Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Studio delle Problematiche ed Evoluzione dello Streaming Adattivo su HTTP

Relatore:
Dott. Luca Bedogni

Presentata da:
Giulio Biagini

Sessione III
Anno Accademico 2016/2017

“Cadendo, la goccia scava la pietra,
non per la sua forza, ma per la sua costanza”
- cit. Lucrezio.

Abstract

Negli ultimi anni abbiamo assistito ad un notevole aumento del **traffico dati mobile**, trend confermato anche per il futuro, generato in gran parte da *dispositivi smart* e perlopiù legato alla metodica di accesso e scambio dati nota con il nome di **video streaming**. Si stima infatti che circa l'80% della bandwidth mobile mondiale nel 2021 sarà utilizzata per lo scambio di dati audio/video. Risulta dunque di fondamentale importanza per i provider di rete e dei servizi di video streaming, soprattutto quelli forniti *on-demand*, cercare di comprendere quali sono i parametri che maggiormente influenzano la cosiddetta **QoE - Quality of Experience** sperimentata dagli utenti finali, in modo da garantire loro il miglior servizio possibile (**QoS - Quality of Service**), mantenendo comunque alti i profitti.
Nell'ambito delle applicazioni e servizi di video streaming, in particolare, la *qualità* percepita dagli utenti finali risulta fortemente influenzata da diversi **effetti di degradazione** che possono verificarsi a causa di *cattive condizioni di rete*, soprattutto nello specifico ambito della *mobilità*.
Da questi studi risulta come, ad esempio, gli utenti preferiscono sperimentare *effetti di degradazione temporale* durante la riproduzione di contenui multimediali piuttosto che *effetti di degradazione della qualità*, risultati da cui derivano scelte implementative quali, ad esempio, l'utilizzo del *protocollo TCP* e la nascita del cosiddetto **HTTP-based adaptive streaming**, ovvero quella tecnica che permette di adattare la qualità dei segmenti scaricati in base al throughput, in modo da minimizzare il numero di interruzioni del servizio e massimizzare, al contempo, la qualità del media in download.

Dopo una rapida analisi delle principali tecnologie oggi utilizzate, si evince come vi sia la necessità di definire un unico formato per il *manifest* e per la struttura dei *segmenti*. Si arriva così alla definizione dello **standard MPEG-DASH** con le proprie **euristiche di adattamento**.
Tutti questi argomenti sono oggetto di studio ed approfondimento all'interno di questa Tesi di Laurea.

Indice

Capitolo 1

Introduzione

Il **traffico dati mobile** è cresciuto molto negli ultimi anni, trend in aumento nel futuro, gran parte del quale risulta essere generato da *dispositivi "smart"*, perlopiù smartphone e phablet, i quali, è stimato, saranno responsabili, di qui ai prossimi 5 anni, dell'*86% del traffico mobile mondiale.*
La stragrande maggioranza di questo traffico è generato dal **video streaming**, ovvero una metodica di condivisione di contenuti multimediali che permette l'accesso ai dati nonostante il download degli stessi non sia completo e continui, pertanto, in background. Le stime che riguardano il consumo della banda circa tale metodica, parlano di un *78% della bandwidth mobile totale* per il 2021, nonostante questa continuerà a crescere negli anni.
Esaminando il trend attuale, ci si rende conto di come i principali provider che basano la fornitura dei propri servizi sul video streaming quali *YouTube*, *Netflix*, *Hulu*, ecc, ne implementano una versione basata sul *protocollo HTTP*, anziché utilizzare protocolli che permettono la distribuzione di servizi in tempo reale come, ad esempio, *RTP - Real-time Transport Protocol.* L'uso del *protocollo TCP*, su cui HTTP si basa, comporta *interruzioni del servizio* che si manifestano come *effetti di degradazione temporale.* Al fine di ridurre questa problematica, sono state implementate strategie che consentono di *adattare la qualità* del contenuto multimediale all'effettiva bitrate. Tale tecnologia prende il nome di **HTTP-based adaptive video streaming**.

1.1 Dati Traffico Mobile

Negli ultimi anni abbiamo assistito ad un costante e continuo aumento della condivisione e richiesta di dati tramite Internet, caratterizzata da un trend in forte crescita nel futuro, soprattutto per quanto riguarda il traffico veicolato su **reti mobili**.

1.1.1 Aumento del Traffico Dati

Secondo quanto riportato nel *Cisco Visual Networking Index (VNI): Global Mobile Data Traffic Forecast Update* [1], parte del *Cisco VNI Forecast*, ovvero un'iniziativa in corso per monitorare e prevedere l'impatto delle applicazioni di visual networking sulle reti globali, il traffico dati mobile:

- nel 2011 ammontava a circa 400 petabyte al mese;
- negli ultimi mesi del 2015 ammontava a circa 4.4 esabyte al mese, facendo registrare un incremento di circa 11 volte in 4 anni;
- negli ultimi mesi del 2016 ammontava a circa 7.2 esabyte al mese (Figura 1.1, prima colonna);
- nel 2021 è stimato attorno ai 49 esabyte al mese, per un totale di più di $\frac{1}{2}$ zettabyte all'anno (Figura 1.1, ultima colonna).

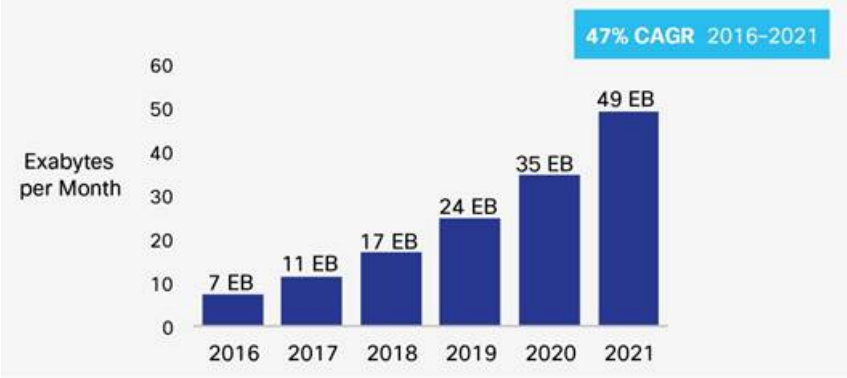

Figura 1.1: Crescita traffico dati mobile [1]

Analizzando questi dati è possibile vedere come negli ultimi 5 anni (dal 2011 al 2016) l'utilizzo della rete mobile ha fatto registrare una crescita di ben 18 volte, con un incremento del 63% solamente nell'ultimo anno (2016) rispetto allo stesso periodo dell'anno precedente (fine 2015).

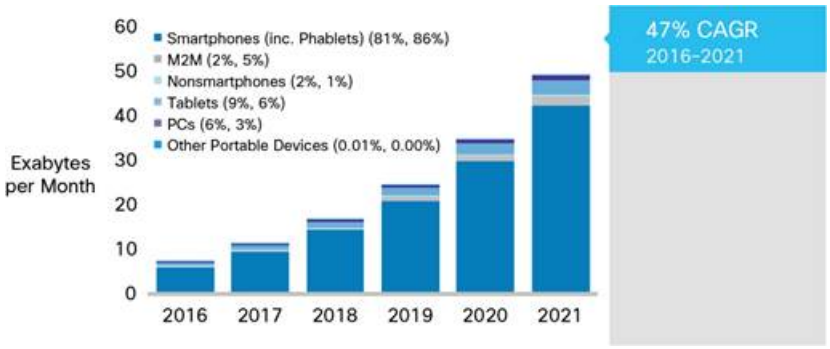

Figura 1.2: Crescita traffico dati "smart" [1]

Secondo la sopra citata ricerca, inoltre, la maggior parte del traffico mobile è generato (e lo sarà anche in futuro) dai cosiddetti **"dispositivi smart"** (nodi mobili con capacità di multimedia/computing avanzate e con tecnologia almeno 3G), perlopiù **smarphone** e **phablet** (Figura 1.2):

- a livello mondiale, gli "smart devices" rappresentavano nel 2016 il 46% delle connessioni dei dispositivi mobili totali (Figura 1.3, prima colonna), con smartphone e phablet al 45% del totale, responsabili dell'89% del traffico dati mobile mondiale (smarphone e phablet dell'81%);

- la stima per il 2021 è che gli "smart devices" arriveranno a toccare il 74.7% delle connessioni totali (Figura 1.3, ultima colonna), dove smartphone e phablet supereranno il 50% del totale, responsabili dell'86% del traffico dati mobile mondiale.

Se si analizzano poi nel dettaglio i dati circa l'utilizzo della bandwith da parte di questi dispositivi, si nota come gran parte di essa sia usata per il **video streaming**, tecnica che permette agli utenti di riprodurre contenuti audio/video mentre sono ancora in download in background, garantendovi

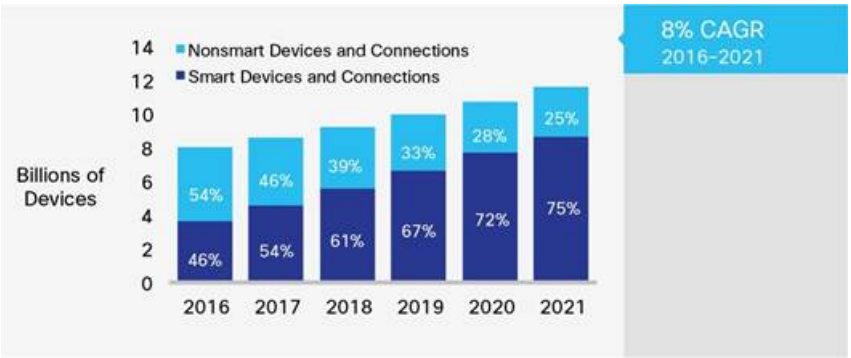

Figura 1.3: Crescita connessioni traffico dati "smart" [1]

un accesso istantaneo, senza attenderne il download completo. Figura 1.4 ne riassume il trend, dove i dati fra parentesi si riferiscono il primo a quanto registrato per il 2016 mentre il secondo indica la previsione per il 2021:

- nel 2016 il traffico dati video via mobile è stato stimato attorno al 60% dell'intero traffico dati mobile;
- la stima per il 2021 è una crescita fino al 78% sull'intero traffico mobile.

Ricapitolando, [1] ci mostra come il traffico dati mobile passerà da una quota di 7.2 esabyte al mese nel 2016 a 49 esabyte nel 2021, come smartphone e phablet passeranno dal rappresentare il 45% del totale dei dispositivi mobili connessi ad oltre il 50% e come essi saranno responsabili dell'86% del traffico mobile mondiale, di cui ben il 78% sarà dovuto al video streaming.

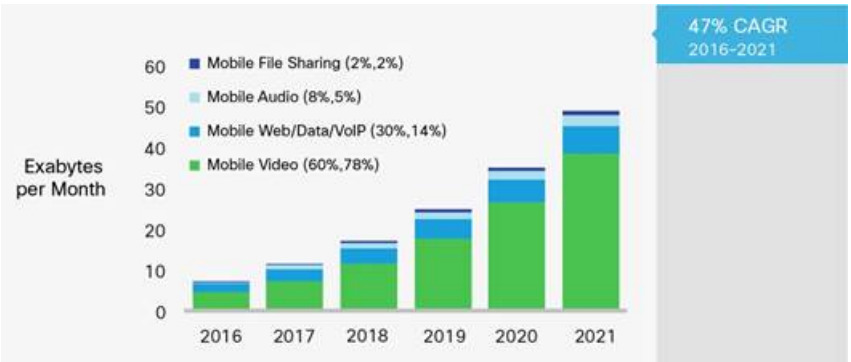

Figura 1.4: Crescita traffico dati video streaming [1]

1.1.2 Aumento delle Connessioni

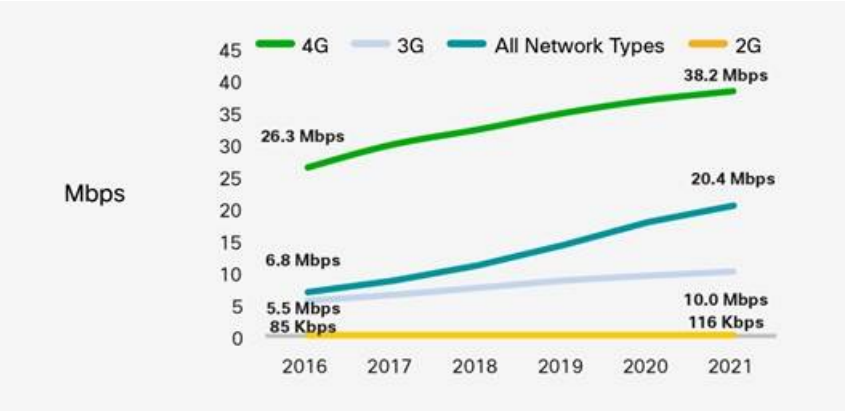

Figura 1.5: Crescita velocità reti mobili [1]

Come abbiamo visto precedentemente, il traffico dati veicolato da reti mobili è cresciuto molto negli utlimi anni e continuerà a farlo anche in futuro, facendo registrare un incremento di crca 40 esabyte al mese di qui ai prossimi 5 anni, ovvero un'aumento di ben 7 volte. Questo potrebbe far pensare ad una saturazione della bandwidth disponibile ma, fortunatamente, sempre secondo quanto riportato in [1], anche la **velocità media delle reti mobili** è in aumento, così come mostrato in Figura 1.5:

- nel 2015 la velocità media delle connessioni su reti mobili è stata registrata attorno a 2 Mbps;
- nel 2016 attorno a 6.8 Mbps, con un incremento di più di 3 volte in un solo anno;
- nel 2021 si stima che tale velocità triplichi ulteriormente, arrivando a superare i 20 Mbps.

Questo fatto è dovuto essenzialmente allo sviluppo di nuove tecnologie come l'**LTE** ed il dislocamento di access point **WiFi** aperti, così come allo sviluppo di nuovi standard quali, ad esempio, quelli di quarta (**4G**) e quinta (**5G**) generazione. Questi ultimi, in particolare, stanno infatti per fare il loro

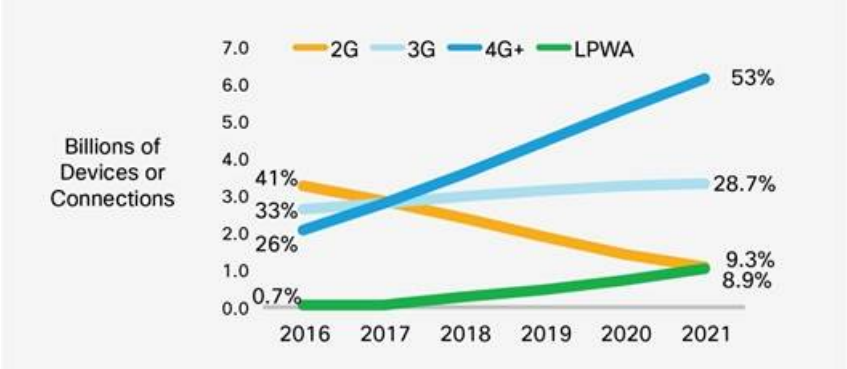

Figura 1.6: Crescita connessioni reti mobili [1]

ingresso nelle nostre vite quotidiane. Si stima che nel 2021, di tutto il traffico generato, il 50% sarà WiFi, il 30% cablato ed il restante 20% sarà mobile. Figura 1.6 riassume molto bene quello che sarà lo scenario che si andrà a delineare nei prossimi anni (2016-2021). Come è possibile osservere:

- le percentuali di utilizzo delle varie tecnologie per la connessione alle reti mobili andranno a ridefinirsi. Si assisterà infatti ad un progressivo passaggio dall'uso di standard via via più obsoleti (2G) verso standard prestazionalmente migliori (4G);

- il numero totale di dispositivi connessi aumenterà negli anni.

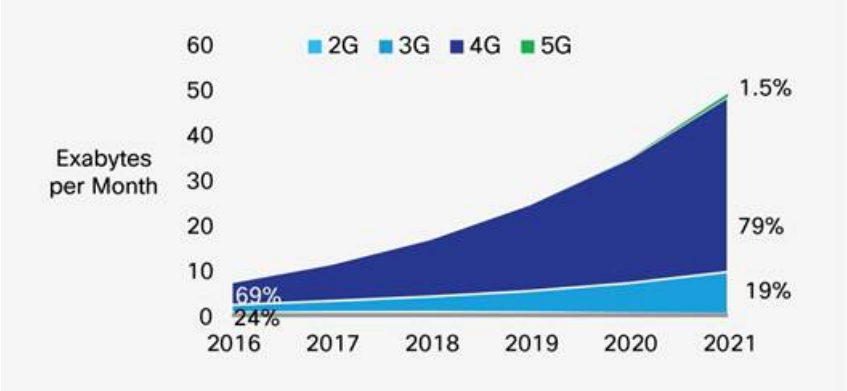

Figura 1.7: Crescita traffico dati mobile per tecnologia [1]

Come mostra Figura 1.7 poi, l'accesso a reti più veloci e che garantiscono più banda genererà inevitabilmente un **traffico dati** molto maggiore.

Figura 1.8: Crescita traffico dati mobile per Paese [1]

Riassumendo, nonostante l'affermarsi di nuove tecnologie e l'incremento della banda che le reti mobili saranno in grado di offrire, il traffico dati continuerà a crescere. Questa situazione è legata al fatto che molti Paesi in via di sviluppo vedranno crescere la propria **economia** ed il proprio **livello di sviluppo tecnologico** negli anni a venire, così come riassunto sia in

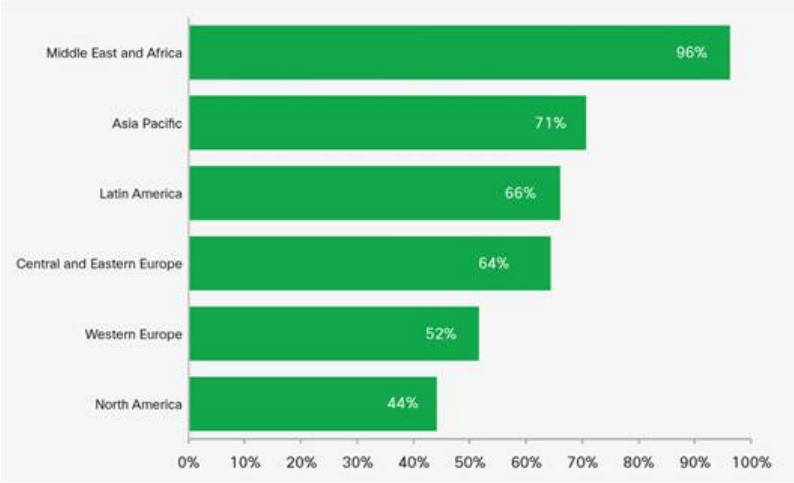

Figura 1.9: Crescita traffico dati mobile per Paese nel 2016 [1]

Figura 1.9 per lo specifico anno 2016, sia in Figura 1.8 come proiezione per gli anni 2017-2021:

- già a fine 2017 la Cina supererà il traffico dati mobile prodotto dagli USA: 1.9 esabyte al mese contro gli 1.6 di questi ultimi;
- le regioni del Medio Oriente e dell'Africa saranno coloro che avranno il più alto tasso di crescita annuale, *CAGR - Compound Annual Growth Rate*, pari al 65%, seguiti dalle regioni dell'Asia che si affacciano sul Pacifico (49%) e dalle regioni dell'America Latina (45%). Questo importante dato emerge già a partire dal 2016 (Figura 1.9) e si conferma anche per gli anni a venire;
- per il 2021 si stima che il numero di dispositivi connessi alla rete mobile pro capite sarà di 1.5.

1.2 Video Streaming

Come abbiamo visto nelle sezioni precedenti, il **video streaming**, ovvero quella metodica di accesso a contenuti audio/video non ancora del tutto scaricati in locale ma il cui download continua in background, è stimato al 78% del traffico mobile mondiale per il 2021 contro il 60% di fine 2016.
Vista la grande importanza che questa metodica di accesso a contenuti multimediali ha tutt'oggi e promette di avere nel futuro, ne saranno presentate in questa sezione le caratteristiche principali.

1.2.1 Video Streaming su HTTP

Grazie all'aumento della banda disponibile ed alla crescita del *World Wide Web*, la necessità di distribuire dati audio e video in pacchetti di piccole dimensioni è andata via via diminuendo. I contenuti multimediali possono tranquillamente essere distribuiti in modo efficiente in segmenti più grandi usando il **protocollo HTTP**. Lo streaming su *HTTP - HyperText Transfert Protocol* ha infatti diversi vantaggi:

- il primo è che l'infrastruttura di Internet si è sviluppata per supportare efficientemente questo protocollo prevedendo, ad esempio, il dislocamento di *cache* allo scopo di ridurre il traffico dati a lungo raggio;

- in secondo luogo è *firewall-friendly*, in quanto quasi tutti i firewall sono configurati per supportarne le connessioni in uscita, a differenza di quanto non accade per *RTP - Real-time Transport Protocol*, ad esempio;

- lo streaming su HTTP, inoltre, può essere gestito dal client senza che esso mantenga *informazioni di stato* relative alla sessione sul server. Per questo motivo, servire un gran numero di client non impone alcun costo aggiuntivo di risorse da parte dei server, i quali possono essere gestiti usando tecniche di ottimizzazione standard.

Per tutti questi motivi, lo streaming su HTTP è diventato un approccio sempre più popolare ed utilizzato per impieghi commerciali.

1.2.2 Effetti di Degradazione Temporale

L'utilizzo del **protocollo TCP** (su cui HTTP si basa) fa sì che nel caso in cui vi siano *cattive condizioni di rete*, l'utente finale di applicazioni e servizi di video streaming forniti *on-demand* quali *YouTube* e *Netflix*, sperimenti **effetti di degradazione temporale**, come *ritardi iniziali* e *pause intermedie*. TCP, infatti, si assicura di consegnare i pacchetti ricevuti nell'ordine in cui il mittente li ha inviati, prevedendo, inoltre, un meccanismo di adattamento della velocità di trasmissione in relazione alle condizioni della rete, al fine di favorirne il de-congestionamento. Questo fatto mette in relazione la presenza di cattive condizioni di rete con il graduale *svuotamento del buffer del video player*, il quale, per riprodurre il video, deve aver scaricato un determinato quantitativo di dati (espresso in tempo di filmato).
Secondo quanto riportato in [2] e [3], una delle situazioni peggiori che si possono verificare durante la visione di un video è la sua interruzione a causa di **stalli**, in quanto gli utenti tendono ad interromperne la riproduzione.

A Longitudinal View of HTTP Video Streaming Performance [2] riporta come il 34% dei video abbia un tempo di download di 3 o più minuti e come, nel caso in cui la qualità sperimentata dagli utenti non sia buona, la loro condivisione scende al 15%. Questi risultati indicano chiaramente come gli stalli scontentino fortemente ed abbastanza velocemente gli utenti, *la metà dei quali* reagisce **stoppando la riproduzione del video**.
Questa situazione suggerisce ai progettisti di applicazioni e servizi in ambito di video streaming on-demand, come non sia una cattiva idea quella di introdurre volontariamente ritardi iniziali al fine di attuare strategie di *pre-buffering* in ottica di evitare l'occorrenza di fastidiose pause intermedie che interrompono bruscamente il servizio. Si è infatti visto come questi infastidiscano gli utenti in maniera minore rispetto al verificarsi di stalli.
Esistono però casi nei quali gli stalli non possono sempre essere evitati. Un esempio è rappresentato dallo specifico ambito della **mobilità** la quale, per sua natura, genera *pattern di stallo non periodici*. La principale problematica che affligge lo streaming audio/video in ambito mobile è rappresentata dal fatto che tali reti possono soffrire di variazioni anche importati nel *data rate* durante gli spostamenti a causa di *effetti di shadowing*, modifica dell'access point a cui si è connessi, ecc... Come è facile intuire, tale situazione ha un'influenza notevole sulla velocità di download alla quale il video viene scaricato dal client, con la possibilità che la riproduzione da parte del player possa essere più volte interrotta a causa della mancanza effettiva dei frammenti video successivi a quelli tutt'ora in riproduzione.

1.2.3 Adaptive Video Streaming

Understanding the Impact of Video Quality on User Engagement [3] mostra come gli utenti tendono, in generale, a reagire più positivamente quando la **qualità del video** in riproduzione **viene ridotta** piuttosto che sperimentare una pausa che interrompe bruscamente la fruizione del servizio.
Al fine di fronteggiare le possibili variazioni di throughput ed impedire, dunque, il verificarsi di interruzioni durante lo streaming, è possibile **monitorare**

come variano le *condizioni della rete* ed **adattare** di conseguenza *la qualità video*: se il throughput aumenta, viene aumentata anche la qualità del filmato in riproduzione, se il throughput diminuisce, diminuirà anche la qualità del video. Questa tecnica è nota con il nome di **adaptive video streaming**.

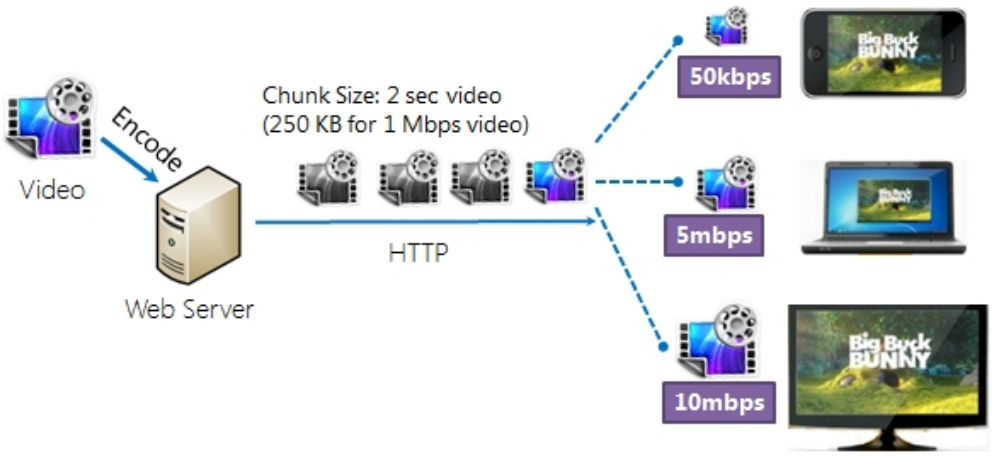

Figura 1.10: Adaptive Video Streaming [4]

Essa prevede, a grandi linee, che i contenuti multimediali vengano divisi in una serie di *segmenti* di una durata temporale determianta, ad esempio 10 secondi ciascuno. Per ogni segmento, i server ne mantengono differenti codifiche, dalle quali ne dipendono la qualità ed "il peso" (in Byte): più è alta la qualità del segmento, più esso "peserà". Quando la qualità della rete migliora, è possibile sfruttare la bitrate maggiore per scaricare, ad esempio, i successivi 10 secondi di video con una qualità migliore, mentre se la qualità della rete è scarsa ed il livello del buffer è in esaurimento, possiamo scaricare il prossimo pacchetto ad una qualità minore. Conoscendo il peso dei pacchetti, il throughput della rete ed il livello del buffer, possiamo attuare qualche tipo di *previsione* e calcolare il peso del prossimo pacchetto da scaricare.
Come è possibile immaginare, le informazioni utilizzate al fine di scegliere la qualità dei pacchetti da scaricare ed il modo in cui sono combinate assieme possono essere le più disparate e definiscono le specifiche istanze degli

algoritmi di adaptive video streaming utilizzati, nonché le cosiddette **euristiche di adattamento** o **di controllo**.

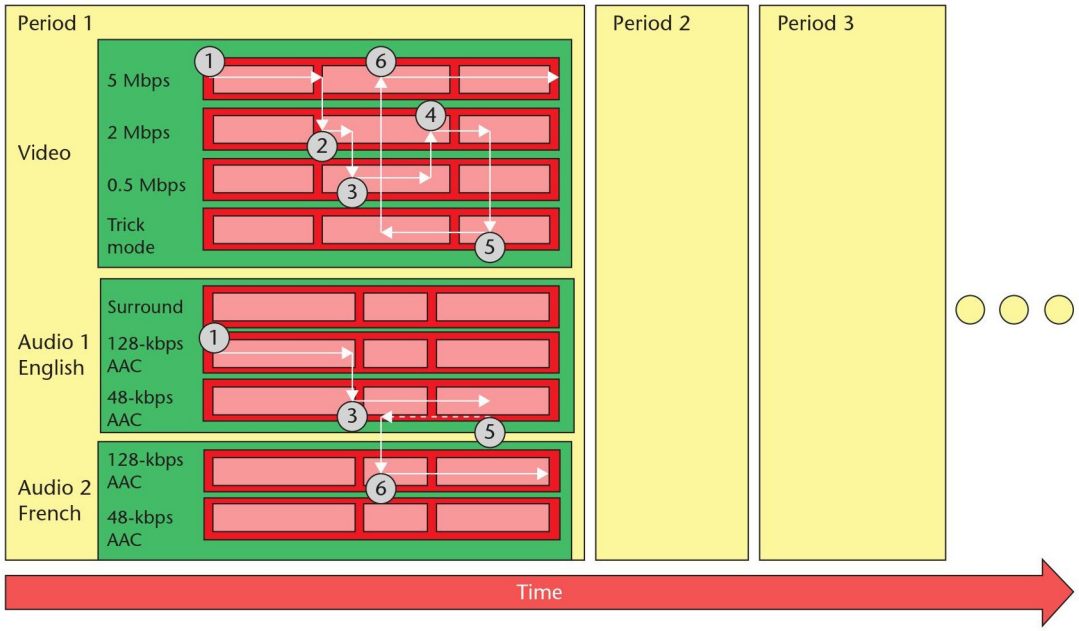

Figura 1.11: Esempio di streaming adattivo dinamico [5]

Figura 1.11 mostra un semplice esempio di streaming adattivo dinamico on-demand,dove il contenuto multimediale è costituito da componenti audio e video. La fonte video è codificata con tre diverse bitrate: 5 Mbps, 2 Mbps e 500 Kbps. In aggiunta, uno stream codificato ad una bassa frame rate è disponibile per la riproduzione in *trick-mode*[1]. L'audio è disponibile in due lingue: l'originale in Francese, codificato in *AAC - Advanced Audio Coding* 128 Kbps e 48 Kbps, la traduzione in Inglese, sia in versione surround che in versione 128 e 48 Kbps AAC:

- lo streaming inizia alla qualità video più alta con audio in lingua Inglese a 128 Kbps (es: la modalità surround non è supportata) (Etichetta 1);

[1]il trick-play manipola lo stream video in modo da includere solo un subset di frame al fine di emulare le operazioni di fast-forward e di rewind

- dopo aver scaricato i primi segmenti audio/video ed aver monitorato la bandwidth, il device si accorge che la banda disponibile è diminuita rispetto ai 5 Mbps iniziali. Al che, al successivo "switching point" (punto al quale è nuovamente possibile scegliere la definizione audio e video dei segmenti da scaricare), decide di passare alla definizione intermedia per il video, 2 Mbps, e lasciare i 128 Kbps per l'audio (Etichetta 2);

- il device continua a monitorare la rete e si accorge che la banda è ulteriormente diminuita. Si trova ora sotto i 2 Mbps, pertanto, al fine di evitare il verificarsi di interruzioni nella fruizione del servizio, decide di scaricare i successivi frame video alla minor qualità possibile, ovvero 500 Kbps con un rate audio di 48 Kbps (Etichetta 3);

- lo streaming continua finché la rete non permette un nuovo switch a qualità più elevate (Etichetta 4);

- ad un certo punto, l'utente decide di mettere la riproduzione video in pausa e di riavvolgere la traccia. Questa operazione, eseguita in trick-mode, causa il silenziamento dell'audio (Etichetta 5);

- i'utente decide di ri-visualizzare parte del video con l'audio in lingua originale ed esegue uno switch dalla traccia Inglese a quella Francese. Il device, inoltre, si accorge che la banda è tornata ai livelli iniziali e riprende lo streaming alla qualità più alta consentita, ovvero con video rate a 5 Mbps ed audio rate di 128 Kbps (Etichetta 6).

Per far fronte alla *dinamicità* dell'ambiente in esame (di cui l'esempio appena riportato rappresenta una semplice ma chiara istanza), la tecnica dello streaming adattivo viene usata da molti fornitori di servizi di video streaming quali *YouTube*, *Hulu*, *Netflix*, ecc...

1.3 Obiettivi

Vista dunque l'importanza che il video streaming ha tutt'ora e si preannuncia avere nel prossimo futuro, questa Tesi di Laurea si pone come **obiettivo** quello di svolgere un'esaustiva fase di ricerca al fine di mettere in luce tutte le motivazioni che hanno concorso all'affermarsi dell'attuale stato dell'arte nell'ambito dell'**HTTP-based adaptive video streaming**, sino alla progettazione ed implementazione del **protocollo MPEG-DASH**.
Questa scelta è stata altresì supportata dal concorrente sviluppo, all'interno della Facoltà, di un *Simulatore MPEG-DASH* sul quale sarà possibile implementare e testare le varie **euristiche di adattamento** previste dal protocollo stesso. Con la presente Tesi di Laurea, si è infatti voluto dare un fondamento teorico a tale progetto, fornendo a chiunque si occuperà di proseguie con questo lavoro, le basi per poterne meglio comprendere il contesto ed il funzionamento. Questa Tesi rappresenta il lavoro di ricerca da me svolto al fine di mostrare quello che è lo **stato dell'arte** circa gli aspetti fondamentali che hanno portato all'implementazione e diffusione dello standard MPEG-DASH e delle varie euristiche di adattamento.

Nel Capitolo 2 ed, in particolare, nelle Sezioni 2.1 e 2.2, vengono introdotti i concetti di *QoS - Quality of Service* e *QoE - Quality of Experience*, fondamentali per comprendere quali sono gli **standard qualitativi** richiesti dagli utenti (QoE) in modo che i service provider di servizi di video streaming on-demand possano offrire loro un servizio di qualità (QoS).
Il Capitolo 3 tratta di tutti quegli aspetti ed annesse **problematiche** che hanno portato alla definizione ed implementazione degli standard per il video streaming adattivo HHTP-based così come li conosciamo oggi.
Sezione 3.1 approfondisce il discorso circa la scelta del **protocollo di trasporto** che è meglio utilizzare in relazione all'influenza che esso esercita sulla qualità percepita dagli utenti. Verrà mostrato come, in caso di cattive condizioni di rete, utilizzando *UDP - User Datagram Protocol* a risultare degradata sarà la *qualità del video*, mentre con *TCP - Transmission Control Protocol* a risultare degradata sarà la *riproduzione del video*. Sarà mostrato

come gli utenti preferiscono sperimentare degradazioni temporali piuttosto che artifatti video quali sfarfallii, salti nello stream, ecc, evidenziando come sia meglio implementare il servizio utilizzando il protocollo TCP. Sarà poi discussa la cosiddetta *IQX hypothesis*, la quale afferma come la QoE decresca in maniera *esponenziale* al crescere di un fattore di degradazione legato alla QoS, giustificando molti dei risultati ottenuti sperimentalmente. Per concludere, una sottosezione mostra come le cattive condizioni di rete siano legate allo svuotamento del buffer del video player *YouTube*, mettendo in luce come si verificano *stalli* ed interruzioni del servizio.
Sezione 3.2 si occupa di approfondire come gli **effetti di degradazione temporale** influenzino la QoE percepita dagli utenti mostrando come, fra il verificarsi di *ritardi iniziali* e *pause intermedie* (*stalli*), gli utenti preferiscano i primi. Questo suggerisce come sia, in generale, una buona strategia quella di introdurre volontariamente ritardi nell'avvio del servizio al fine di attuare, ad esempio, operazioni di *pre-buffering* che permettano di evitare il verificarsi di stalli. In quest'ambito, un importante risultato è dato dalla cosiddetta *WQL hypothesis*, la quale lega il verificarsi dei ritardi iniziali alla degradazione della QoE non da una relazione esponenziale, come accade per gli stalli (IQX hypothesis), ma da una funzione *logaritmica*.
Sezione 3.3 approfondisce ulteriormente l'analisi sulle relazioni che intercorrono tra le percezioni degli utenti circa la qualità alle diverse variabili che influenzano i **pattern di stallo**. Se con le sezioni precedenti abbiamo visto come gli utenti preferiscano il verificarsi di effetti di degradazione temporale piuttosto che effetti di degradazione della qualità e come il verificarsi di ritardi iniziali degradino meno la qualità percepita rispetto al verificarsi di stalli, in questa sezione saranno descritti i contributi che il *numero di stalli*, il *tempo complessivo di stallo*, la *durata di ogni evento di stallo* e la *frequenza di occorrenza degli stalli* hanno sulla QoE.
Una volta identificati i principali fattori in grado di influenzare la qualità percepita dagli utenti e determinato il contributo di ciascuno di essi, nel Capitolo 4 sono elencate le principali **proposte** per l'implementazione di algoritmi di

adaptive video streaming HTTP-based efficaci.
In Sezione 4.1 sono elencate le principali **soluzioni di streaming adattivo proprietarie** che sono state sviluppate da *Microsoft*, *Apple* ed *Adobe*.
Sezione 4.2 racchiude le similitudini che i protocolli precedentemente descritti hanno in comune e presenta lo **standard MPEG-DASH**, descrivendone il formato del *manifest* e dei *segmenti*.
Sezione 4.3 descrive le principali **euristiche di adattamento** che possono essere usate in abbinamento al protocollo MPEG-DASH. Saranno elencate alcune *euristiche throughput-based*, le quali basano le proprie scelte in relazione al throughput istantaneo o mediato su un determinato periodo di tempo ed *euristiche buffer-based*, ovvero algoritmi che guidano la scelta circa la qualità del prossimo segmento da scaricare sulle informazioni relative al livello del buffer del player. Sarà poi esaminato un algoritmo le cui scelte sono guidate da una *logica fuzzy*, ovvero *FDASH* ed un algoritmo il cui comportamento risulta essere pressoché ottimo: *BOLA*.
Il Capitolo 5, infine, conclude questo lavoro di Tesi ricapitolando quello che è lo stato dell'arte circa il video streaming adattivo basato su HTTP e lo standard MPEG-DASH con le proprie euristiche di adattamento.

Capitolo 2

Qualità

Come abbiamo visto nel Capitolo 1, Sezione 1.1, le stime circa l'utilizzo della bandwidth per il prossimo futuro confermano il trend che si è delineato negli utlimi anni, facendone registrare un forte incremento, soprattutto in ambito *mobile*. Di questa, gran parte sarà usata da dispositivi quali *smarphone* e *phablet* soprattutto per lo *streaming di contenuti audio/video*.
Risulta dunque di fondamentale importanza per i fornitori di questa tipologia di contenuti, soprattutto *on-demand*, comprendere a pieno quali sono gli *standard* dei propri utenti al fine di garantire un *servizio di qualità*. Ma cosa si intende per "qualità"? Esistono essenzialmente due tipologie di "qualità":

- la **qualità del servizio** è quella fornita dai service provider tramite la propria applicazione e/o servizio;

- la **qualità dell'esperienza** è quella percepita dagli utenti e deriva dall'utilizzo dell'applicazione e/o servizio.

In questa sezione tratteremo entrambe in modo approfondito, così da fornire al lettore gli strumenti necessari al fine di comprendere meglio quelli che sono i fattori che le influenzano, il cui studio riveste un ruolo importante nell'ambito di questa Tesi di Laurea.

2.1 QoS - Quality of Service

Per comprendere a pieno cosa si intende per **QoS - Quality of Service**, è necessario prima comprendere cosa sia un **servizio**, ovvero un'attività o una serie di attività di natura più o meno intangibile che hanno luogo nell'interazione tra il cliente ed i sistemi del fornitore del servizio e che vengono fornite come soluzioni ai problemi del cliente, es: servizi di *video streaming*. In quest'ambito, per **qualità** ci si riferisce a tutti quei parametri che possono compromettere la fornitura e la fruizione del servizio, in particolare, lavorando con *reti a pacchetto*, con il termine "qualità" sono indicate tutte quelle variabili che possono influenzare questa tipologia di infrastruttura, ovvero:

- *packet loss*: indica la percentuale di pacchetti che la rete nel suo complesso non riesce a consegnare a destinazione. Sebbene essa venga gestita in modi diversi a seconda del protocollo di trasporto utilizzato, questo esula dalla definizione di qualità del servizio: in un protocollo senza riscontro come *UDP* si verificherebbe la mancata ricezione del pacchetto; in un protocollo con riscontro come *TCP*, il destinatario, dopo aver atteso un tempo ragionevole, chiede che l'informazione venga ri-trasmessa, con la possibilità che questo procedimento sia reiterato più volte e causi (anche gravi) ritardi nella trasmissione complessiva;

- *delay*: è il tempo che un pacchetto impiega, una volta immesso sulla rete, a raggiungere la destinazione. Quando questo tempo diventa sostanzioso ed è apprezzabile dall'utente, la qualità del servizio può risentirne. Un esempio è rappresentato dall'uso di protocolli con riscontro come già visto nel caso di TCP;

- *out-of-order*: su alcune reti è possibile che una sequenza di pacchetti arrivi al destinatario in un ordine diverso da quello di origine a causa, ad esempio, dell'instradamento dei pacchetti su percorsi diversi;

- *errori di trasmissione*: può accadere che un pacchetto arrivi a destinazione ma che non sia identico a quello inviato. Molte reti sono in grado

di riconoscere pacchetti errati ed alcune riescono a correggerli. Se il protocollo si accorge dell'errore richiede la ri-trasmissione del pacchetto, altrimenti esso sarà conseganto errato all'applicazione finale;

- *throughput*: la banda disponibile, il cui valore massimo dipende generalmente dal contratto stipulato con il fornitore del servizio, si aggiunge alla lista di parametri in grado di influenzare la qualità del servizio.

2.2 QoE - Quality of Experience

Date le previsioni future circa il grande incremento dell'utilizzo delle reti mobili da parte di dispositivi smart soprattutto per lo straming audio/video, risulta di fondamentale importanza per i fornitori di questa tipologia di servizi comprendere a fondo la cosiddetta **QoE - Quality of Experience**, concetto simile alla *User Experience* ma che affonda le proprie radici nell'ambito delle telecomunicazioni. L'obiettivo è quello di comprendere i requisiti generali di qualità per gli esseri umani, in modo da soddisfare al meglio i propri clienti soprattutto per quanto riguarda i servizi forniti tramite mobile.
Se da una parte abbiamo i clienti, che si trovano in una posizione privilegiata in quanto possono scegliere fra più providers in base alle loro offerte nonché alla qualità esperita ed al costo del servizio, dall'altra abbiamo i fornitori, con la necessità di distribuire i propri contenuti riducendo i costi di trasmissione al fine di incrementare i profitti derivanti dal traffico video.
Secondo quanto riportato nel *Qualinet*[1] *White Paper on Definitions of Quality of Experience* [6]:

- la *qualità* è definita come "il risultato di un processo di confronto e giudizio individuale che include la percezione, la riflessione circa la percezione e la descrizione del risultato stesso";

- l'*esperienza* viene definita come "il flusso individuale di percezioni ed interpretazioni di uno o più eventi".

[1] ICT Action COST IC1003

La QoE viene definita come "il grado di piacere o di noia dell'utente circa la sua esperienza in merito all'utilizzo di un'applicazione o di un servizio", ed ancora: "risulta dal soddisfacimento delle sue aspettative rispetto all'utilità e/o piacere nell'uso dell'applicazione o servizio alla luce della personalità e dello stato attuale dell'utente".
In sintesi, l'**esperienza** riguarda un flusso di percezioni *individuali* che l'utente trasforma in un giudizio di piacere o noia tramite un processo *personale* che prende il nome di **qualità**.
Essendo dunque sia il concetto di qualità che quello di esperienza definiti come *individuali* nel paper sopra citato, risulta evidente come la necessità alla quale si trovano dinanzi gli operatori di dover stabilire una metrica affidabile nella definizione della QoE, in particolar modo per quanto concerne l'erogazione dei servizi di video *on demand*, non sia un compito così semplice.

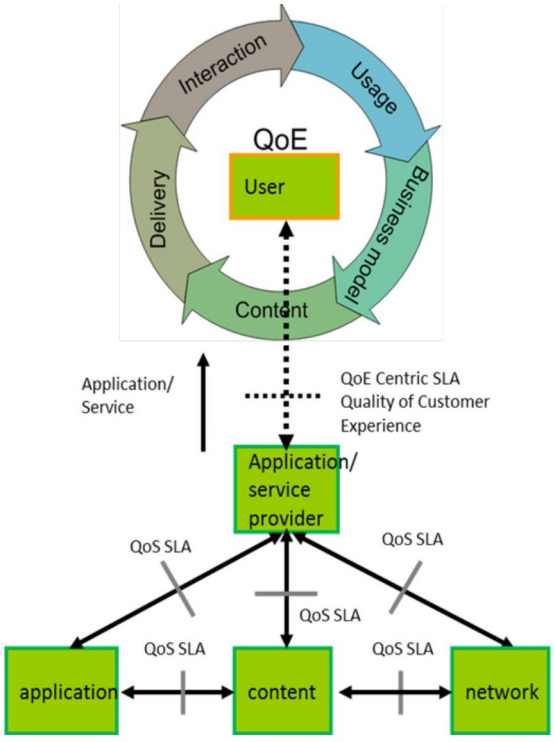

Figura 2.1: QoE nell'ecosistema per i fornitori di servizi/applicazioni [6]

Capitolo 3

Problematiche

In questo capitolo saranno affrontati tutti quegli aspetti che hanno guidato le **scelte** affrontate negli anni, sino alla realizzazione delle varie tecnologie di video streaming così come oggi le conosciamo.
Come abbiamo visto nel capitolo introduttivo ed, in particolare, in Sezione 1.2, i principali fornitori di servizi di video streaming distribuiscono i propri contenuti multimediali utilizzando l'infrastruttura Internet già esistente tramite il *protocollo HTTP*. A questo punto è stato necessario chiedersi quale protocollo di trasporto utilizzare, se **UDP** o **TCP**. Entrambi, da un punto di vista della QoE, in caso di cattive condizioni di rete o se utilizzati nell'ambito della *mobilità*, presentano lati sia positivi che negativi. Con UDP, ad esempio, possono verificarsi perdite di pacchetti o arrivi di pacchetti in un ordine diverso da quello di invio, il ché risulta in *effetti di degradazione della qualità*, come artefatti e salti nello stream. Utilizzando TCP, invece, a risultare degradata sarà la *riproduzione del video*, con il verificarsi di stalli. Occorre dunque effettuare un'indagine accurata da un punto di vista della qualità percepita dagli utenti al fine di poter effettuare le giuste decisioni.
Allo stesso modo, occorre analizzare anche quali, fra le diverse tipologie di **effetti di degradazione temporale** quali *ritardi iniziali* e *pause intermedie*, influenzano maggiormente la qualità percepita dagli utenti finali. Quest'indagine può infatti aiutare nella progettazione di un algoritmo in grado di

soddisfare al meglio le esigenze degli utenti.
La stessa operazione, può poi essere eseguita per quanto riguarda il verificarsi di **stalli**, ricercando dunque quali, fra le diverse variabili in campo (numero di stalli, tempo totale di stallo, frequenza degli stalli, ecc), gioca un ruolo maggiore nella degradazione della qualità percepita dagli utenti finali.

3.1 Transport Protocol

Il *video streaming*, ovvero quella metodica di accesso a contenuti audio/video non ancora del tutto scaricati in locale ma il cui download continua in background, si può distinguere essenzialmente in due tipologie:

- una riguarda la consegna di pacchetti *live* con un encoding *on-the-fly* come, ad esempio, *IPTV - Internet Protocol Television*, un protocollo per la distribuzione di contenuti televisivi su reti basate su IP;
- l'altra riguarda la consegna di pacchetti *pre-encoded* ed è nota con il nome di *VoD - Video on Demand* come, ad esempio, *YouTube*, il quale conta la visualizzazione di più di due miliardi di video al giorno [7].

Il trasporto degli stream video via Internet è attualmente realizzato sia mediante il protocollo **TCP - Transmission Control Protocol**, sia tramite **UDP - User Datagram Protocol**. A causa delle differenze implementative, l'utilizzo dei due protocolli ha impatti diversi sul comportamento dello streaming e, di conseguenza, sulla QoE percepita dall'utente finale:

- l'utilizzo di **TCP** garantisce la consegna di pacchetti inalterati in quanto è lo stesso protocollo ad occuparsi della ri-trasmissione dei pacchetti corrotti o persi. In più, esso adatta la sua velocità ai livelli di congestione di rete, il ché permette di minimizzare la perdita di pacchetti. Nel caso in cui la bandwidth sia minore rispetto alla bitrate richiesta per lo streaming, il video player stopperà la riproduzione del video, il ché risulterà in uno o più *stalli*. Se si sta utilizzando il protocollo TCP, dunque, a risultare disturbata sarà la **riproduzione del video**;

- nel caso in cui si utilizzi il protocollo **UDP**, invece, questo non esegue adattamenti in base alla bandwidth disponibile in modo da garantire la consegna dei pacchetti o la ricezione nell'ordine di trasmissione, ma si occupa soltanto di trasmettere i dati alla stessa bitrate usata dall'applicazione. Perciò, nel caso in cui si verifichino congestioni di rete, esse causeranno la perdita di pacchetti che porteranno ad *artefatti video* come sfarfallii dell'immagine o salti nello stream. Utilizzando UDP, dunque, a risultare degradata sarà la **qualità video**.

La domanda, dunque, riguarda quale tipologia di protocollo di livello trasporto usare dal punto di vista dell'utente finale, cioè, in base alla QoE.

Al fine di rispondere a questa domanda, gli autori di *QoE of YouTube Video Streaming for Current Internet Transport Protocol* [8] hanno definito uno scenario nel quale le capacità della rete sono limitate a causa di una **bottleneck** o **collo di bottiglia**. Perciò, se la *bandwidth* disponibile B è limitata rispetto alla *video bitrate* V richiesta per il download del contenuto multimediale, ovvero:

$$V > B$$

l'utente potrebbe sperimentare *stalli* o *degradazioni della qualità* durante la riproduzione, a seconda che sia usato rispettivamente il protocollo TCP od il protocollo UDP.

3.1.1 UDP e Packet Loss Ratio

Al fine di valutare la qualità percepita dagli utenti durante lo streaming di video *YouTube* utilizzando il **protocollo UDP**, gli autori di [8] si sono affidati ad una banca dati pubblicamente disponibile[1] che fornisce stream video codificati con *H.264*, lo stesso codec utilizzato da YouTube.

[1]EPFL-PoliMI: http://vqa.como.polimi.it

Per i test sono state analizzate 12 diverse sequenze video di cui una metà con risoluzione *CIF* (352x240 pixel) e l'altra metà con risoluzione *4CIF* (704x480 pixel). Per ognuno dei 12 stream orignali H.264, sono stati generati un determinato numero di stream corrotti eliminando pacchetti secondo pattern di errori prestabiliti. I pattern sono stati generati secondo 6 differenti *packet loss ratio R*: 0.1%, 0.4%, 1%, 3%, 5% e 10%. Inoltre, sono state usate due diverse tipologie di errore: errori random ed errori burst (una sequenza di simboli errati). Pertanto, in totale, sono state considerate ben 72 sequenze video codificate in CIF ed altre 72 codificate con metodo 4CIF, tutte afflitte da errori di packet loss, oltre alle 12 sequenze video originali (non corrotte). Gli stream CIF e 4CIF sono stati presentate agli utenti in due sequenze di test separate. Al termine di ogni sequenza, agli utenti era chiesto di valutare la qualità usando una scala *ACR - Absolute Category Rating Scale* [9] divisa in 5 punti: 1 - bad, 2 - poor, 3 - fair, 4 - good, 5 - excellent.

Figura 3.1 mostra il *MOS - Mean Opionion Score* [9], ovvero la media dei voti espressi dagli utenti su scala ACR, per **packet loss ratio**. In figura sono rappresentati i risultati per entrambe le risoluzioni: puntini neri indicano i valori per CIF, i puntini gialli per 4CIF.

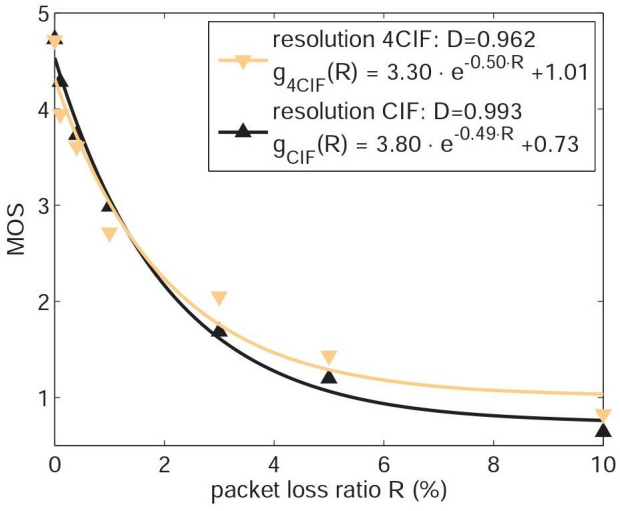

Figura 3.1: Influenza della packet loss ratio sul MOS [8]

Come è possibile vedere, il MOS decresce fortemente all'aumentare dei problemi di rete in termini di perdita di pacchetti. Inoltre, possiamo vedere come la packet loss influenzi fortemente la QoE percepita dagli utenti finali mentre la risoluzione del video (CIF vs. 4CIF) ha un impatto minimo. In ultima analisi, è possibile osservare come i valori trovati siano ben approssimati da **curve esponenziali** e come i *coefficienti di determinazione* D assumano valori molto vicini a quelli di match perfetto ($D = 1$).

3.1.2 TCP e Numero di Stalli

Per quanto riguarda invece la valutazione circa l'impatto che l'uso del **protocollo TCP** ha sulla QoE nella visione di video *YouTube*, gli autori hanno deciso di effettuare i test usando la tecninca del **crowdsourcing** utilizzando, in particolare, la piattaforma *Microworkers.com* [10].
Questa è una tecnica che permettere di sottomettere un task ad una moltitudine di persone provenienti, potenzialmente, da Paesi diversi, riducendo allo stesso tempo i costi ed i tempi per lo svolgimento dei test ed ottenendo risultati statisticamente significativi [11].
Ad ogni modo, questa metodologia può determinare l'inaffidabilità di alcuni risultati a causa dell'anonimato e della lontananza ("remoteness") dei partecipanti [12]. Alcuni soggetti, infatti, possono inviare risultati non corretti tentando di massimizzare i propri profitti cercando, ad esempio, di completare quanti più task possibili; altri invece potrebbero lavorare in modo scorretto a causa della mancanza di un supervisore.
Gli autori mostrano però come in [13] e [14], nonostante la qualità dei risultati sia un problema intrinseco della metodologia del crowdsourcing, essa possa comunque essere notevolmente migliorata grazie al filtraggio dei risultati considerati non attendibili, i quali possono essere identificati sottoponendo agli utenti test addizionali che includono, ad esempio, domande relative al contenuto video mostrato, domande relative alla coerenza e monitorando l'uso dell'applicazione.

Per quanto riguarda lo **scenario**, invece, sebbene la presenza di una bottleneck in una rete dove la bandwidth è costante lo renda relativamente semplice, i pattern legati all'occorrenza degli stalli potrebbero non esserlo a loro volta. Sono molteplici, infatti, i *fattori che interagiscono ed influenzano i pattern di stallo*. Alcuni di essi hanno a che fare con il controllo del flusso a livello applicazione implementato in YouTube [15], altri hanno a che fare con il controllo del flusso a livello trasporto implementato in TCP, alcuni riguardano le variazioni nella bitrate dovute al video encoding, altri all'implementazione del video player ed alla gestione del buffer.
In [16], ad esempio, sono state studiate le caratteristiche della rete derivanti dall'utilizzo dei servizi di video streaming *YouTube* e *Netflix*, mostrando come, in base all'applicazione utulizzata (es: browser, app mobile, ecc) ed al container (es: *Silverlight*, *Flash*, *HTML5*, ecc), possano essere osservate differenti strategie per lo streaming, con la conseguente generazione di pattern di traffico dati diversi fra loro.
È per questo motivo che gli autori hanno deciso di semplificare il modello di osservazione dei pattern degli stalli basandosi solamente sulle misurazioni effettuate a *livello applicazione*.

La campagna di test si è svolta da Luglio ad Agosto 2011, durante la quale sono stati richiesti più di 37.000 video, catturati circa 35 GB di traffico dati ed analizzati in dettaglio più di 1.000 video, fotogramma per fotogramma. Inoltre sono state scaricate da YouTube 266.245 descrizioni per ricavare la durata dei video. La configurazione usata per le misurazioni comprendeva:

- un *bandwidth shaper*: un software di emulazione di rete utilizzato per limitare la larghezza di banda sia in upload che in download. In particolare, lo specifico programma usato è stato *NetLimiter*;

- uno *YouTube user simulator*: un componente che simula un utente mentre guarda video YouTube nel suo browser. A tale scopo è stato configurato un web server *Apache* in locale e sono state generate dinamicamente pagine web dalle quali sono chiamate le *API YouTube* per incorporare e riprodurre i video. L'embedding dei video YouTube al-

l'interno di pagine web si è reso necessario al fine di monitorare la QoS a livello applicazione (si veda il prossimo punto);

- un *monitor per la QoS*: tramite l'uso del linguaggio *Javascript* dalle pagine generate dinamicamente, variabili come lo stato del video player ("playing", "buffering", "ended") e del buffer (in termini di byte caricati per la riproduzione del video) sono state monitorate.

Al termine delle simulazioni, le informazioni relative al monitoring degli stalli ed allo stato del buffer sono state scritte su file di log. Inoltre, sono stati catturati pacchetti usando *Wireshark* e *Tshark*. Come risultato, sono state ottenute informazioni circa la QoS sia a livello network (grzie al packet tracing effettuato con Wireshark) sia a livello applicazione (pattern di stallo).

Figura 3.1 mostra il *MOS* per **numero di stalli** N della durata di 1 e 3 secondi. Anche in questo caso è possibile notare come l'andamento delle curve che approssimano i risultati sia **esponenziale**, con valori per i *coefficienti di determinazione* R^2 molto vicini a quello di match perfetto ($R^2 = 1$).

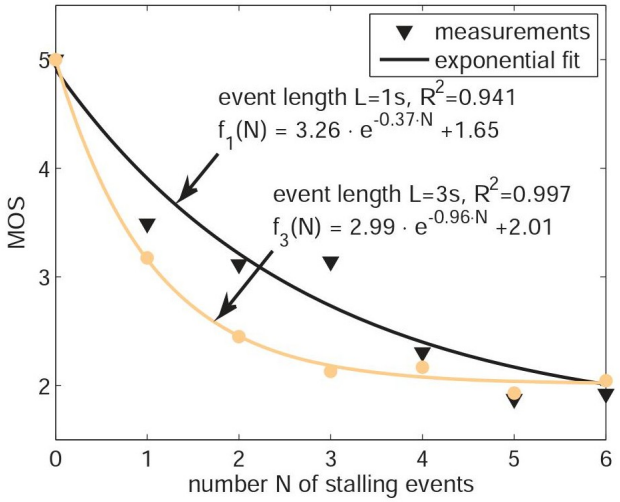

Figura 3.2: Influenza del numero di stalli sul MOS [8]

3.1.3 TCP e Frequenza degli Stalli

Il numero di stalli non è stata l'unica variabile studiata. Si è infatti anche cercato di capire quale fosse l'influenza della **frequenza degli stalli** F e a cosa fosse dovuta.
Per questo studio sono state usate due differenti limitazioni nella bandwidth disponibile B. La prima coincide con la banda tipica delle *celle telefoniche UMTS*: 384 Kbps, mentre la seconda è identificata dalla media delle bitrate osservate in altri studi condotti dagli autori (si veda [17]): 450 Kbps.
La frequenza alla quale occorrono gli stalli è definita come il rapporto fra il numero di stalli N e la durata complessiva del video D:

$$F = \frac{N}{D}$$

Dapprima è stata studiata la correlazione fra F ed altri fattori di influenza. Come *coefficiente di correlazione*, ovvero un indice che esprime un'eventuale relazione di linearità tra due variabili statistiche X ed Y, è stato calcolato il **Coefficiente di correlazione di Pearson** [18] [19]. Generalmente indicato con la lettera ρ_{XY}, è definito come la *covarianza* fra le due variabili X ed Y divisa per il prodotto delle loro *deviazioni standard*.
Assume sempre valori compresi tra -1 ed 1:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \qquad \text{con} \qquad -1 \leq \rho_{XY} \leq 1$$

dove:

- se $\rho_{XY} > 0$ le variabili X ed Y si dicono *direttamente correlate* o *correlate positivamente*;
- se $\rho_{XY} = 0$ le variabili X ed Y si dicono *incorrelate*;
- se $\rho_{XY} < 0$ le variabili X ed Y si dicono *inversamente correlate* o *correlate negativamente*.

Inoltre, per la correlazione diretta (così come, analogamente, per quella inversa):

- se $0 \leq \rho_{XY} < 0.3$ si ha *correlazione debole*;
- se $0.3 \leq \rho_{XY} < 0.7$ si ha *correlazione moderata*;
- se $0.7 \leq \rho_{XY} \leq 1$ si ha *correlazione forte*.

I valori ottenuti dagli autori sono stati i seguenti: frame rate: -0.03, durata del video: -0.35, numero di stalli: 0.47, valore medio della lunghezza degli stalli: -0.58, valore mediano della lunghezza degli stalli: 0.37, video bitrate: 0.87. Come è possibile vedere, la frequenza alla quale gli stalli occorrono è *fortemente correlata* solamente alla **video bit rate**.

Figura 3.3 mostra la *frequenza degli stalli* in base alla **richiesta video normalizzata** x, definita come:

$$x = \frac{V}{B}$$

Anche in questo caso, come è possibile vedere, la relazione è **esponenziale**, con un *coefficiente di determinazione* D pari a 0.943.

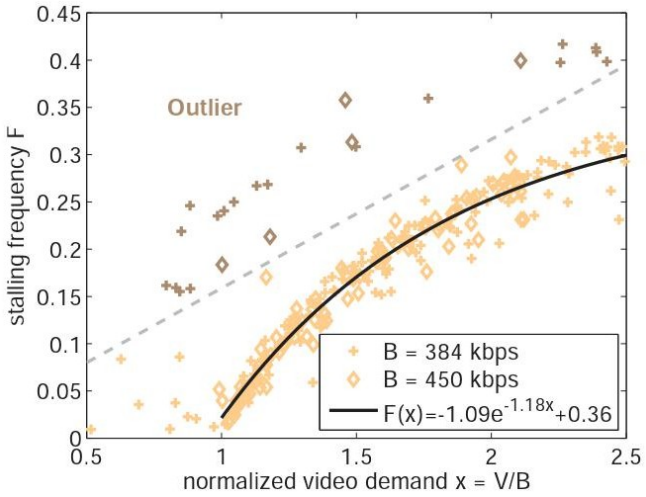

Figura 3.3: Influenza della richiesta video sulla frequenza degli stalli [8]

Ad ogni modo, circa un 15.22% dei video sono *outlier*. Gli autori non hanno trovato una correlazione fra questi valori ed altre variabili. Pensano comunque che essi siano dovuti all'implementazione del video player stesso.

3.1.4 TCP e Durata degli Stalli

Infine, l'ultima variabile studiata dagli autori è stata la **durata di ogni evento di stallo** L. Anche in questo caso essi hanno tentato di stabilire eventuali correlazioni di questa misura con altre variabili come la video frame rate, la bitrate, la frequenza alla quale gli stalli sono occorsi, ecc, ma non è stato ottenuto alcun risultato.

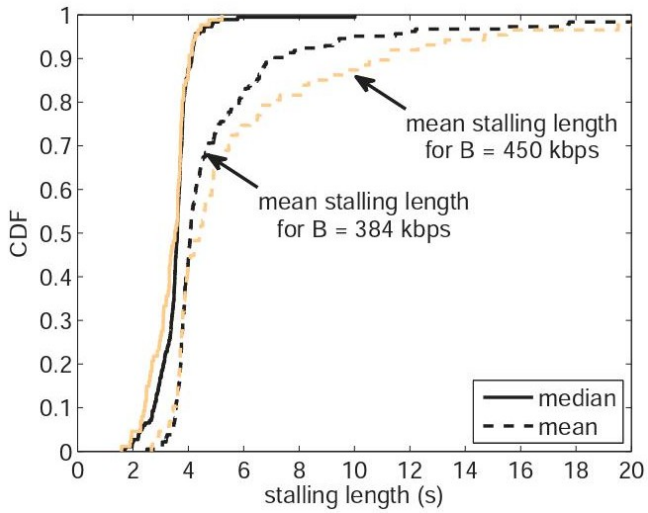

Figura 3.4: Distribuzione della durata media degli stalli [8]

Figura 3.4 mostra la **CDF - Cumulative Distribution Function** o **funzione di ripartizione** che racchiude le informazioni circa la distribuzione di questo fenomeno per entrambe le bandwidth B analizzate.
La media delle durate degli stalli indica come essi si aggirano fra i 2 secondi ed i 6 secondi, anche se ci sono alcuni video che mostrano una media molto maggiore. Analizzando questi risultati, si è scoperto come questi ultimi siano influenzati da un unico lungo evento di stallo e corrispondano agli outlier identificati in Figura 3.3.

In conclusione, come detto precedentemente, anche se lo scenario è semplice, possono essere tanti i fattori che influenzano i pattern relativi all'occorrenza degli stalli. Come abbiamo visto, infatti, in alcuni casi la frequenza e la durata di alcuni di loro può non essere in linea con la media generale. Così come gli autori credono, questo fatto può essere ricondotto a fattori che non hanno strettamente a che fare con lo scenario definito come, in questo caso, all'implementazione del video player stesso.

3.1.5 The IQX Hypothesis

Coma abbiamo visto precedentemente, sia per quanto riguarda la relazione che intercorre fra la *packet loss ratio* ed il *MOS* nel caso in cui il protocollo *UDP* sia usato (Sottosezione 3.1.1, Figura 3.1), sia per quanto riguarda la relazione che intercorre fra il *numero di stalli* ed il *MOS* nel caso in cui sia usato il protocollo *TCP* (Sottosezione 3.1.2, Figura 3.2), così come per quanto riguarda la *frequenza degli stalli* ed il *MOS* sempre nel caso di TCP (Sottosezione 3.1.3, Figura 3.3), abbiamo l'occorrenza di **dipendenze esponenziali**. Ricordando come il MOS sia un meccanismo con il quale gli utenti esprimono il grado di qualità percepita, si evince come questi fattori degradino la QoE in maniera esponenziale.
Questo fatto corrisponde a quanto già postulato in [20] con la cosiddetta **IQX hypothesis**, ovvero la *exponential **i**nterdependency of **q**uality of **exp**erience and quality of service*, uno studio nel quale gli autori hanno mostrato come la QoE e la QoS siano in relazione tra di loro ed, in particolare, come la QoE dipenda dall'occorrenza di un fattore di degradazione legato alla QoS.
Matematicamente questa relazione può essere espressa dalla seguente *equazione differenziale*:

$$\frac{\partial QoE}{\partial QoS} = -\beta(QoE - \gamma)$$

che può essere risolta come una **relazione esponenziale** fra la Quality of Experience e la Quality of Service.

Nel nostro specifico caso, essendo la *packet loss ratio* un fattore di degradazione relativo all'uso del protocollo UDP ed il verificarsi di *stalli* un fattore di degradazione relativo all'utilizzo del protocollo TCP, la QoE in termini di MOS è guidata proprio dalla funzione esponenziale appena definita.

La qualità dell'approssimazione tra i valori misurati e l'istanza della specifica funzione esponenziale utilizzata è rappresentata dal **coefficiente di determinazione** (D in un caso, R^2 nell'altro), i cui valori si trovano all'interno dell'intervallo $[0, 1]$ dove 0 rappresente il peggior match possibile mentre 1 rappresenta la miglior approssimazione possibile.

3.1.6 UDP vs. TCP

Come abbiamo visto in questa sezione, nel caso in cui vi sia un collo di bottiglia nella rete che limita la bandwidth B ad un valore costante e si tenti di visualizzare video YouTube ad una bitrate V dove $V > B$, questo scenario porterà l'utente a scontrarsi con problematiche quali la degradazione della qualità del video riprodotto, sfarfallii e salti nello stream nel caso in cui sia utilizzato UDP come protocollo a livello trasporto. L'utente sperimenterà stalli durante la riproduzione se invece sarà usato TCP come protocollo per

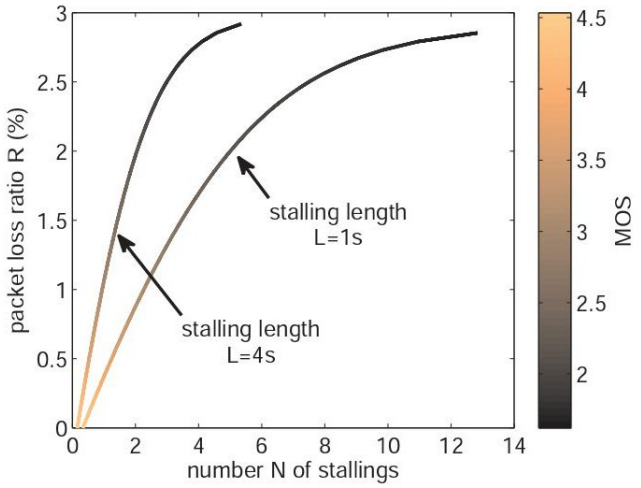

Figura 3.5: Influenza di UDP vs. influenza di TCP sul MOS [8]

il trasporto dei pacchetti via Internet.
Siccome alla base di questo scenario abbiamo il verificarsi delle stesse problematiche a livello di QoS (anche se gli effetti sulla QoE sono nettamente diversi), è possibile determinare quale pattern di occorrenza degli stalli è stato causato da quale packet loss rate siccome, appunto, l'utente finale sperimenterà la stessa QoE.

Figura 3.5 mostra il numero di stalli N sull'asse x ed il corrispondente packet loss ratio R sull'asse delle y, legati dallo stesso valore di MOS (indicato dal colore). Figura 3.6 effettua un confronto fra UDP e TCP usando il valore di throughput normalizzato ρ calcolato come il rapporto fra la bandwidth B e la video bitrate V:

$$\rho = \frac{B}{V}$$

I risultati indicano chiaramente come l'utilizzo del **protocollo TCP** da parte della piattaforma YouTube per il video streaming permetta agli utenti di sperimentare una QoE migliore rispetto all'utilizzo di UDP.
Utilizzando TCP, in definitiva, i problemi di cui la rete può soffrire, qual-

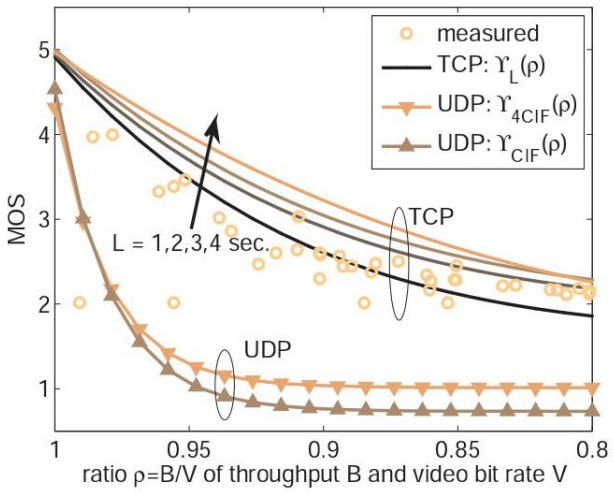

Figura 3.6: Influenza di UDP vs. influenza di TCP sul MOS [8]

siasi essi siano, sono percepiti dall'utente in un unico modo: il verificarsi di un'**interruzione del servizio** durante la fruizione dello stesso.
Questa definizione di QoE richiede inevitabilmente uno studio approfondito da parte dei provider di rete e dei servizi, in modo da comprendere quali siano i requisiti di qualità che secondo gli utenti devono essere soddisfatti.

3.1.7 TCP e Stalli in YouTube

Appurato come utilizzando il *protocollo TCP* per il trasporto dei pacchetti audio/video su Internet nel caso in cui si verifichino problemi di rete (quali bottleneck) la riproduzione dei video sia interrotta da interruzioni quali *stalli*, vedremo ora perché essi si manifestano come influenzano la *gestione del buffer* del *video player* utilizzato dalla piattaforma di video streaming **YouTube**.

Passive YouTube QoE Monitoring for ISPs [21] mostra alcuni approcci usati nel **monitornig della QoE**:

- gli autori di [22] hanno presentato un software client-side per il monitornig del traffico YouTube a livello applicazione, applicato con successo nell'ottimizzazione di reti wireless mesh in modo che, al verificarsi di stalli, la rete fornisca maggiori risorse [23] [24];

- in modo simile, l'approccio *FoG - Forwarding on Gates* [25] è usato in [26] per sviluppare un nuovo stack di rete dinamico basato su blocchi funzionali per ottimizzare la QoE nella riproduzione di video YouTube.

[21], in particolare, si concentra sulla definizione di 3 diversi modelli utilizzabili direttamente da parte degli **ISPs - Internet Service Providers** per monitorare la QoE relativa alla visione di video sulla piattaforma di video streaming YouTube.
Il primo, *M1*, permette di misurare il tempo totale di stallo T a partire dal tempo totale necessario per il download del video Y e la sua durata D:

$$T = max(Y - D, 0)$$

Il secondo modello, *M2*, permette invece di calcolare la frequenza alla quale gli stalli occorrono. Prima di tutto è necessario stimare la capacità del collo di bottiglia B come già mostrato in [27] e [28]. Dopodiché è necessario estrarre la bitrate V dai pacchetti parsando i metadati disponibili nel formato container del file. Può così essere calcolata la richiesta video normalizzata x che permette di stimare il numero di eventi di stallo N dove, in questo caso, Y rappresenta il tempo effettivo per il download e non il tempo totale:

$$N = min(D, Y)F(x) \qquad \text{con} \qquad x = \frac{V}{B}$$

È però il terzo modello, *M3* quello che apporta il maggior contributo circa lo studio sulle cause e le modalità di occorrenza degli stalli in YouTube. Esso si occupa infatti di stimare lo stato del *buffer* del video player partendo da misurazioni a livello network. Come abbiamo già ripetuto più volte, anche precedentemente, infatti, quella del video streaming è una tecnica che permette l'accesso ad un contenuto audio/video senza attenderne il download completo, operazione che sta contemporaneamente avvenendo in background. Nel buffer si trovano i dati già scaricati che attendono di essere riprodotti. L'idea di fondo sfruttata in questo modello è quella di confrontare i tempi di riproduzione dei frame video con i *timestamp* dei pacchetti ricevuti. Tralasceremo i dettagli implementativi del modello in quanto trascendono lo scopo di questa Tesi, per concentrarci solo su un preciso risultato ottenuto dagli autori. Essi hanno infatti scoperto le regole che stanno alla base della **gestione del buffer video** del **player YouTube** e, di conseguenza, le regole che determinano l'occorrenza degli stalli, così come mostrato in Figura 3.7:

- Θ_1 definisce il quantitativo temporale minimo che il buffer deve contenere affinché il video continui la riproduzione una volta avviata. Nel caso in cui si scenda sotto questo threshold, il video stallerà;

- Θ_0 definisce il quantitativo temporale minimo contenuto nel buffer che deve essere superato affinché la riproduzione del video sia ri-avviata.

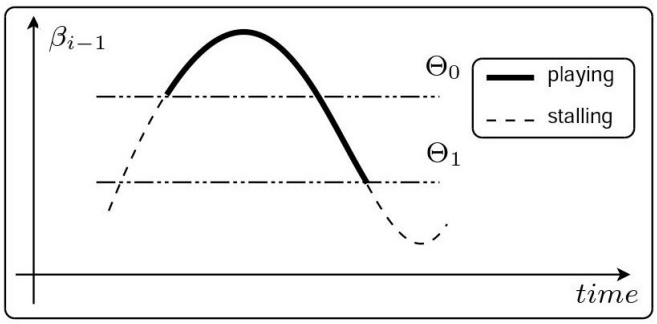

Figura 3.7: Gestione del buffer del player YouTube [21]

Considerando dunque la dimensione del contenuto temporale del buffer β_i all'istante i-esimo, abbiamo che se β_{i-1} scende sotto Θ_1 il video entra in stallo, mentre se β_{i-1} eccede Θ_0 il video riprende la riproduzione. Dunque, uno stallo ha luogo se la seguente condizione è verificata:

$$(\psi_{i-1} \wedge (\beta_{i-1} < \Theta_0)) \vee (\neg\psi_{i-1} \wedge (\beta_{i-1} < \Theta_1))$$

dove ψ_i è una variabile booleana[2] che indica se il video è $(\psi_i = 0)$ o meno $(\psi_i = 1)$ in riproduzione all'istante i.

Gli esperimenti condotti dagli autori (sempre in [21]) in un laboratorio al FTW di Vienna da Giugno ad Agosto 2011, hanno poi permesso di mettere in **relazione** la presenza di **cattive condizioni di rete** con il graduale **svuotamento del buffer** del video player ed il relativo verificarsi di **stalli** secondo la regola precedentemente determinata.
L'ambiente di simulazione è stato ottenuto emulando particolari condizioni di rete grazie all'uso di software già esistenti come *Dummynet* [29], al fine di monitorare specifiche variabili quali la perdita di pacchetti, il verificarsi di ritardi ed il throughput, sia in upload che in download. Come già fatto anche in altri studi, è stato poi implementato uno script in grado di simulare un utente che guarda video YouTube sul suo browser. È stato configurato un web server locale e le pagine web sono state generate dinamicamente in

[2]variabile che può assumere solo due valori: true e false, on e off, 0 e 1, ecc...

modo che chiamassero delle API YouTube per l'embedding e la riproduzione del video. Per il monitoring circa lo stato del player e le condizioni del buffer è stato usato il linguaggio Javascript direttamente dalle pagine web. Alla fine della simulazione le informazioni sugli stalli e sullo stato del buffer sono state scritte su file di log per l'analisi.
I risultati ottenuti esaminando i tempi di riproduzione dei frame video ed i timestamp dei pacchetti ricevuti mostrano come i pattern degli stalli bene approssimino le regole definite nel modello *M3* definito dagli autori.

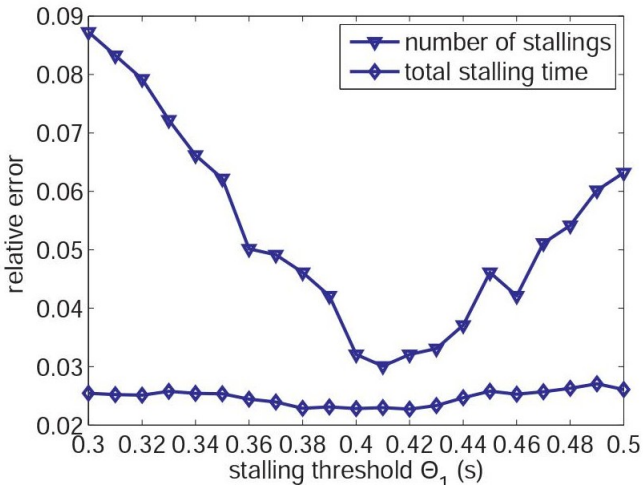

Figura 3.8: Stima del valore Θ_1 del buffer del player YouTube [21]

Un altro risultato importante per questa Tesi di Laurea è rappresentato dalla **stima approssimativa circa i valori di** Θ_0 **e** Θ_1 grazie al monitoring dei parametri relativi alla QoS sia a livello applicazione che a livello network. Figura 3.8 mostra gli errori relativi che intercorrono fra i pattern approssimati e quelli misurati a livello applicazione, dipendenti dalla variabile di threshold Θ_1. Così come Figura 3.8 giustifica l'elezione del valore Θ_1, il valore Θ_0 è stato ottenuto allo stesso modo. Da questa valutazione sono stati trovati i seguenti risultati:

$$\Theta_0 = 2.2s \qquad \text{e} \qquad \Theta_1 = 0.4s$$

Basandosi sui valori di threshold ottenuti, è poi stato possibile ricostruire i pattern relativi agli stalli. Figura 3.9 mostra la dimensione stimata del buffer video allo scorrere del tempo. È possibile osservare come il video inizi la riproduzione non appena il valore Θ_0 sia superato (linee blu). Quando invece il valore del buffer scende sotto il valore Θ_1 il video entra in stallo (linee rosse). Si può inoltre osservare come gli stalli stimati (linee rosse) e quelli reali (barre nere) si sovrappongano con grande precisione.
Il relativo *coefficiente di determinazione* R^2 assume infatti un valore quasi uguale a quello di match perfetto ($R^2 = 1$):

$$R^2 = 0.9996$$

Questo significa che la stima fornita dal modello *M3* risulta molto accurata e che i valori Θ_0 e Θ_1 trovati dagli autori approssimano molto bene quelli realmente usati dalla piattaforma.

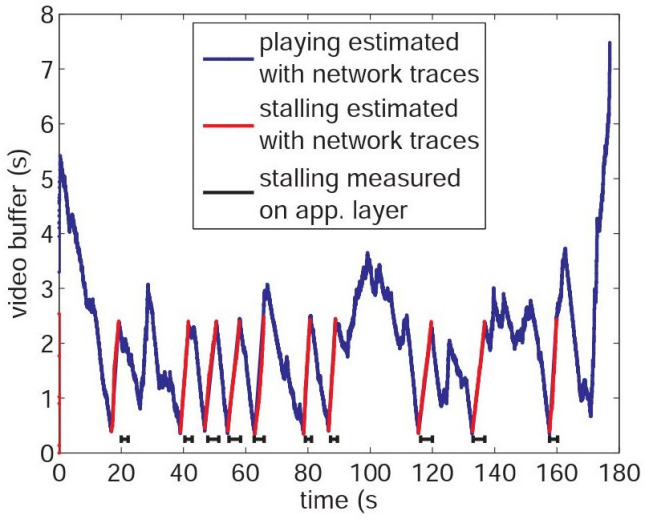

Figura 3.9: Stima del buffer del player YouTube (inferenza degli stalli) [21]

3.2 Effetti di Degradazione Temporale

Come abbiamo visto nella sezione precedente, da un punto di vista dell'utente e quindi della QoE da lui sperimentata, l'utilizzo del *protocollo TCP* a livello trasporto è da preferire rispetto all'utilizzo dei *datagram UDP*.
Questo però fa sì che, indipendentemente da quelle che siano le problematiche che affliggono la rete in un dato momento (es: il verificarsi di un *collo di bottiglia*), l'utente sperimenti solamente delle *interruzioni del servizio*.
Ma quali sono, nel dettaglio, i **fattori di degradazione temporale** che possono verificarsi peggiorando la percezione della QoE?

Molto spesso le percezioni qualitative che gli utenti finali hanno nell'utilizzo delle applicazioni Internet sono caratterizzate da tempi di attesa che intercorrono **prima** del consumo del servizio e da interruzioni **durante** la fruizione dello stesso. Questi effetti temporali diventano sempre più importanti quanto più l'applicazione che si sta utilizzando è *time-sensitive*, ovvero sensibile a quelle che vengono definite *degradazioni temporali*.
L'*audio/video straming*, infatti, è un ambito nel quale il fattore tempo gioca un ruolo fondamentale nell'erogazione del servizio, con forti ricadute sulla QoE sperimentata dagli utenti finali:

- una latenza elevata può essere dovuta a problemi di rete;
- un basso bitrate può essere dovuto ad una insufficienza di risorse;
- l'autenticazione da parte dell'utente o il setup della connessione Internet possono risultare in operazioni che richiedono tempo per essere compiute.

Se uniamo questi elementi al fatto che il servizio viene dispensato tramite l'uso di *reti mobili*, dove le variazioni nel data rate possono essere importanti, così come ricadute negative a causa del verificarsi di *effetti di shadowing* dovuti alla natura intrinseca che caratterizza il tipo di rete in uso (mobilità degli utenti), quello che ne deriva è uno scenario fortemente time-sensitive.

Da un lato i provider di rete e del serivzio si trovano a dover fare i conti tra la necessita di dover investire nel miglioramento della QoE e dei vincoli economici in modo che essi risultino vantaggiosi, dall'altro, se i tempi di attesa sono inevitabili, devono implementare strategie di gestione della QoE in modo che il servizio di distribuzione sia ottimale.
In quest'ottica l'obiettivo della ricerca è quello di rapporatre quantitativamente la QoE con i tempi di attesa che interrompono la fruizione del servizio, siano essi **ritardi iniziali** o **pause intermedie** (**stalli**).

3.2.1 Ritardi Iniziali

Per quanto riguarda l'investagazione circa la relazione che intercorre fra la QoE sperimentata dagli utenti ed il verificarsi di **ritardi iniziali** che occorrono prima dell'utilizzo di un servizio, sono state condotte diverse ricerche.

[30] e [31] trattano di un esperimento circa l'impatto che un ritardo iniziale può avere sulla QoE in ambito di *social networking* dove, tale ritardo, è causato da operazioni relative all'*autenticazione*. L'esperimento si è svolto usando un laptop nel quale è stato installato un web browser. La pagina relativa al social network in esame usa un server remoto *OpenID* per le operazioni di autenticazione il quale completa l'operazione in tempi predeterminati. Una volta terminato l'esperimento, agli utenti viene chiesto di dare una valutazione al servizio tenendo conto dei tempi di risposta usando una scala continua da 0 a 100, poi mappata su una *scala ACR* di 5 punti [9].

[32] e [33], invece, mettono in relazione la QoE esperita dagli utenti in base ai ritardi iniziali dovuti al *setup della connessione Internet 3G*. Anche in questo caso gli utenti che hanno partecipato al test sono stati fatti sedere di fronte ad un laptop sul quale era stato installato un emulatore di rete customizzato in modo che il tempo che intercorresse fra la pressione del bottone per la connessione e la connessione stessa seguisse pattern prestabiliti. Nuovamente, una volta completato il task agli utenti era chiesto di esprimere il proprio grado di soddisfazione usando la scala ACR.

Tipo test	Numero partecipanti	Durata video (s)	Tempi di attesa (s)
lab	36	30	0, 1, 8, 16
		60	1, 8, 16
crowd	32	30	0, 1, 4, 8, 16, 32
	40	60	0, 0.5, 1, 2, 4, 8

Tabella 3.1: Dati del test video YouTube sui ritardi iniziali [34]

Un'ultima ricerca, [34], si è invece occupata di investigare circa le relazioni che intercorrono fra la QoE ed il verificarsi di ritardi iniziali nella *visualizzazione di video YouTube* dovuti ad un primo parziale riempimento del buffer del video player utilizzato. L'esperimento ha raccolto i dati provenienti da due fonti: sia tramite *test in laboratorio*, così come fatto nelle ricerche precedenti, sia tramite uno studio di *crowdsourcing*. I dati utilizzati per queste prove sono riportati in Tabella 3.1.
Per quanto riguarda la parte di *laboratorio*, l'esperimento ha avuto una durata complessiva di 1 ora e 30', dove la parte di testing effettiva legata alla QoE è stata di circa 1 ora. Il test è stato preceduto da una parte di briefing comprensiva e seguito da una parte di debriefing finale, nella quale è stata svolta un'intervista agli utenti ed è stato somministrato loro un questionario demografico. In tale questionario gli utenti dovevano indicare il proprio background, le esperienze tecniche, la condizione attuale, la sensazione di fatica ed il carico cognitivo. Il test si è composto della visualizzazione di brevi clip video della durata variabile fra i 30 ed i 60 secondi. L'argomento dei video poteva variare fra: trailer d'azione, musica, animazioni, documentari e news. Alla fine di ogni clip video agli utenti era chiesto di esprimere una votazione in una scala ACR di 5 punti sulla qualità generale, inclusa la qualità video e le performance di loading.
Per quanto riguarda invece il test effettuato tramite *crowdsourcing*, la metodologia è stata la stessa di quella già descritta nella Sottosezione 3.1.2. Come piattaforma è stata usata *Microworkers.com* e per far sì che l'utente

sperimentasse le condizioni volute dal test, i video venivano completamente scaricati in locale prima di avviarne la riproduzione ed i ritardi erano simulati sfruttando le *API YouTube* ed il *linguaggio Javascript*. Come per i test in laboratorio, alla fine della visione di ogni video, era somministrato agli utenti un questionario che permettesse loro di esprimere il grado di soddisfazione generale e, in aggiunta, domande per verificare la loro coerenza, al fine di individuare i test non affidabili, così come già fatto in precedenza [14]. All'utente era infine chiesto di esprimere il punteggio usando la solita scala ACR su 5 punti.

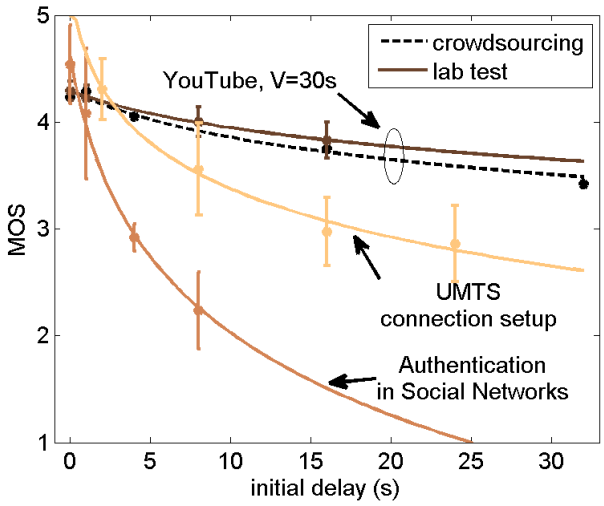

Figura 3.10: Influenza dei ritardi iniziali sul MOS [34]

I risultati delle tre ricerche, raccolti sempre in [34], sono raffigurati in Figura 3.10 dove sono messi in relazione i tempi di attesa iniziali ed il *MOS*, ovvero il valore medio di soddisfazione [9]. Emergono 3 importanti risultati.

Il primo riguarda il fatto che **le 3 curve divergono fortemente** le une dalle altre. Questo significa che i ritardi iniziali sono percepiti in maniera significativamente differente dagli utenti in base alla tipologia del servizio in uso. A parità di tempo atteso, infatti, la QoE risulta essere minore se si sta attendendo l'autenticazione in un social network rispetto al setup della connessione Internet, così come essa sembra rimanere su livelli più accettabili

nel caso in cui si stia attendendo la riproduzione di un video YouTube.
Questa situazione è spiegata dagli autori della ricerca in base al fatto che gli utenti imparano dall'interazione quotidiana con le applicazioni quali siano le tempistiche necessarie al fine di completare ogni operazione. Essi ipotizzano inoltre che la durata della fruizione del servizio influenzi l'esperienza. Ad esempio, all'autenticazione in un social network od al setup della connessione 3G sono associati task di più breve durata rispetto alla visione di un video, il ché spiegherebbe come al verificarsi di un dato tempo di attesa t, nei primi due casi la qualità percepita dagli utenti risulti essere minore rispetto alla qualità associata al task di più lunga durata (la visione di un video).

Il secondo risultato, e forse il più importante, è che tutte e 3 le curve sono approssimate da una **funione logaritmica** in accordo con la cosiddetta **WQL hypothesis** [32]. Essa è basata sulla legge fondamentale della psicologia di *Weber-Fechner* applicata ai ritardi iniziali ed assume che la relazione che intercorre fra i ritardi iniziali (**w**aiting time) e la loro valutazione in termini di **Q**oE usando una scala ACR lineare sia **l**ogaritmica. È infatti possibile vedere come una funzione logaritmica bene approssimi le curve di Figura 3.10. A tal proposito gli autori hanno usato una funzione del tipo:

$$f(T_0) = -a\ log(T_0 + b) + 5$$

La funzione $f(T_0)$ è la funzione di mapping fra i tempi di attesa iniziali T_0 ed i valori di MOS. I parametri a e b sono determinati risolvendo un problema non lineare di minimizzazione dei minimi quadrati fra i valori di MOS al tempo T_0 ed il valore della funzione $f(T_0)$. La qualità dell'approssimazione è rappresentata dal *coefficiente di determinazione* D i cui valori si trovano all'interno dell'intervallo $[0, 1]$ dove 0 rappresente il peggior match possibile mentre 1 rappresenta la miglior approssimazione possibile.
In Tabella 3.2 sono riportati i valori delle funzioni logaritmiche usate dagli autori per il mapping ed il relativo coefficiente di determinazione D. Come è possibile vedere, D assume valori molto vicini a quelli di match perfetto, il

Servizio	Funzione di mapping $f(T_0)$	Goodness-of-fit (D)
Social network	$-2.816\ log(T_0 + 1.378) + 5$	0.9925
3G setup	$-1.557\ log(T_0 + 0.742) + 5$	0.9889
YouTube (lab)	$-0.862\ log(T_0 + 6.718) + 5$	0.9983
YouTube (crowd)	$-0.963\ log(T_0 + 5.381) + 5$	0.9619

Tabella 3.2: Valori parametri a e b di $f(T_0)$ e di D [34]

ché indica come la WQL hypothesis non possa essere rigettata.

Il terzo ed ultimo risultato, marginale rispetto agli altri due in ottica di influenza circa i ritardi iniziali sulla QoE ma importante per quanto riguarda la modalità nella quale possono essere effettuati i test, riguarda i risultati che emergono da un confronto fra i dati ottenuti in laboratorio e quelli ottenuti tramite crowdsourcing. Come è possibile vedere, infatti, **la modalità usata per la raccolta dei dati non influenza il MOS** (linea tratteggiata per le prove in laboratorio e linea continua per il crowdsourcing). Anzi, entrambi i risultati si trovano all'interno delle barre di errore, le quali rappresentano il 95% del valore degli *intervalli di confidenza*.

Gli autori sottolineano poi come, per questioni di leggibilità, non siano stati riportati i valori degli esperimenti che riguardano la visione dei video di 60 secondi in quanto sovrapponibili a quelli per il video di 30 secondi, così come gli intervalli di confidenza dei dati raccolti tramide crowdsourcing, in quanto anch'essi del tutti identici a quelli per i test in laboratorio.

In conclusione, gli autori di [34], dopo aver raccolto i dati relativi alla QoE in relazione al verificarsi di ritardi iniziali pubblicati da altre ricerche sia in ambito di servizi di autenticazione su social network [30] [31] sia di setup di una connesione 3G [32] [33] e dopo aver integrato tali dati conducendo un proprio esperimento in ambito di video streaming, hanno mostrato come, nonostante i tempi di attesa iniziali siano gli stessi, la QoE esperita dagli utenti vari in base alla tipologia del servizio in uso e come tale variazione sia descritta da una specifica **legge logaritmica** definita in ambito psicologico.

3.2.2 Pause Intermedie

Dopo un primo significativo apporto circa la QoE sperimentata dagli utenti in relazione al verificarsi di ritardi iniziali, [34] ci offre un secondo contributo fondamentale: l'analisi di quanto un'altra forma di degradazione temporale influisce sulla QoE, ovvero le interruzioni del servizio dovute a **pause intermedie**, anche definite **stalli**.

Per quanto riguarda questi test, sono state usate le modalità già sperimentate precedentemente, ovvero lo svolgimento di una prova in laboratorio e la raccolta di dati tramite crowdsourcing.
In entrambi i casi sono stati fatti vedere agli utenti due video, uno dove era presente un ritardo iniziale di x secondi ed uno dove era presente uno stallo della stessa durata. Alla fine era chiesto loro di esprimere una valutazione su entrambi. Per quanto riguarda il test in laboratorio sono state usate clip da 30 secondi mentre, per il crowdsourcing, sono stati usati video di 60 secondi. In Tabella 3.3 sono riportati i dati utilizzati per i due test ed anche i dati di un esperimento precedentemente realizzato dagli autori, effettuato sempre tramite crowdsourcing [14].

In Figura 3.11 sono mostrati i risultati dell'esperimento. Le barre riportano sempre il 95% del valore degli intervalli di confidenza. Come si nota dalle due linee colorate, la *QWL hypothesis* che suggerisce una *dipendenza logaritmica* fra i tempi di attesa ed il MOS, non vale più nel caso degli stalli. Come mostrato in Tabella 3.4, infatti, il *coefficiente di determinazione D* nel caso di funzioni logaritmiche assume valori minori rispetto all'utilizzo di **funzioni esponenziali**.

Tipo test	Numero partecipanti	Durata video (s)	Tempi di attesa (s)
lab	36	30	
crowd	48	60	0, 0.5, 1, 2, 4, 8
crowd [14]	44	30	

Tabella 3.3: Dati del test video YouTube sugli stalli [34]

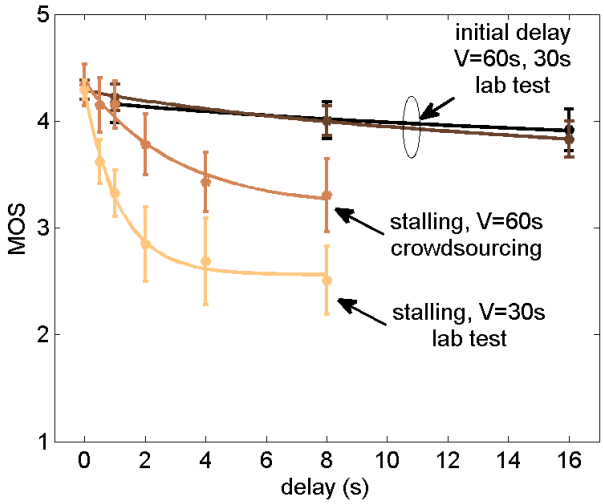

Figura 3.11: Influenza degli stalli sul MOS [34]

Questo corrisponde a quanto già postulato in [20] con la cosiddetta **IQX hypothesis** della quale ci siamo già occupati nella Sottosezione 3.1.5.
Secondo questa legge, la formula che meglio approssima la relazione fra i tempi di attesa sperimentati dagli utenti sottoforma di stalli ed il MOS è una funzione nella forma:

$$f(T_1) = \alpha \cdot e^{-\beta \cdot T_1} + \gamma$$

Lunghezza video (s)	Funzione di mapping $f(T_1)$	Goodness-of-fit (D)
30	logaritmica	0.817
60		0.945
30	$1.710\ e^{-0.860\ T_1} + 2.561$	0.9939
60	$1.175\ e^{-0.334\ T_1} + 3.190$	0.9726

Tabella 3.4: Valori parametri α, β e γ di $f(T_1)$ e di D [34]

3.2.3 Ritardi Iniziali vs. Pause Intermedie

Osservando bene Figura 3.11, possiamo renderci conto di come la QoE percepita dagli utenti sia influenzata in maniera significativamente diversa al verificarsi di ritardi iniziali o stalli.
Innanzitutto si nota come non vi sia una differenza statistica rilevante fra le curve che mettono in relazione i *tempi di attesa iniziali* ed il *MOS* in base alla durata dei video, fatto non altrettanto vero nel caso in cui si verifichino *stalli*: in questo caso, la curva relativa ai video di lunghezza pari a 30 secondi non è sovrapponibile a quella per le clip di 60 secondi. Nel caso in cui si verifichi uno stallo della durata di 8 secondi, ad esempio, se ciò avviene durante la visione di un video di 30 secondi il MOS relativo è di 2.51 mentre, per video di 60 secondi è significamente più elevato: 3.30. Ciò indica che la QoE percepita dagli utenti è peggiore se lo stallo si verifica durante la visione di un video di più breve durata.
Oltre a questo, come già visto precedentemente, rimane il fatto che la relazione che meglio approssima il mapping fra tempi di attesa ed il MOS nel caso di ritardi iniziali è *logaritmica*, mentre per gli stalli è *esponenziale*.
Questi due fatti hanno in realtà un denominatore comune: possono infatti essere spiegati tramite lo stesso concetto, ovvero il cosiddetto **Recency Effect**. Si veda Sottosezione 3.2.4 per maggiori dettagli.

In ogni caso, essendo i valori di MOS più bassi per quanto riguarda gli stalli rispetto ai tempi di attesa iniziali (se confrontati su tempi di attesa della stessa durata), questo fatto ci da un'importante indicazione sul come, in fase di progettazione di un eventuale servizio di video straeming, questi ultimi debbano essere preferiti rispetto ai primi.

Al fine di eliminare ogni dubbio circa la preferenza degli utenti, gli autori di [34] hanno condotto un ulteriore studio.
Nel primo test sono state fatte vedere a degli utenti in laboratorio due coppie di video YouTube (con la tecnica già precedentemente usata) con contenuti differenti che etichetteremo con A e B della durata di 30 s. Per ogni coppia è stato fatto vedere un video nel quale era presente un ritardo iniziale di x

secondi ed uno nel quale era presente uno stallo della stessa durata. Una volta visualizzata la coppia di video, all'utente veniva chiesto di esprimere un'opinione circa la propria preferenza su quale dei due video avesse preferito in base alla tipologia di degradazione temporale.
Il secondo test si è svolto chiedendo semplicemente agli utenti di due test effettuati tramite crowdsourcing, che etichetteremo con *I* e *II*, se preferissero vedere video dove era presente un ritardo iniziale di una determinata durata od un video dove era presente uno stallo della stessa lunghezza.

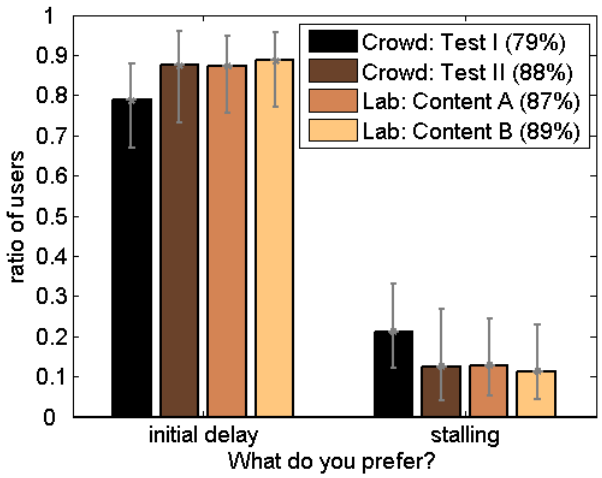

Figura 3.12: Ritardi iniziali vs.Stalli: preferenza degli utenti [34]

Come mostrato in Figura 3.12, la percentuale di utenti che **preferisce sperimentare un ritardo iniziale** prima della visione di un video YouTube rispetto ad uno stallo della stessa durata è quasi il *90%*. Inoltre, non vi è differenza fra i test svolti in maniera rigorosa in laboratorio ed i dati raccolti chiedendo semplicemente agli utenti di esprimere una preferenza tramite crowdsourcing. Resta comunque da spiegare a cosa possa essere dovuta la preferenza di un esiguo *10%* di utenti verso gli stalli. Gli autori suppongono che ciò sia dovuto al fatto che un ritardo iniziale nell'avvio del video elimini la possibilità di ricevere un feedback circa il funzionamento o meno del servizio.

3.2.4 Recency Effect

Un importante fattore usato per spiegare come gli stalli degradino maggiormente la QoE rispetto a quanto non facciano i ritardi iniziali arriva direttamente dalla psicologia. Secondo il **Serial Position Effect** [35] una persona tende a ricordare meglio i primi e gli ultimi elementi di una lista rispetto a quelli intermedi. Questo termine fu coniato da *Hermann Ebbinghaus*[3] attraverso una serie di studi che effettuò su sé stesso e si riferisce alla scoperta di come la precisione dei ricordi vari in funzione della *posizione* degli elementi all'interno di una lista di studio [36].
Si è scoperto come, quando alle persone viene chiesto di ricordare una lista di elementi in un qualsiasi ordine, esse tendono a partire dalla fine della lista in quanto ricordano meglio gli ultimi elementi. È il cosiddetto **Recency Effect**, il quale trova spiegazione nella *working memory* o *memoria di lavoro*, un modello introdotto per descrivere con più accuratezza le dinamiche della *memoria a breve termine*. Si è poi visto che tra gli elementi precedenti, quelli ricordati più frequentemente sono quelli che compaiono all'inizio della lista. In questo caso si parla di **Primary Effect**, un meccanismo relativo alla *memoria a lungo termine* [37] [38].
Questi effetti temporali sulle percezioni umane sembrano giocare un ruolo importante per quanto riguarda l'occorrenza delle diverse tipologie di effetti di degradazione temporale ed il loro timing. In accordo con il principio del recency effect, infatti, si è visto come la qualità generale del servizio che l'utente sperimenta è fortemente influenzata dal verificarsi di un singolo evento negativo che occorre verso la fine della fruizione del servizio piuttosto che verso l'inizio. Siccome questo effetto non può essere atteso per i ritardi iniziali in quanto non chiaramente percepiti come disturbi dagli utenti, esso assume un peso maggiore per quanto riguarda il verificarsi di **stalli**, soprattutto verso la fine. Inoltre, più il video che si sta guardando è lungo e minore sarà l'impatto dello stallo dovuto al recency effect.

[3] 24 Gennaio 1850 - 26 Febbraio 1909, fu uno psicologo tedesco pioniere dello studio sperimentale sulla memoria

3.3 Pattern di Stallo

Come abbiamo visto nella sezione precedente, la ricerca *Initial Delay vs. Interruptions: Between The Devil and The Deep Blue Sea* [34] ben riassume come, in caso di cattive condizioni di rete, fra il verificarsi dei possibili effetti di degradazione temporale che influenzano in maniera negativa la QoE, gli utenti preferiscono i *ritardi iniziali* alle *pause intermedie* che, inevitabilmente, determinano un'interruzione del servizio. Qualora ci si trovi nell'ambito del *video streaming*, infatti, questi risultando in fastidiosi **stalli**.
Sebbene questo risultato fornisca un'importante linea guida per i progettisti di applicazioni e servizi di video streaming in quanto suggerisce loro di sfruttare un eventuale ritardo iniziale per compiere, ad esempio, operazioni di *pre-buffering* al fine di evitare il verificarsi di stalli una volta avviata la riproduzione del video, è altrettanto vero che, in caso di condizioni non favorevoli (es: bottleneck, scarsa bandwidth, situazioni intrinsecamente legate all'ambito della *mobilità*, ecc), non sempre è possibile evitarne la comparsa.

Lo scopo di questa sezione è quello di approfondire nel dettglio quali sono i **fattori chiave di influenza sulla QoE** relativi ai **pattern di stallo** ed, in particolare, quanto il *tempo complessivo di stallo* T, il *numero di stalli* N e la *durata di ogni singolo evento di stallo* L influenzino la qualità percepita dagli utenti finali, soprattutto in ambito di **mobilità**.

3.3.1 Fattori Chiave di Influenza sulla QoE

Come accennato nella sezione precedente, in [14] sono riportati una serie di risultati utili circa lo studio sull'impatto che i ritardi iniziali e le pause intermedie hanno sulla QoE. In particolare, questi dati sono presi da una ricerca condotta tramite *crowdsourcing* dove ben 1349 utenti provenienti da 61 Paesi hanno partecipato ai test sugli stalli, per un totale di 4047 video visualizzati. Uno dei maggiori risultati di questo studio, come abbiamo già visto, è stato quello di dimostrare come non vi fosse differenza fra i risultati raccolti in modo rigoroso in laboratorio e quelli ottenuti proprio tramite

crowdsourcing (dopo un'operazione di filtraggio di quanti non risultassero coerenti), dimostrando chiaramente come questa tecnica possa essere usata a tutto vantaggio di costi minori, un bacino di utenza maggiore ed una riduzione nelle tempistiche di raccolta ed elaborazione dei dati.

In questa ricerca, oltre ai test di cui abbiamo già parlato nella sezione precedente, sono state analizzate diverse variabili candidate ad essere elette come **fattori chiave di influenza sulla QoE**. Fra queste è stato quantificato 1) l'impatto dovuto al *numero di stalli* N, 2) la *lunghezza di ogni singolo evento di stallo* L e 3) i risultanti *tempi di stallo totali* T. Sono stati considerati 4) i risultati provenienti da ogni campagna di raccolta dati e 5) i video mostrati, in modo da tenerne in considerazione la tipologia, così come la risoluzione, il setting dei codec, ecc... È stato inoltre chiesto agli utenti di valutare 6) se avessero gradito il contenuto del video usando la solita scala *ACR* su 5 punti [9]. Sono stati poi raccolti una serie di dati riguardanti il background degli utenti che hanno partecipato al test includendo la compilazione di un questionario demografico circa 7) l'età, 8) il genere, 9)-13) ecc, così come 16)-17) domande riguardanti le abitudini nell'uso di applicazioni Internet. Sono poi stati raccolti dati circa 14) la velocità della rete ed 15) il browser usato, in modo da poter individuare potenziali fattori in grado di influenzare la QoE relativa alla visione di video sulla *piattaforma YouTube*.
I risultati così come ottenuti dagli autori in [14] sono mostrati in Figura 3.13.

Come *coefficiente di correlazione* (barre nere), ovvero un indice che esprime un'eventuale relazione di linearità tra due variabili statistiche X ed Y, è stata calcolata una variante del **Coefficiente di correlazione di Pearson** [18] [19] (già trattato nella Sottosezione 3.1.3). I valori vengono difatti convertiti in *ranghi* prima del calcolo: si tratta del **Coefficiente di correlazione per ranghi di Spearman** [39].
Generalmente indicato con la lettera ρ_s, r_i ed s_i sono rispettivamente i ranghi delle due variabili X ed Y all'i-esima osservazione:

$$\rho_s = \frac{\sum_i (r_i - \overline{r})(s_i - \overline{s})}{\sqrt{\sum_i (r_i - \overline{r})^2} \sqrt{\sum_i (s_i - \overline{s})^2}}$$

Per quanto riguarda invece le barre gialle in Figura 3.13, esse sono generate mediante l'uso di **Support Vector Machines - SVM** o **macchine a vettori di supporto** [40], ovvero metodologie di apprendimento supervisionato per la regressione e la classificazione di pattern sviluppate negli anni '90. In questo caso è stata utilizzata l'implementazione **SMO - Sequential Minimal Optimization** [41] in *WEKA* [42]. Ad ogni variablie è associato un peso dal modello che ne indica l'importanza. Siccome le SVM agiscono solo su *due classi* di problemi, i valori della scala ACR che vanno da 1 a 3 sono stati inseriti nella classe "bad quality", mentre i restanti valori 4 e 5 sono stati inseriti nella classe "good quality".

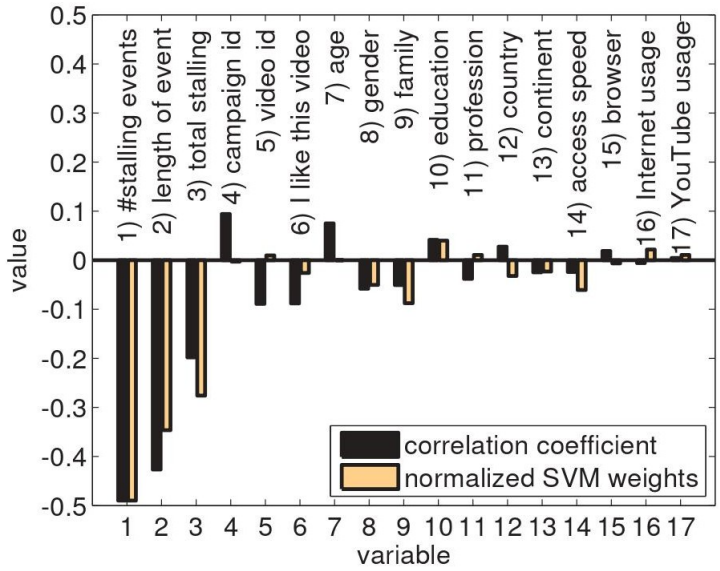

Figura 3.13: Fattori chiave di influenza sulla QoE [14]

Figura 3.13 mostra chiaramente come vi sia una **correlazione medio/forte** fra gli **eventi di stallo** e la **qualità** esperita dagli utenti e come siano proprio essi a farla da padroni come fattori chiave di influenza sulla QoE.
Ciò che sorprende, inoltre, è come parametri quali la risoluzione del video, il suo contenuto, ecc, non risultino essere correlati alle valutazioni degli utenti.

3.3.2 Numero di Stalli vs. Durata Eventi di Stallo

Appurato come a dominare i fattori chiave di influenza sulla QoE siano gli *stalli* ed in particolare il *tempo totale di stallo* T, il *numero di stalli* N e la *durata di ogni evento di stallo* L, vedremo ora come essi concorrono nel degradare la qualità percepita dagli utenti finali.

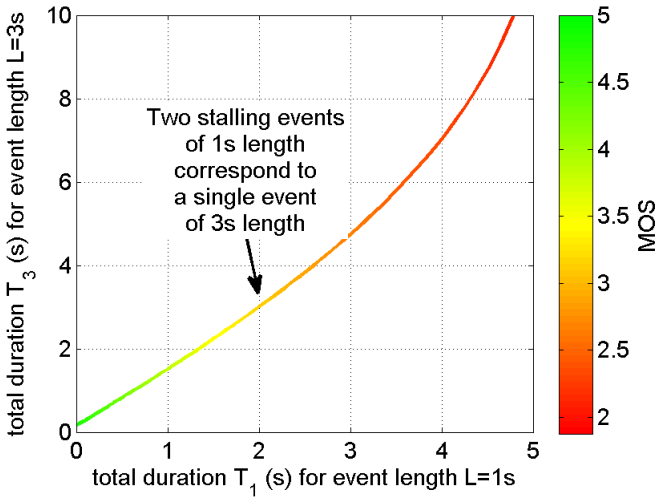

Figura 3.14: Influenza di N vs. influenza di L sul MOS (1 di 3) [43]

Secondo quanto riportato in *Pippi Longstocking Calculus for Temporal Stimuli Pattern on YouTube QoE: 1+1 = 3 and 1·4 ≠ 4·1* [43], infatti, nonostante i tempi totali di attesa che intercorrono durante la visione di un video T siano gli stessi, gli utenti possono percepire una qualità finale diversa in base ai **pattern di occorrenza degli stalli**, ovvero ai parametri N ed L.
Anche in questo caso, gli autori hanno condotto i test usando le modalità più volte viste in precedenza [14], ovvero utilizzando un approccio basato sul *crowdsourcing* dove è stata usata la piattaforma *Microworkers.com*, mostrando video con contenuti diversi e generando gli stalli mediante l'utilizzo del linguaggio *Javascript*, il quale interagisce con un'istanza del player YouTube *Chromeless* integrato nelle pagine web dinamicamente generate.
Per questo test sono stati usati ben 8163 video della lunghezza di 30 secondi, dove è stato variato sia il numero N di eventi di stallo che la durata L di

ciascun evento. Sono stati coinvolti 2035 utenti provenienti da più di 60 Paesi i quali, dopo aver guardato ogni video afflitto da un pre-determinato pattern di stalli, ha espresso un suo giudizio usando la solita scala *ACR* su 5 punti: "Hai percepito queste pause come fastidiose?" 5) impercettibili, 4) percettibili, 3) leggermente fastidiose, 2) fastidiose, 1) molto fastidiose.

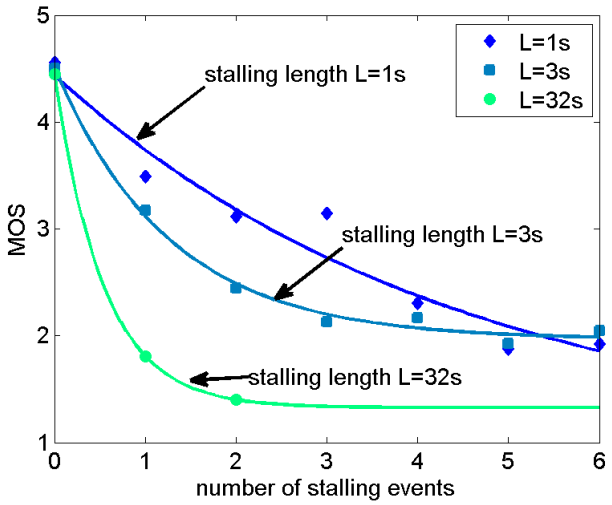

Figura 3.15: Influenza di N vs. influenza di L sul MOS (2 di 3) [43]

Il primo risultato ottenuto dagli autori di questa ricerca è esattamente quanto sintetizzato anche nel titolo, ovvero: **QoE(1+1) = QoE(3)**.
Figura 3.15 conferma quanto già evidenziato in Sezione 3.1, Sottosezione 3.1.2, relativa allo studio circa l'impatto che l'uso del *protocollo TCP* ha sulla QoE in relazione al numero di eventi di stallo che intercorrono durante la visione di filmati YouTube. Figura 3.2 riporta infatti i grafici relativi all'influenza che hanno gli stalli di durata 1 e 3 secondi sul MOS al variare del numero di occorrenze. Figura 3.15 amplia questi risultati mostrando come, all'aumentare della durata L dei singoli stalli, il valore di MOS decresce sempre più rapidamente in base al numero N di eventi di stallo.
In particolare, è possibile vedere come il valore di $MOS = 3$ ("leggermente fastidiosi") è ottenuto sia a partire da da due eventi ($N = 2$) della durata ciascuno di 1 secondo ($L = 1$) che da un singolo evento di stallo ($N = 1$)

della durata di 3 secondi ($L = 3$). Da qui la formula QoE(1+1) = QoE(3), ovvero: la qualità percepita dagli utenti al verificarsi di 2 stalli della durata di 1 secondo ognuno è la stessa che essi sperimenterebbero all'occorrenza di un solo stallo della durata di 3 secondi.
Si noti comunque come per tempi totali di attesa piccoli (sotto i 3 secondi per stalli della durata di 1 secondo e sotto i 4.5 secondi per stalli della durata di 3 secondi) la curva mostra un rapporto lineare fra i valori di MOS percepiti, con $L_3 \approx 1.5L_1$. Per valori maggiori la curva acquista pendenze maggiori.
Questa osservazione mostra come la valutazione di un singolo parametro quale la durata complessiva del tempo di stallo non è sufficiente per modellare adeguatamente la QoE relativa alla visione di video YouTube.
Anche in questo caso, comunque, la **IQX hypothesis** (ovvero la dipendenza esponenziale fra la QoE e le disfunzioni di qualità - QoS - come, in questo caso, gli stalli) è verificata e segue la forma riportata in Sottosezione 3.2.2 (si faccia riferimento alla Sottosezione 3.1.5 per maggiori dettagli):

$$f_L(N) = \alpha_L e^{-\beta_L N} + \gamma_L = y$$

Il secondo risultato ottenuto dagli autori è stato quello di mostrare quale parametro fra il numero N di stalli e la durata L di ciascuno stallo influisce maggiormente sulla qualità finale percepita dell'utente e come, usando sempre la sintassi introdotta precedentemente, $\mathbf{QoE(1{\cdot}4) \neq QoE(4{\cdot}1)}$.
Questa dicitura specifica come la qualità che gli utenti sperimentano al verificarsi di 4 stalli della durata di 1 secondo ciascuno non sia la stessa di quella sperimentata al verificarsi di un solo evento della durata di 4 secondi.
Come mostra Figura 3.16, assumendo il tempo totale di stallo $T = N \cdot L$ costante, più sono le interruzioni che si verificano (curva rossa) e più velocemente decrescerà la qualità percepita dagli utenti. Si noti come per valori di $T = 4$ secondi, si abbia un valore di $MOS \approx 2.3$ nel caso in cui si verifichino 4 eventi di stallo (curva rossa) e $MOS \approx 2.8$ nel caso in cui se ne verifichi solo uno (curva blu). Per $T > 4$ la differenza si assesta a circa 1 MOS.

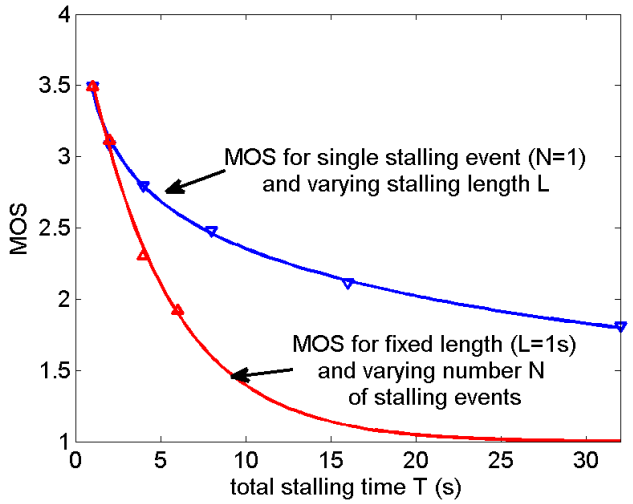

Figura 3.16: Influenza di N vs. influenza di L sul MOS (3 di 3) [43]

Queste scoperte confermano ancora una volta come il tempo complessivo T nel quale il video rimane in stallo non sia l'unico parametro che influenzi la qualità percepita dagli utenti finali e come, fissando questa variabile, sia il **numero N di eventi di stallo** ad avere un peso meggiore nella degradazione della qualità rispetto alla durata di ogni singolo evento di stallo L, come mostrato anche in Figura 3.14.

3.3.3 Stalli e Mobilità

Gli stessi autori di [43], dopo aver analizzato quali aspetti legati all'occorrenza degli stalli apportassero il maggior contributo circa la degradazione della QoE, si sono dedicati allo studio dei pattern di stallo in un ambito che interessa particolarmente gli obiettivi di questa Tesi di Laurea, ovvero quello legato alla **mobilità** ed in particolare all'uso della **tecnologia 3G** per l'accesso ai contenuti di video streaming presenti su Internet.
Innanzitutto si sono occupati di definire un ambiente di studio che fosse il più *realistico* possibile. A tale scopo sono state effettuate misurazioni circa i pattern di stallo che occorrono in YouTube usando la rete 3G fornita da un operatore mobile pubblico tedesco.

Mentre si provvedeva alla visualizzazione di video YouTube, i pattern di stallo erano monitorati a livello applicazione usando le apposite API, le quali hanno permesso l'embedding e la riproduzione dei video. Lo stato del player ("playing", "buffering" e "ended"), così come i relativi timestamp, sono stati catturati usando il linguaggio Javascript. L'utilizzo della rete 3G tramite laptop era consentito grazie all'utilizzo di uno smartphone che forniva connettività via *tethering usb*.

Figura 3.17 mostra i risultati per 5 clip video scelte casualmente, con focus sulla durata di ogni singolo evento di stallo L e sul tempo che intercorre fra due eventi di stallo consecutivi Δt. Nel caso precedentemente descritto in cui era preso in esame lo scenario della bottleneck, si verificavano pattern di stallo periodici, i quali risultavano in tempi di stallo costanti e intervalli fra due interruzioni consecutive determinati, con un valore per il **coefficiente di variazione** pari a 0. In questo caso, invece, tale coefficiente assume valori diversi da zero per tutti e 5 i video in esame, con una media pari a 0.82.

Tali risultati suggeriscono come il verificarsi di **pattern di stalli non periodici** siano la regola per quanto riguarda l'utilizzo della tecnologia 3G per la connessione ad Internet.

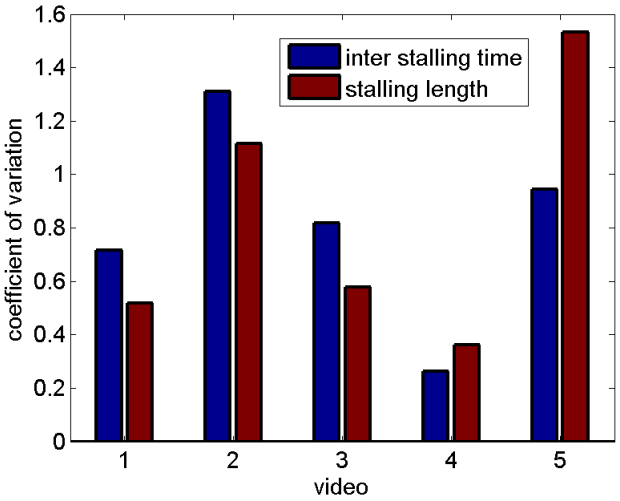

Figura 3.17: Coefficiente di variazione per pattern di stalli usando il 3G [43]

In un altro studio, *Quantifying the Influence of Rebuffering Interruptions on the User's Quality of Experience During Mobile Video Watching* [44], è stata condotta una ricerca simile ma da autori diversi, utilizzando per la connessione ad Internet la tecnologia **WiFi** anziché il 3G.
L'ambiente di studio prevedeva che la connessione fornita tramite 802.11g fosse limitata alla banda tipica del 3G. La media delle misurazioni di throughput era di 564 Kbps, la media dell'audio bitrate dei video utilizzati era 62 Kbps mentre la media del video bitrate era 109 Kbps. Sono stati impiegati 3 video mostrati a 57 utenti, per un totale di 171 test.
Anche in questo caso, per una distribuzione esponenziale il coefficiente di variazione si è rivelato essere pari a 1, sintomo della presenza di **pattern non periodici** nell'ambiente 3G emulato.

Capitolo 4

Proposte

Dopo aver analizzato nel dettaglio da un punto di vista della qualità percepita dall'utente finale di applicazioni e servizi di video streaming:

- quali sono le conseguenze dell'utilizzo dei protocolli *UDP* e *TCP* a livello trasporto;
- quali *effetti di degradazione* sono meglio tollerati fra quelli che influenzano la *qualità* del video in riproduzione e quelli che influenzano proprio la *riproduzione* stessa;
- quali fra gli effetti di degradazione temporale influenzano in modo maggiore la qualità percepita dagli utenti, se i *ritardi iniziali* o le *pause intermedie* (*stalli*);
- quali sono i fattori chiave che l'utente finale maggiormente percepisce come disturbi all'interno dei *pattern di stallo*, se il numero complessivo di stalli, il tempo totale di stallo, la frequenza con la quale le interruzioni occorrono, ecc,

è giunto il momento di analizzare come queste informazioni sono state combinate assieme al fine di implementare alcune **soluzioni** di accesso e scambio dati basate sulla tecnologia del video streaming.
Dapprima saranno analizzate tre **soluzioni proprietarie**: *Microsoft Smooth*

Streaming, *Apple HLS - HTTP Live Streaming* ed *Adobe HDS - HTTP Dynamic Streaming*. Di queste saranno analizzate perlopiù la gestione dei *segmenti* e del *manifest* ovvero, rispettivamente, delle singole parti nelle quali il contenuto multimediale è suddiviso ed il file contenente i metadati, il quale fornisce una rappresentazione del media sul server.
Dopodiché si passerà ad esaminare i punti che le tre tecnologie hanno in comune e si descriverà il funzionamento del **protocollo MPEG-DASH**, uno *standard* per il video streaming.
Infine, saranno mostrate le principali **euristiche di adattamento**, ovvero quegli algoritmi che, analizzati diversi parametri, guidano la scelta del prossimo pacchetto (segmento) da scaricare.

4.1 Soluzioni Proprietarie

Come abbiamo visto nella Sottosezione 1.2.3, lo **streaming adattivo su protocollo HTTP** rappresenta un'ottima soluzione per realizzare servizi nei quali i contenuti audio/video sono forniti in streaming on-demand, con l'obiettivo di ridurre al minimo le interruzioni del servizio a causa di problematiche di rete o strettamente legate allo specifico ambito della mobilità, dove variazioni nella bitrate possono causare lo svuotamento precoce del buffer del video player e risultare in fastidiosi *stalli*.
Grazie a questa tecnica il contenuto multimediale è generalmente diviso in *segmenti* di cui il server ne mantiene diverse codifiche ed il client ne effettua il download tenendone in considerazione la dimensione in relazione alla bitrate disponibile, il livello del buffer del player ed altre informazioni che dipendono dalla specifica *euristica di adattamento* (algoritmo) implementata.
Al fine di realizzare servizi basati su questa tecnologia, sono state implementate varie soluzioni, fra le quali spiccano **Microsoft Smooth Streaming**, **Apple HTTP Live Streaming** ed **Adobe HTTP Dynamic Streaming**.

4.1.1 Microsoft Smooth Streaming

Secondo quanto riportato nel documento *IIS Smooth Streaming Technical Overview* [45], nell'Ottobre del 2008 **Microsoft** annunciò che la *IIS - Internet Information Services*[1] *7.0* sarebbe stata dotata di una nuova estensione per lo streaming adattivo basato su HTTP: **Smooth Streaming**.
Questa tecnologia non prevede la suddivisione del contenuto multimediale in più file, uno per segmento, ma un unico file contiguo per ogni bitrate codificata. Il formato scelto a tale scopo è *MPEG-4* (si veda Sottosezione 4.2.1 per un approfondimento su questo formato). Dunque, la specifica Smooth Streaming definisce ogni segmento come un *MPEG-4 Movie Fragment* e lo memorizza in un unico file *MP4* per facilitarne l'accesso random. Per ogni codifica supportata è presente un solo file MP4. Quando un client richiede al Web server IIS uno specifico segmento, esso ricerca il relativo Movie Fragment all'interno del file MP4 e lo inoltra sulla rete come un file standalone. In pratica, con Smooth Streaming i segmenti sono creati *virtualmente* in base alle richieste del client, ma il video è memorizzato per intero su disco come un unico file per bitrate. Tale scelta implementativa offre grandi vantaggi dal punto di vista della gestione dei file memorizzati sul server, permettendo comunque al client la totale cacheabilità del contenuto scaricato.

Secondo quanto riportato in [45], Smooth Streaming è stato il primo format multimediale Microsoft a non utilizzare *ASF - Advanced Systems Format* per la codifica dei file memorizzati. Come già detto, infatti, la codifica usata è *MP4*. Tale scelta è supportata da diverse ragioni:

- innanzitutto MP4 è un formato più leggero rispetto ad ASF ed introduce un minor overhead;
- in secondo luogo MP4 risulta un formato più semplice da parsare dal codice utilizzato (*.NET*) rispetto ad ASF;
- è basato su uno standard ampiamente usato, il ché ne rende l'adozione ed il supporto da parte di terze parti più semplice;

[1] complesso di servizi server Internet per sistemi operativi Microsoft Windows

- è progettato per supportare il codec video *H.264*, uno standard di compressione video leader nel settore, adottato da una vasta gamma di prodotti Microsoft, inclusi *Silverlight 3*, *Windows 7*, *Xbox 360*, *Zune* e *MediaRoom*;
- infine, MP4 è progettato per supportare in modo nativo la frammentazione del payload all'interno del file.

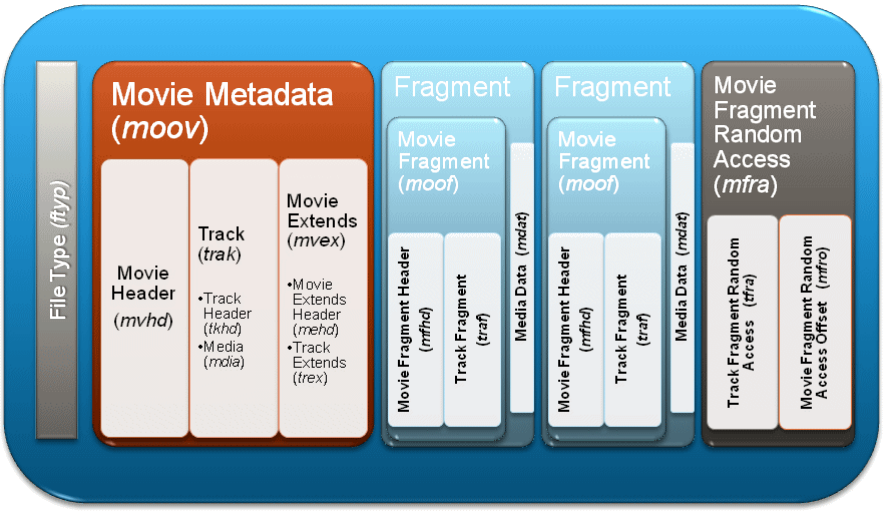

Figura 4.1: Formato del file Smooth Streaming [45]

Figura 4.1 mostra una panoramica ad alto livello di come un file Smooth Streaming è fatto all'interno. Esso inizia con i metadati che descrivono in maniera generale il file: sono i cosiddetti *moov*. Dopodiché troviamo una serie di *Fragment*, i quali costituiscono la maggior parte del file in quanto ne rappresentano il payload. Ogni fragment è costituito da un'altra porzione di metadati che lo descrive in maniera più accurata, *moof*, e dai dati del media veri e propri, *mdat*. In figura sono rappresentati solamente due Movie Fragment, ma in un tipico file Smooth Streaming è generalmente presente un Fragment per ogni 2 secondi di contenuto multimediale (audio/video). Alla fine troviamo un *mfra*, ovvero un'area indicizzata che permette di effettuare ricerche in modo semplice ed accurato all'interno del file.

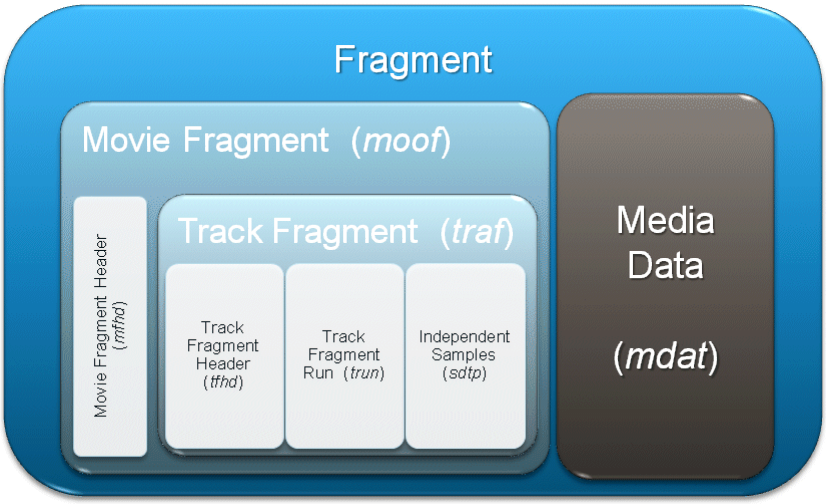

Figura 4.2: Formato del fragment Smooth Streaming [45]

Figura 4.2 mostra più nel dettaglio come è composto un Movie Fragment. Quando un client richiede una porzione di contenuto multimediale al Web server IIS, esso effettua una ricerca del frammento più appropriato all'interno del file e lo consegna al client così com'è. Questa tecnica consente di migliorare notevolmente l'efficienza del server Web IIS in quanto non introduce alcun overhead dovuto ad operazioni di re-muxing o rewriting.

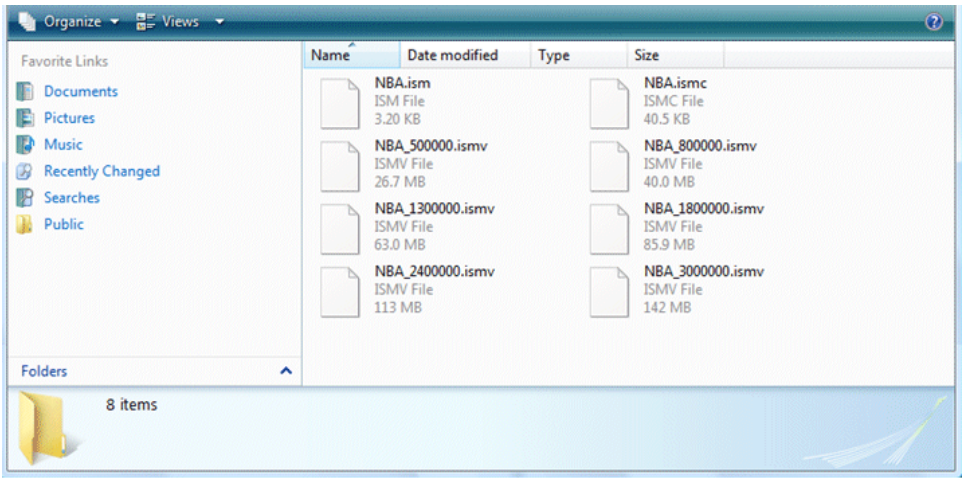

Figura 4.3: Presentazione Smooth Streaming codificata [45]

Un esempio di interazione fra un client ed un server in Smooth Streaming può avvenire nel seguente modo:

- prima di tutto il client scarica il *client manifest *.ismc*, il quale ha lo scopo di descrivere gli stream disponibili: i codec usati, le bitrare disponibili, la risoluzioni video, la lista di frammenti con la marca temporale di inizio e la loro durata, ecc. . . ;

- in base al codec scelto, il client inizializza il relativo decoder e la pipeline per il playback;

- dopodiché il client inizia a richiedere i frammenti usando le apposite API RESTFUL definite tramite appositi URL, passando come parametri la bitrate e l'offset al quale il frammento ha inizio;

- una volta ricevuta la richiesta, il server IIS cerca il frammento (in base alla qualità specificata) nel file *server manifest *.ism*, il quale descrive le relazioni fra i media, le bitrate ed i file su disco, mappandolo nel relativo file fisico **.ismv* o **.isma*, a seconda se questo contiene una componente video o solo audio;

- una volta identificato il file fisico, con il meccanismo descritto in precedenza identifica il relativo frammento, lo estrae e lo inoltra al client.

Per quanto riguarda Microsoft Smooth Streaming, il server non gioca alcun ruolo nel processo di switching della bitrate. È il codice lato client che si occupa di analizzare i tempi di download dei segmenti, il tempo di buffer, i frame rate renderizzati ed altri fattori e decide quando richiedere bitrate superiori o inferiori al server.
Se durante il processo di codifica ci si assicura che tutte le bitrate dello stesso sorgente siano perfettamente allineate al fotogramma (stessa lunghezza, nessun frame omesso, ecc), allora il passaggio da una all'altra avverrà in modo completamente lineare e continuo.

4.1.2 Apple HLS - HTTP Live Streaming

Così come Microsoft, anche **Apple** ha sviluppato una piattaforma per lo streaming di contenuti audio e video che permettesse ai content provider di inviare contenuti multimediali live o pre-encoded a dispositivi quali iPhone, iPad, iPod touch, Apple Tv e Mac. Tale piattaforma è nota con il nome di **HLS - HTTP Live Streaming**, la cui prima pubblicazione avvenne su *IETF - Internet-Draft* nel 2009. La versione 7 del protocollo è descritta nel documento *HTTP Live Streaming: draft-pantos-http-live-streaming-23* [46] e fa riferimento all'*RFC 8216 - HTTP Live Streaming* [47].
Anche in questo caso, la scelta di basare la propria piattaforma sull'uso del protocollo HTTP è stata dettata dalla facilità con la quale è possibile sviluppare lo streaming di contenuti multimediali utilizzando dei comuni web server al posto di server specializzati.

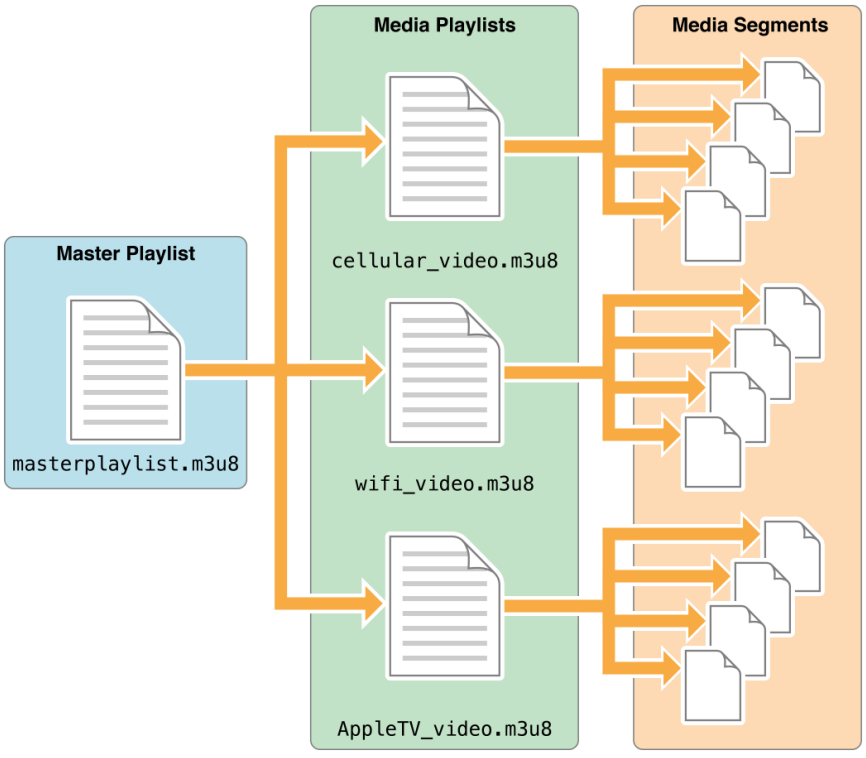

Figura 4.4: Relazioni fra le playlist in HLS [48]

Un contenuto multimediale in HLS è identificato da un *URI - Uniform Resource Identifier* ad una *playlist*. Queste possono essere di 2 differenti tipi, entrambi file di testo codificati UTF-8 che contengono URI e tag descrittivi:

- *Master Playlist*: fornisce una serie di *Variant Streams*, ognuno dei quali descrive una differente versione dello stesso contenuto;
- *Media Playlist*: contiene una lista di *Media Segments* i quali, una volta eseguiti sequenzialmente, formano il contenuto multimediale.

In sostanza, una playlist può essere definita come una *Master Playlist* se tutte le linee in essa contenute che definiscono URI si riferiscono a Variant Streams. Una *Media Playlist*, invece, può essere definita come tale quanto tutte le linee che definiscono URI al suo interno si riferiscono a Media Segments. Tutte le altre playlist non sono considerate valide dal protocollo.
Un *Variant Stream* include una Media Playlist, la quale specifica la bitrate alla quale il media è codificato, il formato del media e la risoluzione dello stesso nel caso in cui esso contenga video.

Quando un client HLS vuole riprodurre un determinato contenuto multimediale, se l'URI della playlist in suo possesso definisce una Master Playlist, allora ne esamina i formati, le bitrate e le risoluzioni delle varianti degli stream in essa contenuti. A quel punto seleziona la specifica variante e ne acquisisce l'URI. Esso definisce l'identifier di una Media Playlist. A questo punto, così come nel caso in cui il client disponga direttamente dell'URI di una Media Playlist senza passare tramite una Master Playlist, ne esamina i Media Segments e ne avvia il download per poterli poi riprodurre.
La durata di ogni *Media Segment* è specificata dal tag *EXTINF* nella Media Playlist la quale può includere, in aggiunta, il range in Byte dello specifico segmento. Nel caso in cui esso contenga dati video, inoltre, dovrebbe includere le informazioni necessarie al fine di inizializzare il relativo decoder.
I formati supportati dal protocollo con i quali i segmenti possono essere codificati sono: *MPEG-2* ed *MPEG-4*, *Packed Audio* (*AAC* con framing *ADTS*, *MP3*, *AC-3* ed *Enhanced AC-3*) e *WebTTV* per i sottotitoli.

4.1.3 Adobe HDS - HTTP Dynamic Streaming

Così come Windows ed Apple, anche **Adobe** ha sviluppato un protocollo per la distribuzione di contenuti audio/video su HTTP con l'obiettivo, come si legge testualmente nell'*HTTP Dynamic Streaming Specification - Version 3.0 Final* [49], di consentire la distribuzione di media in modo completo ed efficiente, senza la necessità di dover utilizzare specifici server dedicati "special purpose". Si tratta del protocollo **HDS - HTTP Dynamic Streaming**.
Come si legge nelle specifiche, sfruttando l'infrastruttura HTTP esistente, HDS è in grado di ridurre in modo significativo il costo di distribuzione dei media su Internet, fornendo comunque accesso a funzionalità quali limitazioni della bandwidth, avvii e ricerche a bassa latenza e protezione del contenuto.

In HDS un *flusso di contenuti* è un flusso multimediale che rappresenta un singolo film, episodio od evento live. Una *presentazione* (presentation) è definita da un gruppo di *rappresentazioni* o *rese* (rendition) che definiscono un contenuto multimediale. La presentazione è descritta da un documento chiamato *manifest*. Esso può essere uno solo o suddiviso in più documenti. Una rappresentazione, invece, è una realizzazione unica di un flusso di contenuti destinata a rappresentare una codifica di una o più tracce dello stesso flusso. È descritta dall'elemento $\langle media \rangle$ nel manifest, il quale include spesso informazioni addizionali quali la bitrate o la lingua nella quale la rappresentazione è codificata. Essa si compone di una sequenza di *frammenti* numerati (identificati da un numero) che possono essere scaricati, ognuno dei quali è composto da contenuti audio e/o video in un dato intervallo. A loro volta, i frammenti possono essere raggruppati in *segmenti*.
Secondo la specifica, un *frammento* è un'unità sufficientemente grande da potersi muovere in modo efficiente nell'infrastruttura HTTP ma, allo stesso tempo, sufficientemente piccola da poter essere trattata come un'unità discreta per il download. Un frammento potrebbe consistere in zero o un canale di contenuti audio, zero o un canale di contenuti video, zero o un canale di contenuti data. Le specifiche suggeriscono come i segmenti dovrebbero essere quanto più simili fra loro per tempo di contenuto e dimensione.

I *segmenti*, invece, sono utilizzati per raggruppare uno o più frammenti in unità contigue più grandi. In questo modo possono essere considerevolmente ridotti il numero di file che devono essere gestiti dal filesystem ed, allo stesso tempo, può essere migliorata l'efficienza della cache HTTP. È infatti permesso ai proxy adibiti al caching, il pre-fetching dell'intero segmento mentre sono elaborate le richieste per i singoli frammenti.
Una *rappresentazione*, dunque, altro non è che una serie di frammenti, ognuno dei quali è identificato da un *fragment number* usato per l'indirizzamento. Allo stesso modo i segmenti sono identificati da una *fragment naming sequence*, una sequenza di numeri non per forza continua. Le discontinuità possono rappresentare "buchi" nel contenuto, generalmente causate dallo stream di contenuti live a causa della codifica, della congestione dei server o al fallimento di un altro elemento nella rete.
Una *presentazione*, infine, può essere composta da più rappresentazioni. È il caso in cui vi sono più versioni di un contenuto audio o video, come la codifica a diverse bitrate, utili nel caso dello streaming adattivo.
Il *manifest* deve essere un documento *XML* contenente i metadati necessari al fine di guidare il client nell'elaborazione di un flusso HDS. Esso deve contenere i metadati relativi al contenuto e agli indici dei frammenti e dei segmenti, le cosiddette *informazioni di bootstrap* e, facoltativamente, i metadati per la protezione del contenuto. Infine, deve rispettare il formato specificato in *[F4MSPEC] - Flash Media Manifest Format Specification Version 3.0* [50].
Le *informazioni di bootstrap* per una rappresentazione indicizzano tutti i frammenti ed i segmenti disponibili, forniscono un mapping dal tempo relativo alla riproduzione del contenuto al fragment number e dal fragment number al segment number. Tali informazioni sono di fondamentale importanza per il client durante il download dei frammenti in relazione al tempo di riproduzione attuale. Nel caso di una live stream, le informazioni di bootstrap devono poi contenere il cosiddetto "live time", ovvero il tempo di riproduzione attuale. Le informazioni di bootstrap sono descritte all'interno del tag $\langle bootstrapinfo \rangle$ presente nel manifest.

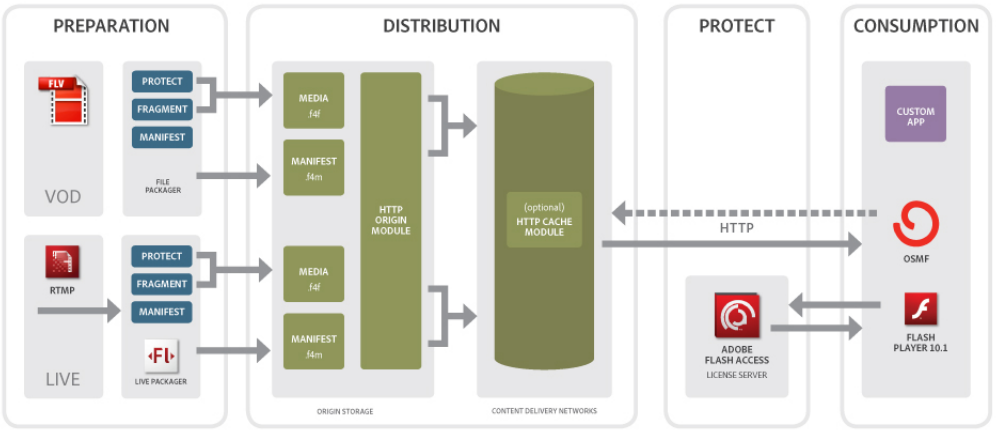

Figura 4.5: Adobe HTTP Dynamic Streaming [51]

Le operazioni che un client deve effettuare per lo streaming di un contenuto multimediale in HDS differiscono a seconda del fatto che esso sia uno stream VoD - Video on Demand, oppure un Live stream.
Nel primo caso il manifest deve essere acquisito solo una volta, all'inizio della sessione di riproduzione, la quale è normalmente avviata a partire dal primo segmento specificato nelle informazioni di bootstrap non appena il player ha riempito il buffer fino al livello richiesto dalla specifica implementazione. Il download dei segmenti viene messo in pausa una volta che il buffer di riproduzione è stato riempito e riprende una volta che lo spazio torna disponibile. Poiché ogni frammento è per definizione decodificabile in modo indipendente, un client può trattare l'inizio di ogni frammento come un random access point, caratteristica favorevole per lo streaming adattivo in quanto non è richiesto il download di acluna informazione aggiuntiva nel caso in cui si decida di passare da una bitrate all'altra.
Nel caso in cui il contenuto in download sia un live stream, sono in primo luogo scaricati i frammenti relativi al "live time" sino al livello del buffer prestabilito. Dopodiché, una volta avviata la riproduzione, il download dei frammenti continua in background. Solamente una volta aver effettuato il download di tutti i frammenti descritti nel manifest, il client scarica la nuova versione di questo documento con le informazioni circa i nuovi frammenti.

Le specifiche sottolineano come un client dovrebbe selezionare una lunghezza del buffer in grado di fornire interruzioni di riproduzione minime, tenendo conto di fattori quali le condizioni di rete previste, la latenza ed i tempi di avvio desiderati e gli effetti sulla scalabilità dei server. Le specifiche di come selezionare una lunghezza del buffer appropriata non rientrano nell'ambito del documento in analisi [49] ma, in assenza di altri fattori, esso consiglia ai client di selezionare una lunghezza del buffer di almeno tre volte la durata del frammento ideale.
In Adobe HDS lo streaming adattivo è permesso quando il manifest contiene la descrizione di più rappresentazioni. In tal caso, un client dovrebbe selezionare una rappresentazione iniziale in base alle caratteristiche note circa il dispositivo di riproduzione (es: potenza di elaborazione dei media) e le precedenti misurazioni sulla larghezza di banda disponibile. In linea con il principio dello streaming adattivo, inoltre, il client dovrebbe tentare di selezionare la rappresentazione che meglio si adatti alla larghezza di banda disponibile in relazione alle attuali condizioni di rete. Anche per una volta, le specifiche su come attuare questo comportamento non rientrano nell'ambito del documento in esame, ma come per il caso precedente, esso fornisce alcune raccomandazioni di base, quali il monitoring dell'attuale livello del buffer di riproduzione e della velocità con cui vengono scaricati i nuovi frammenti, al fine di determinare se la bitrate della rappresentazione corrente è troppo alta o troppo bassa in relazione alle condizioni attuali.

4.2 MPEG-DASH

Come abbiamo visto nella sezione precedente, piattaforme di streaming quali *Microsoft Smooth Streaming*, *Apple HLS* e *Adobe HDS* basano le proprie tecnologie sullo streaming HTTP come metodo di base per la consegna dei dati. Tuttavia ogni implementazione utilizza formati proprietari nella definizione del *manifest* e dei *segmenti video*. Perciò, per ricevere i dati da un server, il client deve supportarne il relativo protocollo.

La definizione di uno **standard** per lo streaming HTTP di contenuti multimediali permetterebbe a tutti i client standard-based di scaricare dati audio/video da qualsiasi server standard-based, garantendo interoperabilità fra client e server di diversi fornitori. È proprio questo quello di cui ci occuperemo in questa sezione.

4.2.1 MPEG

MPEG - Moving Picture Experts Group [52] è un *working group*[2] creato da *ISO - International Organization for Standardization* [53] ed *IEC - International Electrotechnical Commission* [54] con l'obiettivo di sviluppare standard per la compressione e trasmissione audio e video. La sua designazione ufficiale è *ISO/IEC JTC 1/SC 29/WG 11 - Coding of Moving Pictures and Audio* (*ISO/IEC Joint Technical Committee 1, Subcommittee 29, Working Group 11*) [55] [56]. Dal 1988, anno della sua creazione, nonché anno del primo meeting ufficiale ad Ottawa, Canada, il gruppo ha prodotto numerosi standard, fra cui:

- *MPEG-1* [57]: la prima compressione audio e video standard del working group MPEG che include, fra gli altri, il famoso formato di compressione audio *MP3 - MPEG-1 Audio Layer III*;
- *MPEG-2* [58]: introdotto nel '94, riscosse un grande successo in quanto destinato al broadcast televisivo ed adottato come standard per i DVD;
- *MPEG-3*: nato per gestire lo standatd *HDTV - High Definition TV*, successivamente dismesso in quanto risultati simili potevano essere raggiunti modificando in modo marginale lo standard MPEG-2;
- *MPEG-4* [59]: nato nel 1996 e finalizzato nel 1998, racchiude una serie di standard per la videotelefonia, la televisione digitale, la trasmissione di filmati via web e la memorizzazione di CD-ROM.

[2] gruppo di esperti che collaborano assieme per il raggiungimento di un obiettivo comune

Ognuno di questi standard racchiude diverse tecnologie che ne rappresentano i campi d'impiego: *MPEG-A* per i formati delle applicazioni multimediali, *MPEG-C* per le tecnologie video, *MPEG-D* per le tecnologie audio, ecc...

Come abbiamo visto, il working group MPEG si è sempre dimostrato puntuale nella definizione di standard in relazione all'emergere di nuove tecnologie che andavano via via affermandosi.
Dapprima è stato creato uno standard per la compressione audio/video "di base", ovvero MPEG-1. Dopodiché è stato creato uno standard per il broadcast televisivo (MPEG-2) e, con l'avvento dell'alta definizione, uno standard per il broadcast HD (MPEG-3). Una volta arrivata la tv digitale, così come la videotelefonia e la trasmissione di filmati via web, è stato definito un nuovo standard che ha tutt'oggi un grande successo: MPEG-4.
Non è difficile immaginare dunque come, visto l'affermarsi dello streaming audio/video on-demand su protocollo HTTP, MPEG abbia creato uno standard su misura per questa nuova tecnologia, dimostrando ancora una volta quanto sia importante rimanere al passo con i tempi. Tale standard ha nome **DASH - Dynamic Adaptive Streaming over HTTP**. [60].

4.2.2 DASH

Lo standard **MPEG-DASH** [5], generalmente abbreviato **DASH - Dynamic Adaptive Streaming over HTTP**, nasce come protocollo per il video streaming su HTTP.
Osservando le prospettive del mercato e le richieste del settore, nell'Aprile del 2009 MPEG ha pubblicato una *Request for Proposal* al fine di definire uno standard per lo streaming HTTP. A Luglio 2009, data in cui MPEG ha avviato la valutazione delle tecnologie presentate, 15 proposte complete erano state ricevute. Nei due anni successivi, MPEG ha sviluppato le specifiche con la partecipazione di molti esperti e con la collaborazione di altri gruppi occupati nella definizione di standard, come il *3GPP - Third Generation Partnership Project*. Lo standard risultante è noto come **MPEG-DASH**.

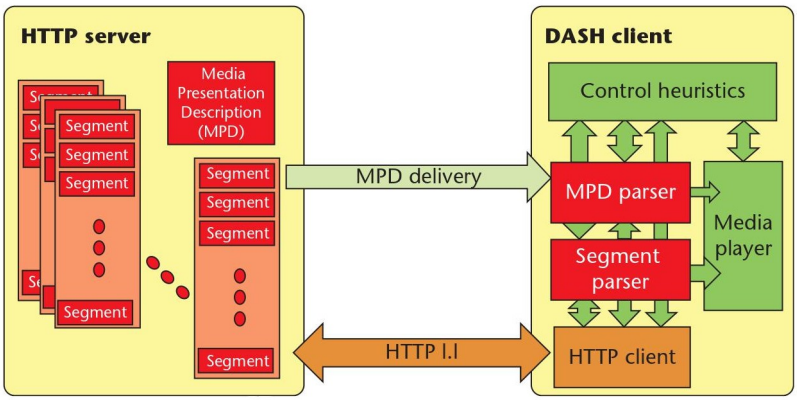

Figura 4.6: Schema di funzionamento generale di MPEG-DASH [5]

In Figura 4.6 è illustrato un semplice scenario di streaming di contenuti fra un *server HTTP* ed un *client DASH*. Il contenuto multimediale è memorizzato in un server e distribuito su protocollo HTTP.
Il contenuto sul server esiste in due parti:

- il **MPD - Media Presentation Description**: descrive il *manifest* del contenuto, le sue alternative, i suoi url ed altre caratteristiche;
- i **segmenti**: contengono gli effettivi bitstream multimediali sottoforma di *chunks* (blocchi), in uno o più file.

Al fine di riprodurre il contenuto, il client DASH deve ottenere prima di tutto il MPD. Questo può essere consegnato al client via HTTP, e-mail, broadcast od altri mezzi di trasporto. Analizzando l'MPD il client apprende la tempistica della programmazione, la disponibilità dei contenuti multimediali, le tipologie dei media, le risoluzioni, la banda minima e massima necessaria, l'esistenza delle varie codifiche alternative, le caratteristiche di accessibilità e la gestione dei diritti digitali necessari (*DRM - Digital Rights Management*), dove i componenti multimediali sono localizzati sulla rete ed altre caratteristiche relative ai contenuti. Grazie a queste informazioni, il client seleziona la codifica appropriata fra le alternative disponibili ed inizia lo streaming del contenuto tramite il fetching dei segmenti usando *richieste HTTP GET*.

Dopo un adeguato *pre-buffering* iniziale in modo da tenere conto delle variazioni del throughput di rete, il client scarica uno dopo l'altro i segmenti necessari, monitorando sempre le fluttuazioni della bandwidth. In relazione alle misurazioni effettuate, questo decide in che maniera adattarsi alla banda disponibile, scegliendo a quale definizione scaricare i segmenti (abbassandone, alzandone o tenendone costante la bitrate), il tutto allo scopo di mantenere un livello di buffer adeguato.

IMPORTANTE: la specifica MPEG-DASH definisce solamente il formato del MPD e dei segmenti. Il metodo di consegna del MPD, le tipologie di formati per la codifica del media e dei relativi segmenti, il comportamento del client per il fetching e la riproduzione dei contenuti, così come l'**euristica di adattamento** (come la qualità del contenuto multimediale deve adattarsi in relazione alle variazioni della banda disponibile e del livello dell buffer) sono al di fuori degli scopi di questo standard.

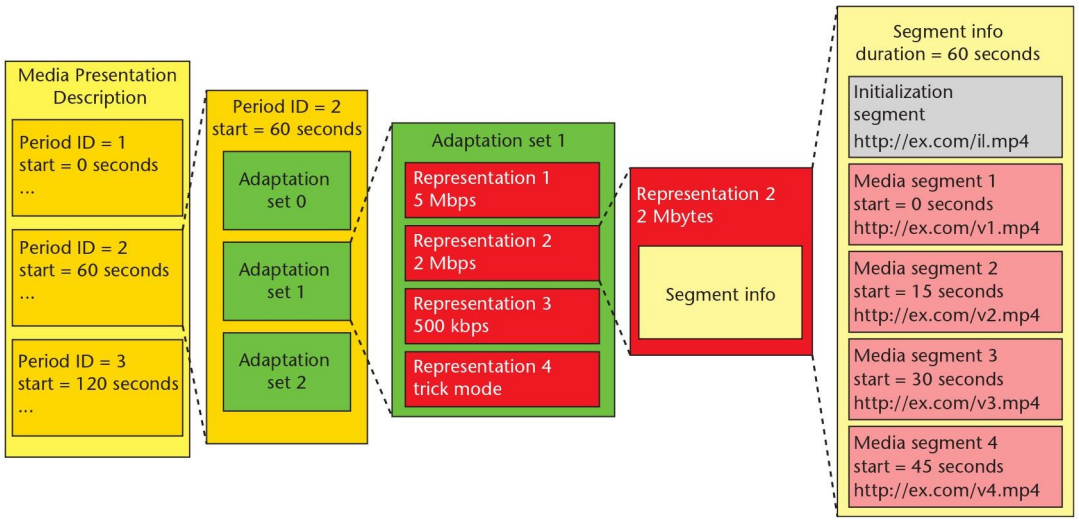

Figura 4.7: Modello gerarchico di MPD [5]

4.2.3 MPD e Segmenti

Affinché lo streaming adattivo (dinamico) funzioni, è necessario che il contenuto multimediale sia mantenuto dai server usando diverse codifiche. Inoltre, tale contenuto può essere costituito da più componenti, ad esempio video, audio, testo, ecc, ognuno dei quali può avere differenti caratteristiche. Tutte queste proprietà sono descritte nel **MPD - Media Presentation Description**, nient'altro che un documento *XML*.
Figura 4.7 mostra la gerarchia del MPD. Esso è costituito da una serie di *period*, ovvero un intervallo lungo l'asse temporale, caratterizzato da un tempo di inizio e da una durata. Ogni period consiste in uno o più *adaptation set*, i quali contengono le informazioni riguardanti i media di uno stesso contenuto multimediale. Ad esempio, un adaptation set può contenere le diverse bitrate video, un altro adaptation set può contenere le diverse codifiche audio e così via, ma sempre tutte riguardanti lo stesso contenuto multimediale. Ogni adaptation set può, a sua volta, includere più *representation*, ovvero un'alternativa nella codifica di uno stesso media, che differisce dalle altre "rappresentazioni" a seconda della bitrate, della risoluzione, del numero di canali od altre caratteristiche. Infine, una rappresentazione si suddivide in una serie di *segmenti*, che sono i *chunks* (frammenti) in sequenza temporale. Ogni segmento ha uno *URI - Uniform Resource Identifier*, ovvero una specifica locazione indirizzabile sul server ed è scaricabile tramite una richiesta HTTP di tipo GET, la quale può includere uno specifico range di byte.
Per utilizzare questo MPD, il client DASH prima di tutto ne parsa l'XML, dopodiché sceglie l'insieme di rappresentazioni per ogni media (adaptation set) che utilizzerà in base alla descrizione che trova nel MPD stesso, alle capacità del client ed alle scelte dell'utente. Il client definisce così una timeline ed inizia a riprodurre il contenuto multimediale richiedendo, per ogni media, i segmenti corrispondenti.

Il contenuto multimediale può dunque essere acceduto tramite una serie di **segmenti**, i quali possono essere definiti come il *body* delle risposte alle richieste HTTP GET (anche parziali, nel caso in cui sia richiesto un deter-

minato range di byte) da parte del client DASH. Il contenuto multimediale è dunque codificato e diviso in una serie di segmenti, il primo dei quali può essere un segmento di inizializzazione, contenente le informazioni necessarie al client DASH per l'inizializzazione del decoder del media e non include, pertanto, alcun dato effettivo.
A ciascun segmento è poi assegnato un URL univoco, un indice ed un tempo di inizio implicito od esplicito, oltre ad una durata. Ogni segmento contiene poi uno *stream access point*, ovvero un punto all'interno dello stream dal quale tale segmento può essere riprodotto. È un "punto di switch" nel quale è possibile passare da un segmento di una data codifica all'altro, in modo che esso possa essere decodificato grazie alle sole informazioni in esso contenute.

Il protocollo MPEG-DASH definisce inoltre sia un formato *MPEG-2 Transport Stream* [58] che *ISO Base Media File Format* [61] come contenitore per i segmenti e risulta *media codec agnostic*, supportando sia contenuti codificati in maniera multiplexed che non multiplexed.

4.2.4 Proprietà del Protocollo

Il protocollo MPEG-DASH è uno standard ricco di **feature**, alcune delle quali includono:

- la possibilità di creare un manifest compatto: gli URL dei segmenti possono essere riportati usando schemi definiti in appositi template, il cui uso risulta in un MPD più compatto;

- al contrario, la possibilità di avere un manifest frammentato: il MPD può essere diviso in più parti o alcuni dei suoi elementi possono essere riferiti esternamente, permettendo il download del MPD in più step;

- la possibilità di specificare, per ogni segmento, più URL: lo stesso contenuto può infatti essere disponibile a più URL, ovvero, su più server ed il client può effettuarne lo stream in parallelo massimizzando, ad esempio, la banda disponibile;

- la possibilità per i segmenti di avere durate differenti: con lo streaming live dei contenuti, la durata del segmento successivo può essere notificata con la consegna del segmento attuale;

- la possibilità di selezionare uno specifico stream e passare da uno all'altro per lo stesso contenuto multimediale: il Media Presentation Description fornisce infatti tutte le informazioni necessarie al client per far sì che sia possibile selezionare uno stream audio fra diverse lingue, uno stream video fra diverse angolature della videocamera, i sottotitoli in una lingua a scelta fra quelle fornite, e così via;

- il supporto di *SVC - Scalable Video Coding* ed *MVC - Multiview Video Coding*: il MPD fornisce tutte le informazioni necessarie riguardanti le dipendenze circa il deconding fra le rappresentazioni, che possono essere usate per il download di stream multilayer, come SVC e MVC;

- la possibilità di inserire, in modo flessibile, una serie di descrittori: essi possono classificare il contenuto, specificare il ruolo dei componenti, descrivere caratteristiche di accessibilità, elencare le viste della telecamera, le modalità di impacchettamento dei frame e la configurazione dei canali audio;

- la possibilità di implementare metriche per il controllo della qualità, al fine di ricevere feed circa l'esperienza sulla sessione: lo standard MPEG-DASH include infatti una serie di metriche di qualità ben definite che consentono all'utente di effettuare misurarazioni e segnalazioni ad un reporting server adibito a tale scopo;

- la possibilità per il client di controllare il proprio *clock-drift* grazie all'inserimento di timestamp UTC all'interno dei segmenti;

- l'aggiunta di inserzioni pubblicitarie: gli annunci pubblicitari possono essere inseriti come period fra i vari periodi o come segmenti fra i vari segmenti, sia su richiesta che on-demand.

Anche la **sicurezza** è un aspetto importante di questo protocollo.
In combinazione con la standardizzazione di MPEG-DASH, il working group MPEG sta sviluppando uno standard comune per la crittografia, *ISO/IEC 23001-7* [62] [63] [64], che definisce uno schema comune per la crittografia dei contenuti multimediali. Utilizzando questo standard, il contenuto può essere crittografato una volta sola e trasmesso in streaming ai client, nonostante questi supportino diversi sistemi DRM - Digital Rights Management per la gestione delle licenze. In particolare, ogni client riceve le chiavi con le quali effettuare le operazioni di decodifica, così come tutte le informazioni extra di cui lo specifico sistema DRM in uso (segnalato nel MPD) necessita. Dopodiché può iniziare lo streaming del contenuto multimediale comunemente crittografato dal server.

4.3 Euristiche di Adattamento

Come detto nella sezione precedente, il protocollo *MPEG-DASH* definisce uno *standard* per lo *streaming adattivo on-demand* di contenuti multimediali su *protocollo HTTP* in grado di garantire interoperabilità fra client e server standard-based di diversi fornitori del servizio.
Il problema nasce dal fatto che le soluzioni precedenti, le quali basano la distribuzione dei propri contenuti sfruttando proprio questa tecnologia, quali Microsoft Smooth Streaming, Apple HLS ed Adobe HDS, basandosi su protocolli proprietari, definiscono un proprio formato sia per i *manifest* che per i *segmenti* nei quali il contenuto multimediale è suddiviso. Il protocollo MPEG-DASH si occupa di standardizzare i formati di queste due entità, stabilendo quale formato manifest e segmenti devono avere.
Come detto precedentemente, però, vi sono alcune cose che il protocollo non definisce in quanto non inquadrate nei propri scopi. Si tratta, ad esempio, del metodo di consegna del *MPD* (il documento che descrive il manifest), le tipologie di formati per la codifica dei media e dei segmenti, il comportamento del client per il fetching e la riproduzione dei contentui, ed altri ancora.

Uno dei comportamenti non definiti all'interno dello standard è appunto l'agire del client in relazione ai cambiamenti della banda di rete ed ai livelli del buffer del player. Si tratta delle cosiddette **euristiche di adattamento** o **euristiche di controllo**, algoritmi che indicano la qualità del prossimo segmento da scaricare in modo che tale misura sia massimizzata, minimizzando, al contempo, il verificarsi di interruzioni del servizio.

4.3.1 Euristiche Throughput-based

Nel corso degli anni sono stati proposti vari algoritmi di adattamento che utilizzano il protocollo MPEG-DASH con l'obiettivo di adattare la bitrate del media richiesto dal client alle condizioni della rete.
Una delle metodologie più semplici per l'implementazione di un'euristica di adattamento è, ad esempio, il download del prossimo segmento ad una qualità tale per cui la propria bitrate sia minore o uguale al throughput attuale della rete. Possiamo chiamare questo algoritmo **istantaneous throughput**.
Va da sé che questa metodica presenta numerose problematiche, soprattutto se la bandwidth non è stabile come nel caso della mobilità:

- se il rilevamento del throughput attuale avviene in un momento "favorevole", ovvero se il valore registrato risulta particolarmente alto, l'euristica indicherà il download del segmento con una qualità molto alta. La banda disponibile può però diminuire provocando numerose interruzioni del servizio;

- al contrario, se la misurazione del throughput avviene in un momento nel quale la copertura di rete non è ottimale, esso risulterà basso. L'algoritmo inidcherà così il download del prossimo segmento ad una scarsa qualità. Se questa situazione era però solamente temporanea, ecco che l'utente vedrà una buona copertura di rete ma il filmato trasmesso sul proprio device risulterà di qualità inappropriata.

Entrambe le situazioni descritte provocano nell'utente una QoE molto bassa. Risulta dunque necessario implementare meglio le politiche di misurazione

del throughput al fine di scaricare il prossimo segmento ad una qualità più appropriata. L'euristica di adattamento, infatti, "fallisce" sia nel caso in cui proponga il download di un segmento ad una qualità troppo alta, provocando dunque stalli, sia quando propone il download di un segmento ad una qualità troppo bassa. Essa infatti deve mediare proponendo la *miglior qualità possibile* con l'occorrenza del *minor numero di stalli possibile*.

I progettisti del protocollo **Adobe OSMF - Open Source Media Framework** [65] hanno capito che una misurazione istantanea del throughput non è ottimale. Essi stimano infatti la larghezza di banda disponibile considerando il tempo di download degli ultimi due segmenti video. La bitrate del prossimo segmento video da scaricare sarà uguale alla miglior codifica possibile fra quelle applicabili per la larghezza di banda stimata.
Così facendo si iniziano a mitigare gli effetti negativi derivanti da campionamenti durante periodi corrispondenti a massimi e minimi locali, i quali portano a sovrastimare o sottostimare la larghezza di banda realmente disponibile. Possiamo classificare questa tipologia di algoritmi con il nome di **smoothed throughput**. Essi stimano il valore di throughput mediandolo lungo un lasso di tempo più o meno lungo.

Esistono poi algoritmi che tentano di stimare la qualità del prossimo segmento da scaricare non solo cercando di evitare il verificarsi di stalli, ma facendo sì che anche i *ritardi iniziali* non si presentino. È il caso di **AAASH - Adaptation Algorithm for Adaptive Streaming over HTTP** [66]. Esso adatta la velocità video richiesta dal client utilizzando un complesso meccanismo di adattamento basato su più condizioni e parametri di configurazione. L'algoritmo seleziona la rappresentazione a più bassa qualità possibile per il primo segmento da scaricare. Così facendo riduce al massimo il tempo di attesa che intercorre tra l'avvio del servizio e l'effettivo inizio della riproduzione del media. Questa fase è nota con il nome di *fast start - avvio veloce* o *avvio rapido*. La qualità del video sarà poi aumentata, provocando il passaggio alla fase successiva, nel caso in cui una delle seguenti condizioni sia soddisfatta:

- è stata raggiunta la risoluzione video più alta;
- il livello del buffer non è aumentato in maniera monotòna;
- la bitrate relativa alla qualità del segmento è prossima al valore del throughput stimato.

4.3.2 Euristiche Buffer-based

Oltre agli algoritmi che basano la propria scelta circa la qualità del prossimo segmento da scaricare sul throughput, sia esso istantaneo o mediato su un determinato intervallo di tempo, esistono algoritmi che tengono in considerazione anche altri parametri, come il *livello del buffer*. Sono i cosiddetti algoritmi **buffer-based**, di cui due esempi sono rappresentati da **SVAA - Smooth Video Adaptation Algorithm** [67] e **RAASH - Rate Adaptation for Adaptive HTTP Streaming** [68].
Il primo stima la velocità con la quale il successivo segmento sarà scaricato utilizzando l'attuale tempo di buffer del client. L'algoritmo tenta di adattare la qualità del video scaricato al throughput di rete disponibile mentre applica un tetto massimo al buffer per evitare overflow.
Il secondo valuta la larghezza di banda della rete confrontando il tempo necessario al fetching del segmento ed il tempo di riproduzione sul client. Viene determinato un tempo di inattività per controllare le richieste di segmenti e limitare il tempo di buffering ad un livello massimo. Sono inoltre usati un *metodo di switch-up* graduale ed un *metodo di switch-down* aggressivo per regolare la bitrate di download in relazione a throughput disponibile.

4.3.3 FDASH

Un algoritmo che utilizza la **logica fuzzy** per controllare il tempo di buffering e la qualità dei segmenti al fine di distribuire un contenuto della migliore qualità possibile fornendo una riproduzione video senza interruzioni ed evitando frequenti cambi della risoluzione è **FDASH** [69] [70] [71].

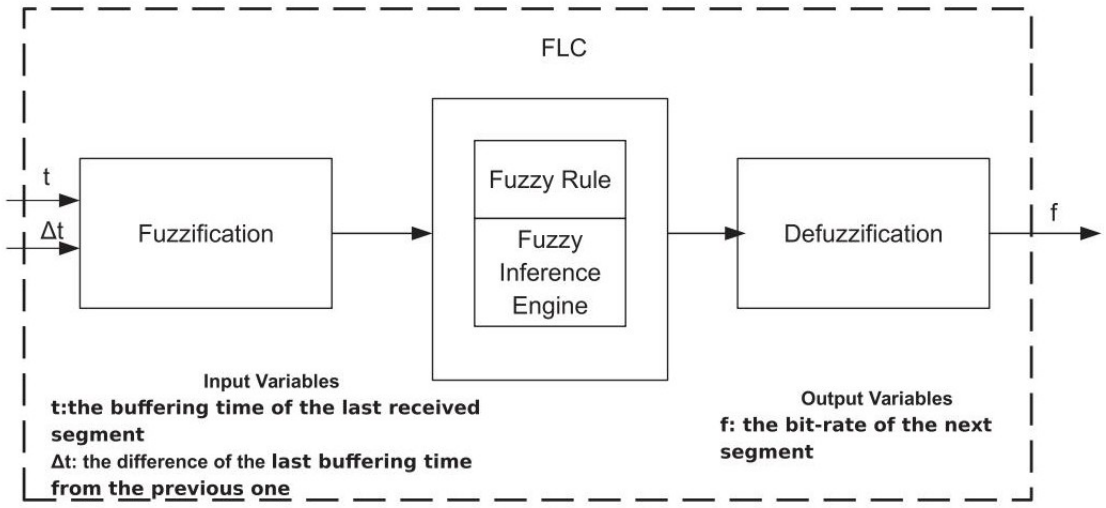

Figura 4.8: Struttura del FLC [71]

Uno degli elementi più importanti nella logica fuzzy è il *FLC - Fuzzy Logic Controller*, il quale consiste dei seguenti elementi, così come mostrato anche in Figura 4.8:

- *fuzzification*: durante questo processo, ogni elemento appartenente ai dati in input viene convertito in "gradi di appartenenza" mediante una ricerca in una o più funzioni di appartenenza [72];
- *fuzzy rule*: una semplice regola di tipo "if-then" con una condizione ed una conclusione. Associa il fuzzy-input al fuzzy-output ed è costruita per controllare il fuzzy-output. Nel caso in cui la regola sia costituita da più parti, specifici operatori fuzzy devono essere usati per combinare più input;
- *fuzzy interference engine*: attua il processo di interferenza fuzzy, computando il grado di attivazione e l'output di ogni regola;
- *defuzzification*: il set fuzzy viene trasformato in un set crisp. I cinque metodi di defuzzificazione sono: il centroid, il bisector, il middle of maximum, il largest of maximum e lo smallest of maximum.

Per quanto riguarda FDASH, nel controller le variabili di input sono:

- il tempo di buffering t_i che l'ultimo segmento ricevuto i attende sul client fino a quando non ne inizia la riproduzione;
- la differenza $\Delta t_i = t_i - t_{i-1}$ dell'ultimo tempo di buffer dal precedente.

Sono state adottate tre variabili linguistiche per classificare il tempo di buffering al fine di descrivere la distanza tra il tempo di buffering attuale da un tempo di buffering target T, fissato in FDASH per evitare lo svuotamento del buffer e mantenere vicina allo zero la differenza fra la risoluzione attuale e quella precedente, così da ridurre i continui cambi di risoluzione:

- S: short;
- C: close;
- L: long.

Per descrivere il comportamento della frequenza tra i successivi tempi di buffering, vengono considerate le seguenti variabili linguistiche:

- F: falling;
- S: steady;
- R: rising.

In modo simile, l'output di FLC rappresenta un fattore di aumento/diminuzione della risoluzione del segmento successivo, le variabili dell'output dunque sono:

- R: reduce;
- SR: small reduce;
- NC: no change;
- SI: small increase;
- I: increase.

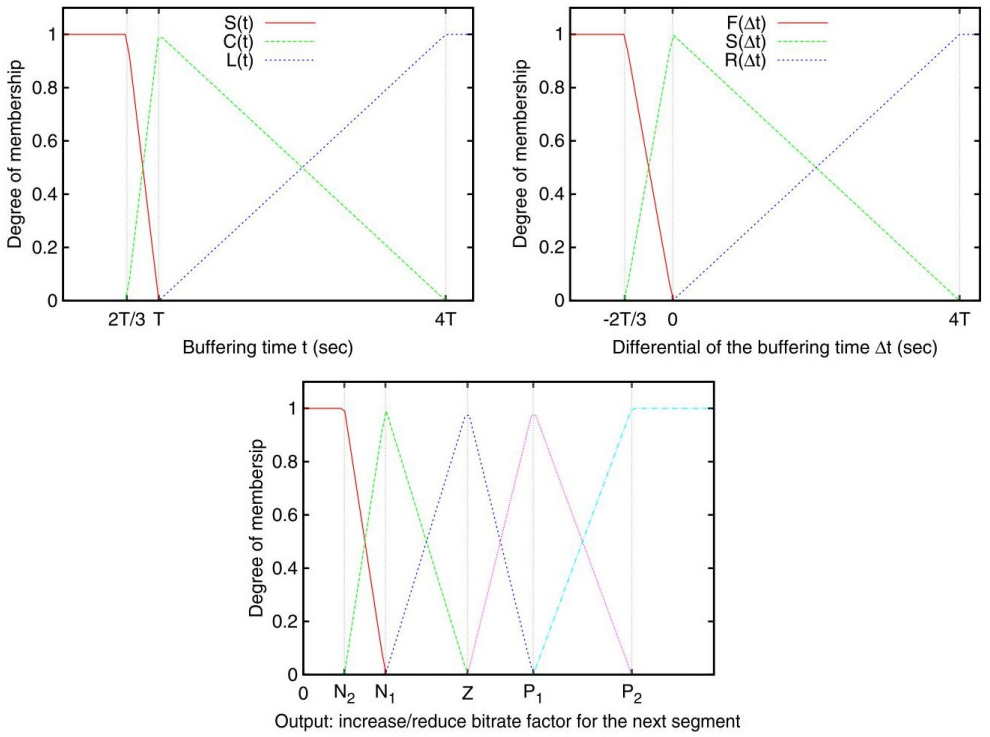

Figura 4.9: Funzioni di appartenenza delle variabili di I/O [71]

Figura 4.9 illustra le funzioni di appartenenza delle variabili di input e output.

Le regole "if-then" per il controller sono le seguenti:

- Rule 1 (r1): if (short) and (falling) then R;
- Rule 2 (r2): if (close) and (falling) then SR;
- Rule 3 (r3): if (long) and (falling) then NC;
- Rule 4 (r4): if (short) and (steady) then SR;
- Rule 5 (r5): if (close) and (steady) then NC;
- Rule 6 (r6): if (long) and (steady) then SI;
- Rule 7 (r7): if (short) and (rising) then NC;

- Rule 8 (r8): if (close) and (rising) then SI;

- Rule 9 (r9): if (long) and (rising) then I.

Il valore di ogni regola viene calcolato come il valore minimo tra le due funzioni di input che lo compongono.
Infine, per la fase di defuzzificazion, viene utilizzato il metodo centroid (o del baricentro). L'uscita f viene pertanto calcolata come:

$$f = \frac{N_2 \times R + N_1 \times SR + Z \times NC + P_1 \times SI + P_2 \times I}{SR + R + NC + SI + I}$$

dove:

$$I = \sqrt{r_9^2}$$

$$SI = \sqrt{r_6^2 + r_8^2}$$

$$NC = \sqrt{r_3^2 + r_5^2 + r_7^2}$$

$$SR = \sqrt{r_2^2 + r_4^2}$$

$$R = \sqrt{r_1^2}$$

Per il modello utilizzato nelle simulazioni, avvenute su *simulatore ns-3*, si è presupposto che un flusso video disponibile su un server fosse costituito da n segmenti di durata τ. Ogni segmento è codificato in più risoluzioni di qualità.

Il throughput dell'i -esimo segmento è stimato al cliente come:

$$r_i = (b_i \times \tau)/(d_i - r_i)$$

dove b_i, d_i ed r_i denotano, rispettivamente, la bitrate del segmento i, il tempo al quale il segmento i ha iniziato ad essere scaricato ed il tempo al quale l'intero segmento i è stato ricevuto dal client.
Ogni client richiede il prossimo segmento, la cui risoluzione deve essere determinata. Questa operazione è effettuata dal FLC, il cui output è un fattore che si riferisce alla bitrate (alla risoluzione) del segmento successivo, cioè b_{i+1}, in relazione al throughput del canale stimato nell'ultimo periodo, ovvero:

$$b_{i+1} = f \times r_d$$

dove r_d denota il throughput disponibile, stimato come media dei throughput degli utlimi k segmenti scaricati durante un determinato lasso di tempo d.
Di conseguenza, il throughput è dato da:

$$r_d = 1/k \times \sum_{i=1}^{k} r_i$$

Inoltre, al fine di evitare fluttuazioni non necessarie nella bitrate, sono state applicate le seguenti politiche:

- se $b_n > b_i$ e selezionando la nuova bitrate b_{i+1}, il livello del buffer è stimato essere meno di T per i prossimi 60 secondi, allora la bitrate rimane invariata;
- se $b_n < b_i$ ma la vecchia bitrate è stimata produrre un livello del buffer per i prossimi 60 secondi che è maggiore di T, allora la bitrate rimane invariata;
- in tutti gli altri casi la bitrate viene settata a b_n.

Come detto precedentemente, le simulazioni sono state fatte su *ns-3* e sono stati valutati diversi parametri, fra cui: il numero di interruzioni del servizio (stalli), il numero di cambiamenti della risoluzione, la qualità media ed il livello del buffer. FDASH è stato confrontato con altri algoritmi di rate adaptation su HTTP, fra cui OSMF, AAASH, SVAA, e RAASH.

Algorithm	Parameters	Value	Definition
FDASH	T	35 sec	Target buffering time
	d	60 sec	Time period estimating the connection throughput
	(N_2, N_1, Z, P_1, P_2)	(0.25, 0.5, 1, 1.5 ,2)	Factors of the output membership functions
AAASH	B_{min}	10 sec	Minimum buffer level
	B_{tar}	$[20, 50]$ sec	Target buffer level interval
	D_β	1 sec	Discretization parameter for the buffer level
	δ_t	10 sec	Average throughput estimation period
	$(a1, .., a5)$	(0.75, 0.33, 0.5, 0.79, 0.9)	Safety margins
SFTM	bmt_{min}	20 sec	Idle period between consecutive requests
	$TBMT$	20 sec	Target buffered media time
	ρ	0.75	Remaining segment fetch time priority factor
SVAA	q_{ref}	20 sec	Target buffer size
	m	dynamic-m function	Responsiveness parameter
	q_{max}	20 sec	The buffer cap
	ρ_υ	$0,05$	Video rate reduction margin
	W	10 segments	Window of the last downloaded segments
RAAHS	γ_d	0.67	Switch down threshold
	t_{min}	9 sec	Minimum buffered media time

Figura 4.10: Parametri di simulazione [71]

Il simulatore è stato configurato con vari nodi che agiscono da client MPEG-DASH ed un nodo che simula il comportamento di un server MPEG-DASH. Ogni nodo client scarica dal server il media alle varie risoluzioni usando gli algoritmi sopra citati. I parametri usati per la simulazione sono riportati in Figura 4.10.

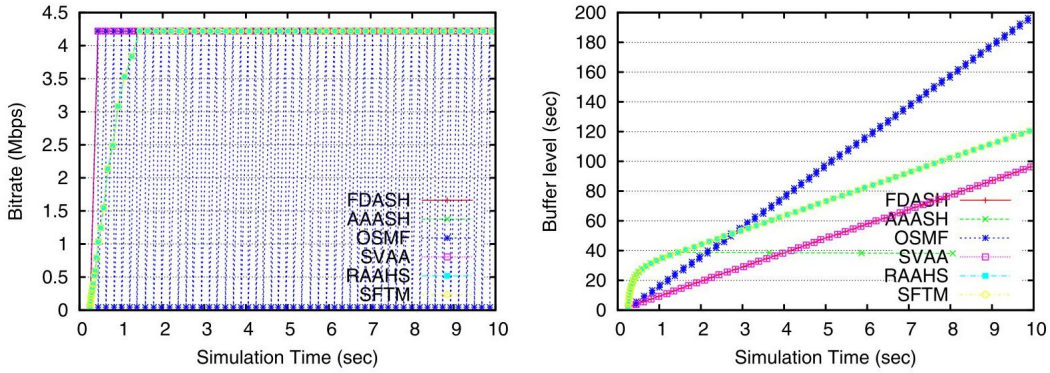

Figura 4.11: Un client, bandwidth illimitata [71]

Il primo scenario è consistino nella simulazione di un singolo client per euristica che agisce in un ambiente dove la bandwidth è illimitata. Per quanto riguarda la bitrate, sia FDASH che SVAA raggiungono la qualità massima in poco meno di 0.3 secondi. Tutti gli algoritmi incrementano il proprio livello di buffering time al di sopra del limite target dopo un breve periodo iniziale della durata di 2 secondi, senza il verificarsi di alcuno stallo. OSMF compie oscillazioni coninue fra la risoluzione più alta e quella più bassa a causa del fatto che l'algoritmo esegue l'adattamento della velocità considerando solo la stima del throughput disponibile ed ignora il tempo di buffer disponibile sul client. I risultati sono riportati in Figura 4.11.

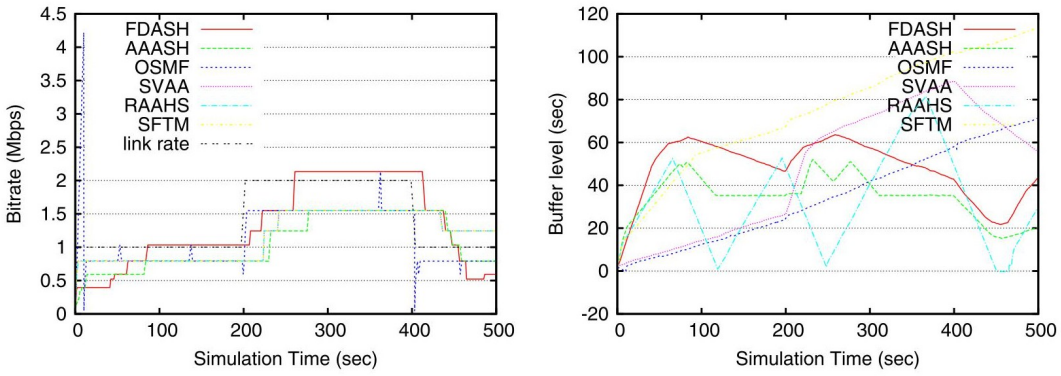

Figura 4.12: Un client, cambiamento costante nella bandwidth [71]

Algoritmo	Video rate media (Kbps)	Interruzioni	Cambi di risoluzione
FDASH	1206.350	0	15
AASH	967.282	0	16
OSMF	1153.380	65	21
SVAA	1107.930	0	5
RAASH	735.868	286	21
SFTM	1062.790	0	16

Tabella 4.1: Un client, cambiamento costante nella bandwidth [71]

Nel secondo scenario è sempre stato fatto agire un solo client per algoritmo ma, questa volta, il throughput ha subito cambiamenti costanti. Durante i primi 200 secondi è stato mantenuto il valore di 2 Mb/s, dopodiché si è passati ad 1 Mb/s per altri 200 s, sino a tornare ad un valore di 2 Mb/s per gli ultimi 100 s. La risoluzione media, il numero di interruzioni ed il numero di volte che la risoluzione è stata modificata è riportato in Tabella 4.1.
Come è possibile vedere anche da Figura 4.12, la risoluzione video fornita da FDASH risulta essere maggiore rispetto a quelle degli altri algoritmi. OSMF si posiziona al secondo posto, ma comporta ben 65 stalli. Come è poi possibile notare, FDASH presenta un ottimo meccanismo di gestione del buffer.

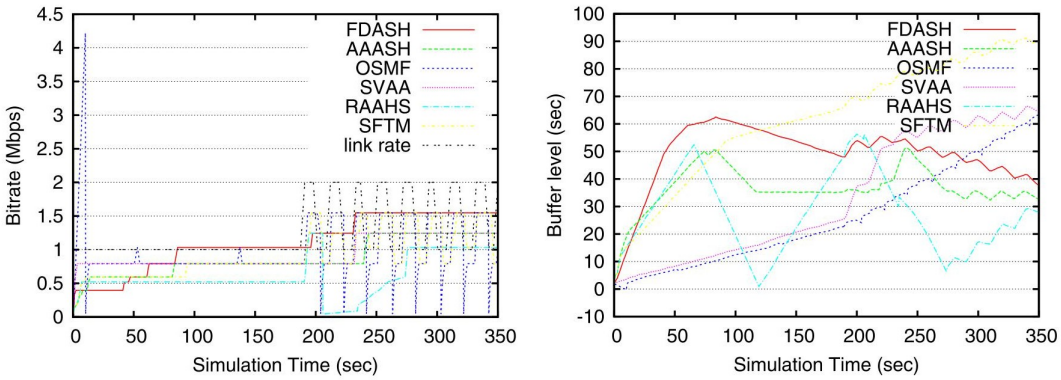

Figura 4.13: Un client, cambiamenti a breve termine nella bandwidth [71]

Algoritmo	Video rate media (Kbps)	Interruzioni	Cambi di risoluzione
FDASH	1069.990	0	10
AASH	907.584	0	12
OSMF	971.918	65	86
SVAA	981.259	0	3
RAASH	787.654	0	25
SFTM	904.044	0	66

Tabella 4.2: Un client, cambiamenti a breve termine nella bandwidth [71]

Nel terzo scenario di simulazione, è sempre lasciato agire un singolo client ma, questa volta, il livello di throughput ha subito variazioni periodiche a breve termine. Esso infatti alterna picchi e cadute della durata di circa 10 secondi. In Figura 4.13 è possibile vedere l'andamento della qualità dei segmenti video scaricati e del livello del buffer mentre, in Tabella 4.2 sono riportati i valori numerici relativi alla bitrate media, al numero di stalli ed al numero di variazioni della risoluzione per ciascuna delle euristiche utilizzate. Anche per una volta, FDASH risulta l'algoritmo che fornisce la bitrate media più elevata, evitando stalli e cambi frequenti non necessari nella risoluzione.

Sono poi stati configurati altri scenari ed, in particolare:

- un client per algoritmo su una rete WiFi con traffico TCP in background: il traffico di sottofondo è generato da 5 flussi TCP;
- più client per algoritmo su una rete a 2 Mb/s: per ogni algoritmo sono usati due client che competono per l'accesso al server;
- più client per algoritmo su una rete WiFi con traffico di sottofondo: è stato replicato lo scenario precedente con 3 flussi TCP in background.

I risultati sono riportati in [71] e tutti mostrano come FDASH spicchi in quanto a bitrate fornita evitando, al contempo, stalli e modifiche non necessarie della risoluzione.

4.3.4 BOLA

Una menzione particolare nel panorama delle euristiche di controllo la merita sicuramente **BOLA - Buffer Occupancy based Lyapunov Algorithm** [73], i cui autori hanno dimostrato avere un'utilità media che rientra in un termine additivo $O(1/V)$ del valore ottimale, per un parametro di controllo V correlato alla dimensione del buffer video. Essi hanno mostrato come BOLA raggiunga un'utilità quasi ottimale, in molti casi significativamente più alta rispetto agli algoritmi di ultima generazione.
Come abbiamo visto nelle sezioni precedenti, la maggior parte degli algoritmi implementati nella pratica utilizza un approccio basato sulla larghezza di banda, *throughput-based* o *bandwidth-based*, in cui il throughput tra il server ed il video player viene stimato ed è utlizzato per determinare la bitrate del prossimo chunk che deve essere scaricato. Un approccio complementare è quello basato sul buffer, *buffer-based*, che non cerca di prevedere la larghezza di banda ma utilizza come parametro decisionale solamente la quantità di dati attualmente archiviati nel buffer del lettore video.
Recentemente sono emerse prove empiriche di come un approccio basato su buffer abbia proprietà migliori rispetto agli approcci basati sulla larghezza di banda. Un algoritmo buffer-based ha infatti proprietà che mancano in approcci throughput-based. È per questo motivo che *Netflix*, ad esempio, ha recentemente adottato euristiche che basano le proprie decisioni in relazione a variabili buffer-related [74].
Un risultato interessante di BOLA è come esso si comporti in maniera *quasi ottimale* e richieda solamente la *conoscenza della quantità di dati nel buffer*, senza alcuna stima della larghezza di banda disponibile. Pertanto, esso fornisce la prima giustificazione teorica del perché gli algoritmi basati su buffer si comportano bene nella pratica ed aggiunge nuovi approfondimenti al dibattito in corso all'interno delle comunità degli standard DASH e di video streaming circa l'efficacia dei due approcci [75]. Inoltre, poiché BOLA è basato su buffer, sono evitati gli overhead relativi alle complesse previsioni della larghezza di banda present nelle attuali implementazioni dei video player,

risultando inoltre più stabile in relazione alle fluttuazioni del throughput. I risultati ottenuti dagli autori implicano che il livello di buffer è una statistica sufficiente, la quale fornisce indirettamente tutte le informazioni sulle variazioni passate della larghezza di banda, informazioni richieste per la scelta della bitrate del prossimo segmento da scricare.
Con BOLA gli autori hanno formulato il problema dell'adattamento della bitrate come un **problema di massimizzazione dell'utilità** che incorpora entrambi i fattori chiave legati alla QoE:

- la bitrate media sperimentata dall'utente durante la visione del video: un aumento della bitrate media aumenta l'utilità;
- la durata degli eventi di re-buffering: un evento di re-buffering diminuisce il valore di utilità.

Un punto di forza di questo framework è che l'utilità può essere definita in modi arbitrari, ad esempio, a seconda del contenuto, del provider dei video o del dispositivo dell'utente. Ciò contrasta con gli algoritmi di adattamento attualmente in uso, i quali non offrono tale flessibilità.

BOLA consiste in un **algoritmo di bitrate adaptation online** derivato dal **metodo di ottimizzazione di Lyapunov** ed è il primo a fornire una garanzia teorica sull'utilità raggiunta.
Il primo passo svolto nella definizione dell'algoritmo è la formalizzazione del **modello video**. Il file video è suddiviso in N chunk indicizzati come $\{1, 2, \ldots, N\}$ dove ogni chunk rappresenta p secondi del video. Sul server ogni segmento è disponibile in M diverse bitrate dove, maggiore è la qualità, maggiore sarà il "peso" del segmento, migliore sarà l'esperienza dell'utente e maggiore sarà l'utilità associata. La dimensione in bit di un chunk codificato con la bitrate di indice m è S_m e l'utilità derivata dalla sua visualizzazione da parte dell'utente è data da v_m, dove $m \in \{1, 2, \ldots, M\}$. Vale dunque quanto segue:

$$v_1 \geq v_2 \geq \cdots \geq v_M \iff S_1 \geq S_2 \geq \cdots \geq S_M$$

Il secondo step atto alla definizione dell'algoritmo è la formalizzazione del **modello del video player**. Il lettore video scarica successivamente i chunk del file video dal server e ne riproduce il contenuto. Ogni segmento deve essere scaricato nella sua interezza prima di poter essere riprodotto ed i chunk vengono scaricati nello stesso ordine in cui sono riprodotti. Il video player ha un buffer finito di dimensioni Q_{max} chunk per l'archiviazione dei segmenti scaricati ma ancora da riprodurre. Misurare il buffer in chunk equivale a misurarlo in secondi poiché la durata di ogni blocco è fissa: p. Se il buffer è pieno, il lettore non può scaricare nuovi blocchi ed attende un periodo di tempo prestabilito, espresso in secondi, prima di tentare di scaricare un nuovo blocco. I blocchi che sono stati scaricati completamente vengono riprodotti a una frequenza fissa di $1/p$ pezzi/secondo senza alcuna attesa. Con l'invio di una richiesta di download per un nuovo chunk, il player specifica anche la bitrate desiderata per quel chunk. Ciò consente al player di bilanciare la qualità video complessiva con la probabilità di re-buffering che si verifica quando non ci sono blocchi nel buffer per la riproduzione.
Si procede poi con la definizione del **modello di rete**. Si presume che la larghezza di banda disponibile (in bit/secondo) tra il server ed il player sito sul client, vari continuamente nel tempo in base ad un processo casuale stazionario $\omega(t)$. Non sono fatte alcune ipotesi circa la conoscenza delle proprietà statistiche o della distribuzione di probabilità di $\omega(t)$, eccetto per il fatto che ha un primo ed un secondo momento finiti ed un secondo momento inverso anch'esso finito. Supponendo che il video player al tempo t inizi a scaricare un chunck la cui bitrate abbia un indice m, allora il tempo t' al termine del download soddisfa quanto segue:

$$S_m = \int_t^{t'} \omega(\tau) d\tau$$

$$Let \;\; \mathbb{E}\{\omega(t)\} = \omega_{avg}. \;\; Then \;\; \mathbb{E}\{t' - t\} = S_m/\omega_{avg}.$$

Si passa poi alla **formulazione del problema**. Si considerano due parametri prestazionali primari che influenzano la QoE complessiva dell'utente:

- qualità media di riproduzione nel tempo: è una funzione della bitrate dei blocchi visualizzati dall'utente;
- frazione del tempo trascorso a non ri-bufferizzare: può essere inteso come una misura di fluidità media relativa alla riproduzione.

Il primo parametro è indicato dal simbolo $\overline{v}_N$ mentre il secondo parametro è indicato con il simbolo $\overline{s}_N$. Seguono una serie di formalizzazioni matematiche che non saranno riportate in quanto non inerenti agli obiettivi fissati per questa Tesi di Laurea.
Sono poi definiti gli **obiettivi** legati alle performance dell'algoritmo, ovvero, ci si pone l'obiettivo di progettare un algoritmo di controllo che massimizzi la *misura di utilità* relativa a:

$$\overline{v}_N + \gamma \overline{s}_N$$

soggetta ai vincoli del modello, dove $\gamma > 0$ definisce un peso che assegna una determinata priorità alla fluidità di riproduzione. Questo problema può essere formulato come un *problema di ottimizzazione stocastica* dove possono essere usati approcci basati sulla programmazione dinamica (DP) per risolverlo [76]. Tuttavia, i metodi tradizionali basati su DP presentano due grossi svantaggi:

- innanzitutto, richiedono la conoscenza della distribuzione del processo $\omega(t)$ che potrebbe essere difficile da ottenere;
- in secondo luogo, anche quando tale conoscenza è disponibile, la DP risultante può avere uno spazio degli stati molto grande.

Questi due fattori portano ad una fase di **rilassamento nella formulazione del problema**. Saranno ora riportate, per completezza, le definizioni formali

dei parametri $\overline{v}$ ed $\overline{s}$ anche se, come detto precedentemente, si sottolinea come esse esulino dagli scopi di questa Tesi di Laurea:

$$\overline{v} \triangleq \lim_{N\to\infty} \overline{v}_N = \frac{\lim_{N\to\infty} \mathbb{E}\{\frac{1}{K_N} \sum_{k=1}^{K_N} \sum_{m=1}^{M} a_m(t_k) v_m\}}{\lim_{N\to\infty} \mathbb{E}\{\frac{1}{K_N} \sum_{k=1}^{K_N} T_k\}}$$

$$\overline{s} \triangleq \lim_{N\to\infty} \overline{s}_N = \frac{\lim_{N\to\infty} \mathbb{E}\{\frac{1}{K_N} \sum_{k=1}^{K_N} \sum_{m=1}^{M} a_m(t_k) p\}}{\lim_{N\to\infty} \mathbb{E}\{\frac{1}{K_N} \sum_{k=1}^{K_N} T_k\}}$$

Infine è possibile utilizzare il metodo noto con il nome di **Lyapunov optimization-over-renewal-frames** [77] per derivare un algoritmo in grado di ottimizzare le metriche definite dalle due equazioni precedenti e dalla seguente equazione:

$$Q(t_{k+1}) = max[Q(t_k) - \frac{T_k}{p}, 0] + \sum_{m=1}^{M} a_m(t_k)$$

la quale formalizza come il livello del buffer $Q(t_{k+1})$ venga aggiornato all'inizio di ogni slot.

IMPORTANTE: In quanto, come già ribadito, non rientra fra gli obiettivi di questa Tesi di Laurea un'analisi circa il modello matematico che sta dietro la definizione di questo algoritmo il quale è stato citato solamente al fine di indicare come sia stata realizzata un'euristica di adattamento che opera in modo quasi ottimale, saranno riportati ora solamente un paio di grafici che ne mostrano l'efficacia.

BOLA è stato fatto operare scaricando un media della durata complessiva di 99 secondi suddiviso in segmenti da 3 secondi l'uno. Figura 4.14 mostra le scelte circa la bitrate in funzione del livello del buffer. Figura 4.15 e Figura 4.16 mostrano, rispettivamente, la bitrate dei chunk scaricati e le variazioni del livello del buffer.

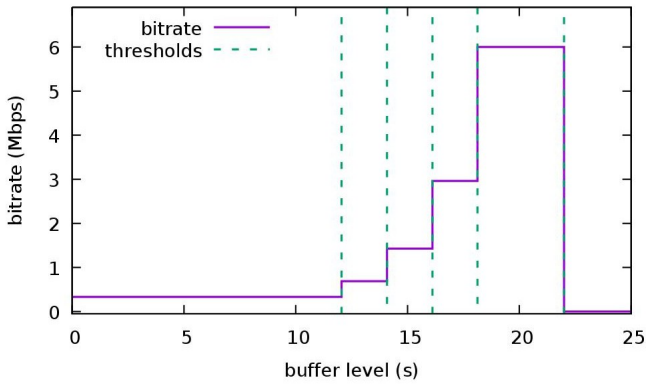

Figura 4.14: Scelte della bitrate come funzione del livello del buffer [73]

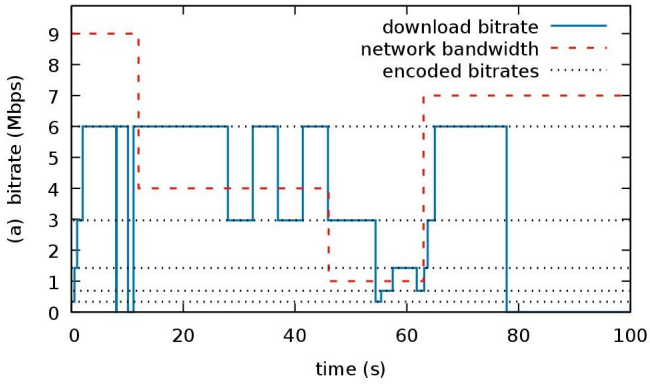

Figura 4.15: Bitrate dei segmenti scaricati durante le simulazioni di BOLA [73]

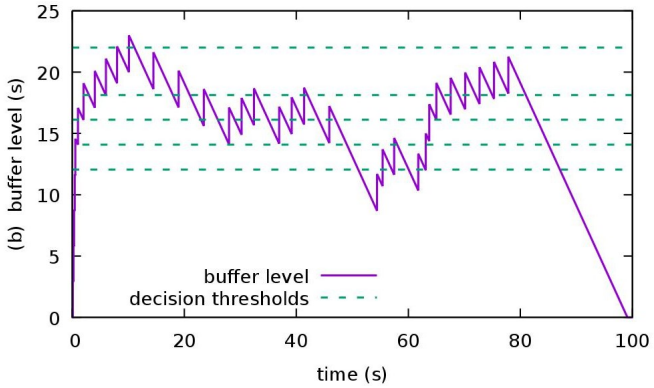

Figura 4.16: Livello del buffer durante le simulazioni di BOLA [73]

Capitolo 5

Conclusioni

Come abbiamo visto, sono state numerose le scelte che hanno portato all'affermarsi dell'attuale stato dell'arte nell'ambito del video streaming adattivo basato su HTTP e tutte hanno avuto in comune un'unica linea guida: la massimizzazione della **QoE - Quality of Experience** sperimentata dagli utenti finali di servizi ed applicazioni basate su questa tecnologia.
Prima di tutto, grazie all'aumento della banda disponibile ed alla crescita del World Wide Web, si è affermato lo streaming su **HTTP**. L'infrastruttura di Internet si è sviluppata per supportare efficacemente questo protocollo prevedendo, ad esempio, il dislocamento di cache allo scopo di ridurre il traffico a lungo raggio. Ulteriori motivazioni sono rappresentate, ad esempio, dal fatto che HTTP è firewall-friendly, a differenza di altri protocolli, come RTP.
Per quanto riguarda il protocollo di livello trasporto, invece, fra UDP e **TCP** si è affermato quest'ultimo. Anche in questo caso la scelta è stata guidata da ricerche sulla QoE percepita dagli utenti finali. UDP, infatti, in caso di cattive condizioni di rete o nello specifico ambito della mobilità, può causare perdita di pacchetti o l'arrivo degli stessi in un ordine diverso da quello di invio. Questo provoca una degradazione nella qualità video che risulta essere peggio tollerata dagli utenti rispetto a **fattori di degradazione temporale**, tipici di TCP. Nonostante sia i fattori di degradazione che occorrono in UDP (packet loss ratio) sia quelli che occorrono in TCP (numero e frequenza

degli stalli) influenzino in modo esponenziale la QoE (IQX hypothesis), si vede come, a parità di condizioni di rete, gli effetti generati dall'uso di TCP risultano essere meglio tollerati.
Ulteriori studi hanno poi mostrato come, fra le due tipologie possibili di degradazione temporale quali i **ritardi iniziali** e le **pause intermedie**, gli utenti preferiscano di gran lunga sperimentare i primi, i quali influenzano la QoE in maniera logaritmica (WQL hypothesis) anziché esponenziale, come invece accade per gli **stalli**. Due importanti fattori usati per spiegare questi effetti sono i cosiddetti Serial Position Effect ed il Recency Effect, i quali agiscono sulla memoria a breve termine e sulla memoria di lavoro. Questa informazione può essere usata dai progettisiti di servizi di video streaming per implementare tecniche quali l'occorrenza di ritardi iniziali volontari al fine di evitare il verificarsi di stalli successivi (es: pre-buffering).
Investigando ulteriormente su quelli che sono i **fattori chiave di influenza sulla QoE** per quanto riguarda i pattern di stallo, si vede come sia il **numero di stalli** il parametro ad avere il peso maggiore sulla qualità percepita dagli utenti rispetto, ad esempio, al tempo complessivo di stallo.
Per quanto riguarda lo specifico ambito della **mobilità**, invece si è visto come ad occorrere siano **pattern di stallo non periodici**.
Tutte queste informazioni sono state aggregate e numerose aziende hanno implementato **soluzioni proprietarie** per la distribuzione di contenuti con la tecnologia del video streaming. Fra queste spiccano Microsoft, con il suo Smooth Streaming, Apple, con HTTP Live Streaming ed Adobe, con HDS - HTTP Dynamic Streaming. Studiando nel dettaglio queste tecnologie, si evince come, a grandi linee, la struttura di base sia sempre la stessa: è presente un file che descrive come è fatto il media sul server e sono presenti una serie di segmenti nei quali il media è suddiviso, codificati con differenti bitrate, le quali ne stabiliscono la qualità.
Al fine di permettere l'interoperabilità fra client e server di diversi fornitori, **MPEG** ha implementato uno standard: **DASH**. Esso permette a tutti i client standard-based di poter scaricare contenuti audio/video da qualsiasi

server standard-based. Il manifest è rappresentato dal MPD - Media Presentation Description ed i segmenti rappresentano gli effettivi bitstream multimediali sottoforma di chunks in uno o più file. La specifica MPEG-DASH definisce solamente il formato del MPD e dei segmenti. Il metodo di consegna del MPD, le tipologie di formati per la codifica del media e dei relativi segmenti, il comportamento del client per il fetching e la riproduzione dei contenuti, così come l'euristica di adattamento sono al di fuori degli scopi di questo standard. MPEG-DASH è uno standard ricco di feature, alcune delle quali includono la possibilità di avere un manifest compatto, così come la possibilità di poter frammentare lo stesso, la possibilità di poer specificare per ciascun segmento più url, la possibilità per i segmenti di avere diverse durate, e così via. Anche la sicurezza è un aspetto importante di questo protocollo. In combinazione con la standardizzazione di MPEG-DASH, il working group MPEG sta infatti sviluppando uno standard comune per la crittografia: ISO/IEC 23001-7, che definisce uno schema comune per la crittografia dei contenuti multimediali.

Per completare il punto della situazione circa lo streaming adattivo HTTP-based, mi sono poi occupato in questa Tesi di Laurea di descrivere in linea generale quello che è il panorama circa le **euristiche di adattamento**, ovvero gli algoritmi che guidano la scelta del prossimo segmento da scaricare in modo da massimizzare la qualità video complessiva e minimizzare, al contempo, il verificarsi di interruzioni del servizio.

Come abbiamo visto, esistono generalmente due approcci: un primo approccio **throughput-based**, dove la decisione circa la qualità del prossimo segmento da scaricare viene presa in relazione a campionamenti del throughput istantanei o mediati su un determinato arco temporale e **buffer-based**, dove invece la scelta circa la bitrate del prossimo segmento da scaricare viene presa esaminando solamente il livello del buffer del video player.

Sono poi descritti due algoritmi più complessi, FDASH e BOLA, in modo da mostrare come, attualmente, le euristiche di adattamento si siano evolute al fine rendere la misura relativa alla QoE più alta possibile. **FDASH** è

un algoritmo che utilizza la logica fuzzy e risulta migliore rispetto a molti semplici algoritmi throughput o buffer based. **BOLA**, invece, risulta essere un algoritmo quasi ottimo. Con esso, gli autori gli autori hanno formulato il problema dell'adattamento della bitrate come un problema di massimizzazione dell'utilità. BOLA consiste in un algoritmo di bitrate adaptation online derivato dal metodo di ottimizzazione di Lyapunov ed è il primo a fornire una garanzia teorica sull'utilità raggiunta.

In conclusione, abbiamo visto quelle che sono state le problematiche affrontate e le scelte che sono state prese sino ad arrivare all'attuale stato dell'arte, dove lo streaming di contenuti multimediali viene fatto su protocollo HTTP usando standard quali MPEG-DASH, guidati da euristiche di adattamento che tentano di massimizzare l'intera qualità del media in download minimizzando, al contempo, il numero di stalli che possono verificarsi, così come parametri secondari che possono influenzare in maniera negativa la QoE sperimentata dall'utente finale quali, ad esempio, il continuo cambiamento della risoluzione video a causa di variazioni nel throughput attuale.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] T. Cisco, "Cisco visual networking index: Global mobile data traffic forecast update." 2016-2021 White Paper, 2017.

[2] L. Plissonneau and E. Biersack, "A longitudinal view of http video streaming performance," in *Proceedings of the 3rd Multimedia Systems Conference*, pp. 203–214, ACM, 2012.

[3] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 362–373, ACM, 2011.

[4] M. Yan, "How does video streaming work? http://mingfeiy.com/adaptive-streaming-video-streaming."

[5] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[6] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, *et al.*, "Qualinet white paper on definitions of quality of experience," in *Output from the Fifth Qualinet Meeting*, 2013.

[7] M. Shiels, ""youtube at five- 2 bn views a day," 2010.

[8] T. Hoßfeld, R. Schatz, and U. R. Krieger, "Qoe of youtube video streaming for current internet transport protocols," in *Measurement,*

Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, pp. 136–150, Springer, 2014.

[9] I. T. Union, "Absolute category rating (acr) and mean opinion score (mos) definitions," *ITU-T P.800.2*, 7 2016.

[10] M. Hirth, T. Hoßfeld, and P. Tran-Gia, "Anatomy of a crowdsourcing platform-using the example of microworkers.com," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pp. 322–329, IEEE, 2011.

[11] K.-T. Chen, C.-J. Chang, C.-C. Wu, Y.-C. Chang, and C.-L. Lei, "Quadrant of euphoria: a crowdsourcing platform for qoe assessment," *IEEE Network*, vol. 24, no. 2, 2010.

[12] M. Hirth, T. Hoßfeld, and P. Tran-Gia, "Cost-optimal validation mechanisms and cheat-detection for crowdsourcing platforms," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, pp. 316–321, IEEE, 2011.

[13] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Crowdtesting: a novel methodology for subjective user studies and qoe evaluation," *University of Würzburg, Tech. Rep*, vol. 486, 2013.

[14] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *Multimedia (ISM), 2011 IEEE International Symposium on*, pp. 494–499, IEEE, 2011.

[15] S. Alcock and R. Nelson, "Application flow control in youtube video streams," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 24–30, 2011.

[16] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, p. 25, ACM, 2011.

[17] T. Hoßfeld, T. Zinner, R. Schatz, M. Seufert, and P. Tran-Gia, "Transport protocol influences on youtube qoe," *University of Würzburg, Tech. Rep*, vol. 482, 2011.

[18] J. Pallant, *SPSS survival manual*. McGraw-Hill Education (UK), 6 ed., 4 2016.

[19] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240–242, 1895.

[20] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, 2010.

[21] R. Schatz, T. Hoßfeld, and P. Casas, "Passive youtube qoe monitoring for isps," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pp. 358–364, IEEE, 2012.

[22] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "Yomo: A youtube application comfort monitoring tool," *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications, Tampere, Finland*, pp. 1–3, 2010.

[23] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "Aquarema in action: Improving the youtube qoe in wireless mesh networks," in *Internet Communications (BCFIC Riga), 2011 Baltic Congress on Future*, pp. 33–40, IEEE, 2011.

[24] B. Staehle, F. Wamser, M. Hirth, D. Stezenbach, and D. Staehle, "Aquareyoum: application-and quality of experience-aware resource management for youtube in wireless mesh networks," *PIK-praxis der Informationsverarbeitung und Kommunikation*, vol. 34, no. 3, pp. 144–148, 2011.

[25] F. Liers, T. Volkert, and A. Mitschele-Thiel, "Demonstrating forwarding on gates with first applications," *EuroView2010, Würzburg, Germany*, 2010.

[26] T. Hoßfeld, F. Liers, T. Volkert, and R. Schatz, "Fog and clouds: Optimizing qoe for youtube," *KuVS 5thGI/ITG KuVS Fachgespräch NG Service Delivery Platforms*, 2011.

[27] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth.," in *USITS*, vol. 1, pp. 11–11, 2001.

[28] T. En-Najjary and G. Urvoy-Keller, "Pprate: A passive capacity estimation tool," in *End-to-End Monitoring Techniques and Services, 2006 4th IEEE/IFIP Workshop on*, pp. 82–89, IEEE, 2006.

[29] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010.

[30] C. Lorentzen, M. Fiedler, and H. Johnson, "On user perception of safety in online social networks," *International Journal of Communication Networks and Distributed Systems*, vol. 11, no. 1, pp. 77–91, 2013.

[31] C. Lorentzen, M. Fiedler, H. Johnson, J. Shaikh, and I. Jørstad, "On user perception of web login—a study on qoe in the context of security," in *Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian*, pp. 84–89, IEEE, 2010.

[32] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, ""time is bandwidth"? narrowing the gap between subjective time perception and quali-

ty of experience," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 1325–1330, IEEE, 2012.

[33] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo, "The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment," in *Communications (ICC), 2010 IEEE International Conference on*, pp. 1–5, IEEE, 2010.

[34] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: Between the devil and the deep blue sea," in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*, pp. 1–6, IEEE, 2012.

[35] A. M. Coleman, "Oxford dictionary of psychology," *New York: Oxford UniversityPress*, p. 688, 2006.

[36] H. Ebbinghaus, *Memory: A contribution to experimental psychology*. No. 3, University Microfilms, 1913.

[37] J. Deese and R. A. Kaufman, "Serial effects in recall of unorganized and sequentially organized verbal material.," *Journal of experimental psychology*, vol. 54, no. 3, p. 180, 1957.

[38] B. B. Murdock Jr, "The serial position effect of free recall.," *Journal of experimental psychology*, vol. 64, no. 5, pp. 482–488, 1962.

[39] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 15, no. 1, pp. 72–101, 1904.

[40] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[41] J. C. Platt, "Using analytic qp and sparseness to speed training of support vector machines," in *Advances in neural information processing systems*, pp. 557–563, 1999.

[42] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[43] T. Hossfeld, D. Strohmeier, A. Raake, and R. Schatz, "Pippi longstocking calculus for temporal stimuli pattern on youtube qoe: 1+1 = 3 and 1·4≠ 4·1," in *Proceedings of the 5th Workshop on Mobile Video*, pp. 37–42, ACM, 2013.

[44] T. De Pessemier, K. De Moor, W. Joseph, L. De Marez, and L. Martens, "Quantifying the influence of rebuffering interruptions on the user's quality of experience during mobile video watching," *IEEE Transactions on Broadcasting*, vol. 59, no. 1, pp. 47–61, 2013.

[45] A. Zambelli, "Iis smooth streaming technical overview," *Microsoft Corporation*, vol. 3, p. 40, 2009.

[46] R. Pantos, "Sito ufficiale di apple http live streaming (draft): https://tools.ietf.org/html/draft-pantos-http-live-streaming-23," 2017.

[47] "Sito ufficiale di rfc 8216 - apple http live straming: https://tools.ietf.org/html/rfc8216."

[48] "Sezione "guides and sample code" del sito *developer.apple.com* dedicato ad apple http live streaming: https://developer.apple.com/library/content/referencelibrary/GettingStarted/AboutHTTPLiveStreaming/about/about.html."

[49] "Sito ufficiale di http dynamic streaming specification - version 3.0 final: https://wwwimages2.adobe.com/content/dam/acom/en/devnet/hds/pdfs/adobe-hds-specification.pdf."

[50] "Sito ufficiale di flash media manifest (f4m) format specification - version 3.0 final:
https://wwwimages2.adobe.com/content/dam/acom/en/devnet/hds/pdfs/adobe-media-manifest-specification.pdf."

[51] "Blog denivip:
http://blog.denivip.ru/index.php/2010/12/http-dynamic-streaming-configuring-web-server-cluster/?lang=en."

[52] "Sito ufficiale di mpeg:
http://mpeg.chiariglione.org."

[53] "Sito ufficiale di iso:
http://www.iso.org."

[54] "Sito ufficiale di iec:
http://www.iec.ch."

[55] "Sito ufficiale iso di iso/iec jct 1/sc 29:
http://www.iso.org/committee/45316.html."

[56] "Sito ufficiale iec id iso/iec jct 1/sc 29:
http://www.iec.ch/dyn/www/f?p=103:7:0::::FSP_ORG_ID:3403."

[57] "Sito ufficiale di mpeg-1:
http://mpeg.chiariglione.org/standards/mpeg-1."

[58] "Sito ufficiale di mpeg-2:
http://mpeg.chiariglione.org/standards/mpeg-2."

[59] "Sito ufficiale di mpeg-4:
http://mpeg.chiariglione.org/standards/mpeg-4."

[60] "Sito ufficiale di mpeg-dash:
http://mpeg.chiariglione.org/standards/mpeg-dash."

[61] "Sito ufficiale di mpeg iso base media file format: http://mpeg.chiariglione.org/standards/mpeg-4/iso-base-media-file-format."

[62] "Sito ufficiale iso di iso/iec 23001-7:2016 https://www.iso.org/standard/68042.html."

[63] "Sito ufficiale iec di iso/iec 23001-7:2016 https://webstore.iec.ch/publication/24226."

[64] "Sito ufficiale mpeg di iso/iec 23001-7 https://mpeg.chiariglione.org/standards/mpeg-b/common-encryption-iso-base-media-file-format-files/text-isoiec-23001-7-pdam-1."

[65] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "Qdash: a qoe-aware dash system," in *Proceedings of the 3rd Multimedia Systems Conference*, pp. 11–22, ACM, 2012.

[66] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for adaptive streaming over http," in *Packet Video Workshop (PV), 2012 19th International*, pp. 173–178, IEEE, 2012.

[67] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in dynamic http streaming," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 109–120, ACM, 2012.

[68] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 169–174, ACM, 2011.

[69] D. J. Vergados, A. Michalas, A. Sgora, and D. D. Vergados, "A control-based algorithm for rate adaption in mpeg-dash," in *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, pp. 438–442, IEEE, 2014.

[70] D. J. Vergados, A. Michalas, A. Sgora, and D. D. Vergados, "A fuzzy controller for rate adaptation in mpeg-dash clients," in *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on*, pp. 2008–2012, IEEE, 2014.

[71] D. J. Vergados, A. Michalas, A. Sgora, D. D. Vergados, and P. Chatzimisios, "Fdash: a fuzzy-based mpeg/dash adaptation algorithm," *IEEE Systems Journal*, vol. 10, no. 2, pp. 859–868, 2016.

[72] Z. Bingül and O. Karahan, "A fuzzy logic controller tuned with pso for 2 dof robot trajectory control," *Expert Systems with Applications*, vol. 38, no. 1, pp. 1017–1031, 2011.

[73] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*, pp. 1–9, IEEE, 2016.

[74] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[75] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 325–338, ACM, 2015.

[76] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena scientific Belmont, MA, 1995.

[77] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

Ringraziamenti

Desidero innanzitutto ringraziare il mio relatore, Dott. Luca Bedogni, per l'opportunità datami nel realizzare questa Tesi di Laurea e per il supporto ricevuto durante il periodo di scrittura della stessa.

Voglio ringraziare coloro con cui ho condiviso questo percorso di studi universitari ed, in particolare, coloro i quali più degli altri mi hanno dato una mano ogni qualvolta se ne sia palesato il bisogno, a partire da Gianluca Iselli, a Giacomo Bergami, sino a Davide Berardi, dimostratisi amici ancor prima che compagni di studio.

Desidero altresì ringraziare la mia famiglia ed i miei parenti tutti, grazie ai quali ho potuto intraprendere e portare a termine questo percorso.

Un ringraziamento speciale a tutte le persone che quotidianamente mi sono vicine e con le quali condivido la maggior parte del tempo, le quali hanno compreso le mie esigenze legate a questo percorso di studi che, in quest'ultimo periodo, si sono sommate a quelle lavorative. Desidero qui ringraziare espressamente Katia e gli amici di una vita.

Grazie a coloro con cui condivido interessi comuni, i quali hanno fatto sì che questo "viaggio" potesse essere un po' più leggero, dai ragazzi della palestra ai ragazzi del poligono.

Desidero infine ringraziare chiunque, in un modo o nell'altro, si sia sempre fatto trovare pronto e non mi abbia mai fatto mancare il suo sostegno, con un discorso di incoraggiamento o semplicemente per una serata in compagnia.