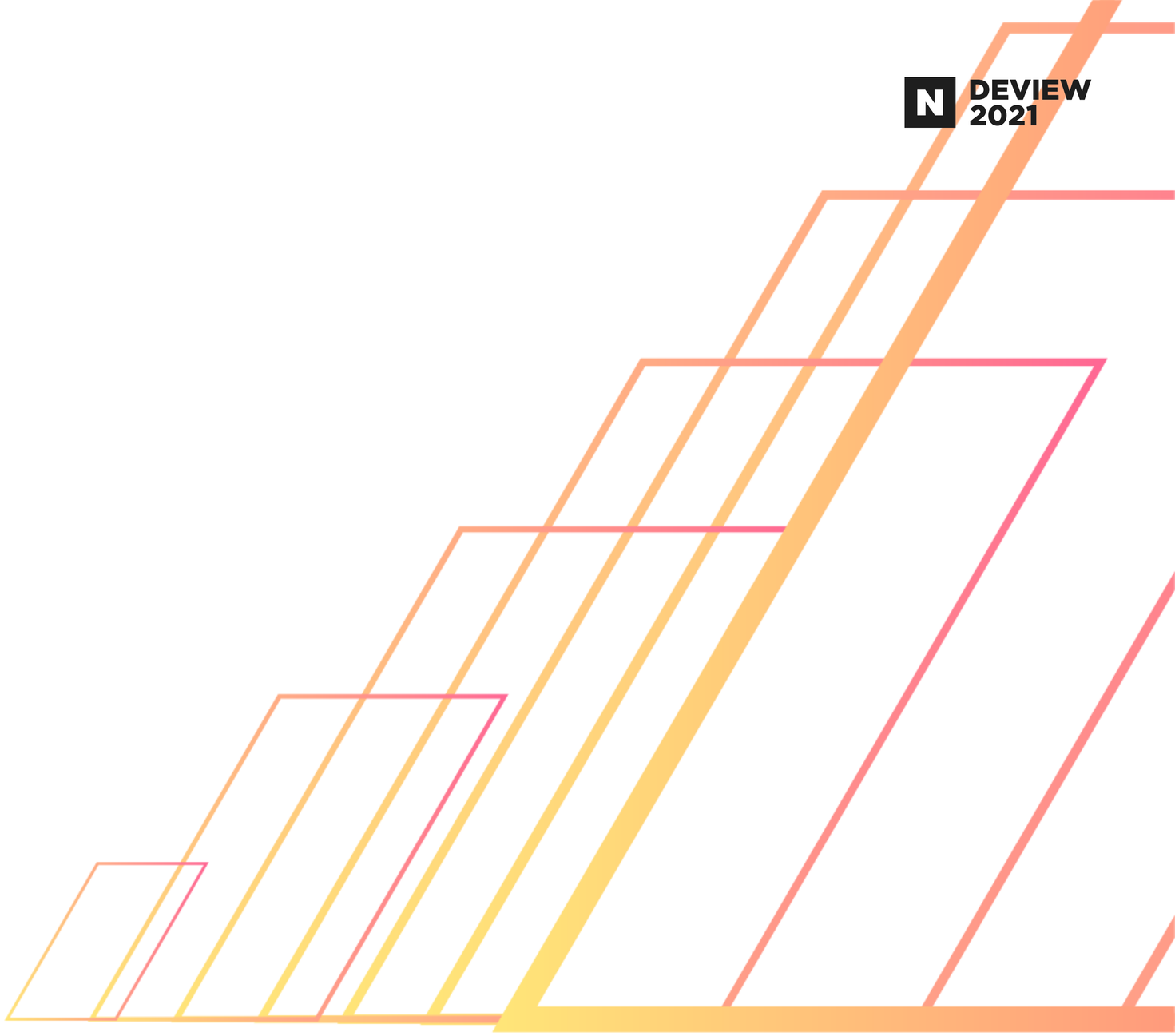
```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXT-X-DISCONTINUITY
#EXTINF:4.0,
index0.ts
#EXTINF:4.0,
index1.ts
#EXTINF:4.0,
index2.ts
#EXTINF:4.0,
index3.ts
#EXTINF:4.0,
index4.ts
#EXT-X-ENDLIST
```

MediaComponent1

MediaComponent2

HLS 편집

- edit master.m3u8
- edit media.m3u8
- merge media.m3u8



3. DASH 편집
(미 지원 프로토콜)

3.1 DASH의 이해

DEVIEW 2021

MPEG-DASH 특징

- AVPlayer에서 지원되지 않는 포맷

```
// 2. Play DASH Video
func playDASHVideo() {
    guard let dashURL = Test.Path.dash.toURL else { return }

    let urlAsset = AVURLAsset(url: dashURL);

    play(playItem: AVPlayerItem(asset: urlAsset))
}
```

```
Error occurred: Operation Stopped
Domain: AVFoundationErrorDomain
Code: -11850
```

3.1 DASH의 이해

DASH 특징

- MPEG, XML

```
#EXTM3U


#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=232370,CODECS="mp
gear1/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=649879,CODECS="mp
gear2/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="m
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=41457,CODECS="mp4
gear0/prog_index.m3u8
```

HLS - m3u8

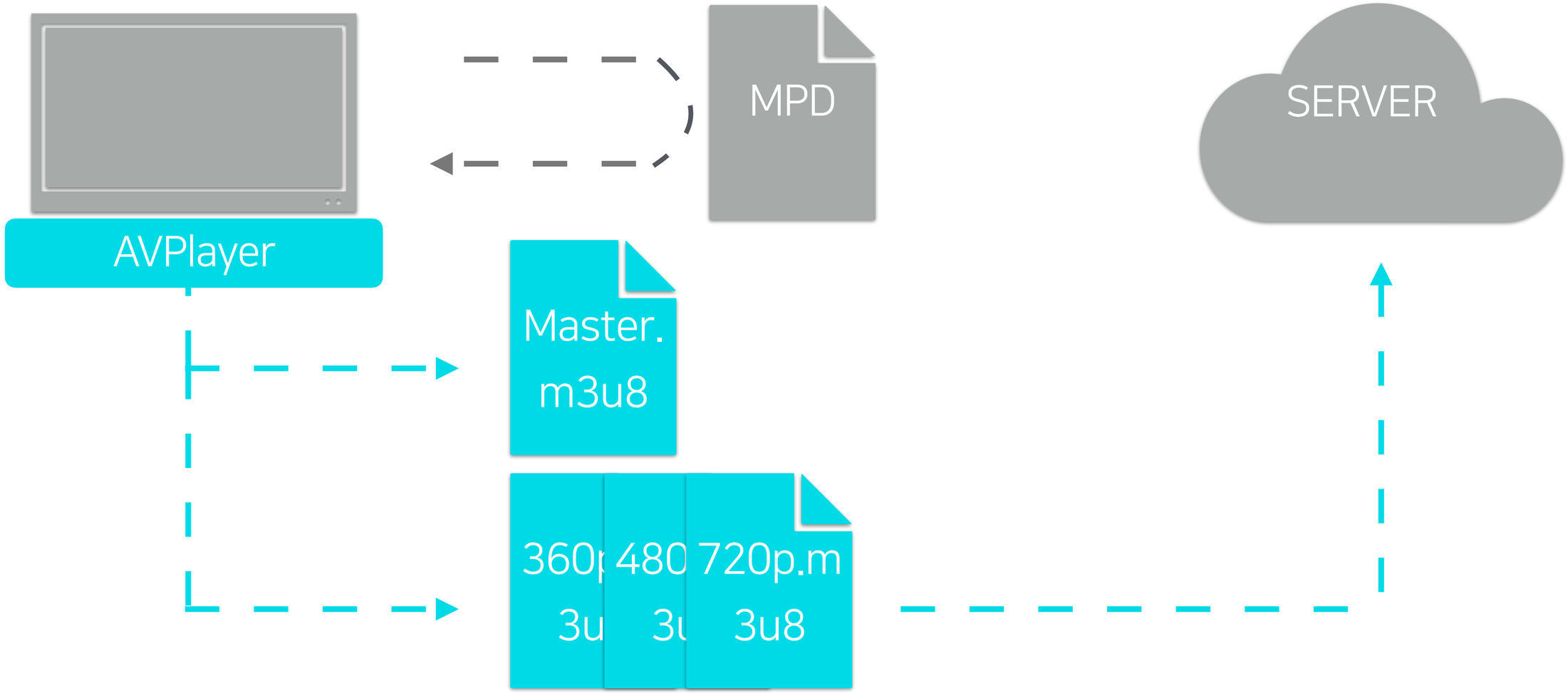
```
<MPD mediaPresentationDuration="PT634.566S"
    minBufferTime="PT2.00S" profiles="urn:hbbtv:dash
    :profile:isoff-live:2012,urn:mpeg:dash:profile
    :isoff-live:2011" type="static"
    xmlns="urn:mpeg:dash:schema:mpd:2011"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema
        -instance" xsi:schemaLocation="urn:mpeg:DASH
        :schema:MPD:2011 DASH-MPD.xsd">
    <BaseURL>./</BaseURL>
    <Period>
        <AdaptationSet mimeType="video/mp4"
```

DASH - mpd

3.2 HLS Format ideation

DEVIEW 2021

3.2 HLS Format ideation



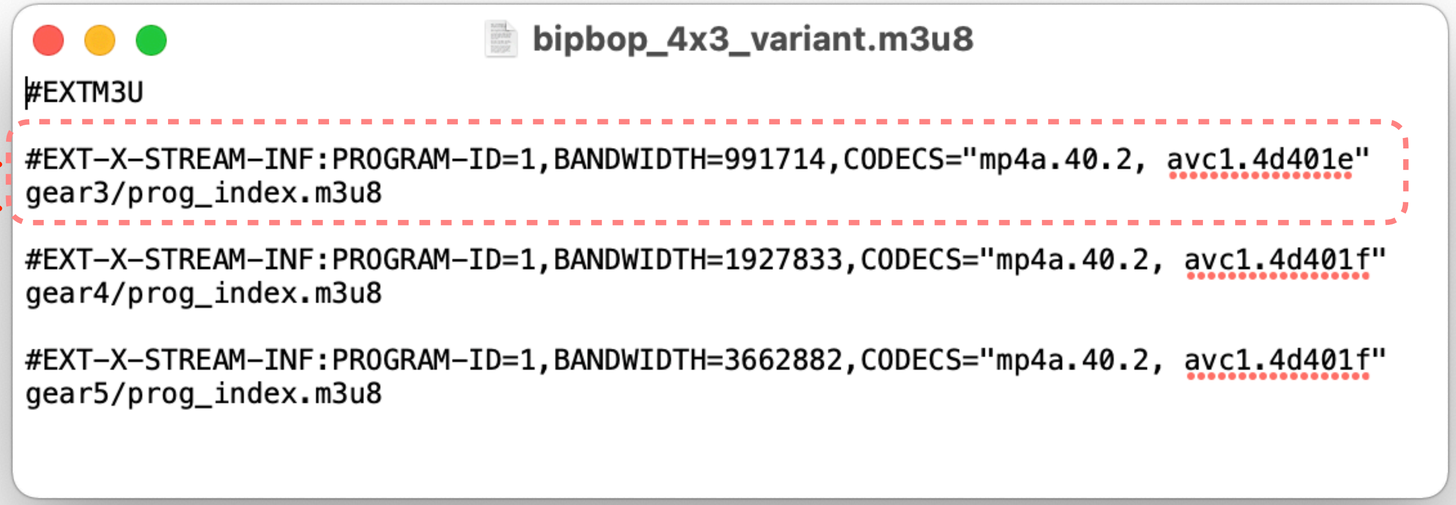
3.2 HLS Format ideation

DEVIEW 2021

MediaComponent

bandwidth, codec

m3u8 path



3.2 HLS Format ideation

DEVIEW 2021

MediaComponent

bandwidth, codec

m3u8 path

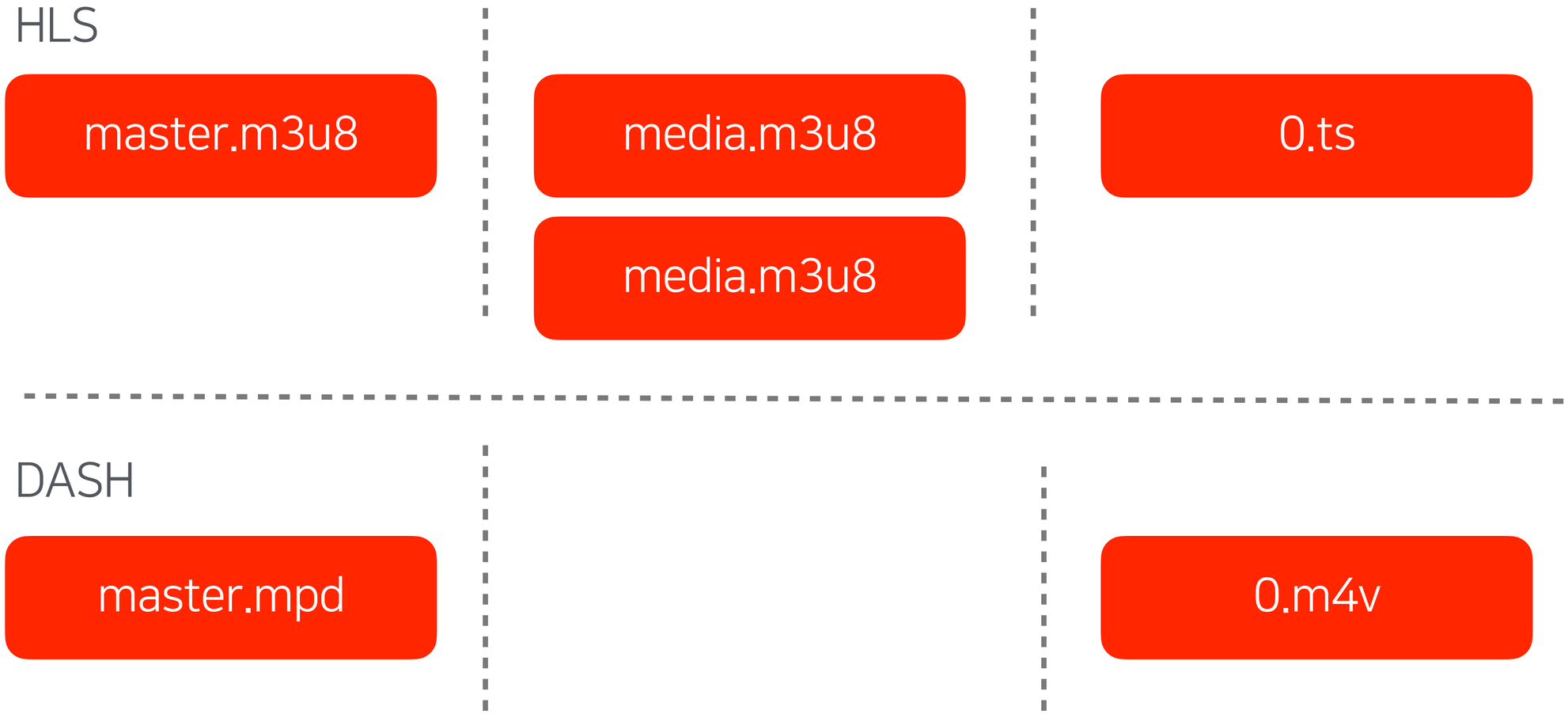
```
<MPD mediaPresentationDuration="PT634.566S" minBufferTime="PT2.00S"
    profiles="urn:hbbtv:dash:profile:isoff-live:2012,urn:mpeg:dash
    :profile:isoff-live:2011" type="static"
    xmlns="urn:mpeg:dash:schema:mpd:2011"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
        :schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd">
    <BaseURL>./</BaseURL>
    <Period>
        <AdaptationSet mimeType="video/mp4" contentType="video"
            subsegmentAlignment="true" subsegmentStartsWithSAP="1" par
            ="16:9">
            <SegmentTemplate duration="120" timescale="30" media
                ="$RepresentationID$/$RepresentationID$_$Number$.m4v"
                startNumber="1" initialization="$RepresentationID$
                /$RepresentationID$_0.m4v"/>
            <Representation id="bbb_30fps_1024x576_2500k" codecs="avc1
                .64001f" bandwidth="3134488" width="1024" height="576"
                frameRate="30" sar="1:1" scanType="progressive"/>
            <Representation id="bbb_30fps_1280x720_4000k" codecs="avc1
                .64001f" bandwidth="4952892" width="1280" height="720"
                frameRate="30" sar="1:1" scanType="progressive"/>
            <Representation id="bbb_30fps_1920x1080_8000k" codecs="avc1
                .640028" bandwidth="9914554" width="1920" height="1080"
                frameRate="30" sar="1:1" scanType="progressive"/>
            <Representation id="bbb_30fps_320x180_200k" codecs="avc1
```

codecs

bandwidth

```
<Representation id="bbb_30fps_1024x576_2500k" codecs="avc1
    .64001f" bandwidth="3134488" width="1024" height="576"
    frameRate="30" sar="1:1" scanType="progressive"/>
```

3.2 HLS Format ideation



3.2 HLS Format ideation

DEVIEW 2021

```
return xml
    .filter { $0.tag == “Representation” }
    .makeM3UAttribute($0, basePath: “₩($0.height).m3u8”)
    .makeM3UHeader
    .joined(separator: "₩n")
```

```
<MPD mediaPresentationDuration="PT634.566S" minBufferTime="PT2.00S"
    profiles="urn:hbbtv:dash:profile:isoff-live:2012,urn:mpeg:dash
    :profile:isoff-live:2011" type="static"
    xmlns="urn:mpeg:dash:schema:mpd:2011"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
        :schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd">
    <BaseURL>./</BaseURL>
    <Period>
        <AdaptationSet mimeType="video/mp4" contentType="video"
            subsegmentAlignment="true" subsegmentStartsWithSAP="1" par
            ="16:9">
            <SegmentTemplate duration="120" timescale="30" media
                presentationID$/$RepresentationID$_$Number$.m4v"
                Number="1" initialization="$RepresentationID$
                resentationID$_0.m4v"/>
                tation id="bbb_30fps_1024x576_2500k" codecs="avc1
                1f" bandwidth="3134488" width="1024" height="576"
                Rate="30" sar="1:1" scanType="progressive"/>
                tation id="bbb_30fps_1280x720_4000k" codecs="avc1
                1f" bandwidth="4952892" width="1280" height="720"
                Rate="30" sar="1:1" scanType="progressive"/>
                tation id="bbb_30fps_1920x1080_8000k" codecs="avc1
                28" bandwidth="9914554" width="1920" height="1080"
                Rate="30" sar="1:1" scanType="progressive"/>
                tation id="bbb_30fps_320x180_200k" codecs="avc1
                0d" bandwidth="254320" width="320" height="180"
                    frameRate="30" sar="1:1" scanType="progressive"/>
        </AdaptationSet>
    </Period>
</MPD>
```

3.2 HLS Format ideation

DEVIEW 2021

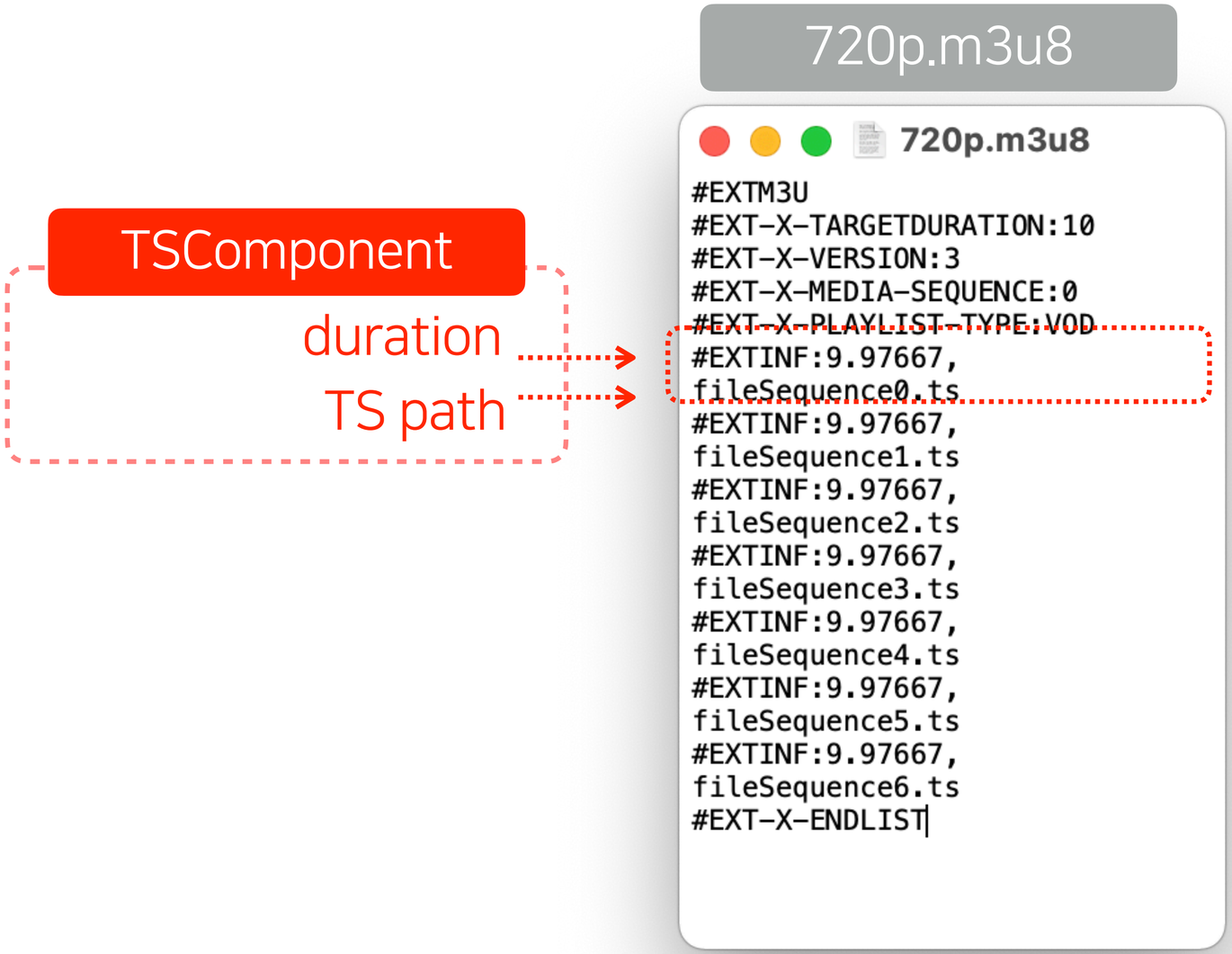
```
return xml
    .filter { $0.tag == “Representation” }
    .makeM3UAttribute($0, basePath: “₩($0.heigh
    .makeM3UHeader
    .joined(separator: "₩n")
```

Master.m3u8

bipbop_4x3_variant.m3u8

```
#EXTM3U

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=991714,CODECS="mp4a.40.2, avc1.4d401e"
gear3/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1927833,CODECS="mp4a.40.2, avc1.4d401f"
gear4/prog_index.m3u8

#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=3662882,CODECS="mp4a.40.2, avc1.4d401f"
gear5/prog_index.m3u8
```

3.2 HLS Format ideation



3.2 HLS Format ideation

total_duration

TSComponent

duration

TS path

```
<MPD mediaPresentationDuration="PT634.566S"
    profiles="urn:hbbtv:dash:profile:isoff-l
    :profile:isoff-live:2011" type="static"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
    :schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd">
    <BaseURL>./</BaseURL>
    <Period>
        <AdaptationSet mimeType="video/mp4" contentType="video"
            subsegmentAlignment="true" subsegmentStartsWithSAP="1" par
            ="16:9">
            <SegmentTemplate duration="120" timescale="30" media
                ="$RepresentationID$/$RepresentationID$_$Number$.m4v"
                startNumber="1" initialization="$RepresentationID$
                /$RepresentationID$_0.m4v"/>
            <Representation id="bbb_30fps_1024x576_2500k" codecs="avc1
                .64001f" bandwidth="3134488" width="1024" height="576"
                frameRate="30" sar="1:1" scanType="progressive"/>
            <Representation id="bbb_30fps_1280x720_4000k" codecs="avc1
                .64001f" bandwidth="4952892" width="1280" height="720"
                frameRate="30" sar="1:1" scanType="progressive"/>
```

duration

```
<SegmentTemplate duration="120" timescale="30" media
    ="$RepresentationID$/$RepresentationID$_$Number$.m4v"
    startNumber="1" initialization="$Representa
    /$RepresentationID$_0.m4v"/>
```

path

```
</MPD>
```

3.2 HLS Format ideation

DEVIEW 2021

```
let filePath = xml.segmentTamplate.media
let maxDuration = xml.mediaPresentationDuration

return xml.Representation[720]
    .makeMediaM3U(maxDuration: maxDuration,
                  path: filePath)
    .makeM3UHeaderBottoms
    .joined(separator: "₩n")
```

duration

total_duration

```
<MPD mediaPresentationDuration="PT634.566S"
  profiles="urn:hbbtv:dash:profile:isoff-l
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi
  :schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd">
<BaseURL>./</BaseURL>
<Period>
  <AdaptationSet mimeType="video/mp4" contentType="video"
    subsegmentAlignment="true" subsegmentStartsWithSAP="1" par
    ="16:9">
    <SegmentTemplate duration="120" timescale="30" media
      ="$RepresentationID$/$RepresentationID$_$Number$.m4v"
      startNumber="1" initialization="$RepresentationID$
      /$RepresentationID$_0.m4v"/>
    <Representation id="bbb_30fps_1024x576_2500k" codecs="avc1
      .64001f" bandwidth="3134488" width="1024" height="576"
      frameRate="30" sar="1:1" scanType="progressive"/>
    <Representation id="bbb_30fps_1280x720_4000k" codecs="avc1
      .64001f" bandwidth="4952892" width="1280" height="720"
      frameRate="30" sar="1:1" scanType="progressive"/>
duration="120" timescale="30" media
="$RepresentationID$/$RepresentationID$_$Number$.m4v"
startNumber="1" initialization="$Representa
/$RepresentationID$_0.m4v"/>
</MPD>
```

path

3.2 HLS Format ideation

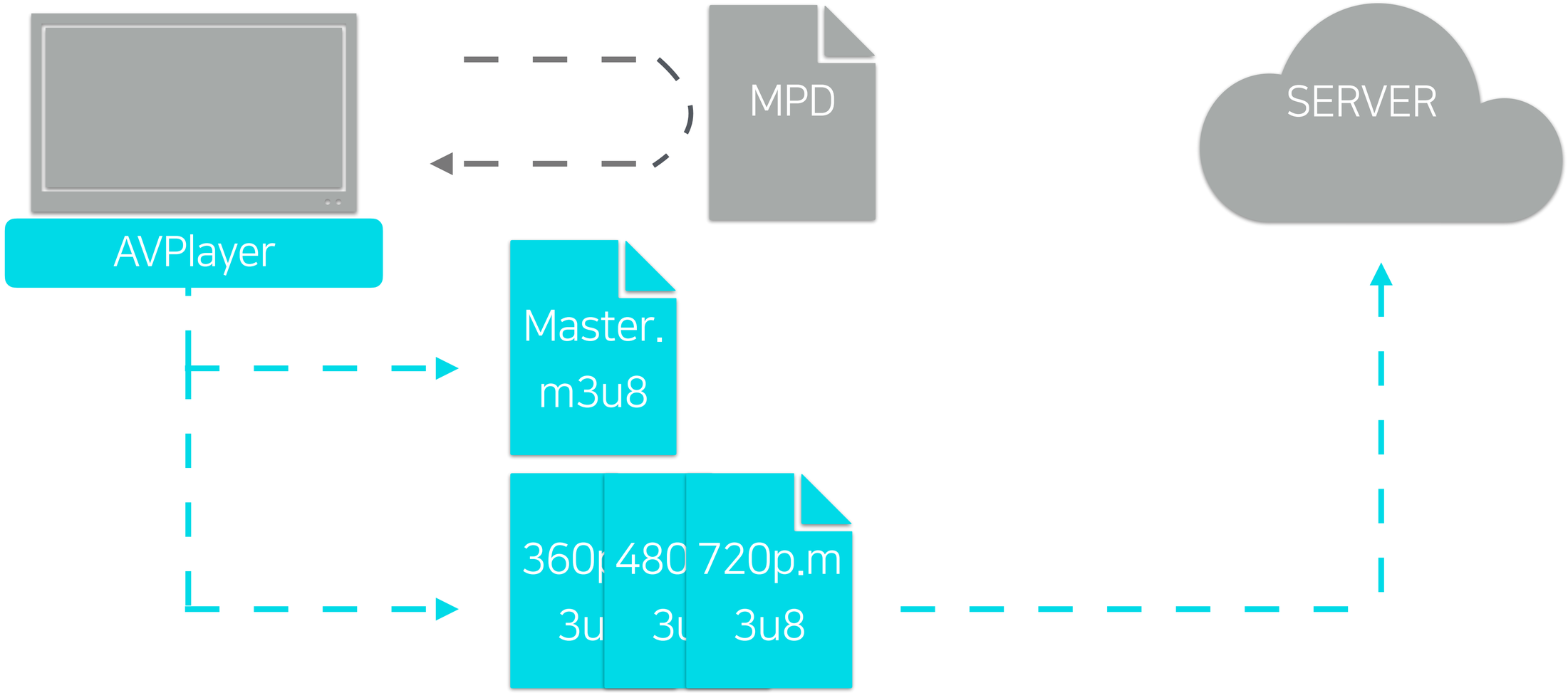
DEVIEW 2021

```
let filePath = xml.segmentTamplate.media
let maxDuration = xml.mediaPresentationDuration

return xml.Representation[720]
    .makeMediaM3U(maxDuration: maxDuration,
                  path: filePath)
    .makeM3UHeaderBottoms
    .joined(separator: "₩n")
```

720p.m3u8

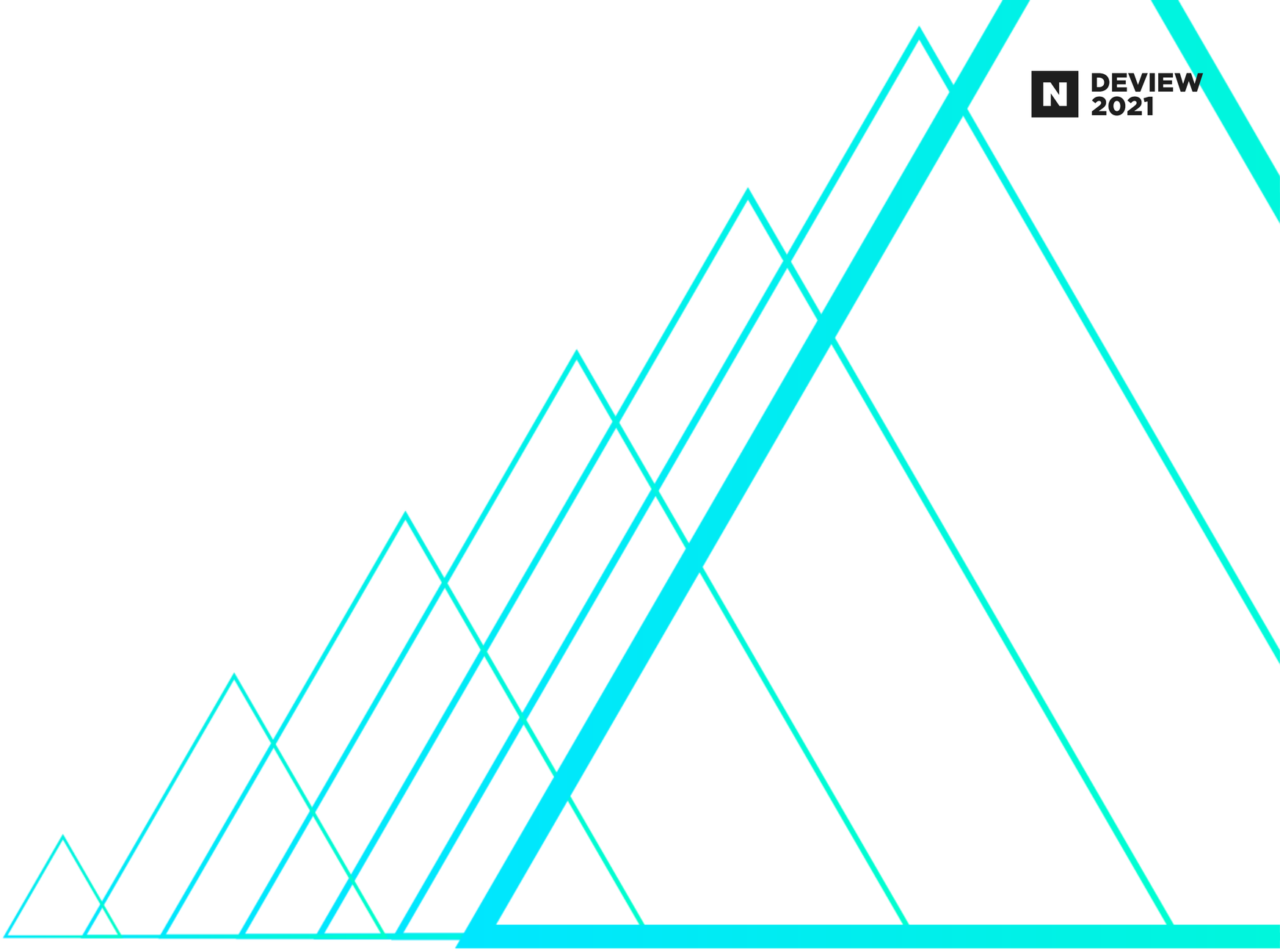
```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:9.97667,
fileSequence0.ts
#EXTINF:9.97667,
fileSequence1.ts
#EXTINF:9.97667,
fileSequence2.ts
#EXTINF:9.97667,
fileSequence3.ts
#EXTINF:9.97667,
fileSequence4.ts
#EXTINF:9.97667,
fileSequence5.ts
#EXTINF:9.97667,
fileSequence6.ts
#EXT-X-ENDLIST
```

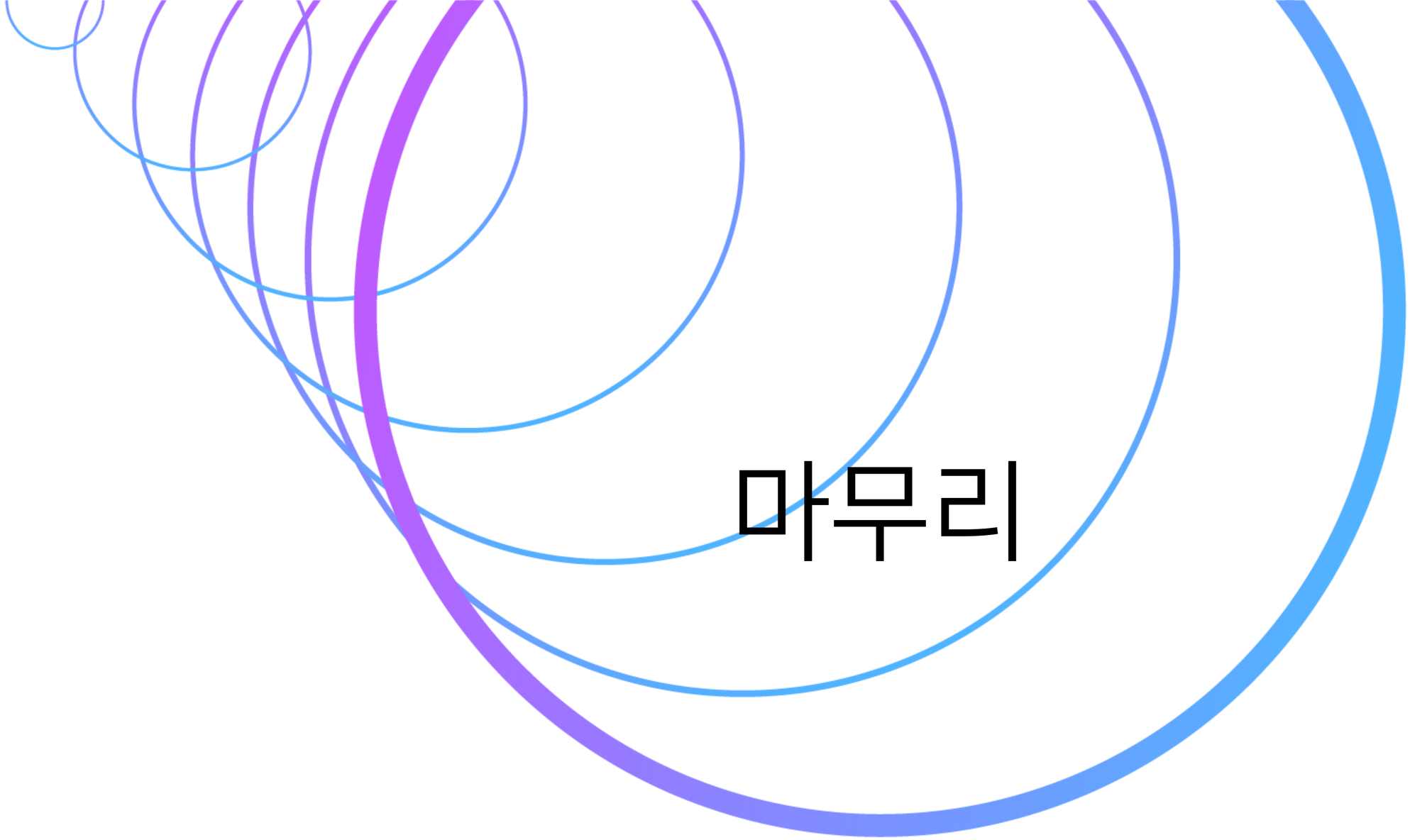
3.2 HLS Format ideation



HLS 생성

- mpd to m3u8





마무리

미디어 재생시 iOS 커스터마이징 방법

- AVPlayer CustomResourceLoader
- CustomPlayer
- Local Proxy

미디어 재생시 iOS 커스터마이징 방법

- AVPlayer CustomResourceLoader
- CustomPlayer
- Local Proxy

CustomResourceLoader

- Custom Scheme
- AVAssetResourceLoaderDelegate
- HLS m3u8 수정으로 다양한 커스터마이징

Demo 소스코드

NAVER
Emerging
TECHnology

함께 성장하실 동료분을 모십니다

DEVIEW 2021

FE / BE / 앱 / SDK 개발 기술과 함께 멀티미디어의 핵심 기술을 배우고,

포토 / 오디오 / 비디오 / UGC 기술 도메인 전문가로 성장할 수 있도록 적극 지원하겠습니다.

ETECH 직무 소개

- ETECH 직무 소개 : https://naver-career.gitbook.io/kr/service/etech
- ETECH 기술 문의 : etech@navercorp.com
- ETECH 채용 문의 : etech-recruit@navercorp.com

Thank You