Protocolli streaming: Contribution e ingest

Scritto da Febogamedev

Nel capitolo precedente abbiamo visto che *RTP* è particolarmente adatto al trasporto di audio e video in tempo reale.
Potrebbe quindi sembrare naturale utilizzarlo anche per inviare una diretta da un software come *OBS* verso una piattaforma di streaming.
RTP può essere utilizzato anche per il trasporto professionale di flussi live. Per l'ingest da encoder, piattaforme come Amazon IVS e Twitch supportano RTMP/RTMPS; IVS supporta anche SRT. Esistono inoltre soluzioni di ingest basate su WebRTC, come WHIP: RTP non è quindi escluso da questa fase (Amazon Web Services, s.d.; Twitch, s.d.; Garcia Murillo e Gouaillard, 2025).
La scelta dipende dall'architettura del servizio, dalle tecnologie supportate e dai requisiti del collegamento.

RTP viene utilizzato molto spesso sopra *UDP*.
Questa soluzione privilegia la bassa latenza, ma può essere più complessa da utilizzare attraverso Internet perché bisogna gestire aspetti come perdita di pacchetti, porte *UDP*, *NAT*, firewall e configurazione della sessione.

RTP, inoltre, si concentra principalmente sul trasporto del media e normalmente viene affiancato da altri protocolli o meccanismi per configurare e controllare la sessione (Schulzrinne et al., 2003).
Per l'invio di una diretta verso un server centralizzato si è quindi diffuso un modello più semplice: Encoder → Connessione persistente → Streaming Server

Nel live ingest, infatti, il contenuto non esiste ancora come file completo: audio e video vengono prodotti continuamente e devono essere trasportati mentre vengono generati. Il protocollo deve quindi gestire un flusso temporale continuo di dati multimediali, mantenendo la trasmissione attiva per tutta la durata dell'evento.

Nel Video on Demand, invece, la situazione è differente. Il contenuto esiste già come file completo prima dell'ingest e non deve essere riprodotto mentre viene trasferito alla piattaforma. Per questo il caricamento può avvenire attraverso normali protocolli di trasferimento dati, tipicamente basati su *HTTP*, senza la necessità di utilizzare un protocollo di streaming multimediale come *RTP* o *RTMP*.

Il modello diventa quindi:
- *VOD*: File completo → Upload *HTTP* → Server / infrastruttura di ingest → Elaborazione
- *LIVE*: Frame prodotti continuamente → Protocollo di streaming → Live ingest → Elaborazione in tempo reale

La differenza fondamentale è quindi che nel *VOD* l'ingest trasferisce un oggetto già completo, mentre nel live deve trasportare un flusso che viene creato progressivamente nel tempo.

RTMP

Uno dei protocolli storicamente più utilizzati per l'ingest è ***RTMP*** **(Real-Time Messaging Protocol)**.

RTMP è un protocollo nato per trasportare dati multimediali e messaggi tra un client e un server attraverso una connessione persistente (Parmar e Thornburgh, 2012).
È stato sviluppato originariamente da Macromedia e successivamente utilizzato da Adobe nell'ecosistema Flash.
Adobe ha interrotto il supporto a Flash Player alla fine del 2020; RTMP continua invece a essere supportato per l'ingest live da piattaforme come Twitch e Amazon IVS (Adobe, 2021; Twitch, s.d.; Amazon Web Services, s.d.).

Esempio
Un esempio tipico è:
Camera → OBS → RTMP → Streaming Platform.

Il software di encoding acquisisce audio e video, li codifica e utilizza *RTMP* per inviarli continuamente al server della piattaforma (Amazon Web Services, s.d.).
A differenza di *RTP*, *RTMP* viene normalmente utilizzato sopra *TCP (Parmar e Thornburgh, 2012)*.

TCP garantisce la consegna ordinata dei dati e gestisce automaticamente la ritrasmissione di quelli persi (Eddy, 2022).
Questo rende *RTMP* semplice da utilizzare per stabilire una connessione stabile tra encoder e server, anche se può introdurre più latenza rispetto a soluzioni basate su *UDP.*
L'accettabilità di questo ritardo dipende dal servizio: anche l'ingest live può avere requisiti di latenza molto stringenti (Eddy, 2022; Garcia Murillo e Gouaillard, 2025).

RTMP stabilisce una connessione persistente tra il client e il server.
Quando, per esempio, *OBS* deve iniziare una diretta, semplificando molto, avviene questo processo:

```
OBS
 ↓
Connection
 ↓
Authentication
 ↓
Publish stream
 ↓
```

```
Audio + Video
 ↓
Streaming Server
```

L’encoder non invia quindi semplicemente una serie indipendente di file.
Mantiene una sessione continua con il server e gli invia progressivamente i dati della diretta.
Durante questa comunicazione *RTMP* può trasportare:
Video + Audio + Messaggi di controllo + Metadati (Parmar e Thornburgh, 2012).

Per esempio, oltre ai dati audiovisivi possono essere trasmesse informazioni relative alle caratteristiche dello stream.
Quando si utilizza un software di streaming è frequente incontrare una **stream key**.
La piattaforma può fornire, per esempio:

- Server: rtmp://live.example.com/app
- Stream key: ABCD-1234-EFGH

Il software utilizza queste informazioni per sapere:

- A quale server collegarsi.
- A quale stream pubblicare i dati.

La stream key funziona quindi come identificatore e, in molti sistemi, anche come credenziale che permette alla piattaforma di associare il flusso al corretto account o canale (Twitch, s.d.; Amazon Web Services, s.d.).
Per questo una stream key deve essere trattata come un'informazione riservata.

L’utilizzo di *TCP* rappresenta contemporaneamente un vantaggio e un limite.
Supponiamo che durante la trasmissione un segmento di dati venga perso.
Con *TCP*:

```
1 → 2 → X → 4

     ↓

ritrasmissione di 3
```

Il protocollo cerca di recuperare il dato mancante.
Questo aumenta l’affidabilità, ma può anche introdurre ritardo.
In una rete instabile si può quindi verificare un accumulo di latenza perché *TCP* cerca di consegnare i dati correttamente e nell’ordine previsto (Eddy, 2022).

Per questo *RTP* può essere più adatto ad alcune comunicazioni estremamente sensibili alla latenza, mentre *RTMP* è stato storicamente molto conveniente per l’invio stabile di una diretta verso un server.

La differenza fondamentale può essere riassunta così:

	RTP	***RTMP***

Funzione principale	Trasporto multimediale real-time.	Trasporto di stream e messaggi client-server
Trasporto tipico	*UDP*	*TCP*
Priorità	Bassa latenza	Connessione stabile e affidabile
Gestione perdita	Può tollerare perdite	*TCP* ritrasmette i dati persi
Scenario tipico	VoIP, videoconferenza, sistemi real-time	Ingest di streaming live
Esempio	Endpoint ↔ Endpoint	OBS → Streaming Server

RTMP nella sua versione originale non cifra necessariamente la comunicazione.
Per proteggere il collegamento può essere utilizzato ***RTMPS***, cioè *RTMP* trasportato attraverso il protocollo **TLS (Amazon Web Services, s.d.)**.
Il modello diventa:
RTMP → TLS → TCP → IP.

In questo modo il traffico tra encoder e server viene cifrato.
Amazon IVS, per esempio, raccomanda RTMPS rispetto a RTMP per proteggere il collegamento di ingest (Amazon Web Services, s.d.).

SRT

RTMP rimane molto diffuso, ma presenta alcuni limiti.
Essendo basato principalmente su *TCP*, può soffrire quando la rete presenta perdita di pacchetti o variazioni importanti delle condizioni di trasmissione.
La ritrasmissione dei dati può infatti aumentare la latenza.

Inoltre *RTMP* è nato in un periodo molto diverso dall'attuale Internet e non è stato progettato specificamente per affrontare alcuni scenari moderni di contribution attraverso reti mobili, Wi-Fi instabili o collegamenti geografici complessi.
Per questo sono nate tecnologie più recenti come ***SRT* (Secure Reliable Transport)**, che cercano di combinare:
bassa latenza + recupero delle perdite + cifratura + maggiore resilienza alle reti instabili.
Quindi:

- *RTMP* privilegia una connessione semplice e affidabile tra encoder e server.
- *SRT* nasce invece per rendere il trasporto live più robusto anche quando la rete è instabile o presenta perdita di pacchetti (Haivision, s.d.-c).

L'SRT è un protocollo progettato per trasportare audio, video e altri dati in tempo reale attraverso reti *IP* anche quando queste presentano condizioni non ideali, come:

- Perdita di pacchetti.
- Jitter.
- Variazioni della banda disponibile.

- Congestione.
- Ritardi variabili.

SRT è stato sviluppato originariamente da Haivision e successivamente pubblicato come tecnologia open source (Haivision, s.d.-c).
È pensato soprattutto per collegamenti di contribution e distribuzione professionale attraverso reti non controllate, come Internet pubblico.

A differenza di *RTMP*, che utilizza normalmente *TCP*, *SRT* è costruito sopra *UDP (Haivision, s.d.-c)*.
Il modello è quindi: audio/video → *SRT* → *UDP* → *IP*.

UDP non introduce propri riscontri di ricezione o ritrasmissioni; è SRT ad aggiungere questi meccanismi. Anche *TCP* può inviare più segmenti prima di ricevere conferma, quindi la differenza non consiste nell'attendere un riscontro per ogni singolo pacchetto *(Postel, 1980; Eddy, 2022).*
Questo è utile per mantenere la latenza bassa.
UDP, però, non garantisce da solo:

- Consegna dei pacchetti.
- Ordine dei pacchetti.
- Ritrasmissione dei dati persi.

SRT aggiunge quindi sopra *UDP* una serie di meccanismi che cercano di ottenere affidabilità senza rinunciare completamente alla bassa latenza.

Possiamo pensarlo così:

```
UDP
↓
Velocità e bassa latenza
+
SRT
↓
Recupero perdite
Controllo temporale
Cifratura
```

Uno dei meccanismi principali utilizzati da *SRT* è l'***ARQ* (Automatic Repeat reQuest)**.
Quando il ricevitore rileva che manca un pacchetto, può richiederne la ritrasmissione al mittente (Haivision, s.d.-c).

Per esempio:

```
1 → 2 → X → 4 → 5
        ↑
   Packet 3 missing
```

Il ricevitore richiede di ritrasmettere il pacchetto 3 e il mittente può inviarlo nuovamente.

Questo comportamento ricorda in parte *TCP*, ma c'è una differenza fondamentale.

Nella modalità live, SRT usa i timestamp di trasmissione e la latenza configurata per gestire le scadenze di consegna, senza dover interpretare il contenuto audiovisivo (Haivision, s.d.-a; Haivision, s.d.-b).
Se un pacchetto arriva troppo tardi per poter essere utilizzato, continuare ad aspettarlo non avrebbe senso.
Con il meccanismo Too-Late Packet Drop abilitato, SRT può scartare i pacchetti che non possono più essere consegnati in tempo. Questa opzione è attiva per impostazione predefinita nella modalità live (Haivision, s.d.-a).

Internet non garantisce che i pacchetti arrivino con gli stessi intervalli con cui sono stati inviati.
Per esempio, una sorgente potrebbe inviare un pacchetto ogni 20 ms:
Sorgente:

- Packet 1 → 0 ms
- Packet 2 → 20 ms
- Packet 3 → 40 ms
- Packet 4 → 60 ms

A causa del jitter della rete, però, potrebbero arrivare al ricevitore in questo modo:
Arrivo dalla rete:

- Packet 1 → 0 ms
- Packet 2 → 20 ms
- Packet 3 → 63 ms
- Packet 4 → 68 ms

Se i pacchetti venissero immediatamente consegnati al decoder, anche la temporizzazione del flusso risulterebbe irregolare.
SRT utilizza quindi un meccanismo chiamato ***TSBPD* (TimeStamp Based Packet Delivery)**.
Il *TSBPD* utilizza i timestamp assegnati ai pacchetti dalla sorgente per capire quando ciascun pacchetto dovrebbe essere consegnato all'applicazione (Haivision, s.d.-b).
Invece di inoltrare immediatamente ogni pacchetto quando arriva dalla rete, *SRT* introduce intenzionalmente un certo ritardo e mantiene temporaneamente i pacchetti nel buffer di ricezione.
Per esempio, in un modello semplificato con latenza configurata di 100 ms e riferimenti temporali già allineati, trascurando il ritardo di propagazione di base:

Pacchetto	**Timestamp originale**	**Arrivo della rete**	**Consegna all'applicazione**
Packet 1	0 ms	0 ms	100 ms
Packet 2	20 ms	20 ms	120 ms
Packet 3	40 ms	63 ms	140 ms

Packet 4	60 ms	68 ms	160 ms

Anche se i pacchetti arrivano dalla rete in modo irregolare, vengono quindi consegnati all'applicazione rispettando nuovamente gli intervalli temporali originali.
Il percorso diventa:
Rete irregolare → buffer *SRT*/*TSBPD* → temporizzazione ricostruita → decoder

Questa latenza aggiuntiva ha anche un secondo vantaggio fondamentale: fornisce tempo per recuperare eventuali pacchetti persi.
Se manca un pacchetto, infatti, *SRT* deve avere il tempo necessario per:
Rilevare la perdita → richiedere la ritrasmissione → ricevere il pacchetto → inserirlo nella sequenza.
Se il pacchetto ritrasmesso arriva prima del momento previsto per la sua consegna, può essere recuperato senza interrompere il flusso (Haivision, s.d.-b).

Da qui nasce il compromesso fondamentale di *SRT*:

- Latenza configurata maggiore → più tempo nel buffer per compensare jitter e recuperare perdite (Haivision, s.d.-b).
- Latenza configurata minore → meno tempo per recuperare i pacchetti. La sola capacità del buffer in byte non determina il ritardo di consegna (Haivision, s.d.-a; Haivision, s.d.-b).

Questo riduce gli effetti del jitter sul decoder.

I pacchetti *IP* non devono necessariamente arrivare nello stesso ordine in cui sono stati inviati.
SRT può riconoscere la sequenza corretta e ricostruirla prima di consegnare i dati all'applicazione (Haivision, s.d.-c).
Questa funzione è particolarmente importante quando il flusso attraversa reti complesse o congestionate.

SRT monitora continuamente le condizioni della connessione e utilizza meccanismi di controllo della congestione e della velocità di trasmissione (Haivision, s.d.-a).

Per esempio, in una situazione del genere:

- Il video richiede 20 Mbit/s.
- La rete riesce a sostenere soltanto 8 Mbit/s.

SRT non può aumentare la banda fisica disponibile né ricodificare autonomamente il video per ridurne il bitrate.
Se la sorgente produce stabilmente 20 Mbit/s e il collegamento ne sostiene soltanto 8, il flusso non è sostenibile così com'è (Haivision, s.d.-a).
L'encoder o l'applicazione, se prevedono questa funzione, possono ridurre il bitrate in risposta alle condizioni osservate; in alternativa occorre un collegamento con capacità sufficiente, considerando anche l'overhead e le ritrasmissioni.

SRT supporta la cifratura del payload tramite *AES*, quando abilitata e configurata sui due endpoint (Haivision, s.d.-a).
Questo è particolarmente importante quando un flusso professionale attraversa Internet pubblico.

Una connessione *SRT* coinvolge normalmente due endpoint.
Uno può funzionare come Caller e l'altro come Listener. Questi ruoli riguardano l'avvio della connessione e non impongono chi debba inviare il contenuto; è disponibile anche la modalità Rendezvous (Haivision, s.d.-a).

Il Caller avvia la connessione:

```
Caller ──────────────────→ Listener
```

Il Listener rimane in attesa di connessioni in ingresso.
Per esempio:

```
OBS / Encoder
    Caller
      ↓
Internet
      ↓
Streaming Server
    Listener
```

Quando la connessione viene stabilita, i due endpoint possono iniziare a scambiarsi il flusso *SRT*.

Uno degli utilizzi principali di *SRT* è nella contribution.
RTMP e *SRT* possono entrambi essere utilizzati per portare un flusso live verso una piattaforma, ma seguono filosofie differenti.

	RTMP	***SRT***
Trasporto sottostante	*TCP*.	*UDP*.
Recupero perdite	Gestito da *TCP*	*ARQ* controllato da *SRT.*
Jitter	Gestito principalmente attraverso il buffering applicativo	Gestione temporale integrata nel protocollo.
Cifratura	*RTMPS* tramite *TLS*	Supporto *AES* integrato.
Reti instabili	Può accumulare latenza	Progettato specificamente per gestirle.
Scenario tipico	Ingest Internet semplice.	Contribution professionale su Internet.

Latenza	Generalmente contenuta, ma sensibile alle ritrasmissioni *TCP*.	Configurabile in funzione delle condizioni della rete.

La differenza concettuale più importante è questa:

- RTMP si affida a TCP per ottenere affidabilità.
- SRT utilizza UDP e costruisce sopra di esso i propri meccanismi di affidabilità e temporizzazione.

Perché non utilizzare semplicemente *RTP*?
Anche RTP utilizza frequentemente *UDP* ed è progettato per trasportare contenuti real-time.
La differenza è che *RTP* fornisce soprattutto gli strumenti necessari a:

- Identificare i pacchetti.
- Ordinarli.
- Temporizzarli.
- Consentire, insieme a RTCP, la sincronizzazione tra flussi differenti.

RTP, da solo, non implementa un sistema completo di recupero delle perdite, cifratura e gestione della latenza pensato specificamente per attraversare reti Internet instabili (Schulzrinne et al., 2003).

RIST

Un'altra tecnologia utilizzata nella contribution professionale è ***RIST*** **(Reliable Internet Stream Transport).**
Come *SRT*, è progettato per trasportare flussi audiovisivi attraverso reti *IP* non affidabili, come Internet pubblico, utilizzando meccanismi di recupero dei pacchetti persi e mantenendo una latenza contenuta.

RIST nasce dalle specifiche del Video Services Forum per il trasporto professionale interoperabile. Nel Simple Profile utilizza RTP come base per i media e RTCP per il feedback, aggiungendo il recupero delle perdite (Video Services Forum, 2020).
Per questo *SRT* e *RIST* possono essere considerati due approcci moderni allo stesso problema: trasportare video professionale attraverso Internet senza richiedere una rete dedicata.

- *RIST* nasce perché il settore broadcast aveva bisogno di una specifica comune che garantisse l'interoperabilità tra apparati di produttori differenti (Video Services Forum, 2020).
- *SRT* nasce invece come tecnologia sviluppata da Haivision e successivamente resa open source e adottata da un ampio ecosistema.

Esempio
Immagina una televisione che utilizza un encoder del produttore A per inviare un contributo video da uno stadio e un decoder del produttore B nella sede centrale.

Se entrambi supportano profili e funzioni RIST compatibili, con formati audiovisivi e configurazioni coerenti, possono scambiarsi il flusso anche se sono prodotti da aziende differenti (Video Services Forum, 2020).
In questo modo l'emittente non è costretta ad acquistare encoder e decoder dello stesso marchio.

Ciò non significa, tuttavia, che SRT sia incompatibile con apparati di produttori differenti: un encoder di un produttore può inviare un flusso *SRT* a un server o a un decoder realizzato da un altro produttore, purché entrambi implementino correttamente il protocollo (Haivision, s.d.-c).

Materiale di supporto

Esercizi

Esercizio 1 — Quale protocollo utilizzeresti?
Una produzione deve trasmettere un evento live da una sede temporanea verso una piattaforma di streaming.

Sono disponibili tre possibili situazioni:

- Scenario A. Connessione in fibra stabile. La piattaforma accetta un normale flusso proveniente da OBS e fornisce indirizzo del server e stream key.
- Scenario B. Collegamento attraverso una rete mobile che presenta occasionalmente jitter, perdita di pacchetti e variazioni della banda disponibile.
- Scenario C. Collegamento professionale tra l'encoder di una troupe esterna e l'infrastruttura broadcast di un'emittente. I due apparati sono prodotti da aziende differenti e l'interoperabilità rappresenta un requisito importante.

Per ciascuno scenario scegli tra RTMP/RTMPS, SRT e RIST e motiva la decisione.

- Distingui innanzitutto contribution e ingest.
- Spiega perché, per l'ingest da software come OBS, il solo RTP richieda ulteriori meccanismi di configurazione e gestione, e perché possa essere utile una soluzione integrata come RTMP/RTMPS o SRT.
- Per lo Scenario A spiega perché RTMP o RTMPS può rappresentare una soluzione semplice.
- Descrivi il percorso: Camera/Microfono → OBS → _____ → Streaming Server.
- Quale funzione svolgono l'indirizzo del server e la stream key?
- Perché la stream key deve essere trattata come informazione riservata?
- Spiega il vantaggio di TCP per RTMP e il possibile svantaggio quando vengono persi dei dati.
- Quale vantaggio aggiunge RTMPS rispetto a RTMP?
- Per lo Scenario B spiega perché SRT può essere più appropriato.
- Perché SRT utilizza UDP ma non rinuncia completamente all'affidabilità?
- Descrivi il ruolo dell'ARQ.
- Spiega perché SRT non tenta necessariamente di recuperare indefinitamente ogni pacchetto perso.
- Che funzione svolge il TSBPD quando i pacchetti arrivano con intervalli irregolari?
- Commenta il compromesso: Latenza SRT configurata maggiore → più tempo per recuperare le perdite ↔ maggiore ritardo di consegna.
- Spiega inoltre la differenza tra Caller e Listener in una connessione SRT.
- Per lo Scenario C spiega perché RIST può risultare particolarmente interessante nel settore broadcast.
- Quale importanza assume l'interoperabilità tra apparati di produttori differenti?
- Costruisci infine una tabella comparativa con:

Protocollo	Trasporto	Recupero	Sicurezza	Scenario più

		perdite		adatto
RTMP/RTMPS				
SRT				
RIST				

- Perché non esiste un protocollo di contribution universalmente migliore, ma la scelta dipende dalle condizioni della rete, dalla latenza richiesta, dall'affidabilità necessaria e dall'infrastruttura coinvolta?

Esercizio 2 — Realizzare un ingest live con OBS
Devi utilizzare OBS Studio per simulare la postazione di produzione di una piccola diretta. Prepara una scena contenente almeno:
- Una sorgente video, che può essere una webcam oppure un file preregistrato.
- Una sorgente audio.
- Un titolo o un elemento grafico.
- Una seconda scena da utilizzare durante la trasmissione.

Procurati un endpoint *RTMP*/*RTMPS* di ingest composto da:
- Server: indirizzo del server di streaming.
- Stream key: identificatore assegnato alla trasmissione.

Nota: Per una prova tecnica su Twitch puoi usare la modalità di test con Twitch Inspector, aggiungendo «?bandwidthtest=true» alla stream key. Il flusso raggiunge l'ingest ma non viene mostrato in diretta agli spettatori: è un test di banda, non una diretta privata. Controlla la connessione tramite Inspector; per verificare visivamente il video remoto usa un endpoint di prova che consenta tale visualizzazione (Twitch, s.d.).

Il tuo obiettivo è realizzare concretamente il percorso studiato nella dispensa: Sorgenti → OBS → Encoding → RTMP/RTMPS → Ingest Server.
- Configura in OBS le sorgenti audio e video e verifica che vengano correttamente acquisite.
- Imposta una codifica adatta alla trasmissione e annota:
  - Risoluzione.
  - Frame rate.
  - Bitrate video.
  - Bitrate audio.
- Inserisci in OBS l'indirizzo del server e la stream key forniti.
- Avvia la trasmissione e verifica che il flusso raggiunga correttamente la piattaforma.
- Durante il test cambia scena e verifica che la connessione continui. Se usi un endpoint con visualizzazione remota, controlla anche il cambio di scena; la modalità bandwidthtest di Twitch non offre una diretta visibile agli spettatori.
- Individua concretamente nel tuo esperimento:
  - Acquisizione.
  - Encoding.
  - Contribution.

    - Ingest.
- Spiega quale operazione viene effettuata da OBS e quale invece dal server della piattaforma.
- Osserva le statistiche mostrate da OBS durante la trasmissione e registra eventuali problemi relativi alla connessione o alla trasmissione.
- Interrompi la diretta e modifica significativamente il bitrate video. Ripeti quindi il test.
- Confronta i due tentativi:
    - Il collegamento rimane stabile?
    - Compaiono perdite o difficoltà di trasmissione?
    - La banda disponibile è sufficiente rispetto al bitrate impostato?
- Immagina ora di dover realizzare la stessa diretta attraverso una connessione mobile instabile. Spiega, senza necessariamente configurarlo realmente, perché potresti valutare SRT al posto di RTMP/RTMPS.
- Indica quali problemi della rete SRT cercherebbe di gestire meglio:
    - Packet loss.
    - Jitter.
    - Variazioni della banda.
    - Ritardi variabili.
- Spiega infine perché il protocollo utilizzato da OBS per raggiungere l'ingest server non determina necessariamente il protocollo con cui gli spettatori riceveranno successivamente la diretta.

Bibliografia

Adobe (2021). Adobe Flash Player EOL General Information. https://www.adobe.com/products/flashplayer/end-of-life.html

Amazon Web Services (s.d.). Amazon IVS Streaming Configuration. Documentazione Amazon IVS. https://docs.aws.amazon.com/ivs/latest/LowLatencyUserGuide/streaming-config.html

Eddy, W. (a cura di) (2022). Transmission Control Protocol (TCP). RFC 9293. RFC Editor. https://www.rfc-editor.org/rfc/rfc9293.html

Garcia Murillo, S., & Gouaillard, A. (2025). WebRTC-HTTP Ingestion Protocol (WHIP). RFC 9725. RFC Editor. https://www.rfc-editor.org/rfc/rfc9725.html

Haivision (s.d.-a). API Socket Options. Documentazione ufficiale SRT. https://github.com/Haivision/srt/blob/master/docs/API/API-socket-options.md

Haivision (s.d.-b). Latency. Documentazione ufficiale SRT. https://github.com/Haivision/srt/blob/master/docs/features/latency.md

Haivision (s.d.-c). SRT: Secure, Reliable, Transport. Repository ufficiale e documentazione del progetto. https://github.com/Haivision/srt

Parmar, H., & Thornburgh, M. (a cura di) (2012). Adobe's Real Time Messaging Protocol. Specifica RTMP 1.0. Adobe Systems.https://veovera.org/docs/legacy/rtmp-v1-0-spec.pdf

Postel, J. (1980). User Datagram Protocol. RFC 768. RFC Editor. https://www.rfc-editor.org/rfc/rfc768.html

Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). RTP: A Transport Protocol for Real-Time Applications. RFC 3550. RFC Editor. https://www.rfc-editor.org/rfc/rfc3550.html

Twitch (s.d.). Video Broadcast. Twitch Developers. Consultato il 18 ottobre 2026. https://dev.twitch.tv/docs/video-broadcast/

Video Services Forum (2020). Reliable Internet Stream Transport (RIST): Protocol Specification — Simple Profile. TR-06-1:2020. https://static.vsf.tv/download/technical_recommendations/VSF_TR-06-1_2020_06_25.pdf

Approfondimenti

- Amazon Web Services, Configurazione dello streaming di Amazon IVS, Documentazione Amazon Interactive Video Service — Una guida tecnica dedicata alla configurazione dei flussi audiovisivi inviati dagli encoder alle piattaforme di streaming. La documentazione approfondisce i protocolli RTMP, RTMPS e SRT, illustrando le modalità di collegamento agli endpoint di ingest, l'utilizzo delle stream key e i parametri necessari per stabilire una trasmissione. È particolarmente pertinente alla prima parte della dispensa perché permette di comprendere concretamente come un software di produzione, come OBS, possa inviare audio e video verso un server centralizzato. Il confronto tra RTMPS e SRT consente inoltre di osservare alcune differenze tra protocolli basati su TCP e soluzioni progettate per il trasporto affidabile attraverso UDP.
- Amazon Web Services, Cos'è lo streaming a bassa latenza di Amazon IVS?, Documentazione Amazon Interactive Video Service — Una panoramica delle tecnologie utilizzate per acquisire, elaborare e distribuire contenuti audiovisivi live attraverso un'infrastruttura cloud. La documentazione introduce RTMPS e SRT come protocolli di ingest, evidenziando le esigenze di affidabilità, sicurezza e gestione delle condizioni di rete. È particolarmente pertinente alla distinzione sviluppata nella dispensa tra il trasporto del flusso dalla sorgente alla piattaforma e la successiva distribuzione agli spettatori. Permette inoltre di comprendere perché la latenza complessiva di una diretta dipenda da più componenti e non possa essere attribuita esclusivamente al protocollo utilizzato durante la contribution.
- Amazon Web Services, Acquisizione dei flussi in IVS: streaming in tempo reale, Documentazione Amazon IVS — Una documentazione tecnica che esamina il processo di acquisizione di flussi audiovisivi destinati a servizi interattivi in tempo reale. Il testo presenta RTMP, RTMPS e WHIP come possibili tecnologie di ingest e approfondisce i parametri di codifica necessari per rendere compatibili gli stream con l'infrastruttura di ricezione. È particolarmente pertinente alla distinzione proposta nella dispensa tra protocolli di trasporto e requisiti applicativi della piattaforma. Offre inoltre un'estensione del quadro teorico, mostrando come l'ingest possa utilizzare tecnologie differenti a seconda che il servizio privilegi la distribuzione live tradizionale oppure la comunicazione interattiva a latenza molto ridotta.
- Amazon Web Services, Contenitori, protocolli e sistemi downstream, Documentazione AWS Elemental MediaLive — Una documentazione tecnica dedicata ai formati e ai protocolli utilizzati per trasferire contenuti audiovisivi tra sistemi di produzione, elaborazione e distribuzione. Il testo presenta differenti

modalità di output, tra cui RTMP, RTMPS, SRT e UDP, mostrando come ciascuna possa essere utilizzata per collegare componenti di un'infrastruttura streaming. È particolarmente pertinente alla sezione della dispensa dedicata alla contribution professionale, perché permette di osservare come i protocolli vengano impiegati concretamente per trasportare un flusso verso un sistema downstream. La documentazione aiuta inoltre a distinguere il protocollo utilizzato per il trasporto dal formato dei contenuti e dalle caratteristiche del sistema ricevente.

- Amazon Web Services, Porte delle sorgenti nei flussi MediaConnect, Documentazione AWS Elemental MediaConnect — Un approfondimento tecnico sulla configurazione delle porte di rete utilizzate dai protocolli supportati da AWS Elemental MediaConnect. La documentazione include riferimenti a RIST, SRT, RTP e altri protocolli impiegati nel trasporto professionale dei contenuti audiovisivi. È particolarmente pertinente alla sezione finale della dispensa dedicata a RIST, perché permette di osservare come le differenti tecnologie possano richiedere configurazioni specifiche delle porte e dell'infrastruttura di ricezione. Il documento costituisce un approfondimento operativo, utile per comprendere alcune implicazioni concrete dell'interoperabilità tra encoder, server e sistemi di distribuzione, pur non sostituendo una trattazione completa delle specifiche RIST.
- IBM, Cos'è il computer networking?, IBM Think — Un'introduzione ai principi fondamentali delle reti di calcolatori, con particolare attenzione ai livelli di comunicazione, ai protocolli e al trasferimento dei dati attraverso Internet. L'articolo approfondisce il ruolo di TCP e UDP, evidenziando le differenze tra un trasporto orientato alla consegna affidabile e ordinata dei dati e uno basato su datagrammi che non prevede autonomamente meccanismi di recupero delle perdite. È particolarmente pertinente al confronto della dispensa tra RTMP e SRT, perché permette di comprendere le caratteristiche dei protocolli di trasporto sui quali si basano. Fornisce inoltre le basi necessarie per spiegare perché SRT debba introdurre propri meccanismi di ritrasmissione e controllo temporale sopra UDP.
- IBM, Che cos'è l'ottimizzazione della rete?, IBM Think — Un approfondimento sui meccanismi che influenzano le prestazioni e l'efficienza delle comunicazioni attraverso reti IP. L'articolo affronta il funzionamento di TCP, il controllo del flusso, le finestre di trasmissione e le differenze rispetto a UDP, mostrando come le scelte effettuate a livello di trasporto possano influenzare il comportamento delle applicazioni. È particolarmente pertinente alla discussione della dispensa sui compromessi tra affidabilità e latenza. Permette di comprendere perché il recupero dei pacchetti persi possa introdurre ritardi e perché i protocolli progettati per il trasporto multimediale live debbano gestire attentamente il tempo disponibile per completare le ritrasmissioni.
- Server Room, Tutorial sullo streaming video in diretta con OBS Studio, Documentazione Server Room — Una guida operativa dedicata alla configurazione di OBS Studio per trasmettere contenuti audiovisivi verso un server RTMP. Il tutorial illustra le operazioni necessarie per configurare le sorgenti, selezionare le impostazioni di codifica e inserire l'indirizzo del server e le credenziali dello stream. È particolarmente pertinente agli esempi della dispensa relativi al percorso Camera → OBS → RTMP → Streaming Server. Permette di collegare le caratteristiche teoriche di RTMP a un'applicazione concreta, mostrando come il software di produzione stabilisca una sessione con il server e inizi a pubblicare progressivamente i dati della diretta.

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.

