Protocolli di Streaming: Delivery adattivo

Scritto da Febogamedev

I primi sistemi di streaming utilizzavano spesso protocolli specializzati come *RTP* e *RTSP*, pensati per trasportare e controllare flussi multimediali in tempo reale.
Queste tecnologie erano efficaci, ma presentavano alcuni limiti quando lo streaming iniziò a diffondersi su larga scala attraverso Internet.

In particolare, potevano richiedere server di streaming dedicati, utilizzare modalità di comunicazione differenti dal normale traffico Web e incontrare maggiori difficoltà nell'attraversamento di firewall, *NAT* e reti aziendali.

Inoltre, distribuire lo stesso contenuto a milioni di utenti richiedeva infrastrutture specificamente progettate per lo streaming.
Con la diffusione degli smartphone e delle connessioni mobili emerse anche un altro problema: la capacità della rete poteva cambiare continuamente.
Un video trasmesso con un bitrate fisso poteva funzionare correttamente in un momento e iniziare a interrompersi pochi secondi dopo.

Per questo fu necessario creare nuovi protocolli che potessero affrontare queste sfide.

HLS

Apple sviluppò ***HLS* (*HTTP* Live Streaming)** con questo approccio: invece di mantenere un flusso multimediale continuo tra server e client, il contenuto viene suddiviso in segmenti e distribuito utilizzando il normale protocollo *HTTP (Pantos e May, 2017)*.

Il cambiamento può essere rappresentato così:

- Streaming real-time tradizionale: Streaming Server → RTP → Player
- con HLS diventa: Web Server / *CDN* → *HTTP* → Segmenti → Player

Il vantaggio è che *HLS* può sfruttare gran parte dell'infrastruttura già utilizzata dal Web tramite *l'HTTP (Apple, s.d.-c)*.

Questo permette di distribuire contenuti a un numero molto elevato di utenti attraverso richieste di risorse HTTP. Le connessioni HTTP possono essere persistenti e riutilizzate: la distinzione riguarda il modello di distribuzione, non l'assenza di connessioni persistenti (Pantos e May, 2017).
HLS sacrifica quindi parte della bassissima latenza tipica dei protocolli real-time in cambio di:

- Elevata scalabilità.
- Compatibilità con le infrastrutture Web.
- Facile distribuzione attraverso *CDN*.

- Maggiore semplicità nell'attraversamento di firewall e *NAT*.
- Supporto all'*ABR*.

Questo mette in evidenza una distinzione fondamentale:
- *RTMP*, *SRT* e *RIST* portano principalmente il contenuto verso la piattaforma.
- *HLS* porta principalmente il contenuto dalla piattaforma verso un grande numero di utenti.

Il successo di *HLS* deriva proprio da questa capacità di trasformare lo streaming in un sistema compatibile con l'infrastruttura Web.
In questo modo è possibile distribuire video on demand e live a un numero molto elevato di utenti, adattando contemporaneamente la qualità alle condizioni della rete e alle caratteristiche del dispositivo (Apple, s.d.-c).

HLS utilizza principalmente tre elementi:
- Playlist.
- Segmenti multimediali.
- *HTTP*.

Playlist
Le playlist *HLS* sono file di testo normalmente identificati dall'estensione ".m3u8" (Pantos e May, 2017).
Descrivono quali contenuti sono disponibili e dove possono essere recuperati.
Queste playlist si possono trovare sull'origin server o sugli edge server.

Esistono principalmente due tipi di playlist.

La Multivariant Playlist, storicamente chiamata anche Master Playlist, descrive le diverse versioni disponibili dello stesso contenuto (Apple, s.d.-d).
Per esempio:
- 1080p → 6 Mbit/s
- 720p → 3 Mbit/s
- 480p → 1 Mbit/s

Può inoltre indicare:
- Tracce audio differenti.
- Lingue differenti.
- Sottotitoli.
- Codec differenti.
- Versioni video alternative.

Ogni variante può fare riferimento a una propria Media Playlist.

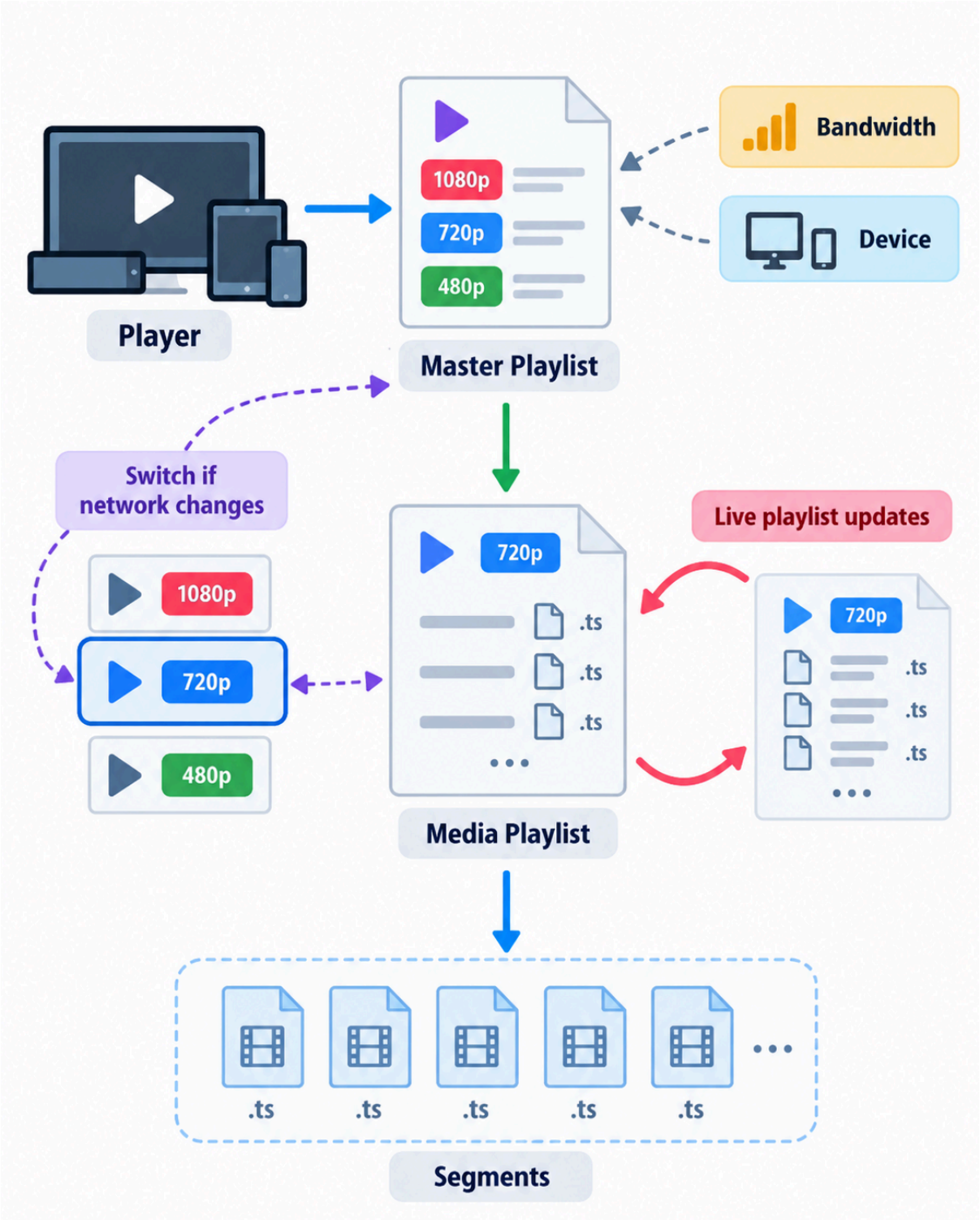
La Media Playlist contiene invece l'elenco dei segmenti che appartengono a uno specifico flusso (Pantos e May, 2017).
Per esempio:
- segment01.m4s.
- segment02.m4s.

- segment03.m4s.
- segment04.m4s.

Il player legge la playlist e richiede progressivamente questi segmenti.

Quindi, il player riceve inizialmente la Multivariant Playlist, che descrive le diverse versioni disponibili del contenuto.
Sulla base delle condizioni della rete e delle caratteristiche del dispositivo, seleziona una Media Playlist specifica e da questa ottiene gli indirizzi dei segmenti audio/video da scaricare e riprodurre.
Se durante la riproduzione la connessione peggiora o migliora, il player può selezionare una diversa Media Playlist tra quelle indicate dalla Multivariant Playlist (Pantos e May, 2017).
Nel live streaming, inoltre, la Media Playlist viene aggiornata continuamente con i nuovi segmenti prodotti.

Nota:
La Multivariant Playlist non è obbligatoria. Se esiste una sola qualità, il player può ricevere direttamente una Media Playlist (Pantos e May, 2017).



Segmenti multimediali
Il contenuto viene suddiviso in porzioni temporali relativamente brevi.
Per esempio:

```
Video
|
├── Segment 1
├── Segment 2
```

```
├── Segment 3
└── Segment 4
```

Storicamente *HLS* ha utilizzato soprattutto segmenti *MPEG-TS*, generalmente con estensione: ".ts".
Nei sistemi moderni viene frequentemente utilizzato anche *fMP4* (Fragmented *MP4*), con segmenti con estensione ".m4s" (Pantos e May, 2017).

Audio e video possono essere presenti nello stesso segmento oppure essere distribuiti attraverso flussi separati.
Per esempio:

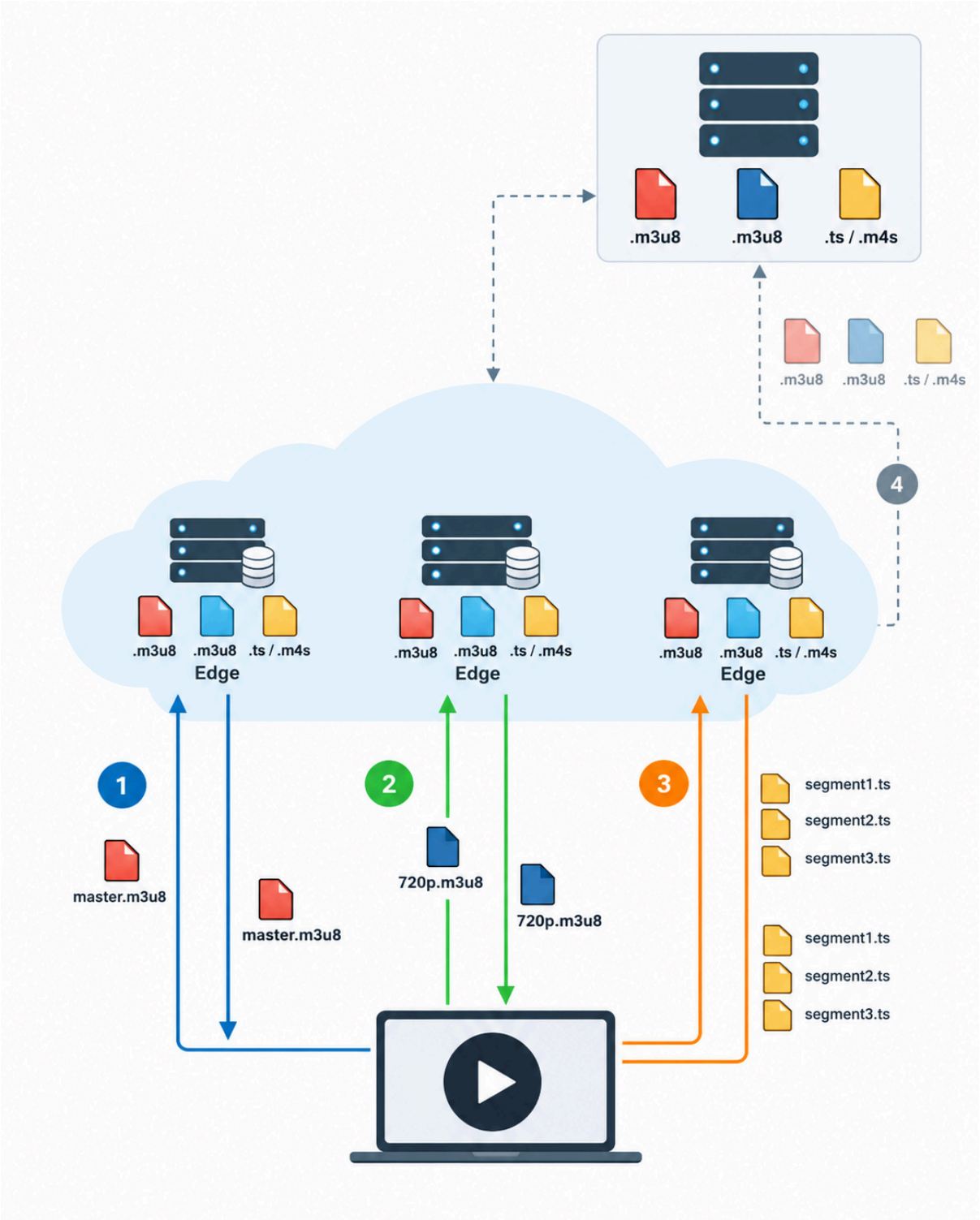
- Video 1080p.
- Video 720p.
- Video 480p.
- Audio Italian.
- Audio English.
- Audio French.

Questo permette al player, per esempio, di cambiare qualità video mantenendo la stessa traccia audio (Apple, s.d.-d).

HTTP
Playlist e segmenti vengono recuperati dal player attraverso normali richieste *HTTP*.
Per esempio:

```
Player → GET playlist.m3u8
Server → playlist
Player → GET segment01.m4s
Server → segment
Player → GET segment02.m4s
Server → segment
```

Si tratta quindi principalmente di un modello client pull: è il player a richiedere le risorse quando ne ha bisogno (Pantos e May, 2017).
Non è necessario che il server "spinga" continuamente il flusso verso il client.
Questo rende HLS particolarmente compatibile con le CDN: un segmento in cache può essere riutilizzato se le regole HTTP lo consentono. Una copia scaduta può richiedere una validazione prima di essere servita; non basta quindi che il file sia presente nella cache (Fielding et al., 2022).



Una delle caratteristiche principali di *HLS* è il supporto all'*ABR*.
Il contenuto viene preparato in più versioni. Nella terminologia HLS, una rendition è una versione di un media, mentre un Variant Stream può combinare rendition video, audio e sottotitoli riproducibili insieme (Pantos e May, 2017).
Per esempio:

- 1080p → 6 Mbit/s

- 720p → 3 Mbit/s
- 480p → 1 Mbit/s

Il player può scegliere dinamicamente quale versione scaricare.

HLS può essere utilizzato anche per trasmettere eventi live.
In questo caso i segmenti non esistono tutti prima dell'inizio della riproduzione, ma vengono generati progressivamente.
La Media Playlist viene quindi aggiornata continuamente man mano che vengono prodotti nuovi segmenti.
Il player ricarica periodicamente la playlist per scoprire quali nuovi segmenti sono disponibili.
Questo modello viene spesso rappresentato come una sliding window:

```
Tempo →
[100][101][102]
     [101][102][103]
          [102][103][104]
```

I segmenti nuovi vengono aggiunti mentre quelli più vecchi possono progressivamente uscire dalla finestra disponibile (Pantos e May, 2017).

Il principale limite dell'*HLS* tradizionale è la latenza.
Prima che un utente possa vedere un determinato momento della diretta devono avvenire diverse operazioni:

```
Acquisizione
↓
Encoding
↓
Packaging
↓
Produzione del segmento
↓
Distribuzione
↓
Buffering
↓
Decoding
↓
Playback
```

La latenza complessiva non dipende quindi soltanto dalla rete.
Dipende anche da:

- Durata dei segmenti.
- Tempo necessario all'encoder.
- Packaging.
- Frequenza con cui vengono aggiornate le playlist.
- CDN.

- Buffer mantenuto dal player.
- Decodifica.

HLS tradizionale tende quindi ad avere una latenza superiore rispetto a protocolli progettati specificamente per comunicazioni real-time, come *RTP* o WebRTC.
Il compromesso dipende dalla configurazione: attendere segmenti completi e mantenere più contenuto nel buffer può aumentare il ritardo e la tolleranza alle variazioni di rete. La scalabilità dipende soprattutto dalla distribuzione HTTP e dalle CDN; non cresce automaticamente aumentando il buffer (Apple, s.d.-b; Apple, s.d.-c).

Per ridurre questo problema è stato sviluppato **Low-Latency *HLS*** (*LL-HLS*) (Apple, s.d.-b).
LL-*HLS* non sostituisce il modello *HLS*.
Continua infatti a utilizzare la struttura *HTTP* + Playlist + Segmenti + *CDN*.
Introduce però meccanismi che permettono al player di ricevere il contenuto molto più vicino al momento in cui viene prodotto.
Tra i principali meccanismi troviamo:
- Partial Segments.
- Blocking Playlist Reload.
- Preload Hints.
- Playlist Delta Updates.
- Rendition Reports.

Partial Segments
Nel normale *HLS* il player può dover attendere che un segmento sia completamente prodotto prima di poterlo scaricare.
LL-HLS permette invece di suddividerlo ulteriormente:



Il player può iniziare a scaricare le prime parti mentre il resto del segmento viene ancora prodotto (Apple, s.d.-b).

Blocking Playlist Reload
Nel normale *HLS* il client controlla periodicamente se la playlist è stata aggiornata.
Con il Blocking Playlist Reload, il client può invece effettuare una richiesta e lasciare che il server la mantenga aperta fino alla disponibilità di nuovi dati (Apple, s.d.-b).
Il modello diventa:
- Player → "Dammi il prossimo aggiornamento"
- Server → attende
- Server → nuovo contenuto disponibile
- Server → risposta immediata

In questo modo si riduce il tempo perso nel controllo ripetuto della playlist.

Preload Hints
Il server può inoltre suggerire al player quale sarà probabilmente la prossima risorsa disponibile.
Il player può inviare in anticipo la richiesta della risorsa indicata; il server può mantenerla in attesa fino alla disponibilità dei dati (Apple, s.d.-b).
Questo permette di avvicinarsi ulteriormente al live edge, cioè al punto più recente del flusso disponibile.

Playlist Delta Updates
Nel live streaming la Media Playlist viene aggiornata molto frequentemente.
Se il player dovesse riscaricare ogni volta l'intera playlist, finirebbe per ricevere continuamente molte informazioni che possiede già.
Con le Playlist Delta Updates, il client può invece richiedere soltanto la parte più recente della playlist (Apple, s.d.-b).
Il principio è:

- Player → "Mandami solo gli aggiornamenti recenti".
- Server → omette le parti vecchie già note al client.
- Server → invia soltanto i nuovi segmenti e le informazioni necessarie.

In questo modo si riduce la quantità di dati trasferiti per aggiornare continuamente la playlist, cosa particolarmente utile nelle live di lunga durata.

Rendition Reports
In uno streaming adattivo esistono più rendition, cioè versioni dello stesso contenuto con bitrate o qualità differenti.
Per esempio:

- 1080p.
- 720p.
- 480p.

Se il player sta utilizzando la Media Playlist 720p e decide di passare alla 480p, deve sapere a quale punto temporale è arrivata la playlist 480p.
I Rendition Reports permettono a una Media Playlist di fornire informazioni aggiornate sulle altre rendition disponibili (Apple, s.d.-b).

Per esempio:
720p Media Playlist
può contenere informazioni come:

- 1080p → ultimo segmento disponibile: 125, Part 3.
- 480p → ultimo segmento disponibile: 125, Part 3.

In questo modo, se la connessione peggiora, il player può passare rapidamente:
720p → 480p
senza dover prima effettuare richieste aggiuntive per scoprire quale sia il punto più recente disponibile nella playlist 480p.

Questo è particolarmente importante perché quando si cerca di rimanere molto vicini al live edge anche uno o due round trip aggiuntivi possono aumentare sensibilmente la latenza.

HLS Ingest

Sebbene HLS sia stato sviluppato principalmente per distribuire contenuti multimediali agli spettatori, può essere utilizzato anche nella fase di ingest, cioè per trasportare il contenuto dall'encoder verso l'infrastruttura della piattaforma.
Questa modalità viene chiamata ***HLS* Ingest**.
La differenza principale rispetto alla distribuzione tradizionale riguarda il modo in cui vengono trasferiti i segmenti.
Nella delivery *HLS*, il player richiede playlist e segmenti attraverso *HTTP* GET. Si tratta quindi di un modello client pull.
Nell'*HLS* Ingest, invece, l'encoder può inviare direttamente playlist e segmenti al server attraverso richieste HTTP PUT o POST. Si tratta quindi di un modello push (DASH Industry Forum, 2026).

Il funzionamento può essere rappresentato così:

- *HLS* Delivery: Server / *CDN* → *HTTP* GET → Player
- *HLS* Ingest: Encoder → *HTTP* PUT / POST → Ingest Server

Durante una diretta, l'encoder comprime il contenuto audiovisivo, lo suddivide in segmenti e aggiorna progressivamente la Media Playlist.
Successivamente invia questi elementi al server di ingest, che li riceve e li rende disponibili alle successive operazioni della piattaforma.

Per esempio:

- Encoder → PUT segment01.m4s.
- Encoder → PUT segment02.m4s.
- Encoder → PUT playlist.m3u8 aggiornata.
- Encoder → PUT segment03.m4s.
- Encoder → PUT playlist.m3u8 aggiornata.

L'ordine effettivo delle richieste e le modalità di aggiornamento dipendono dalle specifiche del servizio utilizzato.
Una volta ricevuto il contenuto, la piattaforma può elaborarlo ulteriormente, preparare altre versioni oppure distribuirlo agli spettatori attraverso *HLS* o altri sistemi.

HLS Ingest presenta alcuni vantaggi: utilizza *HTTP*, può integrarsi con infrastrutture Web e permette di trasmettere contenuti già organizzati in segmenti.
Tuttavia, poiché il contenuto viene generalmente trasferito attraverso unità multimediali segmentate, può introdurre una latenza maggiore rispetto a protocolli come *RTMP* o *WebRTC*, a seconda della configurazione.

HLS non deve quindi essere considerato esclusivamente un protocollo di delivery: la stessa tecnologia può essere utilizzata in fasi differenti della pipeline, anche se con modalità di comunicazione diverse.

Un esempio concreto è YouTube, che supporta *HLS* Ingest per ricevere contenuti live dagli encoder attraverso *HTTP* PUT o POST (Google, s.d.).

MPEG-DASH

Abbiamo visto *HLS*, una tecnologia nata in ambito Apple che ha permesso di trasformare lo streaming in un sistema basato su *HTTP*, manifest, segmenti e *ABR*.
Questo approccio si è dimostrato molto efficace perché permette di utilizzare normali server Web e *CDN* per distribuire contenuti a un numero molto elevato di utenti.

Accanto a HLS, sviluppato da Apple, MPEG-DASH costituisce una soluzione standardizzata a livello internazionale nell'ambito ISO/IEC.
Nasce così ***MPEG-DASH*** **(Dynamic Adaptive Streaming over HTTP)**, uno standard sviluppato nell'ambito *MPEG* e pubblicato come ISO/IEC 23009 (ISO/IEC, 2026).

Dal punto di vista concettuale *DASH* e *HLS* sono quindi molto simili.
Entrambi permettono inoltre al player di scegliere dinamicamente tra più versioni dello stesso contenuto.
La differenza principale non è quindi nel principio di funzionamento, ma soprattutto nella standardizzazione e nella struttura con cui vengono descritti contenuti e segmenti.

In un servizio DASH adattivo, la pipeline di codifica e packaging prepara il contenuto in più versioni: lo standard DASH ne descrive l'organizzazione e l'accesso, non esegue la codifica (ISO/IEC, 2026).
Per esempio:

- 1080p → 6 Mbit/s
- 720p → 3 Mbit/s
- 480p → 1 Mbit/s

Queste versioni vengono chiamate **Representations**.
Il player può iniziare utilizzando una Representation: 720p → 3 Mbit/s.

- Se la rete peggiora, può richiedere il segmento successivo dalla Representation: 480p → 1 Mbit/s.
- Se la rete migliora, può invece passare a: 1080p → 6 Mbit/s.

Il player può prendere questa decisione considerando velocità di download, livello del buffer, bitrate disponibili e altri parametri (DASH Industry Forum, s.d.-a).

L'equivalente della playlist *HLS* è l'***MPD*** **(Media Presentation Description)**.
È un documento *XML* che descrive la struttura della presentazione multimediale e permette al player di sapere quali contenuti sono disponibili e come recuperarli (ISO/IEC, 2026; DASH Industry Forum, s.d.-b).
Una struttura DASH semplificata può essere rappresentata così:
Period → AdaptationSet → Representation.



Un Period rappresenta una parte temporale della presentazione (DASH Industry Forum, s.d.-b).
Un contenuto può avere un solo Period oppure più Period.
Per esempio:

- Period 1 → Film
- Period 2 → Advertising
- Period 3 → Film

I Period possono anche avere codec, tracce o caratteristiche differenti.

Un Adaptation Set raggruppa contenuti alternativi che svolgono una funzione simile.
Per esempio:

- Video Adaptation Set
- Audio Italian Adaptation Set
- Audio English Adaptation Set
- Subtitles Adaptation Set

Un Adaptation Set video può contenere più Representation video con caratteristiche differenti.

Una Representation è una specifica versione codificata del contenuto (DASH Industry Forum, s.d.-b).
Per esempio:

- Representation A: 1920×1080 -> 6 Mbit/s
- Representation B: 1280×720 -> 3 Mbit/s
- Representation C: 854×480 -> 1 Mbit/s

È tra queste Representation che il player può effettuare il cambio di qualità durante l'*ABR*.

In ogni Representation, *DASH* divide il contenuto in unità temporali chiamati **segments**.
L'MPD contiene gli indirizzi, oppure le regole necessarie per costruire gli indirizzi, da cui il player può scaricare questi segmenti (DASH Industry Forum, s.d.-b).

Nei workflow basati su fMP4/*CMAF*, i segmenti multimediali utilizzano comunemente l'estensione .m4s.

Per esempio:

```
Representation 1080p
├── Segment 1
├── Segment 2
├── Segment 3
└── Segment 4

Representation 720p
├── Segment 1
├── Segment 2
├── Segment 3
└── Segment 4

Representation 480p
├── Segment 1
├── Segment 2
├── Segment 3
└── Segment 4
```

Le varie Representation vengono preparate in modo da permettere al player di passare da una all'altra in punti appropriati della timeline (DASH Industry Forum, s.d.-b).

Nei workflow moderni è particolarmente comune l'utilizzo di fragmented *MP4* (fMP4) e di ***CMAF* (Common Media Application Format)**.
CMAF è uno standard pensato per rendere più uniforme la distribuzione dei contenuti multimediali sulle diverse piattaforme di streaming (Apple, s.d.-a).
CMAF può permettere di riutilizzare le stesse risorse multimediali basate su fMP4 con playlist HLS e manifest DASH differenti. La condivisione richiede però compatibilità di codec, struttura dei segmenti e, quando presente, cifratura con i dispositivi e i player destinatari: non è automatica in ogni workflow (Apple, s.d.-a).
Questo riduce la duplicazione dei file, semplifica la gestione dell'infrastruttura e migliora l'interoperabilità tra sistemi diversi.

Questo è interessante perché significa che un'infrastruttura può, in determinati workflow, utilizzare gli stessi file multimediali segmentati e cambiare principalmente il manifest utilizzato per descriverli:

```
             HLS → .m3u8
            /
CMAF Media
            \
             DASH → .mpd
```

MPD e i segmenti vengono recuperati tramite *HTTP (ISO/IEC, 2026)*.

Il funzionamento è principalmente **client pull**:

- Player → GET manifest.mpd.
- Server → MPD.
- Player → GET segment.
- Server → Segment.

DASH può essere utilizzato anche per contenuti live.
Come per l'*HLS*, i segmenti vengono prodotti progressivamente mentre l'evento è in corso. In questo caso il *MPD* è dinamico e descrive una presentazione che evolve nel tempo (DASH Industry Forum, s.d.-b).
Possiamo quindi distinguere:

- Static *MPD* → contenuto on demand.
- Dynamic *MPD* → contenuto live.

Il player continua a recuperare le informazioni necessarie per identificare i nuovi segmenti disponibili e mantenersi vicino al live edge.

Come *HLS* tradizionale, anche *DASH* basato su segmenti completi e buffering può introdurre una latenza significativa.
Per ridurla possono essere utilizzate modalità comunemente definite **Low-Latency DASH**.

Uno degli elementi fondamentali è l'utilizzo di ***CMAF* chunks (DASH Industry Forum, s.d.-c)**.
Un segmento può essere prodotto e trasferito progressivamente in CMAF chunks, permettendo al player di elaborare i dati disponibili prima del completamento dell'intero segmento. Servono il supporto del player e dell'infrastruttura e una segnalazione coerente della disponibilità nel *MPD*; la sola suddivisione in chunk non basta a garantire bassa latenza (DASH Industry Forum, s.d.-c):

```
Segment
│
├── Chunk 1
├── Chunk 2
├── Chunk 3
└── Chunk 4
```

MPEG-DASH Ingest

Sebbene MPEG-DASH sia utilizzato principalmente per distribuire contenuti multimediali agli spettatori, può essere impiegato anche nella fase di ingest, cioè per trasferire il contenuto dall'encoder verso l'infrastruttura della piattaforma.
Questa modalità viene comunemente chiamata **DASH Ingest** ed è descritta nelle specifiche DASH-IF Live Media Ingest Protocol (DASH Industry Forum, 2026).
Come per HLS, la differenza principale rispetto alla distribuzione tradizionale riguarda il modo in cui vengono trasferiti i segmenti.

Nella delivery MPEG-DASH, il player legge il manifest *MPD* e richiede i segmenti multimediali attraverso *HTTP* GET. Si tratta quindi di un modello client pull.
Nel DASH Ingest, invece, l'encoder può inviare direttamente i segmenti e il manifest al server attraverso richieste *HTTP* POST o PUT. Si tratta quindi di un modello push.

Il funzionamento può essere rappresentato così:
- MPEG-DASH Delivery: Server / CDN → HTTP GET → Player
- MPEG-DASH Ingest: Encoder → HTTP POST / PUT → Ingest Server

Durante una diretta, l'encoder comprime il contenuto audiovisivo, lo organizza in segmenti e può produrre un manifest *MPD* che descrive le Representation disponibili.
Successivamente trasferisce questi elementi al server di ingest, che li riceve e li rende disponibili alle successive operazioni della piattaforma.

Per esempio, in una trasmissione con una singola Representation:
- Encoder → POST manifest.mpd.
- Encoder → PUT init.mp4.
- Encoder → PUT segment01.m4s.
- Encoder → PUT segment02.m4s.
- Encoder → PUT segment03.m4s.

Questo esempio è semplificato: l'ordine effettivo delle operazioni, la struttura dei nomi e l'eventuale aggiornamento del manifest dipendono dall'interfaccia di ingest adottata.
Le specifiche DASH-IF distinguono infatti due modalità principali:
- DASH/HLS Ingest (Interface-2): l'encoder invia al server manifest e segmenti multimediali attraverso richieste *HTTP*.
- CMAF Ingest (Interface-1): l''encoder trasferisce tracce multimediali organizzate secondo lo standard *CMAF*. Il server ricevente può successivamente generare i manifest necessari alla distribuzione DASH o *HLS*.

La seconda modalità è particolarmente interessante perché permette di separare maggiormente la produzione dei contenuti dalla loro distribuzione.

Per esempio:
Encoder → CMAF Ingest → Packager → MPEG-DASH / HLS → CDN → Player

Un ulteriore vantaggio riguarda la possibilità di trasferire progressivamente i dati di un segmento, utilizzando meccanismi di trasferimento *HTTP* che consentono al server di iniziare a ricevere il contenuto prima che l'intero segmento sia stato completato.
Questo può contribuire a ridurre la latenza, purché anche le altre componenti della pipeline supportino tale modalità.

DASH Ingest consente quindi di utilizzare tecnologie *HTTP* anche per trasportare contenuti verso la piattaforma, integrandosi con infrastrutture che utilizzano *CMAF*, packaging adattivo e distribuzione attraverso *CDN*.
MPEG-DASH non deve quindi essere considerato esclusivamente una tecnologia di delivery: attraverso le specifiche DASH-IF può essere utilizzato anche nella fase di ingest,

mantenendo una distinzione tra il trasferimento dei contenuti verso la piattaforma e la loro successiva distribuzione agli spettatori.

Materiale di supporto

Esercizi

Esercizio 1 — Progettare una diretta con *HLS*
Una piattaforma deve trasmettere in diretta una cerimonia di premiazione internazionale a decine di migliaia di spettatori.
Il contenuto viene preparato in tre versioni:

- 1080p → 6 Mbit/s.
- 720p → 3 Mbit/s.
- 480p → 1 Mbit/s.

La piattaforma vuole utilizzare *HLS* e una *CDN*.
Durante i test emergono però due esigenze:

- Gli spettatori devono poter continuare a guardare anche quando la connessione cambia rapidamente.
- Il ritardo rispetto all'evento reale deve essere il più ridotto possibile.

Rispondi a questi quesiti:

- Ricostruisci il percorso generale della distribuzione: Produzione → encoding → packaging → … → CDN → player.
- Spiega perché HLS distribuisce il contenuto attraverso HTTP e segmenti invece di mantenere necessariamente un unico flusso continuo tra server e client.
- Individua il ruolo della Multivariant Playlist.
- Individua il ruolo delle Media Playlist.
- Rappresenta una possibile struttura: Multivariant Playlist
    - → 1080p Media Playlist
    - → 720p Media Playlist
    - → 480p Media Playlist
- Spiega come il player può utilizzare questa struttura per realizzare l'Adaptive Bitrate Streaming.
- Durante la diretta il player utilizza inizialmente 1080p. La connessione peggiora e il buffer inizia a ridursi. Descrivi che cosa dovrebbe fare il player.
- Perché i segmenti appartenenti alle diverse rendition devono rappresentare intervalli temporali compatibili?
- Nel live streaming, perché la Media Playlist non può essere considerata un documento statico?
- Spiega il funzionamento della sliding window.
- Immagina questa situazione: Playlist: [205][206][207]. Dopo alcuni secondi diventa: Playlist: [206][207][208]. Che cosa è successo?
- Spiega perché HLS tradizionale introduce una certa latenza anche se la connessione dello spettatore è molto veloce.
- Individua almeno quattro componenti del percorso che possono contribuire alla latenza.
- La piattaforma decide quindi di adottare Low-Latency HLS. Spiega il problema che cerca di ridurre.

- Associa ciascun meccanismo alla sua funzione:
    - Partial Segments.
    - Blocking Playlist Reload.
    - Preload Hints.
    - Playlist Delta Updates.
    - Rendition Reports.
- Perché i Partial Segments permettono al player di avvicinarsi maggiormente al momento reale dell'evento?
- Perché il Blocking Playlist Reload può essere più efficiente rispetto a controllare continuamente se la playlist è cambiata?
- Lo spettatore passa da 720p a 480p. Spiega come i Rendition Reports possono facilitare il cambio mantenendo il player vicino al live edge.
- Confronta infine:
    - HLS tradizionale → maggiore semplicità/stabilità ma maggiore latenza.
    - LL-HLS → minore latenza ma maggiore complessità di coordinamento.
- Concludi spiegando perché HLS è particolarmente adatto alla distribuzione verso un pubblico molto numeroso, ma non nasce principalmente per comunicazioni interattive real-time.

Esercizio 2 — Costruire un servizio MPEG-DASH
Una piattaforma di formazione deve distribuire un corso video internazionale attraverso MPEG-DASH.
Il corso contiene:

- Video 1080p, 720p e 480p.
- Audio italiano e inglese.
- Sottotitoli.
- Un intervallo pubblicitario inserito tra due parti del corso.

La piattaforma vuole utilizzare gli stessi segmenti multimediali anche per una versione HLS del servizio.

- Spiega perché MPEG-DASH e HLS possono essere considerati concettualmente simili.
- Individua la principale differenza generale indicata nella dispensa tra i due sistemi.
- In DASH, che cos'è il MPD?
- Ricostruisci la gerarchia: MPD → Period → Adaptation Set → Representation.
- Progetta una possibile struttura MPD per il corso, indicando almeno:
    - Un Adaptation Set video.
    - Due Adaptation Set audio.
    - Un Adaptation Set per i sottotitoli.
- Inserisci all'interno dell'Adaptation Set video le tre Representation:
    - 1080p → 6 Mbit/s.
    - 720p → 3 Mbit/s.
    - 480p → 1 Mbit/s.
- Spiega che cosa rappresenta una Representation.
- Lo studente sta guardando a 1080p, ma la rete peggiora. Descrivi come il player può passare alla Representation 720p o 480p.
- Quali informazioni può considerare il player nel decidere il cambio di qualità?
- Spiega perché il cambio deve avvenire tra segmenti temporalmente compatibili.

- Il corso è composto da: Parte 1 del corso → pubblicità → Parte 2 del corso. Mostra come potresti utilizzare più Period per rappresentare questa struttura.
- Perché Period differenti possono avere caratteristiche differenti.
- Distingui:
    - Static MPD.
    - Dynamic MPD.
- Quale dei due utilizzeresti per un contenuto on demand?
- Quale per una diretta?
- In uno streaming live, perché il MPD deve descrivere una presentazione che evolve nel tempo?
- La piattaforma vuole distribuire lo stesso contenuto anche tramite HLS senza duplicare tutti i file multimediali. Spiega il ruolo di CMAF.
- Completa lo schema:
    - HLS → _____
    - CMAF Media → segmenti fMP4 comuni
    - DASH → _____
- Spiega perché questa soluzione può ridurre:
    - Duplicazione dei file.
    - Complessità dell'infrastruttura.
    - Problemi di interoperabilità.
- La piattaforma vuole inoltre ridurre la latenza della versione live DASH. Spiega quale funzione possono svolgere i CMAF chunks.
- Concludi confrontando sinteticamente.
    - HLS → playlist .m3u8.
    - DASH → manifest .mpd.
    - Entrambi → HTTP + segmenti + ABR.
    - CMAF → possibilità di condividere gli stessi segmenti multimediali.

Bibliografia

Apple (s.d.-a). About the Common Media Application Format with HTTP Live Streaming (HLS). Apple Developer Documentation.Chttps://developer.apple.com/documentation/http-live-streaming/about-the-common-media-application-format-with-http-live-streaming-hls

Apple (s.d.-b). Enabling Low-Latency HTTP Live Streaming (HLS). Apple Developer Documentation. https://developer.apple.com/documentation/http-live-streaming/enabling-low-latency-http-live-streaming-hls

Apple (s.d.-c). HTTP Live Streaming (HLS). Apple Developer. Consultato il 18 ottobre 2026. https://developer.apple.com/streaming/

Apple (s.d.-d). HTTP Live Streaming (HLS) authoring specification for Apple devices. Apple Developer. https://developer.apple.com/streaming/hls-authoring-specification-for-apple-devices.html

DASH Industry Forum (s.d.-a). ABR Settings. Documentazione dash.js. https://dashif.org/dash.js/pages/usage/abr/settings.html

DASH Industry Forum (s.d.-b). DASH-IF implementation guidelines: restricted timing model. https://dashif.org/Guidelines-TimingModel/

DASH Industry Forum (s.d.-c). Low Latency Streaming. Documentazione dash.js https://dashif.org/dash.js/pages/usage/low-latency.html

Fielding, R., Nottingham, M., & Reschke, J. (a cura di) (2022). HTTP Caching. RFC 9111. RFC Editor. https://www.rfc-editor.org/rfc/rfc9111.html

ISO/IEC (2026). Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats. ISO/IEC 23009-1:2026. Scheda ufficiale dello standard. https://www.iso.org/standard/23009-1

Pantos, R., & May, W. (2017). HTTP Live Streaming. RFC 8216. RFC Editor. https://www.rfc-editor.org/rfc/rfc8216.html

DASH Industry Forum (2026). DASH-IF Live Media Ingest Protocol, versione 1.2. Specifiche tecniche. Fonte primaria.

Google (s.d.). Delivering Live YouTube Content via HLS. Google for Developers. Documentazione ufficiale. Fonte primaria.

DASH Industry Forum (2026). DASH-IF Live Media Ingest Protocol, versione 1.2. Technical Specification, 1 settembre 2026. Documentazione ufficiale. Fonte primaria.

Approfondimenti

- James F. Kurose e Keith W. Ross, Reti di calcolatori e Internet. Un approccio top-down, 8ª edizione, Pearson, 2022 — Un manuale universitario che approfondisce il funzionamento delle applicazioni Internet, dei protocolli di trasporto e delle infrastrutture di comunicazione. Il volume analizza le problematiche legate alla distribuzione dei contenuti audiovisivi, alle prestazioni della rete e all'adattamento della qualità durante la riproduzione. È particolarmente pertinente alla dispensa perché permette di comprendere le ragioni tecnologiche che hanno favorito la diffusione dello streaming basato su HTTP rispetto alle precedenti soluzioni che utilizzavano protocolli specializzati. Fornisce inoltre le basi teoriche per comprendere il rapporto tra banda disponibile, buffering, latenza e distribuzione di contenuti a numerosi utenti.
- Cloudflare, Cos'è HTTP Live Streaming? HLS Streaming, Cloudflare Learning Center — Un'introduzione tecnica al funzionamento di HTTP Live Streaming, la tecnologia sviluppata da Apple per distribuire contenuti audiovisivi attraverso HTTP. L'articolo approfondisce il processo di codifica, la suddivisione dei contenuti in segmenti, l'utilizzo delle playlist e la distribuzione mediante server Web e CDN. È particolarmente pertinente alla sezione della dispensa dedicata a HLS, perché permette di comprendere come il player possa recuperare progressivamente le risorse necessarie alla riproduzione e modificare la qualità in relazione alle condizioni della rete. Aiuta inoltre a chiarire perché HLS sia adatto tanto ai contenuti on demand quanto alle trasmissioni live destinate a un pubblico numeroso.
- Cloudflare, Cos'è MPEG-DASH? HLS e DASH, Cloudflare Learning Center — Un approfondimento su MPEG-DASH, standard internazionale progettato per distribuire

contenuti audiovisivi adattivi attraverso HTTP. L'articolo esamina le principali caratteristiche della tecnologia e le confronta con quelle di HLS, mostrando come entrambe utilizzino segmenti multimediali e informazioni descrittive per consentire al player di selezionare versioni differenti dello stesso contenuto. È particolarmente pertinente alla seconda parte della dispensa, perché permette di comprendere quali principi siano condivisi dai due sistemi e quali differenze emergano dalla loro standardizzazione e organizzazione. Offre inoltre una panoramica dei vantaggi dello streaming adattivo rispetto alla distribuzione di contenuti con bitrate fisso.

- Cloudflare, Cos'è lo streaming con bitrate adattivo?, Cloudflare Learning Center — Un approfondimento sul funzionamento dell'Adaptive Bitrate Streaming, il meccanismo attraverso cui un player può modificare dinamicamente la qualità dei contenuti durante la riproduzione. L'articolo descrive la preparazione di diverse versioni dello stesso video, la segmentazione e il processo di selezione della qualità in funzione della capacità della rete e delle caratteristiche del dispositivo. È particolarmente pertinente alla spiegazione delle varianti HLS e delle Representation MPEG-DASH presenti nella dispensa. Permette di comprendere perché le differenti versioni debbano essere organizzate in modo da consentire il passaggio tra segmenti temporalmente corrispondenti e come l'adattamento della qualità contribuisca a limitare le interruzioni.
- Amazon Web Services, Proprietà del manifesto, Documentazione AWS Elemental MediaPackage — Una documentazione tecnica dedicata alle proprietà dei manifest utilizzati nei sistemi di streaming live e on demand. Il testo presenta alcuni elementi che consentono di distinguere un contenuto già completo da una trasmissione ancora in corso, come il tag EXT-X-ENDLIST nelle playlist HLS e l'attributo type degli MPD MPEG-DASH. È particolarmente pertinente alla distinzione sviluppata nella dispensa tra playlist aggiornate progressivamente e manifest statici. Permette di osservare come le informazioni contenute nei documenti descrittivi influenzino il comportamento del player e perché le modalità di distribuzione live e on demand richiedano una differente gestione della disponibilità dei segmenti.
- Amazon Web Services, Opzioni del manifesto DASH in AWS Elemental MediaPackage, Documentazione AWS — Un approfondimento tecnico sulla struttura dei manifest MPEG-DASH e sugli elementi XML che descrivono le risorse multimediali disponibili. La documentazione presenta la gerarchia MPD, Period, AdaptationSet, Representation, SegmentTemplate e SegmentTimeline, illustrando come questi elementi collaborino per identificare le diverse tracce e i relativi segmenti. È particolarmente pertinente alla sezione della dispensa dedicata all'organizzazione dei manifest DASH. Fornisce esempi concreti che permettono di passare dalla rappresentazione concettuale della struttura a quella effettivamente utilizzata nelle infrastrutture streaming, chiarendo come il player possa individuare e richiedere le risorse necessarie.
- Amazon Web Services, DASH multiperiodo in AWS Elemental MediaPackage, Documentazione AWS — Una documentazione tecnica dedicata all'utilizzo di più Period all'interno di uno stesso manifest MPEG-DASH. Il testo spiega come una presentazione audiovisiva possa essere suddivisa in intervalli temporali distinti, ciascuno caratterizzato da informazioni e risorse specifiche. È particolarmente pertinente alla descrizione del Period sviluppata nella dispensa, perché mostra come questa struttura possa essere utilizzata in applicazioni concrete, per esempio per distinguere il contenuto principale dagli intervalli pubblicitari. Permette inoltre di

comprendere perché un manifest DASH non debba necessariamente descrivere un unico contenuto omogeneo, ma possa organizzare una sequenza articolata di elementi audiovisivi.

- Amazon Web Services, Latenza dei video nella diretta streaming, AWS Media & Entertainment — Un approfondimento sul problema della latenza nella distribuzione di contenuti audiovisivi live e sulle strategie per ridurre il ritardo tra produzione e riproduzione. Il materiale esamina il rapporto tra le caratteristiche dell'infrastruttura, la preparazione dei segmenti e l'utilizzo di tecnologie come HLS, DASH e CMAF. È particolarmente pertinente alle sezioni della dispensa dedicate a Low-Latency HLS e Low-Latency DASH, perché permette di comprendere come la riduzione della latenza richieda interventi coordinati lungo l'intero percorso di distribuzione. Offre inoltre una prospettiva applicativa sui compromessi tra riduzione del ritardo, stabilità e capacità di distribuire lo stesso evento a molti spettatori.
- Microsoft, Streaming adattivo, Microsoft Learn — Una guida tecnica dedicata all'integrazione dello streaming adattivo nelle applicazioni Windows attraverso le API multimediali della piattaforma. La documentazione illustra come un media player possa utilizzare manifest HLS e MPEG-DASH per individuare e riprodurre contenuti audiovisivi distribuiti in segmenti. È particolarmente pertinente alla dispensa perché mostra il funzionamento dello streaming adattivo dal punto di vista dell'applicazione client, completando la descrizione delle operazioni svolte dai server e dalle infrastrutture di distribuzione. Permette inoltre di comprendere come le informazioni contenute nei manifest vengano utilizzate concretamente da un componente software incaricato della riproduzione.

Nota editoriale

Per la realizzazione di questi materiali ho utilizzato strumenti di intelligenza artificiale generativa come supporto alla scrittura, in particolare per migliorare la forma del testo, riorganizzare i contenuti, correggere la formulazione e velocizzare alcune attività redazionali.

Lavorando autonomamente alla produzione di questi materiali, cerco infatti di automatizzare tutte le attività che possono esserlo, così da poter dedicare più tempo alla ricerca, alla progettazione e allo sviluppo dei contenuti.

L'intelligenza artificiale non determina però i contenuti dell'opera: la scelta degli argomenti, la struttura, le idee, le interpretazioni, gli esempi e l'impostazione didattica sono elaborati da me. L'IA viene quindi utilizzata principalmente come strumento di supporto alla produzione e alla revisione formale, mentre la responsabilità autoriale e progettuale dei contenuti rimane mia.

Salvo diversa indicazione, questo materiale è distribuito con licenza Creative Commons Attribuzione – Non commerciale – Condividi allo stesso modo 4.0 Internazionale (CC BY-NC-SA 4.0).

È quindi possibile condividere, ridistribuire, modificare e creare opere derivate a partire dal materiale, a condizione di attribuirne correttamente la paternità, di non utilizzarlo per scopi commerciali e di distribuire eventuali versioni modificate o derivate con la stessa licenza.

