Live video streaming in the FIFA World Cup

Juanjo Ramos
Hudl

Hi, I'm Troy McClure.

HUDL SELECTED AS REPLAY SOLUTION PROVIDER FOR THE FIFA WOMEN'S WORLD CUP FRANCE 2019™

FIFA continues to develop in-game technology solutions.

For the upcoming FIFA Women's World Cup France 2019™ (7 June – 7 July), Hudl has been selected as Replay Solution provider.

Hudl Replay software will be used by team analysts to support their tactical decisions as well as the medical representatives to evaluate and assess potential injuries.

Hudl Replay

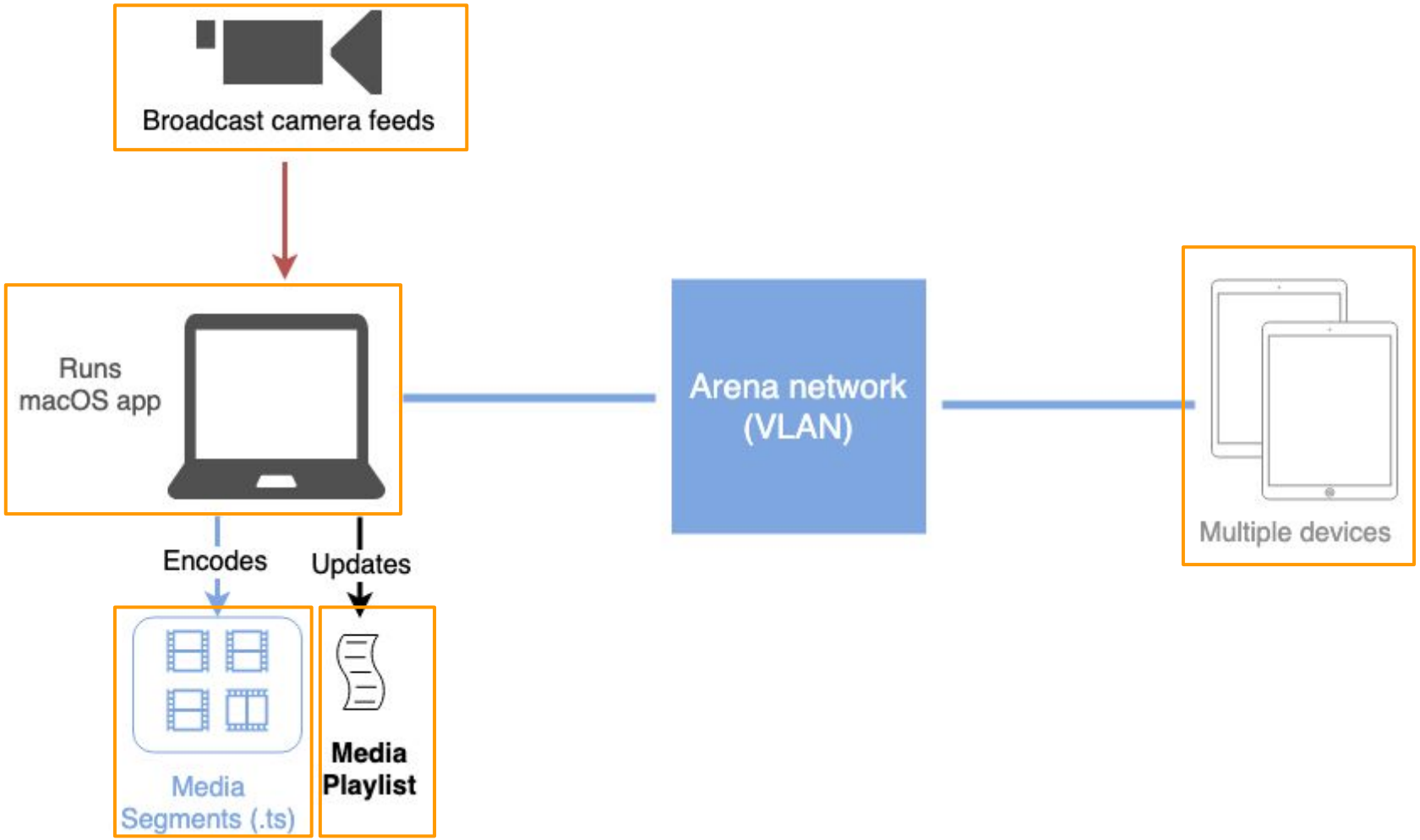
- **iOS** app (iPad only)
- **Live** video streaming client
- Server on the same (V/W)**LAN**
- It uses **HLS**

What's HLS and what do you do with it?

HLS

- HTTP Live Streaming
- Adaptive bitrate streaming protocol
- Implemented by **Apple**

HLS in action


Broadcast camera feeds
Runs
macOS app
Encodes
Updates
Media
Segments (.ts)
Media
Playlist
Arena network
(VLAN)
Multiple devices

(Master) Playlist

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-START:TIME-OFFSET=-60


#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="English",DEFAULT=YES,AUTOSELECT=YES,FORCED=NO,LANGUAGE="en",URI="https://apple-events.akamaized.
net/hls/live/681800/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/cc/eng.m3u8"
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs-b",NAME="English",DEFAULT=YES,AUTOSELECT=YES,FORCED=NO,LANGUAGE="en",URI="https://apple-events.akamaized
.net/hls/live/681800-b/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/cc/eng.m3u8"


#EXT-X-STREAM-INF:BANDWIDTH=1343570,AVERAGE-BANDWIDTH=1390400,CODECS="avc1.4d401e,mp4a.40.2",RESOLUTION=640x360,FRAME-RATE=29.970,SUBTITLES="subs"
https://apple-events.akamaized.net/hls/live/681800/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/1200/1200.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=567560,AVERAGE-BANDWIDTH=660000,CODECS="avc1.4d401e",RESOLUTION=640x360,URI="https://apple-events.akamaized.
net/hls/live/681800/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/1200/1200_I-Frame.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=1343570,AVERAGE-BANDWIDTH=1390400,CODECS="avc1.4d401e,mp4a.40.2",RESOLUTION=640x360,FRAME-RATE=29.970,SUBTITLES="subs-
b"
https://apple-events.akamaized.net/hls/live/681800-b/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/1200/1200.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=567560,AVERAGE-BANDWIDTH=660000,CODECS="avc1.4d401e",RESOLUTION=640x360,URI="https://apple-events.akamaized.
net/hls/live/681800-b/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/1200/1200_I-Frame.m3u8"

#EXT-X-STREAM-INF:BANDWIDTH=725540,AVERAGE-BANDWIDTH=730400,CODECS="avc1.4d401e,mp4a.40.2",RESOLUTION=640x360,FRAME-RATE=29.970,SUBTITLES="subs"
https://apple-events.akamaized.net/hls/live/681800/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/0600/0600.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=283780,AVERAGE-BANDWIDTH=330000,CODECS="avc1.4d401e",RESOLUTION=640x360,URI="https://apple-events.akamaized.
net/hls/live/681800/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/0600/0600_I-Frame.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=725540,AVERAGE-BANDWIDTH=730400,CODECS="avc1.4d401e,mp4a.40.2",RESOLUTION=640x360,FRAME-RATE=29.970,SUBTITLES="subs-b"
https://apple-events.akamaized.net/hls/live/681800-b/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/0600/0600.m3u8
#EXT-X-I-FRAME-STREAM-INF:BANDWIDTH=283780,AVERAGE-BANDWIDTH=330000,CODECS="avc1.4d401e",RESOLUTION=640x360,URI="https://apple-events.akamaized.
net/hls/live/681800-b/0208kmksrrgukmmpvlwqmnzbuhaylttzxoazcamnfmmnni/master/0600/0600_I-Frame.m3u8"
```

Media Playlist

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:EVENT
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-ALLOW-CACHE:YES
#EXTINF:2.000000
0000_00000.ts
#EXTINF:2.000000
0000_00001.ts
#EXTINF:2.000000
0000_00002.ts
#EXTINF:2.000000
0000_00003.ts
#EXTINF:2.000000
0000_00004.ts
#EXTINF:2.000000
0000_00005.ts
#EXTINF:2.000000
0000_00006.ts
#EXTINF:2.000000
0000_00007.ts
#EXTINF:2.000000
0000_00008.ts
```

1. EVENT or VOD

2. Segment duration
3. Segment URI

Server - Client lag >= ~2.5 x segment_duration

Random Access

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-ALLOW-CACHE:YES
#EXTINF:2.000000
0000_00000.ts
#EXTINF:2.000000
0000_00001.ts
#EXTINF:2.000000
0000_00002.ts
#EXTINF:2.000000
0000_00003.ts
#EXTINF:2.000000
0000_00004.ts
#EXTINF:2.000000
0000_00005.ts
#EXTINF:2.000000
0000_00006.ts
#EXTINF:2.000000
0000_00007.ts
#EXT-X-ENDLIST
```

1. VOD

2. Segment duration
3. Segment URI

4. End of list

Hudl Replay in action


Schlotmann
container-owl.de
kran.immo
UNIVERSITÄT PADERBORN
JASPER
Fanartikel in Blau-Schwarz
ZAUN KREISEL
Zäune aller Art
ZAUN-KUHLMANN.DE
„Tore. Siege. Punkte"
Ihre Volksbanken im Kreis Paderborn
BÄCKEREI LANGE
finke
DAS ERLEBNIS-EINRICHTER
MEDIACOM
Warsteiner
10:02
LIVE


LIVE

Our problem

Why does that happen?

1. Return requested segments in a timely fashion
2. Live video ⇒ Playlist file must be updated frequently

Content always available

What didn’t work

01 Remote server only

02 Offline playback

03 Live offline playback

1. Remote server only

```swift
import UIKit
import AVFoundation
import AVKit

class ViewController: AVPlayerViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        let urlString = "https://mnmedias.api.telequebec.tv/m3u8/29880.m3u8"
        let url = URL(string: urlString)!

        let asset = AVURLAsset(url: url)
        let playerItem = AVPlayerItem(asset: asset)
        let player = AVPlayer(playerItem: playerItem)
        self.player = player
    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)

        player?.play()
    }
}

```

2. Offline playback

Sample Code

Using AVFoundation to Play and Persist HTTP Live Streams

Play HTTP Live Streams and preserve streams on disk for offline play back.

SDKs

iOS 11.0+

Xcode 10.0+

https://apple.co/2kxetI3

```
func setupAssetDownload(url: URL) -> AVURLAsset {
    // Create new background session configuration.
    let configuration = URLSessionConfiguration.background(withIdentifier: "Replay-test")

    // Create a new AVAssetDownloadURLSession with background configuration, delegate, and queue
    let downloadSession = AVAssetDownloadURLSession(configuration: configuration,
                                                    assetDownloadDelegate: self,
                                                    delegateQueue: OperationQueue.main)

    let asset = AVURLAsset(url: url)

    // Create new AVAssetDownloadTask for the desired asset
    let downloadTask = downloadSession.makeAssetDownloadTask(asset: asset,
                                                             assetTitle: "Demo",
                                                             assetArtworkData: nil,
                                                             options: nil)

    // Start task and begin download
    downloadTask?.resume()

    return downloadTask!.urlAsset
}
```

```
86      func urlSession(_ session: URLSession, task: URLSessionTask, didCompleteWithError error: Error?) {
87          guard let error = error else {
88              print("The download failed with unknown error")
89              return
90          }
91          print(error.localizedDescription)                                   Thread 1: breakpoint 5.1
92      }
```

OfflineDownloader 〉 Thread 1 〉 0 OfflineDownloader.urlSession(_:task:didCompleteWithError:)

```
Printing description of error:
▿ Optional<Error>
  - some : Error Domain=AVFoundationErrorDomain Code=-11800 "The operation could not be completed"
      UserInfo={NSLocalizedDescription=The operation could not be completed,
      _NSURLErrorFailingURLSessionTaskErrorKey=BackgroundAVAssetDownloadTask <CA4C2515-DDE8-4166-844D-FAE898D611DF>.<1>,
      _NSURLErrorRelatedURLSessionTaskErrorKey=(
    "BackgroundAVAssetDownloadTask <CA4C2515-DDE8-4166-844D-FAE898D611DF>.<1>"
), NSLocalizedFailureReason=An unknown error occurred (-16655)}
```

“We’ll think about it”

Apple AVFoundation Engineer

Sample Code

Using AVFoundation to Play and Persist HTTP Live Streams

Play HTTP Live Streams and preserve streams on disk for offline play back.

SDKs

iOS 11.0+

Xcode 10.0+

NOTE

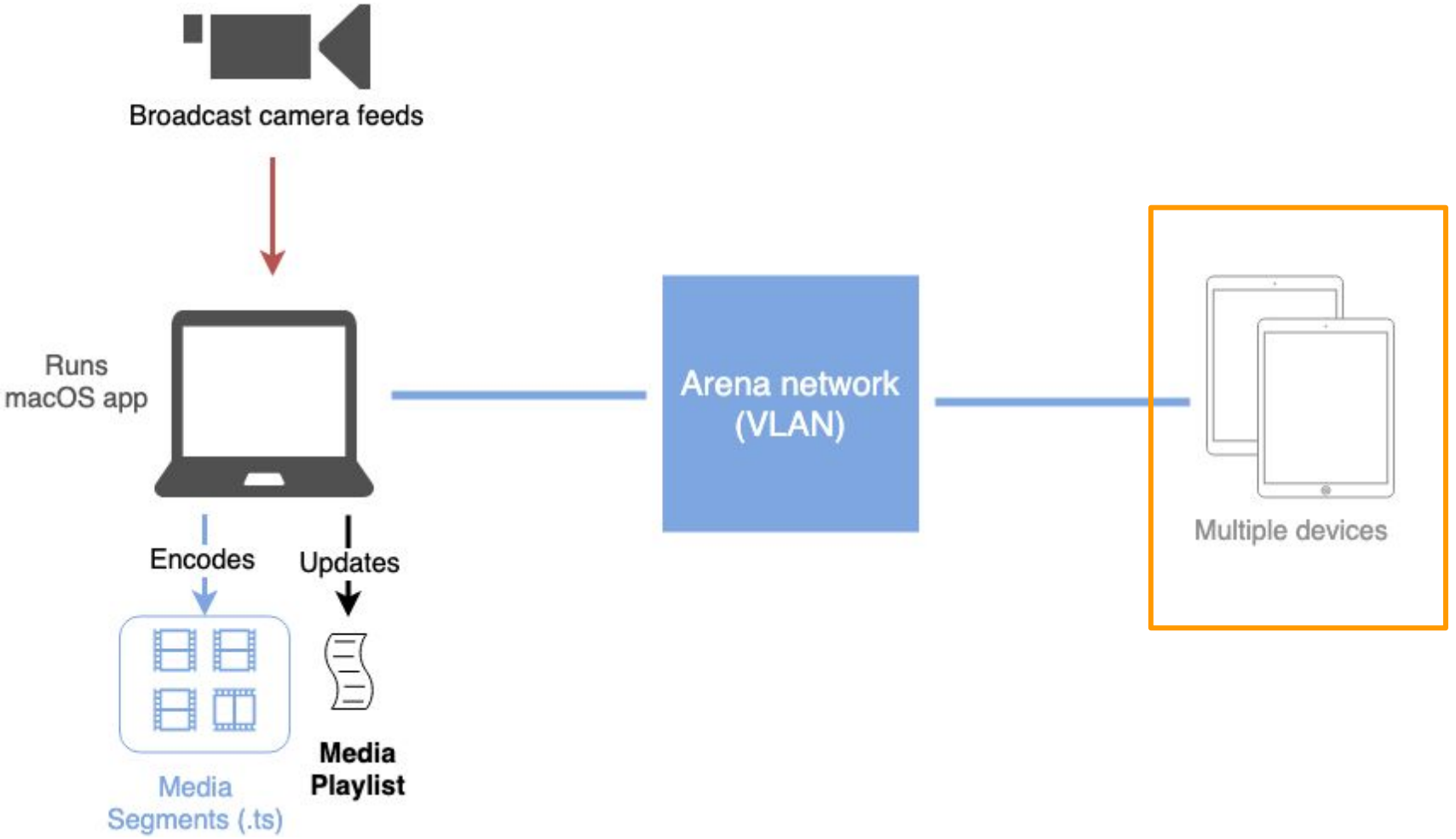
- You cannot save a live HLS stream while it is in progress. If you try to save a live HLS stream, the system throws an exception. Only Video On Demand (VOD) streams support offline playback.

3. Live offline playback

- High complexity
- Suboptimal scrubbing experience
- Getting the worst of both worlds

What did work: Custom offline playback

Always play local content


Broadcast camera feeds
Runs
macOS app
Arena network
(VLAN)
Multiple devices
Encodes
Updates
Media
Segments (.ts)
Media
Playlist

Why does that happen?

1. Return requested segments in a timely fashion
2. Live video ⇒ Playlist file must be updated frequently

Problems to solve

1. How to download and store media segments
2. How to play local media segments (.ts files)
3. How to ensure the player never perma loads

Play .ts files

```
19      override func viewDidLoad() {
20          super.viewDidLoad()
21          let path = Bundle.main.path(forResource: "video", ofType:"ts")!
22          let url = URL(fileURLWithPath: path)
23
24          let urlAsset = AVURLAsset(url: url)
25          let playerItem = AVPlayerItem(asset: urlAsset)
26          let player = AVPlayer(playerItem: playerItem)
27          self.player = player
28      }
29
30      override func viewDidAppear(_ animated: Bool) {
31          super.viewDidAppear(animated)
32
33          self.view.addSubview(customView)
34
35          self.player?.play()
36      }
```

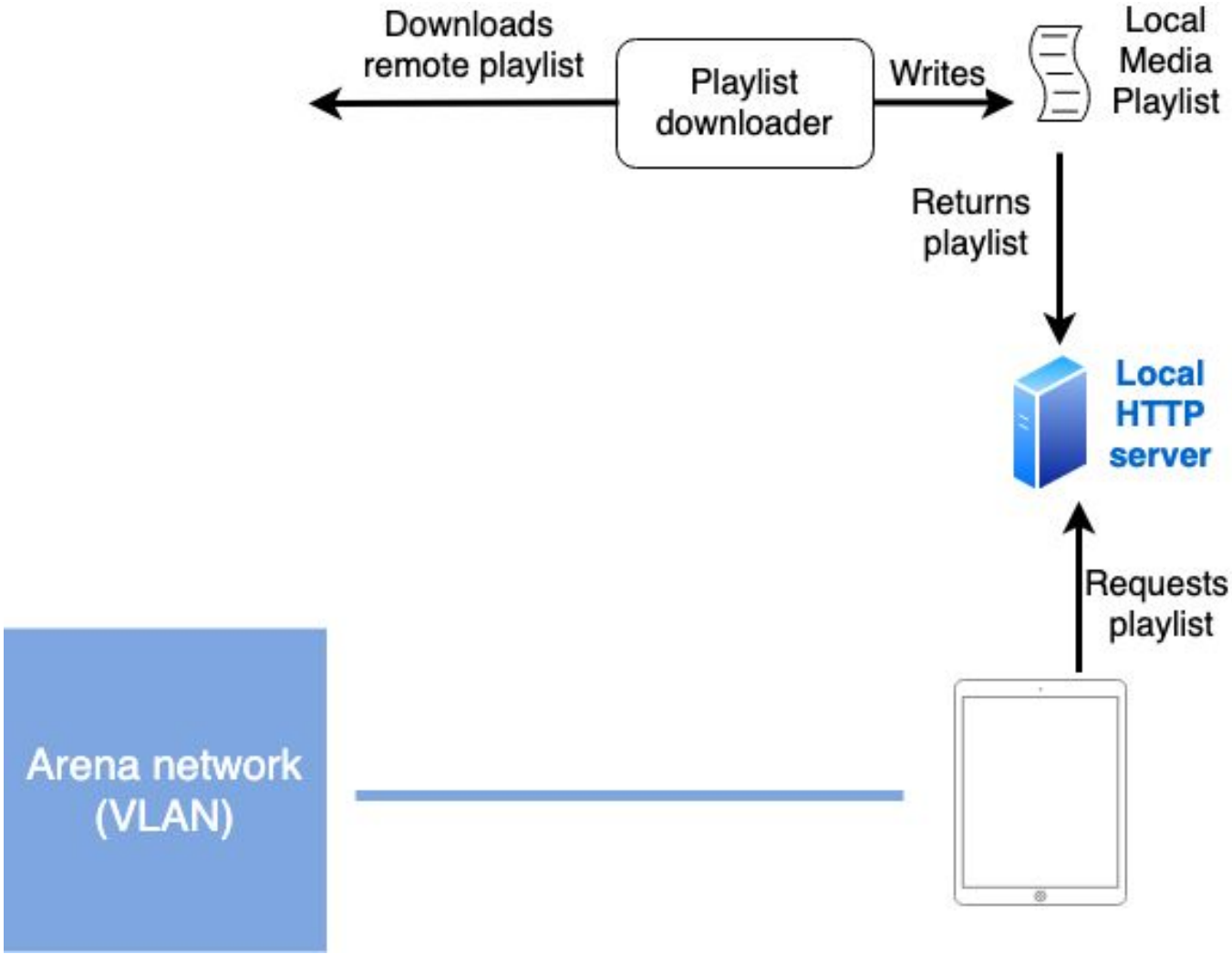
Arena network
(VLAN)


Downloads
remote playlist
Playlist
downloader
Writes
Local
Media
Playlist

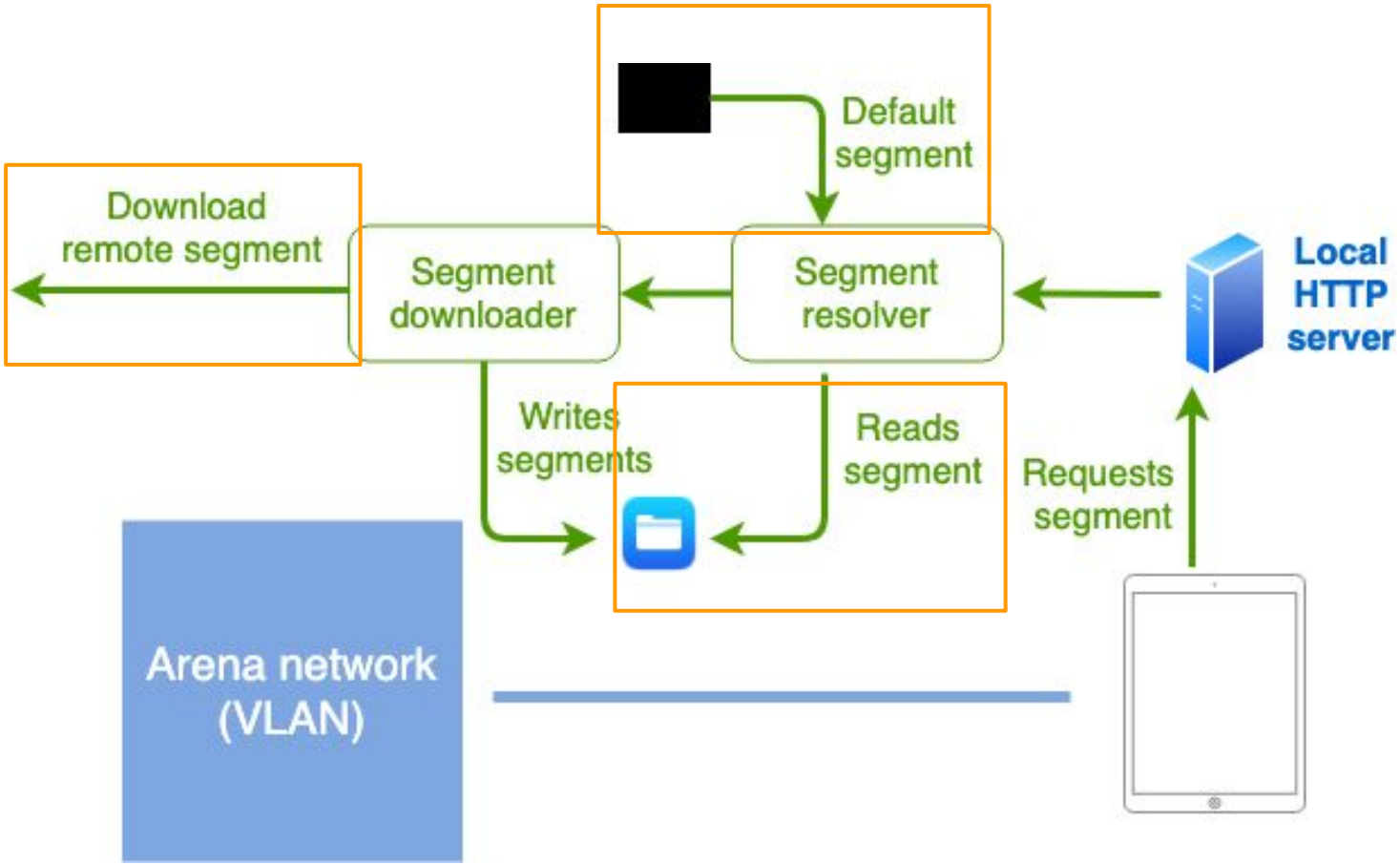

Local
HTTP
server


Arena network
(VLAN)

```
override func viewDidLoad() {
    super.viewDidLoad()
    let urlString = "https://localhost:8080/local-playlist.m3u8"
    let url = URL(string: urlString)!

    let urlAsset = AVURLAsset(url: url)
    let playerItem = AVPlayerItem(asset: urlAsset)
    let player = AVPlayer(playerItem: playerItem)
    self.player = player
}

override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)

    self.view.addSubview(customView)

    self.player?.play()
}
```


Downloads
remote playlist
Playlist
downloader
Writes
Local
Media
Playlist
Returns
playlist
Local
HTTP
server
Requests
playlist
Arena network
(VLAN)

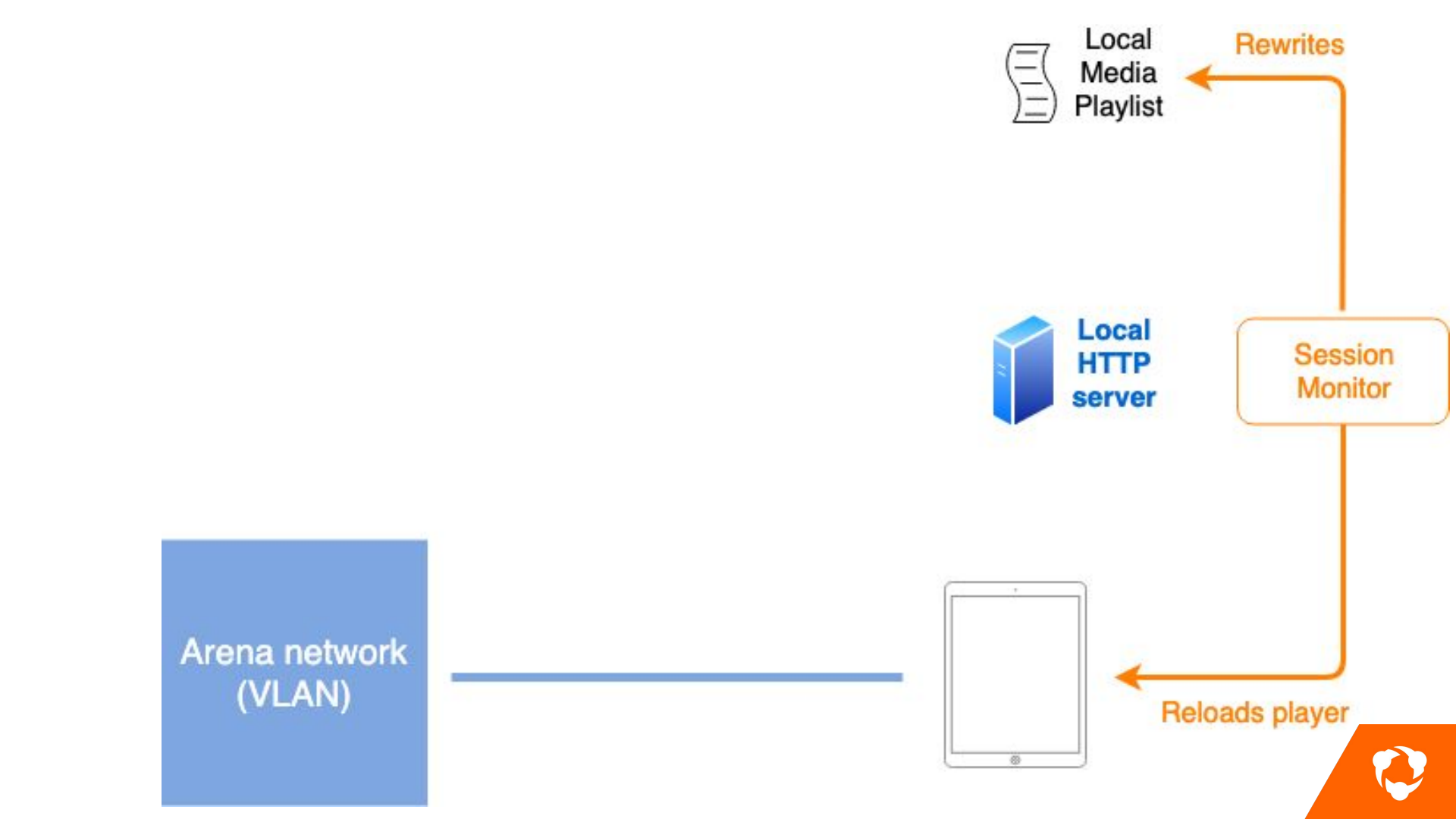
Download media segments


Default segment
Download remote segment
Segment downloader
Segment resolver
Local HTTP server
Writes segments
Reads segment
Requests segment
Arena network (VLAN)

Why does that happen?

1. Return requested segments in a timely fashion ✔️
2. Live video ⇒ Playlist file must be updated frequently

No permaloading

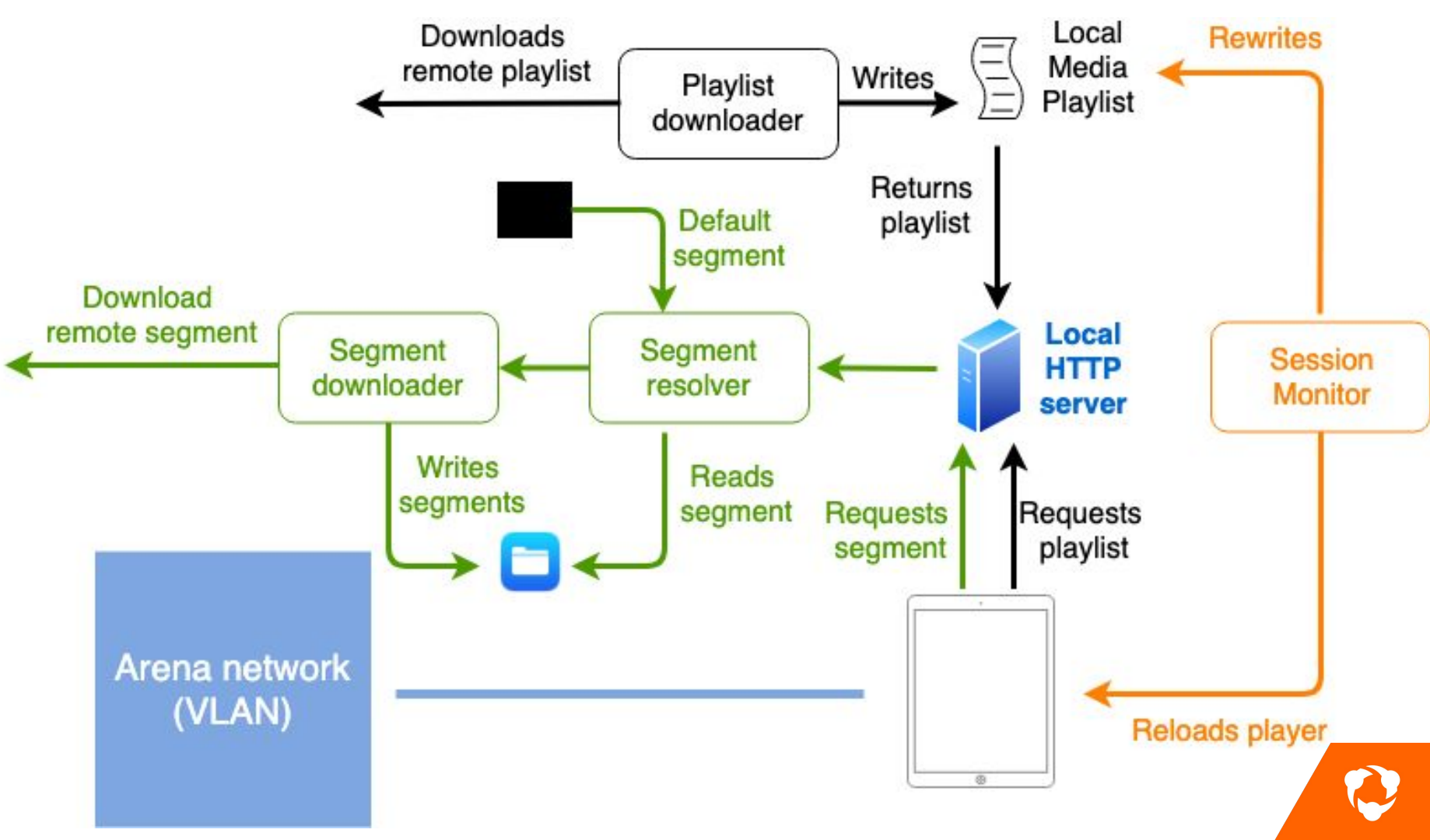

Local
Media
Playlist
Rewrites
Local
HTTP
server
Session
Monitor
Arena network
(VLAN)
Reloads player

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:EVENT
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-ALLOW-CACHE:YES
#EXTINF:2.000000
0000_00000.ts
#EXTINF:2.000000
0000_00001.ts
#EXTINF:2.000000
0000_00002.ts
#EXTINF:2.000000
0000_00003.ts
#EXTINF:2.000000
0000_00004.ts
#EXTINF:2.000000
0000_00005.ts
…
…
...
```

#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-ALLOW-CACHE:YES
#EXTINF:2.000000
0000_00000.ts
#EXTINF:2.000000
0000_00001.ts
#EXTINF:2.000000
0000_00002.ts
#EXTINF:2.000000
0000_00003.ts
#EXTINF:2.000000
0000_00004.ts
#EXTINF:2.000000
0000_00005.ts
…
…
…
#EXT-X-ENDLIST

Why does that happen?

1. Return requested segments in a timely fashion ✔️
2. ~~Live video ⇒ Playlist file must be updated frequently~~


Downloads remote playlist
Playlist downloader
Writes
Local Media Playlist
Rewrites
Returns playlist
Default segment
Download remote segment
Segment downloader
Segment resolver
Local HTTP server
Session Monitor
Writes segments
Reads segment
Requests segment
Requests playlist
Arena network (VLAN)
Reloads player

Juanjo Ramos

@JuanjoRamos82

Thank you!

Q&A