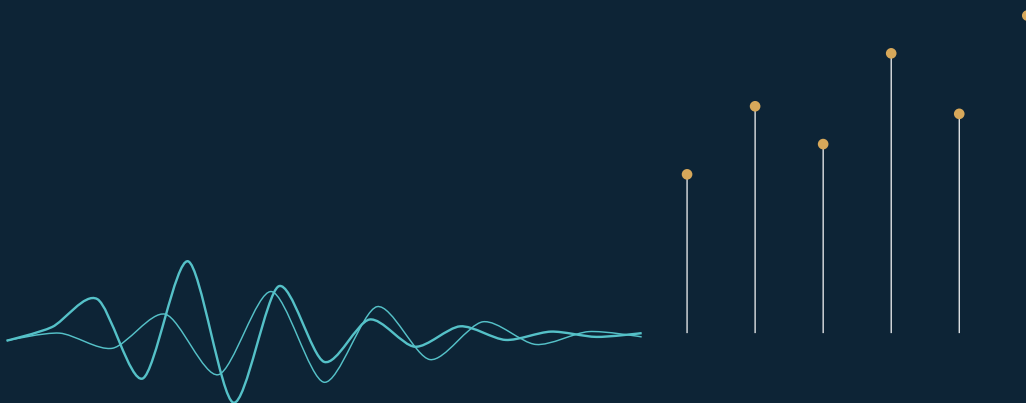
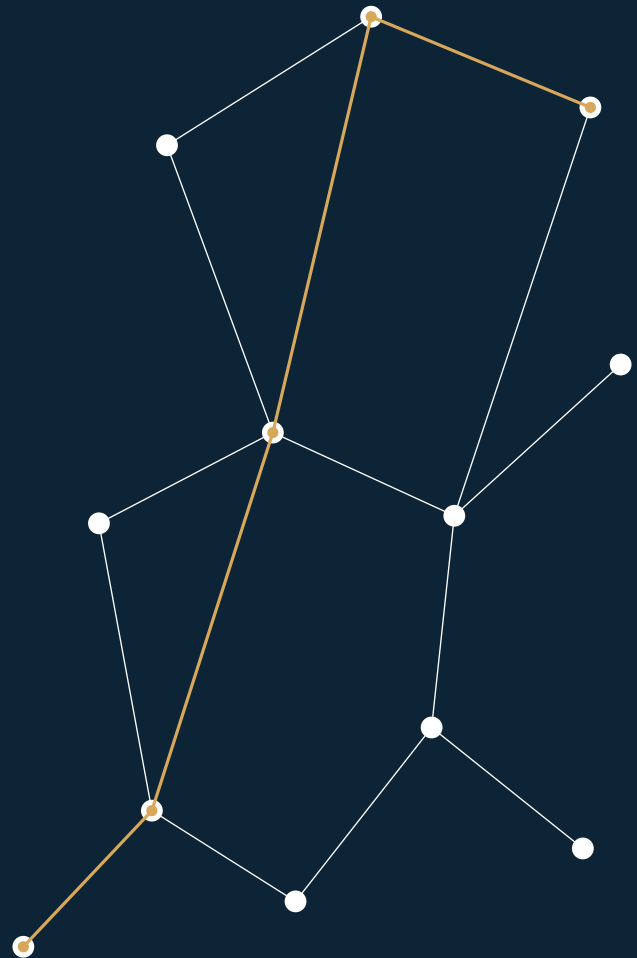
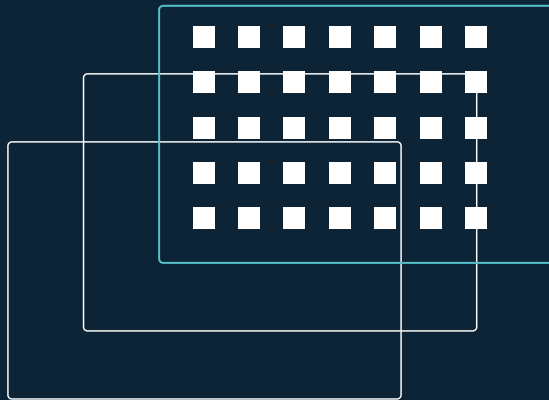


Comunicazioni Multimediali



© 2012–2024 Gianluca Mazzini e Chiara Taddia

© 2025– Gianluca Mazzini

Licenza Creative Commons CC BY-NC-ND 4.0

[Release 2.1](#)

*“There is a single light of science, and to brighten it anywhere
is to brighten it everywhere.”*

— Isaac Asimov

Indice

Premessa	ix
1 Introduzione ai sistemi multimediali	1
1.1 Modello OSI	1
1.2 Sorgenti multimediali	2
1.2.1 Sorgenti statiche	3
1.2.2 Sorgenti dinamiche	3
1.2.3 Ridondanza delle sorgenti	3
1.3 Reti multimediali	4
1.3.1 Banda	4
1.3.2 Ritardo	6
1.3.3 Jitter	6
1.3.4 Perdite	7
1.3.5 Qualità di servizio	7
1.3.6 Reti ibride	7
1.3.7 CDN ed edge	8
1.3.8 P2P e distribuzione ibrida	8
1.4 Sistemi multimediali	9
1.4.1 Requisiti applicativi e compromessi	9
1.4.2 E-learning e collaborazione	10
1.4.3 Gaming, realtà virtuale e realtà aumentata	10
1.4.4 Streaming e televisione	11
1.4.5 Teleconferenze e comunicazione in tempo reale	11
2 Sorgenti	13
2.1 Testo	13
2.1.1 ASCII	13
2.1.2 ISO-8859	14
2.1.3 Unicode, UCS e UTF-8	14
2.2 Immagini	15
2.2.1 Pixel	15
2.2.2 Piani colore	15
2.2.3 Colorimetria moderna e HDR	19
2.2.4 Dimensione, densità e risoluzione	21
2.2.5 Fattore di forma	22
2.2.6 Monitor	22
2.2.7 Elaborazione grafica e GPU	25
2.2.8 Formati raster non compressi	26
2.2.9 Grafica vettoriale e SVG	27
2.3 Audio	27
2.3.1 Conversione A/D	28
2.3.2 Formati audio non compressi	33
2.3.3 MIDI	36

2.4	Video	36
2.4.1	Frequenza di quadro, esposizione e moto	37
2.4.2	Conversione fra cadenze cinematografiche e televisive	37
2.4.3	Scansione progressiva e interlacciata	38
2.4.4	Televisione analogica monocromatica	38
2.4.5	Introduzione del colore: compatibilità e intercalamento	40
2.4.6	Dal segnale analogico al video digitale	41
2.4.7	Sottocampionamento della cromaticità	41
2.4.8	Formati video digitali storici	43
2.4.9	HD, UHD, HDR e frequenza di quadro elevati	43
3	Compressione delle sorgenti statiche	45
3.1	Elementi di codifica di sorgente	45
3.1.1	Entropia	45
3.1.2	Classificazione	48
3.2	Testo	48
3.2.1	RLE	49
3.2.2	Huffman statico	49
3.2.3	Huffman dinamico	55
3.2.4	Codifica aritmetica	58
3.2.5	Famiglia Lempel–Ziv	60
3.3	Compressione delle immagini	64
3.3.1	Misure di distorsione e qualità	65
3.3.2	Curve rate–distortion	66
3.3.3	GIF	66
3.3.4	PNG	68
3.3.5	JPEG baseline	69
3.3.6	JPEG 2000 e HTJ2K	75
3.3.7	JPEG XS: compressione a latenza estremamente bassa	77
3.3.8	WebP	77
3.3.9	HEIF e HEIC: separare contenitore e codec	78
3.3.10	AVIF	78
3.3.11	JPEG XL	78
3.3.12	JPEG AI: trasformazioni apprese	79
3.3.13	Confronto fra le principali famiglie	80
3.3.14	Evoluzione delle idee	80
4	Compressione delle sorgenti dinamiche	81
4.1	Audio	81
4.1.1	DPCM	81
4.1.2	APCM e ADPCM	83
4.1.3	LPC	84
4.1.4	CELP	85
4.1.5	FLAC: compressione audio lossless	86
4.1.6	MPEG-1	87
4.1.7	MPEG-2	94
4.1.8	AAC	95
4.1.9	Opus	96
4.1.10	EVS e LC3: altri obiettivi moderni	97
4.1.11	Dolby	97

4.1.12	Audio spaziale: canali, oggetti e scene	99
4.2	Video	101
4.2.1	MJPEG: il caso senza predizione temporale	101
4.2.2	Ridondanza temporale e predizione	101
4.2.3	H.261: la struttura ibrida diventa standard	107
4.2.4	H.263: più efficienza e più strumenti	109
4.2.5	MPEG-1 e MPEG-2 Video	111
4.2.6	MPEG-4 Visual e l'idea degli oggetti	112
4.2.7	Confrontare le generazioni storiche	113
4.2.8	Dall'AVC ai codec contemporanei	113
4.2.9	Metriche di qualità video moderne	119
4.2.10	Come confrontare i codec video	119
5	Reti	121
5.1	Canale	121
5.1.1	Buffering e playout	122
5.1.2	Perdita, ritrasmissione e FEC	124
5.2	Congestione	129
5.2.1	FIFO e classificazione	130
5.2.2	Fair Queuing e GPS	130
5.2.3	Weighted Fair Queuing	131
5.2.4	Drop, marking e Active Queue Management	131
5.3	Qualità di servizio	133
5.3.1	Code a priorità	133
5.3.2	Leaky Bucket e Token Bucket	135
5.3.3	IntServ e RSVP	136
5.3.4	DiffServ	136
5.3.5	L4S: latenza bassa e segnalazione ECN frequente	137
5.4	RTP e RTCP	137
5.4.1	Sorgenti e mixer	137
5.4.2	Header RTP	138
5.4.3	RTCP: feedback e sincronizzazione	139
5.4.4	Overhead e pacchettizzazione	139
5.5	Streaming multimediale su HTTP	140
5.5.1	HLS e MPEG-DASH	141
5.5.2	Adaptive Bitrate Streaming	141
5.5.3	Low-Latency HLS e Low-Latency DASH	141
5.5.4	CDN, cache e live edge	142
5.6	Distribuzione P2P e ibrida	142
5.6.1	Discovery, overlay e ruoli	143
5.6.2	Albero, mesh e swarming	143
5.6.3	Push, pull e selezione dei chunk	143
5.6.4	Capacità aggregata e carico sull'infrastruttura	144
5.6.5	Offload P2P	145
5.6.6	Disseminazione e latenza	146
5.6.7	Vincolo di fan-out nel live	147
5.6.8	Churn e disponibilità dei chunk	148
5.6.9	Incentivi e fairness	148
5.6.10	NAT, browser e relay	148

5.6.11	Integrità, sicurezza e contenuti ostili	148
5.6.12	P2P per on-demand e live	149
5.6.13	Architettura ibrida CDN+P2P	149
5.7	HTTP/2, QUIC e HTTP/3	150
5.8	WebTransport	150
5.9	Media over QUIC	151
5.10	Contribution professionale: SRT e RIST	151
5.11	Dalla QoS di rete all'adattamento applicativo	152
6	Sistemi	153
6.1	Sistemi di conferenza su reti a circuito e a servizio garantito	153
6.1.1	H.320	153
6.1.2	H.324	155
6.1.3	H.321	157
6.1.4	H.310	157
6.1.5	H.322	157
6.2	Conferenze a commutazione di pacchetto	157
6.2.1	H.323	157
6.2.2	SIP e SDP	160
6.3	WebRTC	163
6.3.1	ICE, STUN e TURN	163
6.3.2	Sicurezza dei media	164
6.3.3	Negoziiazione, RTP/RTCP e adattamento	164
6.3.4	DataChannel	164
6.3.5	Conferenze multiparty: mesh, SFU e MCU	164
6.3.6	WebRTC e WebTransport	166
6.4	Contenitori multimediali	166
6.4.1	RIFF e AVI	166
6.4.2	OpenDML e AVI 2.0	169
6.4.3	MPEG-2 Transport Stream	170
6.4.4	ISO Base Media File Format e MP4	172
6.4.5	Fragmented MP4	172
6.4.6	CMAF	173
6.4.7	Matroska e WebM	173
6.4.8	Confronto dei contenitori	173
6.5	Pipeline multimediale nel Web	174
6.5.1	Media Source Extensions	174
6.5.2	Timed text e accessibilità	175
6.5.3	Web Audio	175
6.6	Protezione del trasporto e protezione del contenuto	175
6.7	WebCodecs e pipeline applicative	176
6.8	Architettura multimediale moderna	177

Premessa

Pochi settori dell'informatica e delle telecomunicazioni sono cambiati quanto il multimedia. Non si è trattato soltanto di un aumento della qualità delle immagini, della capacità delle reti o dell'efficienza dei codec: è cambiato il modo stesso in cui i contenuti vengono prodotti, distribuiti e fruiti. Tecnologie che nella prima fase della storia del multimedia apparivano come applicazioni particolari sono diventate infrastrutture quotidiane, spesso trasparenti per chi le utilizza.

Lo streaming è forse l'esempio più evidente di questa trasformazione. La televisione, per decenni legata a un palinsesto e a una trasmissione lineare, è diventata anche un servizio on demand, accessibile in ogni momento da reti e terminali differenti. Per musica e video si è passati, in larga parte, dal possesso di un supporto o di una copia locale — disco, CD, DVD, file — all'accesso remoto a servizi continuamente disponibili, capaci di offrire cataloghi sterminati. È un cambiamento più profondo di una semplice evoluzione del mezzo: il contenuto non deve più trovarsi dove si trova l'utente e non deve necessariamente essere posseduto da chi lo fruisce. Può essere conservato altrove, replicato, codificato in più forme, adattato alla rete e al terminale e consegnato quando viene richiesto. Anche videoconferenza, collaborazione, gioco, produzione e condivisione dei contenuti sono diventati servizi di rete ordinari.

Questa evoluzione rende particolarmente interessante tornare sui fondamenti. Le tecnologie cambiano rapidamente, ma i problemi che devono risolvere restano riconoscibili: rappresentare una sorgente, eliminare ridondanza, scegliere che cosa possa essere perso, sincronizzare media differenti, trasportare dati rispettando una deadline, reagire alla congestione, adattare la qualità, proteggere il contenuto e renderlo disponibile a molti utenti. Una nuova tecnologia è davvero comprensibile solo quando la si collega a questi problemi e alle soluzioni che l'hanno preceduta.

La prima edizione di questo libro fu realizzata insieme a Chiara Taddia. Già allora l'obiettivo era mettere in relazione temi che spesso venivano studiati separatamente: caratterizzazione delle sorgenti, compressione, reti e sistemi multimediali. Questa nuova edizione conserva quell'impostazione, ma il testo è stato completamente riletto, verificato, ampliato, riorganizzato e aggiornato. Non si tratta quindi della semplice revisione di un libro precedente: è anche il racconto di come un settore sia cambiato mentre diventava parte dell'infrastruttura ordinaria della società digitale.

L'obiettivo non è costruire un catalogo di standard. Il testo parte dalla percezione e dalla rappresentazione di testo, immagini, audio e video, introduce i principi della compressione e segue poi l'informazione attraverso il trasporto e la distribuzione fino ai sistemi completi. Per ogni tecnologia si cerca di ricostruire il percorso progettuale: quale problema esisteva, quale nuova idea viene introdotta, come funziona, che cosa migliora e quale costo comporta in termini di bitrate, qualità, latenza, complessità, affidabilità e scalabilità.

Per questa ragione il libro mantiene intenzionalmente anche tecnologie storiche. CRT, televisione analogica, PAL, NTSC, H.261, H.263, MPEG-1, H.320, H.323, AVI e altri sistemi non vengono descritti perché rappresentino lo stato dell'arte, ma perché rendono visibile la nascita di idee che ricompaiono continuamente: scansione, luma e cromaticità, sottocampionamento, trasformazione, predizione, compensazione del moto, multiplexing, descrizione delle capacità, controllo della sessione, timeline e indicizzazione. Quando una tecnologia

storica non aiuta più a costruire questo percorso, viene invece ridotta o ricontestualizzata.

La stessa impostazione guida le parti matematiche. Le ipotesi dei modelli vengono rese esplicite e ogni risultato non immediato che viene derivato è accompagnato dalla dimostrazione. Una relazione empirica viene presentata come modello o approssimazione; una regola imposta da uno standard viene indicata come specifica; un risultato matematico viene distinto da entrambi. Questa separazione è essenziale per capire che cosa possa essere generalizzato e che cosa dipenda invece da una particolare scelta progettuale o normativa.

Le figure fanno parte della spiegazione. I grafici quantitativi sono costruiti a partire da relazioni e parametri dichiarati; gli schemi concettuali rendono visibili pipeline, strutture di codifica, timeline, packetizzazione, call flow e architetture di distribuzione. Non hanno quindi una funzione decorativa, ma accompagnano il ragionamento e permettono di collegare la descrizione formale al comportamento del sistema.

Il percorso arriva alle tecnologie contemporanee senza perdere la continuità storica. Codifiche moderne per immagini, audio e video si affiancano alle tecniche che ne hanno preparato i principi; la distribuzione comprende streaming adattativo, CDN, bassa latenza, QUIC e HTTP/3; la comunicazione realtime porta a WebRTC e alle architetture SFU; i contenitori e le API Web mostrano come codifica, timeline, trasporto e riproduzione siano diventati componenti di una stessa infrastruttura. L'obiettivo resta sempre capire l'architettura, non memorizzare un elenco di tecnologie.

Il libro è pensato soprattutto per studenti universitari che vogliano capire e studiare il funzionamento dei sistemi, non soltanto acquisirne una visione intuitiva. Richiede i normali fondamenti di matematica, probabilità, segnali e reti; quando serve, i passaggi necessari vengono sviluppati esplicitamente. L'intento è fornire una base sufficientemente rigorosa da poter analizzare e confrontare le prestazioni, ma anche abbastanza concreta da mostrare come i singoli meccanismi cooperino in un sistema reale.

Ringrazio Chiara Taddia, con la quale fu realizzata la prima edizione di questo libro. Una parte importante dell'impostazione originaria del volume nasce da quel lavoro comune. La ringrazio anche per avere insegnato per alcuni anni questo corso all'Università di Ferrara, portandone i contenuti in aula e contribuendo a farli maturare nel confronto con generazioni diverse di studenti.

Desidero inoltre ricordare con gratitudine il Prof. Oreste Andrisano. Dal suo insegnamento ho imparato che il rigore tecnico acquista valore quando si accompagna alla curiosità e alla capacità di guardare oltre la singola tecnologia. È un modo di affrontare i problemi che ha lasciato un segno profondo nel mio percorso e che continua a riemergere anche nel modo in cui cerco di studiare e raccontare questi argomenti.

Gianluca Mazzini

Introduzione ai sistemi multimediali

1.1 Modello OSI

La progettazione di un sistema di comunicazione viene normalmente suddivisa in livelli: l'applicazione esprime un servizio, i protocolli di trasporto e di rete forniscono connettività, i livelli inferiori realizzano il trasferimento sul mezzo fisico. Questa separazione resta fondamentale anche nei sistemi multimediali, ma non implica che i livelli possano essere progettati ignorando completamente i requisiti degli altri.

Audio, video e applicazioni interattive rendono evidente questa dipendenza. Il codec produce un flusso con una certa bitrate e una certa sensibilità a ritardo e perdita; la rete offre capacità, ritardo, jitter e affidabilità variabili; il ricevitore possiede una propria capacità di elaborazione e un proprio display. Il sistema deve quindi adattare i parametri applicativi al servizio effettivamente disponibile. Questa interazione viene spesso indicata come adattamento cross-layer quando utilizza informazioni che attraversano i confini tradizionali fra livelli.

Il modello OSI rimane utile per separare le funzioni e ragionare sulle interfacce. Non va però interpretato come l'obbligo di realizzare sette blocchi indipendenti e immutabili. Le architetture reali possono introdurre sottolivelli, più pile protocollari parallele e canali di controllo che collegano funzioni collocate in punti diversi della pila. WebRTC, per esempio, combina protocolli di trasporto, sicurezza, controllo e descrizione della sessione senza cancellarne la separazione concettuale.

La stratificazione introduce anche overhead. Ogni livello può aggiungere header, informazioni di controllo o protezione. L'overhead non è necessariamente uno spreco: sequence number, timestamp, indirizzi e controlli di integrità rendono possibili funzioni indispensabili. Diventa inefficiente quando il costo della segnalazione è grande rispetto al payload utile, come può accadere con pacchetti molto piccoli. Nel seguito distingueremo quindi sempre fra informazione applicativa e informazione necessaria al trasporto.

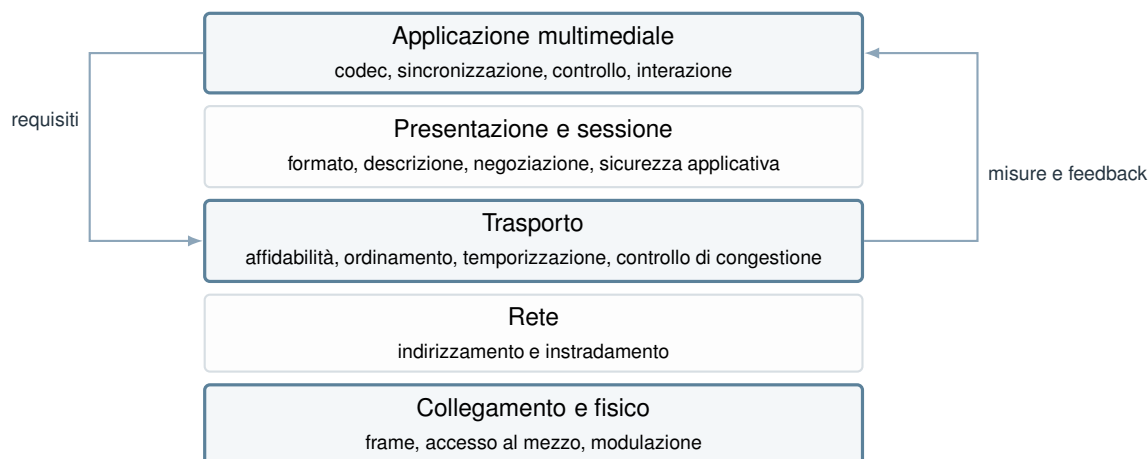


Figura 1.1. Separazione funzionale dei livelli e retroazione applicativa in un sistema multimediale. Il modello OSI resta una guida concettuale, ma i sistemi reali usano feedback fra trasporto e applicazione.

1.2 Sorgenti multimediali

Un sistema multimediale parte da una sorgente di informazione. Testo, immagini, audio e video sono le sorgenti più comuni perché riguardano direttamente vista e udito, ma lo stesso metodo si estende a dati tattili, sensori, posizione e altre informazioni di contesto. Il problema generale è trasformare un fenomeno o una rappresentazione simbolica in dati che possano essere elaborati, memorizzati e trasmessi.

Una classificazione elementare, utile per introdurre i meccanismi del libro, distingue sorgenti statiche e sorgenti dinamiche. Una sorgente statica viene considerata come un oggetto completo, senza una temporizzazione intrinseca necessaria alla fruizione. Una sorgente dinamica produce invece una sequenza il cui ordine e i cui tempi fanno parte essenziale dell'informazione. La distinzione dipende dal modello: una immagine è statica, mentre una sequenza di immagini temporizzata è video.

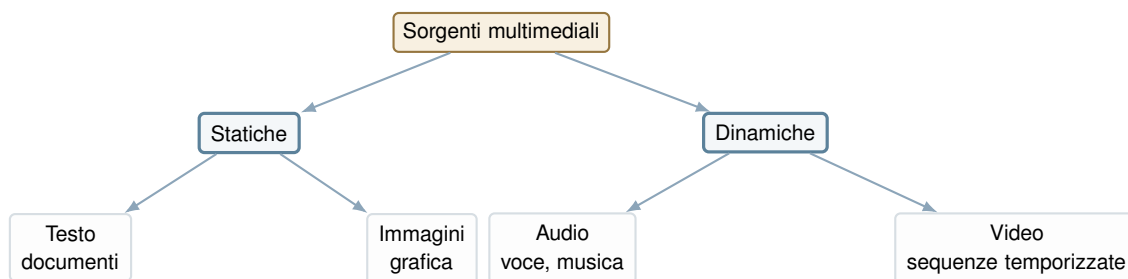


Figura 1.2. Classificazione introduttiva delle sorgenti. La distinzione non è assoluta: lo stesso oggetto può diventare dinamico quando la temporizzazione entra nel significato.

La classificazione statica/dinamica è soltanto il primo livello. Per progettare un sistema occorre anche chiedersi se la sorgente sia *discreta* o *continua* prima della digitalizzazione, quale sia la sua dimensionalità e quale parte dell'informazione sia percettivamente rilevante. Il testo nasce già come sequenza di simboli; un microfono e una telecamera osservano invece grandezze fisiche continue che devono essere campionate e quantizzate. Un'immagine è bidimensionale nello spazio, l'audio è essenzialmente monodimensionale nel tempo, il video aggiunge una dimensione temporale a due dimensioni spaziali. Questa struttura determina il tipo di ridondanza che sarà possibile sfruttare.

1.2.1 Sorgenti statiche

Il testo è una sequenza di simboli appartenenti a un insieme definito. La rappresentazione digitale deve associare ai simboli dei codici binari e deve specificare come tali codici vengono interpretati. ASCII, Unicode e UTF-8, analizzati nel capitolo successivo, mostrano come una stessa informazione linguistica possa essere rappresentata mediante convenzioni differenti.

Il significato di un testo non è contenuto soltanto nei singoli simboli. Lessico, sintassi e contesto introducono dipendenze fra i caratteri e fra le parole. Queste dipendenze costituiscono ridondanza e possono essere sfruttate dalla compressione.

Le immagini rappresentano una distribuzione spaziale di informazione visiva. Un'immagine digitale raster è descritta da un insieme finito di campioni organizzati nello spazio; a ciascun campione sono associate una o più componenti numeriche, per esempio le componenti di colore.

Non è corretto associare all'occhio umano un unico numero equivalente di pixel. La densità dei fotorecettori varia sulla retina e la percezione finale dipende dall'ottica dell'occhio, dal contrasto, dalla luminanza, dai movimenti oculari e dall'elaborazione neurale. Come ordine di grandezza, in condizioni favorevoli, dettagli separati da circa un primo d'arco possono essere distinti nella regione di massima acuità. Questo valore è un riferimento percettivo, non una risoluzione digitale dell'occhio.

È utile anche distinguere la risoluzione percettiva dai limiti fisici di un'apertura ottica. Relazioni come il limite empirico di Dawes e il criterio di Rayleigh descrivono problemi ottici specifici e non devono essere usate come misura completa della risoluzione del sistema visivo. Per una pupilla di alcuni millimetri, la diffrazione nella banda visibile porta a un limite angolare dell'ordine di alcune decine di secondi d'arco; la risoluzione effettiva dell'occhio è normalmente peggiore perché intervengono aberrazioni, retina ed elaborazione neurale.

1.2.2 Sorgenti dinamiche

L'audio rappresenta variazioni temporali di pressione o, più in generale, grandezze acustiche trasformate da un trasduttore in un segnale. Nell'aria, a temperatura ambiente, la velocità del suono è dell'ordine di 340 m/s. La voce possiede inoltre una forte struttura linguistica e fisiologica, che verrà sfruttata dai codec predittivi.

Il video è una sequenza temporizzata di immagini. La percezione della continuità del moto non è governata da una singola frequenza di quadro universale: dipende da contenuto, luminanza, moto, tempo di esposizione e caratteristiche del display. Le frequenze di quadro tradizionali dell'ordine di 24–30 frame/s sono adeguate a molte forme di produzione, mentre valori maggiori migliorano la rappresentazione del moto rapido e riducono alcuni artefatti temporali. La scelta è quindi un compromesso fra qualità temporale, bitrate e complessità.

1.2.3 Ridondanza delle sorgenti

Le sorgenti naturali presentano dipendenze statistiche e percettive. Pixel vicini tendono spesso ad avere valori simili, campioni audio consecutivi sono correlati, i frame di un video condividono gran parte della scena e le lingue naturali impongono vincoli forti sulle sequenze di simboli. Queste regolarità costituiscono ridondanza.

La ridondanza non è tutta dello stesso tipo. Una parte può essere rimossa senza alcuna perdita, ottenendo una codifica lossless. Una parte può invece essere resa meno accurata sfruttando limiti percettivi, ottenendo una codifica lossy. Nel secondo caso la compressione introduce una distorsione e il problema diventa scegliere il compromesso fra bitrate e qualità.

La compressione non consiste quindi semplicemente nel “togliere dati”. Cerca una rappresentazione che utilizzi meno risorse preservando esattamente l'informazione richiesta, nel caso lossless, oppure preservandola entro un livello di distorsione accettabile, nel caso lossy.

I capitoli successivi formalizzeranno questi concetti mediante entropia, predizione, trasformazioni, quantizzazione e modelli percettivi.

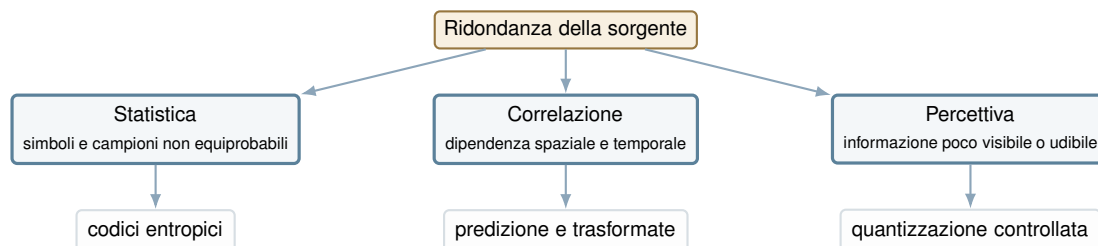


Figura 1.3. Tre forme di ridondanza che ricorrono nella compressione multimediale e gli strumenti tipici usati per sfruttarle.

1.3 Reti multimediali

La storia delle reti multimediali nasce da infrastrutture separate: rete telefonica, reti dati, reti televisive, reti cellulari e sistemi a commutazione di circuito. Ogni infrastruttura disponeva di propri meccanismi di segnalazione, formati di sorgente e tecniche di trasporto. Questa separazione è importante dal punto di vista storico perché spiega l'origine di molti protocolli, ma non descrive più il modello prevalente.

La convergenza si è infatti realizzata soprattutto attorno a IP. La stessa infrastruttura a pacchetto trasporta oggi voce, video, dati, controllo e contenuti interattivi attraverso accessi in fibra, reti mobili, reti wireless e collegamenti satellitari. La differenza fra i servizi non è più determinata principalmente dalla rete fisica usata, ma dal modo in cui applicazione e trasporto reagiscono alle caratteristiche disponibili.

Questa evoluzione modifica anche il problema progettuale. In una rete a circuito la risorsa può essere dimensionata prima dell'inizio della comunicazione. In una rete IP condivisa, invece, banda disponibile, ritardo, jitter e perdita possono cambiare nel tempo. Il sistema multimediale moderno deve quindi misurare o stimare le condizioni di rete e adattare codifica, bitrate, buffering e modalità di trasporto. L'adattamento non è un'aggiunta accessoria: è parte del funzionamento normale del sistema.

Accanto alla rete di accesso sono diventati fondamentali anche i sistemi di distribuzione. Un contenuto richiesto da molti utenti non viene normalmente servito da un unico elaboratore centrale: copie o segmenti vengono distribuiti su Content Delivery Network (CDN), cache ed elaboratori edge. Lo scopo è avvicinare i dati al ricevitore, ridurre il traffico sulle tratte lunghe e diminuire la latenza di accesso. Nei capitoli successivi vedremo come questa architettura si combina con lo streaming adattativo e con i protocolli HTTP moderni.

I parametri fondamentali con cui descriveremo una rete multimediale sono banda utile, ritardo, jitter, perdita e disponibilità. Nessuno di questi parametri, preso isolatamente, determina la qualità dell'applicazione: ciò che conta è il loro effetto congiunto sul meccanismo di trasporto e sul comportamento della sorgente.

1.3.1 Banda

La banda disponibile indica la capacità di trasferimento che una comunicazione può effettivamente utilizzare. Va distinta dalla capacità nominale del collegamento: protocolli, header, ritrasmissioni, codifica di canale e traffico concorrente riducono la quota realmente disponibile per i dati applicativi. È quindi utile distinguere almeno fra capacità di linea, throughput osservato e goodput, cioè la velocità con cui arrivano all'applicazione dati utili non duplicati.

Consideriamo, come modello elementare, pacchetti tutti uguali con P byte di payload

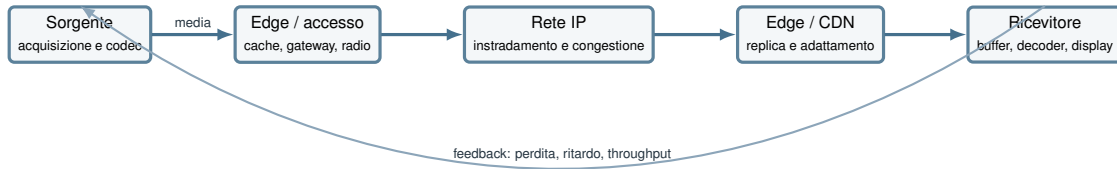


Figura 1.4. Catena end-to-end di una comunicazione multimediale moderna. Il flusso dei media procede verso il ricevitore, mentre misure e feedback possono modificare codifica, trasporto e buffering.

applicativo e H byte complessivi di header. Se il link trasporta i pacchetti al rate R bit/s e non vi sono perdite o ritrasmissioni, l'efficienza di incapsulamento è

$$\eta = \frac{P}{P + H}$$

e il massimo goodput applicativo è

$$R_{app} = R \frac{P}{P + H}.$$

Dimostrazione. In un intervallo nel quale vengono trasmessi N pacchetti, il numero di byte inviati sul collegamento è $N(P + H)$, mentre i byte utili consegnati all'applicazione sono NP . Il rapporto fra dati utili e dati trasmessi vale quindi

$$\frac{NP}{N(P + H)} = \frac{P}{P + H} = \eta.$$

Poiché nello stesso intervallo il collegamento trasporta R bit/s, la quota attribuibile al payload è $R\eta$, da cui segue

$$R_{app} = R \frac{P}{P + H}.$$

Questo conclude la dimostrazione.

La figura 1.5 mostra la sola efficienza di incapsulamento per un esempio con $H = 40$ byte. Il valore di H è scelto soltanto per rendere visibile l'andamento: protocolli reali possono avere overhead differenti e variabili.

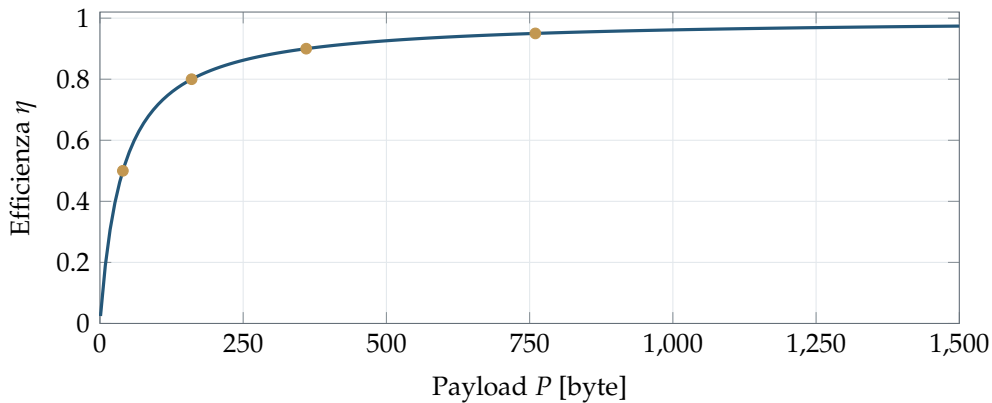


Figura 1.5. Efficienza di incapsulamento $P/(P + H)$ al crescere del payload, con $H = 40$ byte come esempio. Il beneficio di aumentare P presenta rendimenti progressivamente decrescenti.

La relazione mostra un principio generale: a parità di header, pacchetti molto piccoli hanno efficienza inferiore. Nei sistemi in tempo reale, tuttavia, aumentare troppo la dimensione

dei pacchetti può aumentare la latenza di pacchettizzazione e la quantità di informazione persa quando un singolo pacchetto viene eliminato. Anche la dimensione del pacchetto è quindi un compromesso.

La capacità può inoltre essere simmetrica o asimmetrica. Questo è rilevante non soltanto per il download di contenuti: videoconferenza, streaming prodotto dall'utente, backup multimediale e sistemi immersivi possono richiedere una capacità di upload comparabile a quella di download. Non è quindi corretto assumere in generale che il traffico multimediale sia prevalentemente discendente.

1.3.2 Ritardo

Il ritardo end-to-end è il tempo che separa la generazione di una unità informativa dalla sua disponibilità al ricevitore. Comprende tempi di acquisizione e codifica, pacchettizzazione, accodamento, trasmissione, propagazione, elaborazione nei nodi, eventuali ritrasmissioni, buffering e decodifica. Aumentare la banda riduce il tempo necessario a serializzare un pacchetto, ma non elimina le altre componenti.

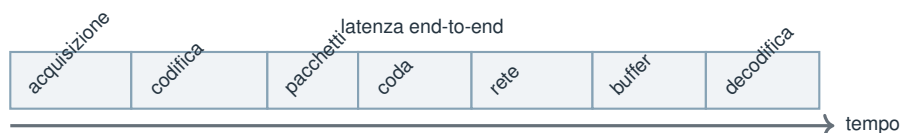


Figura 1.6. Scomposizione concettuale della latenza. L'aumento della capacità di linea riduce soltanto alcune componenti; codifica, propagazione e buffering possono restare dominanti.

Questa distinzione è essenziale. Un collegamento molto veloce può avere latenza elevata se il percorso è lungo o se sono presenti code; un collegamento con capacità inferiore può invece risultare più adatto a un'applicazione interattiva se ha un percorso breve e stabile. La progettazione multimediale deve quindi considerare separatamente capacità e latenza.

Le applicazioni on-demand possono tollerare un ritardo iniziale più elevato purché il flusso successivo sia regolare. Le applicazioni interattive, invece, richiedono che l'intera catena di acquisizione, rete, buffering e riproduzione mantenga la latenza contenuta. Ridurre il ritardo spesso significa rinunciare a parte del buffering e quindi essere più sensibili al jitter: anche in questo caso le prestazioni sono il risultato di un compromesso.

1.3.3 Jitter

Il jitter descrive la variabilità del ritardo fra pacchetti. Può essere caratterizzato mediante deviazione standard, percentili o stimatori filtrati, a seconda del protocollo e dell'applicazione. Non va confuso con il ritardo medio: una comunicazione può avere ritardo relativamente elevato ma quasi costante, oppure ritardo medio ridotto e fortemente variabile.

La contromisura classica è il playout buffer. Il ricevitore ritarda la riproduzione per accumulare un margine temporale e trasformare un flusso di arrivo irregolare in un flusso di uscita regolare. Aumentare il buffer riduce la probabilità che un pacchetto arrivi dopo la propria deadline, ma aumenta la latenza. Nel capitolo dedicato alle reti questo compromesso verrà formalizzato e dimostrato per modelli di ritardo uniformi e gaussiani.

Nei sistemi moderni il buffer è spesso adattativo: la sua dimensione cambia in funzione delle misure recenti della rete. Questa idea è generale e ricomparirà nello streaming adattativo, dove non si adatta soltanto il buffer ma anche la bitrate del contenuto selezionato.

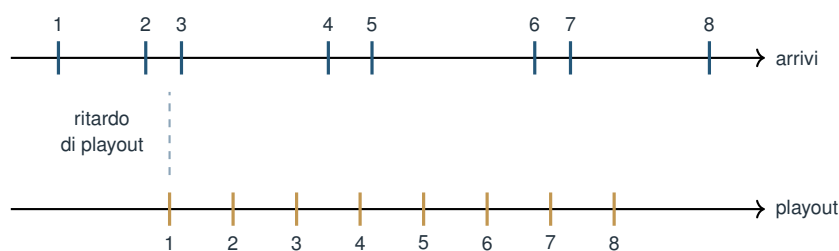


Figura 1.7. Il playout buffer trasforma arrivi irregolari in una cadenza di riproduzione regolare. Un pacchetto che supera la propria deadline diventa inutilizzabile nel modello a deadline rigida.

1.3.4 Perdite

Un pacchetto può essere perso per errori sul collegamento, congestione, politiche di coda o cambiamenti del percorso. La conseguenza dipende dal tipo di applicazione e dal tempo disponibile per reagire.

Un'applicazione affidabile può richiedere la ritrasmissione del dato mancante. Un'applicazione in tempo reale può invece non avere il tempo di attendere: un pacchetto arrivato dopo la propria deadline può essere inutile quanto un pacchetto perso. Per questo i sistemi multimediali combinano, a seconda del caso, ritrasmissione selettiva, Forward Error Correction (FEC), concealment nel decodificatore e adattamento della sorgente.

Le perdite influenzano anche il controllo di congestione. Nei protocolli affidabili la perdita o altri segnali di congestione possono ridurre il rate di invio; nei protocolli moderni la stima della rete può utilizzare anche misure di ritardo e segnali espliciti. Non esiste una percentuale universale oltre la quale un protocollo di trasporto “entra in deadlock”: la prestazione dipende dall'algoritmo di controllo, dal round-trip time, dalla capacità del percorso e dal pattern delle perdite.

1.3.5 Qualità di servizio

La qualità di servizio di rete non è un singolo numero. Banda, ritardo, jitter, perdita e disponibilità descrivono aspetti diversi e un'applicazione può attribuire loro pesi differenti. Una videoconferenza privilegia la latenza e la continuità; un download di un file video privilegia affidabilità e throughput; lo streaming on-demand può assorbire variazioni della rete mediante buffering.

La commutazione di circuito rappresenta storicamente la soluzione più diretta per ottenere risorse prevedibili, ma utilizza male la statistica quando la sorgente è intermittente. Le reti a pacchetto condividono invece le risorse e possono introdurre meccanismi di priorità, scheduling, shaping e controllo di congestione. IntServ e RSVP costituiscono un importante modello di prenotazione esplicita; DiffServ realizza una differenziazione per classi e rimane utile come principio di trattamento aggregato dei pacchetti.

Nelle architetture moderne una parte importante della qualità non è però ottenuta chiedendo alla rete di garantire rigidamente una risorsa. È l'applicazione stessa che si adatta: modifica bitrate, risoluzione, frequenza di quadro, ridondanza e profondità del buffer. Questa evoluzione, dalla sola QoS di rete alla qualità end-to-end adattativa, è una delle idee centrali dei sistemi multimediali contemporanei.

1.3.6 Reti ibride

Broadcast e rete IP non sono modelli necessariamente alternativi. Un sistema televisivo può distribuire in broadcast il contenuto comune a molti utenti e utilizzare una connessione IP bidirezionale per segnalazione, personalizzazione, autenticazione o contenuti on-demand.

Analogamente, una rete satellitare può essere integrata con reti terrestri o mobili.

Il principio generale è separare, quando conviene, il canale più efficiente per la distribuzione massiva dal canale necessario all'interazione individuale. Questa idea storica del “canale di ritorno” resta utile anche nei sistemi moderni: ciò che cambia sono le tecnologie con cui i due percorsi vengono realizzati.

1.3.7 CDN ed edge

Il modello client-server centralizzato incontra un limite quando lo stesso contenuto deve raggiungere un numero molto elevato di utenti o quando la distanza dal server aumenta la latenza. Le CDN replicano o cachano i contenuti in molti punti della rete e indirizzano il client verso una copia vicina o conveniente.

L'idea fondamentale è diversa dalla semplice aggiunta di banda al server: si riduce la distanza logica e fisica fra contenuto e utente e si distribuisce il carico su più nodi. L'edge computing estende ulteriormente il modello consentendo di eseguire vicino all'utente anche elaborazioni, transcodifica, personalizzazione o funzioni di controllo. Questa architettura è particolarmente importante nei servizi live e interattivi, dove trasferire ogni dato verso un data center distante aggiungerebbe latenza evitabile.

1.3.8 P2P e distribuzione ibrida

Nel peer-to-peer (P2P) un terminale non è soltanto consumatore del contenuto: può mettere a disposizione banda di upload, dati già ricevuti, memoria e, in alcuni sistemi, capacità di elaborazione. I peer costruiscono quindi una rete logica o *overlay* sopra la rete IP sottostante. La capacità di distribuzione non dipende più soltanto dall'origin o dai server del servizio, ma anche dalle risorse aggregate dei partecipanti.

Questa idea produce architetture differenti. In una struttura ad *albero* ciascun peer riceve tipicamente da un genitore e inoltra a uno o più figli: il percorso è semplice e l'overhead di controllo può essere contenuto, ma l'uscita di un nodo interno interrompe un intero sottoalbero finché la struttura non viene riparata. In una struttura a *mesh* o *swarm*, invece, un peer può ottenere parti del contenuto da più vicini e fornirne altre ad altri peer. Il sistema è più robusto al churn ma richiede discovery, scelta dei vicini, selezione dei chunk e controllo della disponibilità.

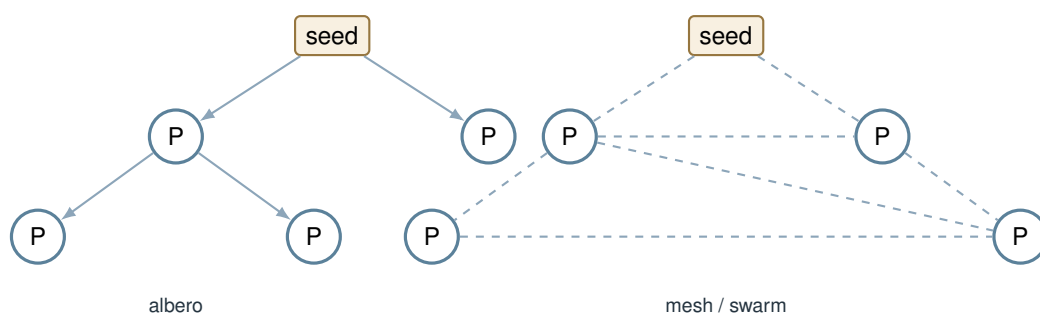


Figura 1.8. Due organizzazioni P2P concettuali. L'albero riduce la molteplicità dei percorsi; la mesh distribuisce i chunk attraverso più relazioni fra peer e tollera meglio il churn.

Per il multimedia la disponibilità del contenuto non basta: ogni chunk ha anche una *deadline*. Una politica efficace per distribuire un file completo può quindi essere inadatta a un flusso live, nel quale un chunk arrivato dopo il playout non ha più valore. Nei sistemi video P2P diventano centrali dimensione dei chunk, buffer, velocità di upload, selezione dei peer, replica delle parti rare e priorità delle parti prossime alla riproduzione.

Il vantaggio potenziale è la scalabilità. Se un nuovo utente porta con sé anche capacità di

upload utilizzabile, l'aumento della domanda è parzialmente compensato da nuova capacità di distribuzione. Questo non significa che il server possa sempre scomparire: startup, peer dietro NAT, churn, parti rare, autenticazione, controllo della qualità e utenti con upload insufficiente richiedono spesso infrastruttura stabile.

Per questo la forma operativamente più interessante è spesso *ibrida*. Origin e CDN garantiscono disponibilità, bootstrap e prestazioni minime; i peer assorbono una parte del traffico quando possono contribuire utilmente. L'infrastruttura non viene sostituita: viene alleggerita.

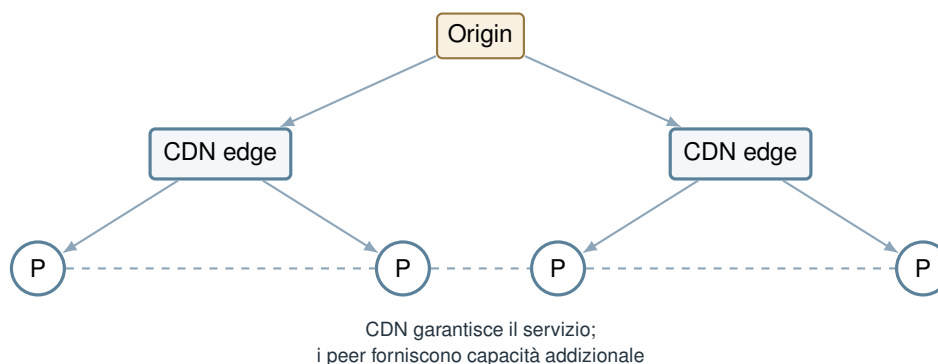


Figura 1.9. Distribuzione ibrida CDN+P2P. Il piano infrastrutturale mantiene disponibilità e controllo, mentre l'overlay P2P può ridurre il traffico servito direttamente dagli edge.

La Sezione 5.6 riprenderà la distribuzione P2P dal punto di vista quantitativo, analizzando capacità aggregata, offload, disseminazione dei chunk, churn e prestazioni dei sistemi live e on-demand.

1.4 Sistemi multimediali

Un sistema multimediale non è soltanto una sorgente collegata a una rete. Comprende acquisizione, codifica, trasporto, eventuale elaborazione intermedia, sincronizzazione, decodifica e interazione con l'utente. Nei sistemi moderni queste funzioni possono essere realizzate in dispositivi differenti e distribuite fra terminale, edge e cloud.

Una prima distinzione utile è fra servizi on-demand, live e interattivi. Nel servizio on-demand il contenuto è già disponibile e si può usare un buffer relativamente profondo. Nel live il contenuto viene prodotto mentre viene distribuito e il ritardo rispetto alla sorgente diventa importante. Nel sistema interattivo entrambe le estremità producono informazioni e la latenza end-to-end influenza direttamente l'esperienza dell'utente. Le tre classi non sono rigide: un medesimo servizio può combinare contenuti registrati, flussi live e interazione in tempo reale.

Questa classificazione chiarisce perché non esiste una sola architettura multimediale ottima. Lo streaming on-demand può privilegiare efficienza e qualità; una videoconferenza deve limitare la latenza; un gioco remoto deve reagire rapidamente agli input; una trasmissione live verso milioni di utenti deve soprattutto scalare. Codec, protocollo di trasporto, dimensione dei buffer e architettura di distribuzione devono essere scelti in funzione del servizio.

1.4.1 Requisiti applicativi e compromessi

Per confrontare architetture differenti conviene separare i requisiti in almeno cinque assi: qualità percettiva, bitrate, latenza, affidabilità e scalabilità. Essi non sono indipendenti. Una codifica più aggressiva può ridurre la bitrate ma aumentare complessità e latenza; un buffer

più profondo riduce il rischio di interruzioni ma allontana il contenuto dall'istante reale; una protezione FEC più forte riduce l'effetto delle perdite ma consuma banda aggiuntiva.

Servizio	Latenza	Continuità	Affidabilità	Scalabilità
Download file	bassa priorità	non critica	molto alta	media
Streaming on-demand	moderata	molto alta	alta	molto alta
Live broadcast IP	bassa-media	molto alta	alta	molto alta
Videoconferenza	molto bassa	alta	selettiva	alta
Cloud gaming / XR	estremamente bassa	alta	selettiva	alta

Tabella 1.1. Priorità qualitative tipiche. Non sono valori normativi: servono a mostrare perché protocolli e buffer differenti risultano adatti a servizi differenti.

Per questo la qualità dell'esperienza (Quality of Experience, QoE) non coincide con un singolo parametro di rete. Due sistemi con la stessa bitrate possono essere percepiti in modo molto differente se uno introduce interruzioni o ritardo interattivo; due codec con la stessa qualità visiva possono essere inadatti allo stesso servizio se hanno complessità e tempi di codifica molto diversi. Il resto del libro userà quindi le prestazioni come grandezze multidimensionali, non come una graduatoria assoluta.

1.4.2 E-learning e collaborazione

L'e-learning mostra bene la convergenza fra contenuto e comunicazione. Una lezione può comprendere video del docente, audio, presentazioni, lavagna condivisa, chat, documenti, quiz e registrazione. Alcune componenti sono in tempo reale, altre possono essere distribuite in modo asincrono.

Il passaggio importante rispetto al semplice "video della lezione" è che i vari media hanno requisiti differenti. La voce deve arrivare con ritardo ridotto e continuità; una slide deve essere leggibile ma può tollerare una latenza maggiore; un documento può essere trasferito in modo affidabile; una registrazione può essere bufferizzata e fruita successivamente. Il sistema multimediale combina quindi più trasporti e più politiche di qualità all'interno della stessa applicazione.

Lo stesso principio vale per il lavoro collaborativo. La videoconferenza è soltanto una componente: condivisione dello schermo, coediting di documenti, controllo remoto, messaggistica e trasferimento di file richiedono semantiche diverse. Le architetture moderne tendono perciò a integrare media in tempo reale e dati applicativi in un'unica sessione logica pur mantenendo meccanismi di trasporto appropriati ai diversi flussi.

1.4.3 Gaming, realtà virtuale e realtà aumentata

Il gaming è una delle applicazioni nelle quali la latenza è più visibile. Un giocatore genera un comando, il sistema aggiorna lo stato del mondo virtuale e il risultato deve essere mostrato con ritardo sufficientemente basso da non compromettere il controllo. A questa catena si aggiungono grafica, audio spaziale, comunicazione vocale e, nei sistemi distribuiti, sincronizzazione fra più giocatori.

Il cloud gaming modifica ulteriormente l'architettura. Il rendering può essere eseguito su un server remoto; il terminale invia gli input e riceve un flusso video compresso. La potenza di calcolo richiesta al client diminuisce, ma il percorso input-rete-rendering-codifica-rete-decodifica diventa parte del tempo di risposta. In questo caso la compressione più efficiente non è necessariamente la migliore se introduce una latenza di codifica troppo elevata.

Realtà virtuale e aumentata aggiungono requisiti ancora più severi. Il sistema deve aggiornare la scena in funzione dei movimenti dell'utente e può integrare video, grafica sintet-

ica, audio spaziale e dati di sensori. Il principio generale resta lo stesso: qualità percettiva, bitrate, complessità e latenza devono essere ottimizzati insieme.

1.4.4 Streaming e televisione

La televisione tradizionale è un esempio naturale di distribuzione broadcast: lo stesso flusso viene inviato contemporaneamente a molti ricevitori. Questo modello è estremamente efficiente quando un numero elevato di utenti guarda lo stesso contenuto nello stesso momento, ma non permette una personalizzazione individuale del tempo di fruizione.

La televisione digitale ha introdotto multiplexing di più programmi, servizi dati e migliore uso dello spettro, ma il cambiamento più profondo è arrivato con la distribuzione su IP. Nel modello on-demand ogni utente può scegliere il contenuto e il momento di riproduzione. Il prezzo di questa flessibilità è che la rete deve sostenere molte sessioni potenzialmente differenti.

La soluzione moderna combina tre idee. La prima è la CDN, che porta copie dei contenuti vicino agli utenti. La seconda è la suddivisione del contenuto in segmenti indipendenti trasferibili con protocolli web. La terza è lo streaming adattativo: lo stesso contenuto viene preparato a qualità e bitrate differenti e il client sceglie nel tempo la rappresentazione più adatta alla capacità disponibile e al livello del buffer.

Questa strategia è concettualmente diversa dalla televisione broadcast. Nel broadcast il trasmettitore sceglie un flusso e tutti i ricevitori lo accettano; nello streaming adattativo il server rende disponibili più alternative e ogni client effettua una selezione locale. Nei capitoli successivi questo principio verrà collegato a HLS, MPEG-DASH, CMAF e ai protocolli HTTP moderni.

Broadcast e streaming possono anche coesistere. Eventi seguiti da un pubblico molto ampio possono sfruttare una distribuzione massiva, mentre servizi personalizzati, replay, contenuti aggiuntivi e interazione usano un canale IP bidirezionale. Si ritrova così, in forma moderna, il principio delle reti ibride introdotto in precedenza.

1.4.5 Teleconferenze e comunicazione in tempo reale

La teleconferenza nasce storicamente dalla conferenza telefonica e successivamente integra video e dati. Il principio di base è semplice: più partecipanti devono ricevere in tempo utile media prodotti dagli altri partecipanti. La difficoltà cresce rapidamente quando aumentano numero di utenti, qualità video e varietà delle reti di accesso.

In un collegamento punto-punto i media possono essere inviati direttamente fra i due terminali. In una conferenza multiparty è possibile replicare ogni flusso verso tutti gli altri partecipanti, oppure introdurre un nodo intermedio che inoltra selettivamente o compone i media. Il secondo approccio riduce il numero di connessioni che ogni terminale deve sostenere e consente al servizio di adattare la qualità alle capacità dei singoli partecipanti.

Il passaggio dalla telefonia tradizionale alla comunicazione web ha richiesto di risolvere problemi che RTP da solo non affronta: attraversamento di NAT e firewall, scoperta del percorso, negoziazione dei media e sicurezza end-to-end del trasporto. WebRTC nasce proprio come integrazione di questi meccanismi per rendere la comunicazione in tempo reale utilizzabile direttamente nelle applicazioni web. In seguito verranno analizzati RTP/RTCP come fondamento, quindi ICE, STUN, TURN e SRTP, evidenziando che cosa aggiungono rispetto alle architetture SIP e H.323 precedenti.

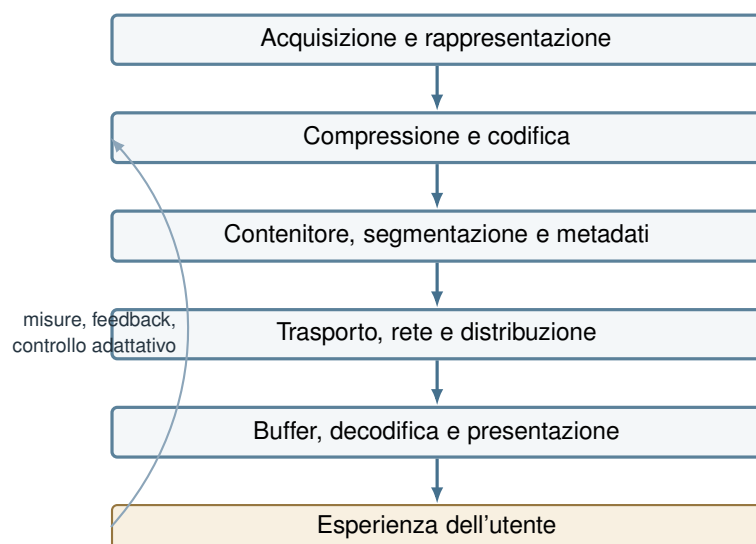


Figura 1.10. Filo conduttore del libro: la qualità finale nasce dall'intera catena, dalla sorgente alla presentazione, con retroazioni che modificano dinamicamente codifica e trasporto.

Sorgenti

2.1 Testo

Il testo di un documento può essere composto da caratteri dell'alfabeto o da ideogrammi, a seconda della lingua di riferimento. Nel caso degli ideogrammi il simbolo grafico definisce integralmente un concetto o una parola, mentre nel caso di alfabeti la parola viene costruita con la giustapposizione dei vari caratteri che la compongono, rispettando un insieme di regole predefinite, tipiche della lingua alla quale ci si riferisce.

Essendo molteplici i linguaggi con cui l'uomo si esprime nelle varie parti del globo, molteplici sono anche gli alfabeti e le regole per comporre le parole o per disegnare gli ideogrammi. Volendo rappresentare queste sorgenti di informazione mediante stringhe di bit è necessario identificare a priori dei sistemi di rappresentazione per i caratteri e per gli ideogrammi delle varie lingue.

2.1.1 ASCII

Considerando che molte delle strutture informatiche sono nate negli USA, il sistema base di codifica dei caratteri è l'American Standard Code for Information Interchange (ASCII), pubblicato nel 1963 e revisionato successivamente nel 1967, 1968, 1977 e 1986.

NUL dec. 0 nullo	BEL dec. 7 beep	BS dec. 8 backspace	HT dec. 9 tab	LF dec. 10 line feed	CR dec. 13 carriage return	ESC dec. 27 escape	DEL dec. 127 delete
-------------------------------	------------------------------	----------------------------------	----------------------------	-----------------------------------	---	---------------------------------	----------------------------------

Figura 2.1. Alcuni caratteri di controllo ASCII storicamente significativi. I valori 0–31 e 127 non rappresentano normali caratteri stampabili.

Il codice ASCII è basato su stringhe di 7 bit, per un totale di 128 diversi caratteri rappresentabili, di cui i primi 33 sono non stampabili e dedicati ad azioni di controllo, mentre i successivi 95 sono stampabili. I caratteri di controllo hanno storicamente avuto funzioni legate alla formattazione del testo e sono stati ereditati da sistemi di editing basati su terminali composti solo da una scheda grafica e da un sistema di trasmissione, senza nessuna capacità di elaborazione a bordo.

I 33 caratteri di controllo vanno dal codice 0 al codice 31, a cui si somma il codice 127. Nella figura 2.1 vengono riportati questi codici di controllo. Risulta interessante commentare alcuni caratteri particolari: 0 per indicare nessun carattere; 7 per fare emettere un beep; 8 per cancellare il carattere precedente; 127 per cancellare il carattere successivo; 9 per un tabulazione orizzontale; 11 per una tabulazione verticale; 10 per andare alla riga successiva; 13 per far tornare a capo il carrello (stampante) o il cursore (monitor); 6 e 21 per fornire ACK e NACK in meccanismi per il controllo degli errori. Particolarmente interessante il 27, denominato escape, che consente di rappresentare una stringa complessa di caratteri di controllo, indicati immediatamente dopo tale codice, ed è in grado di garantire trasparenza, cioè di evitare confusione di interpretazione tra i caratteri dopo l'escape e gli stessi caratteri usati liberamente nel testo.

I 95 caratteri stampabili, come mostrato in figura 2.2, contemplano le lettere maiuscole

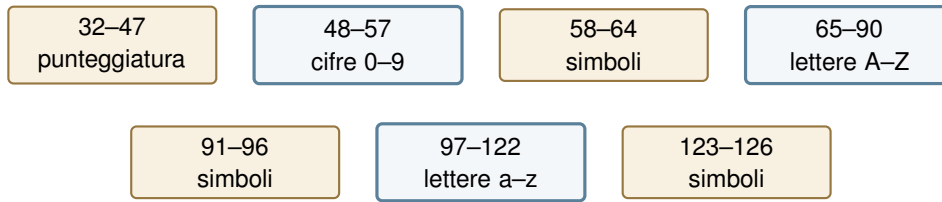


Figura 2.2. Organizzazione dei 95 caratteri stampabili ASCII in intervalli contigui di codici.

da 65 a 90, quelle minuscole da 97 a 122, i numeri da 48 a 57. Tutti gli altri intervalli sono caratterizzati da segni di interpunzione, simboli e parentesi.

Il codice ASCII non è capace di rappresentare caratteri particolari legati a lingue diverse dall'inglese.

2.1.2 ISO-8859

La famiglia ISO-8859 rappresenta un passaggio storico importante fra ASCII e Unicode. Ogni parte usa parole di 8 bit. I valori da 0 a 127 mantengono la compatibilità con ASCII; la metà superiore dello spazio di codifica viene dedicata ai caratteri necessari a un determinato gruppo di lingue. Il vantaggio è che ogni carattere occupa un solo byte; il limite è che nessuna singola parte può rappresentare tutte le scritture contemporaneamente.

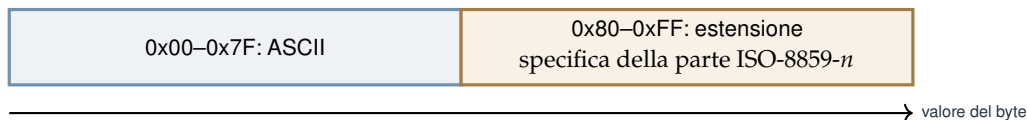


Figura 2.3. Principio della famiglia ISO-8859: la metà bassa resta compatibile con ASCII, la metà alta cambia significato secondo la parte scelta.

Fra le parti più note vi sono ISO-8859-1 per molte lingue dell'Europa occidentale, ISO-8859-2 per l'Europa centrale, ISO-8859-5 per il cirillico, ISO-8859-6 per l'arabo, ISO-8859-7 per il greco, ISO-8859-8 per l'ebraico, ISO-8859-9 per il turco e ISO-8859-11 per il thai. Le parti arrivano fino a ISO-8859-16; il numero 12 non fu mai pubblicato come parte dello standard.

La frammentazione è strutturale: lo stesso valore di byte può rappresentare caratteri differenti se cambia la parte ISO-8859 usata. Un file privo dell'informazione sulla codifica non è quindi interpretabile in modo univoco. Unicode nasce anche per separare il concetto di *carattere* dalla particolare codifica in byte e per assegnare un repertorio globale di punti di codice.

2.1.3 Unicode, UCS e UTF-8

UTF-8 (8-bit Unicode Transformation Format) è una codifica a lunghezza variabile compatibile con US-ASCII. Nella definizione attuale, adottata anche da RFC 3629, i punti di codice compresi tra U+0000 e U+10FFFF sono rappresentati mediante sequenze da uno a quattro byte.

Nel caso a un solo byte il bit più significativo è posto a 0 e i sette bit rimanenti contengono direttamente il valore ASCII. Per una sequenza lunga $k > 1$ byte, il primo byte contiene k bit a 1 seguiti da uno 0; i byte successivi iniziano sempre con il prefisso 10. I bit rimanenti sono quelli disponibili per il punto di codice. Pertanto il numero b di bit disponibili nella struttura della sequenza è

$$b = \begin{cases} 7 & \text{se } k = 1, \\ (7 - k) + 6(k - 1) = 5k + 1 & \text{se } 2 \leq k \leq 4. \end{cases}$$

Si ottengono quindi 7, 11, 16 e 21 bit strutturalmente disponibili rispettivamente per $k = 1, 2, 3, 4$. Nel caso a quattro byte non tutte le configurazioni dei 21 bit sono valide: il massimo punto di codice ammesso è U+10FFFF. Le forme storiche UTF-8 a cinque e sei byte non fanno parte della definizione moderna.

La corrispondenza fra punti di codice e caratteri è definita dallo Universal Character Set (UCS) di ISO/IEC 10646 e dallo standard Unicode. Il Basic Multilingual Plane (BMP) comprende i punti di codice U+0000–U+FFFF: in UTF-8 essi occupano da uno a tre byte, a seconda del valore del punto di codice. I punti di codice oltre il BMP richiedono quattro byte.

2.2 Immagini

Le immagini in formato digitale sono composte da un insieme di elementi giustapposti in uno spazio bidimensionale. Questi elementi sono chiamati pixel, acronimo che nasce dalle parole picture element, dove picture viene abbreviato in pix.

2.2.1 Pixel

Il pixel viene spesso anche chiamato "punto" dell'immagine: in realtà rappresenta un concetto astratto, non legato alla geometria dell'elemento bensì al contenuto di informazione che deve rappresentare. Il pixel rappresenta quindi un campione dell'immagine che viene fisicamente rappresentato con una determinata geometria (quadrato, cerchio, etc) dipendente dal sistema di riproduzione.

Ad ogni pixel è associata un'intensità variabile. Nel caso di immagini in bianco e nero l'intensità del pixel può essere solo binaria, cioè il pixel può essere solo acceso (bianco) o spento (nero). Nel caso di immagini a scale di grigio l'intensità del pixel è un valore scalare che può variare in modo discreto all'interno di un certo range, ad esempio da 0 a 255, dove 0 rappresenta il nero e 255 il bianco, con 254 ulteriori possibili tonalità di grigio intermedie. Nel caso di immagini a colori, l'intensità del pixel è un vettore a 3 o 4 dimensioni, dipendentemente dal sistema utilizzato per la rappresentazione del colore.

2.2.2 Piani colore

La rappresentazione numerica del colore nasce da un fatto fisiologico: la retina umana possiede tre classi di coni, normalmente indicate come S, M e L. Le rispettive sensibilità spettrali sono ampie e fortemente sovrapposte. Non è quindi corretto pensare a tre sensori che misurano una singola lunghezza d'onda "blu", "verde" e "rossa": ciascuna classe integra energia su una regione estesa dello spettro e il colore percepito dipende dal rapporto fra le tre risposte.

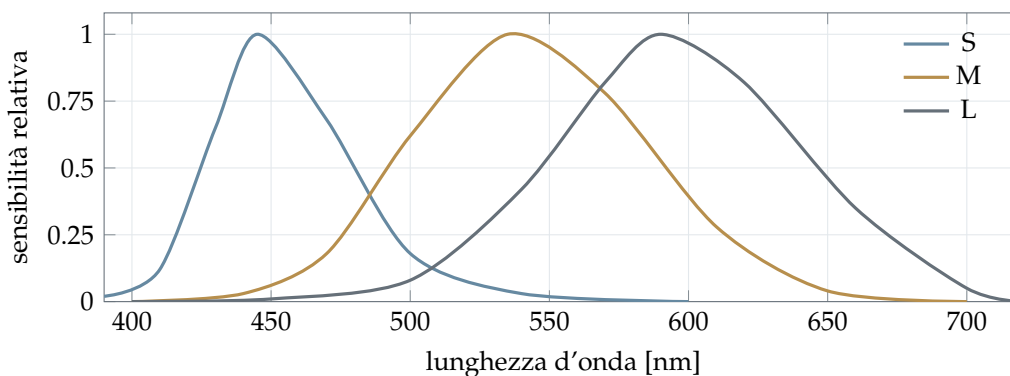


Figura 2.4. Andamento qualitativo delle tre famiglie di coni. Le curve sono schematiche e servono a mostrare la forte sovrapposizione delle sensibilità, non a fornire dati fisiologici di riferimento.

Due distribuzioni spettrali fisicamente differenti possono produrre risposte S, M e L sufficientemente simili da essere percepite come lo stesso colore. Questo fenomeno è detto *metamerismo*. È la ragione per cui un display non deve riprodurre lo spettro di una sorgente reale per dare la stessa sensazione cromatica: può generare un opportuno mix di tre primari. La tricromia è quindi una proprietà del sistema di visione e della scelta dei primari, non una decomposizione unica della luce fisica.

RGB

RGB è il modello additivo più comune nei sistemi di acquisizione e visualizzazione. Le tre componenti descrivono la quantità dei primari rosso, verde e blu. Sommare luce aumenta l'energia emessa; nel modello normalizzato $(0,0,0)$ rappresenta il nero e $(1,1,1)$ il bianco dello spazio colore scelto.

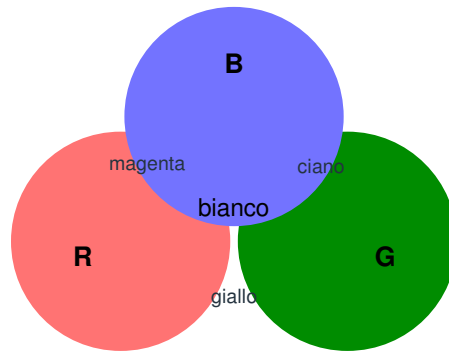


Figura 2.5. Sintesi additiva RGB: la sovrapposizione dei primari produce colori secondari e, nel punto di equilibrio definito dallo spazio colore, il bianco.

Con 8 bit per ciascuna componente sono disponibili

$$N_{RGB} = 2^8 2^8 2^8 = 2^{24} = 16,777,216$$

terne numeriche differenti.

Dimostrazione. Ogni componente a 8 bit possiede $2^8 = 256$ configurazioni. Le scelte di R, G e B sono indipendenti, quindi per il principio di moltiplicazione il numero delle terne è $256^3 = (2^8)^3 = 2^{24} = 16,777,216$. Questo conclude la dimostrazione. Il numero conta configurazioni digitali, non colori certamente distinguibili dall'occhio: la discriminabilità dipende da luminanza, contesto, osservatore e spazio colore.

La profondità di colore si esprime spesso in bit per pixel (bpp). 24 bpp corrisponde tipicamente a 8 bit per componente RGB; 32 bpp può aggiungere un canale alpha o semplicemente allineamento; 48 bpp usa 16 bit per componente ed è utile nelle catene di elaborazione in cui più operazioni successive potrebbero amplificare gli errori di arrotondamento. Formati professionali e HDR usano spesso 10 o 12 bit per componente.

I valori memorizzati non sono necessariamente proporzionali alla luce fisica. In un modello semplificato a legge di potenza, con $C, L \in [0, 1]$,

$$L = C^\gamma, \quad C = L^{1/\gamma}.$$

Dimostrazione. La prima relazione definisce il modello. Poiché $\gamma > 0$, la funzione $C \mapsto C^\gamma$ è monotona su $[0, 1]$ e possiede inversa. Elevando entrambi i membri alla potenza $1/\gamma$ si ottiene $L^{1/\gamma} = (C^\gamma)^{1/\gamma} = C$. Questo conclude la verifica dell'inversa.

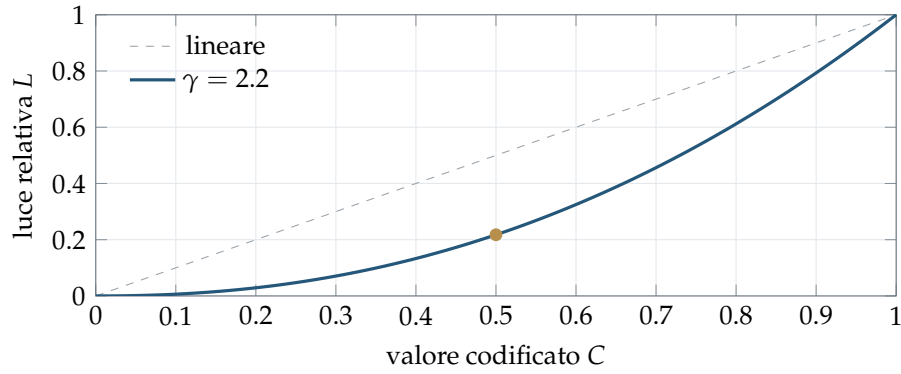


Figura 2.6. Esempio di modello a legge di potenza. Un valore codificato pari a 0.5 non corrisponde necessariamente al 50% della luminanza fisica.

Il valore $\gamma \simeq 2.2$ è un'utile approssimazione didattica, non la definizione esatta di sRGB. sRGB usa una funzione di trasferimento a tratti, scelta per motivi di compatibilità, codifica percettiva ed efficienza numerica.

CMY e CMYK

Nella stampa il meccanismo fisico è sottrattivo: pigmenti e inchiostri assorbono parte della luce incidente. Nel modello ideale normalizzato CMY le componenti sono i complementi di RGB:

$$C = 1 - R, \quad M = 1 - G, \quad Y = 1 - B,$$

con inversa

$$R = 1 - C, \quad G = 1 - M, \quad B = 1 - Y.$$

Dimostrazione. Dalla relazione $C = 1 - R$ segue immediatamente $R = 1 - C$ sottraendo C da entrambi i membri; lo stesso vale per M/G e Y/B . Applicare due volte la trasformazione restituisce quindi il valore iniziale. Questo conclude la dimostrazione.

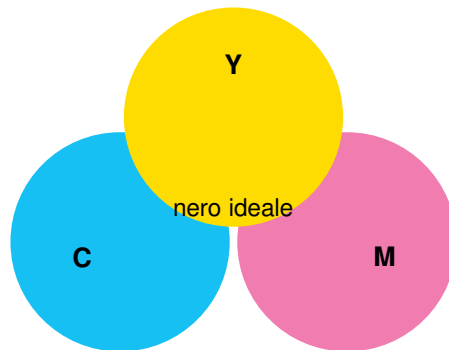


Figura 2.7. Sintesi sottrattiva ideale CMY. Nella stampa reale la sovrapposizione dei tre inchiostri non produce un nero perfetto e conviene introdurre esplicitamente la componente K.

CMYK introduce appunto una componente nera K . La conversione non è unica: le strategie di black generation e under-color removal sono parte del processo di stampa. In un modello algebrico semplice,

$$R = (1 - C)(1 - K), \quad G = (1 - M)(1 - K), \quad B = (1 - Y)(1 - K).$$

Una possibile scelta inversa è

$$K = 1 - \max\{R, G, B\},$$

e, per $K < 1$,

$$C = \frac{1 - R - K}{1 - K}, \quad M = \frac{1 - G - K}{1 - K}, \quad Y = \frac{1 - B - K}{1 - K}.$$

Dimostrazione. Dalla prima relazione diretta, $R = (1 - C)(1 - K)$; se $K < 1$, dividendo per $1 - K$ si ottiene $1 - C = R/(1 - K)$ e quindi $C = (1 - R - K)/(1 - K)$. Gli altri due casi sono identici. La scelta di K garantisce che almeno una fra C , M e Y sia zero e che tutte le componenti restino nel range normalizzato. Per $R = G = B = 0$ risulta $K = 1$ e le frazioni diventano indeterminate: si adotta allora la convenzione $C = M = Y = 0$, $K = 1$. Questo conclude la dimostrazione per il modello scelto.

YUV, Y'PbPr e Y'CbCr

Le sigle YUV, Y'PbPr e Y'CbCr sono spesso usate impropriamente come sinonimi. YUV appartiene storicamente alla codifica analogica del colore televisivo; Y'PbPr descrive componenti analogiche; Y'CbCr è la forma numerica usata in gran parte del video digitale. In tutti i casi l'idea centrale è separare una componente di luma da due componenti di differenza di colore, così da poter trattare con precisione diversa la struttura luminosa e quella cromatica.

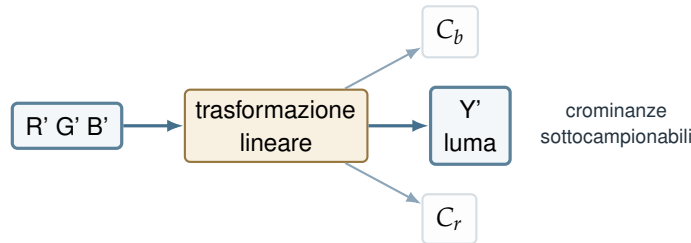


Figura 2.8. Separazione della luma dalle differenze di colore. L'apice ricorda che le componenti R'G'B' sono normalmente non lineari.

Posti K_r e K_b i coefficienti definiti dal sistema e $K_g = 1 - K_r - K_b$, una rappresentazione normalizzata è

$$Y = K_r R + K_g G + K_b B,$$

$$C_b = \frac{B - Y}{2(1 - K_b)}, \quad C_r = \frac{R - Y}{2(1 - K_r)}.$$

Queste relazioni sono la definizione del modello normalizzato; le codifiche digitali effettive aggiungono offset, scale e range numerici specificati dallo standard.

Le trasformazioni inverse sono

$$R = Y + 2(1 - K_r)C_r,$$

$$B = Y + 2(1 - K_b)C_b,$$

$$G = Y - 2 \frac{K_r(1 - K_r)C_r + K_b(1 - K_b)C_b}{K_g}.$$

Dimostrazione. Dalla definizione di C_r si ha $R - Y = 2(1 - K_r)C_r$, da cui segue la prima relazione. Analogamente $B - Y = 2(1 - K_b)C_b$ produce la seconda. Sostituendo R e B in $Y = K_r R + K_g G + K_b B$ e usando $K_r + K_g + K_b = 1$, i termini in Y si raccolgono e si cancellano; isolando G si ottiene la terza relazione. Questo conclude la dimostrazione.

Per BT.601 si usano $K_r = 0.299$ e $K_b = 0.114$; per BT.709 $K_r = 0.2126$ e $K_b = 0.0722$. Sono valori di specifica. Il fatto che i coefficienti cambino mostra che una terna numerica Y'CbCr non è interpretabile completamente senza sapere quale sistema colorimetrico la accompagni.

Un grigio neutro ha $R = G = B = g$. Allora $Y = g$ e le due crominanze sono nulle.

Dimostrazione. Poiché $K_r + K_g + K_b = 1$, $Y = (K_r + K_g + K_b)g = g$. Ne segue $B - Y = 0$ e $R - Y = 0$, quindi $C_b = C_r = 0$. Questo conclude la dimostrazione.

HSV

HSV separa tre quantità utili all'interazione grafica: Hue, Saturation e Value. La tonalità H descrive la posizione sul ciclo cromatico; la saturazione S misura la distanza dall'asse dei grigi; il valore V è legato alla massima componente RGB. È una costruzione geometrica utile per selettori di colore e algoritmi grafici, non un modello fisiologico completo della percezione.

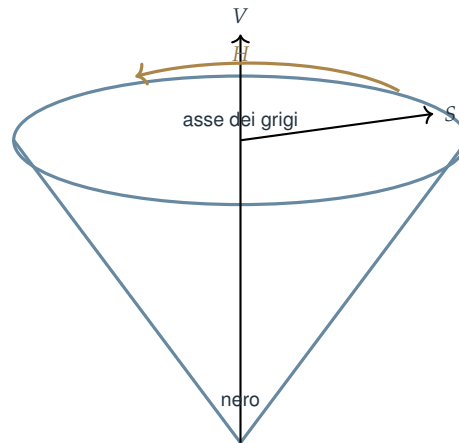


Figura 2.9. Rappresentazione geometrica qualitativa HSV: l'altezza controlla il valore, la distanza dall'asse la saturazione e l'angolo la tonalità.

Per una terna $R, G, B \in [0, 1]$ si definiscono $M = \max\{R, G, B\}$, $m = \min\{R, G, B\}$ e $C = M - m$; quindi $V = M$ e, se $V > 0$, $S = C/V$, mentre $S = 0$ quando $V = 0$. La tonalità si ottiene con una definizione a tratti che identifica quale componente raggiunge il massimo e misura la posizione relativa delle altre due. Quando $C = 0$ il colore è acromatico e H non ha significato fisico: qualsiasi valore numerico scelto è una convenzione software.

Il dettaglio algoritmico della conversione RGB-HSV dipende dalla convenzione sugli intervalli numerici e non verrà usato nelle derivazioni successive. È più importante riconoscere che HSV è una ri-parametrizzazione non lineare di RGB, utile per manipolazione e interfacce, mentre Y'CbCr nasce per rappresentazione e compressione del video.

2.2.3 Colorimetria moderna e HDR

Una terna RGB, da sola, non identifica in modo completo un colore. Per interpretarla occorre conoscere almeno i primari usati per rosso, verde e blu, il punto di bianco e la funzione che collega i valori numerici alla luce emessa o catturata. Due dispositivi possono quindi ricevere la stessa terna numerica e produrre colori diversi se utilizzano spazi colore differenti.

Questo chiarisce il ruolo degli spazi colore standardizzati. sRGB è stato progettato per fornire un riferimento comune per immagini e applicazioni informatiche; BT.709 ha svolto un ruolo analogo nel video HD tradizionale. Sistemi più recenti utilizzano gamut più estesi, come quelli associati a Display P3 o BT.2020, per rappresentare colori che non rientrano nel triangolo dei primari precedenti.

L'altra evoluzione fondamentale è l'High Dynamic Range (HDR). Nello Standard Dynamic Range (SDR) la scena viene mappata in un intervallo di luminanza relativamente limitato. L'HDR cerca invece di conservare una separazione maggiore fra zone scure e zone molto luminose. Questo richiede non soltanto display capaci di produrre livelli di luminanza più estesi, ma anche una rappresentazione numerica e una funzione di trasferimento adeguate.

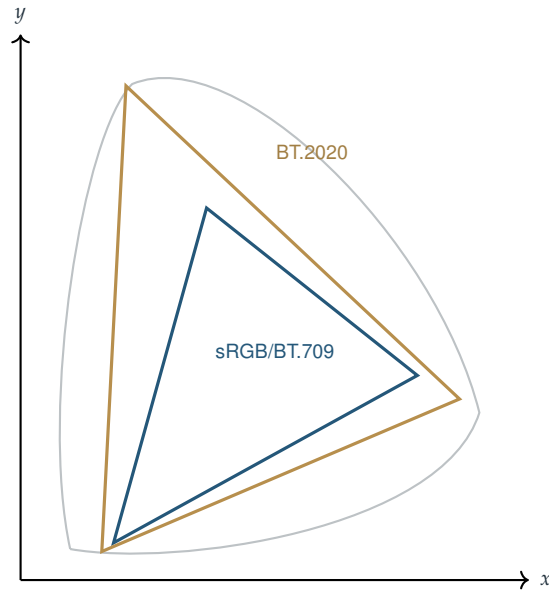


Figura 2.10. Confronto schematico fra gamut nel diagramma cromatico CIE xy . I vertici BT.709 e BT.2020 sono riportati in posizione approssimativa per mostrare l'estensione relativa; il contorno spettrale è volutamente didattico.

Le due famiglie oggi più importanti sono PQ, che descrive una codifica legata alla luminanza assoluta, e HLG, che è stata pensata per una migliore compatibilità con catene di produzione e distribuzione televisiva. Le relative funzioni sono definite dagli standard: sono quindi specifiche, non risultati matematici da dimostrare.

L'aumento della dinamica rende più importante anche la profondità di quantizzazione. Con 8 bit per componente sono disponibili 256 livelli numerici; con 10 bit i livelli diventano 1024 e con 12 bit 4096. Una maggiore profondità riduce la distanza fra livelli consecutivi e quindi il rischio di banding nelle variazioni graduali, soprattutto quando il segnale deve coprire un intervallo percettivo più ampio.

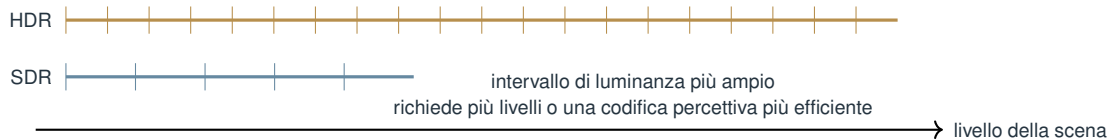


Figura 2.11. Rappresentazione concettuale del rapporto fra dinamica e profondità numerica. HDR non significa semplicemente “più bit”: funzione di trasferimento, range della scena e capacità del display devono essere coerenti.

Per valutare il costo grezzo della maggiore profondità consideriamo un'immagine video di $W \times H$ pixel, frequenza di quadro F e campionamento 4:4:4 con tre componenti, ciascuna rappresentata con b bit. La bitrate non compressa è

$$R_{4:4:4} = 3bWHF.$$

Dimostrazione. Ogni quadro contiene WH pixel. Nel campionamento 4:4:4 ogni pixel ha un campione per ciascuna delle tre componenti, quindi contiene $3WH$ campioni. Ogni campione richiede b bit, per cui un quadro occupa $3bWH$ bit. Trasmettendo F quadri al secondo si ottiene

$$R_{4:4:4} = F(3bWH) = 3bWHF.$$

Questo conclude la dimostrazione.

Nel campionamento 4:2:0, invece, per ogni quattro campioni di luma vi sono un campione C_b e un campione C_r . Il numero medio di campioni per pixel è quindi $1 + 1/4 + 1/4 = 3/2$ e la bitrate grezza diventa

$$R_{4:2:0} = \frac{3}{2}bWHF.$$

Dimostrazione. Su un quadro $W \times H$ il piano di luma contiene WH campioni. Le due crominanze hanno ciascuna meta' della risoluzione orizzontale e meta' di quella verticale, quindi contengono ciascuna $WH/4$ campioni. Il numero totale di campioni è

$$WH + \frac{WH}{4} + \frac{WH}{4} = \frac{3}{2}WH.$$

Moltiplicando per b bit per campione e per F quadri al secondo si ottiene

$$R_{4:2:0} = \frac{3}{2}bWHF.$$

Questo conclude la dimostrazione.

A parita' di tutte le altre condizioni, passare da 8 a 10 bit per campione aumenta quindi la bitrate grezza del fattore

$$\frac{R_{10}}{R_8} = \frac{10}{8} = 1.25.$$

Dimostrazione. Dalle due relazioni appena dimostrate la bitrate grezza è direttamente proporzionale a b . Indicando con K tutti i fattori indipendenti dalla profondità, si può scrivere $R_b = Kb$. Ne segue

$$\frac{R_{10}}{R_8} = \frac{K10}{K8} = \frac{10}{8} = 1.25.$$

Quindi l'aumento è del 25%, prima di qualsiasi compressione. Questo conclude la dimostrazione.

La maggiore profondità non implica automaticamente maggiore qualità: se sensore, elaborazione, codec o display non conservano realmente l'informazione aggiuntiva, i bit supplementari non producono un beneficio percettivo. HDR, gamut, profondità e capacità del display vanno perciò considerati come una catena unica.

2.2.4 Dimensione, densità e risoluzione

Tre grandezze vengono spesso confuse. La *dimensione raster* indica il numero di campioni orizzontali e verticali, per esempio 1920×1080 . La *dimensione fisica* indica larghezza e altezza sul supporto o sul display. La *densità* indica quanti campioni o punti fisici vengono collocati per unità di lunghezza.

PPI (pixel per inch) è appropriato per descrivere la densità di una griglia di pixel. DPI (dots per inch) è invece particolarmente usato per la stampa e descrive i punti fisici prodotti dal dispositivo. Un pixel dell'immagine può richiedere più punti di stampa per riprodurre correttamente colore e intensità; PPI e DPI non sono quindi sinonimi.

Un pollice misura esattamente 25.4 mm. Una densità di d DPI equivale pertanto a

$$d_{\text{mm}} = \frac{d}{25.4} \text{ punti/mm.}$$

Dimostrazione. Se in 25.4 mm sono presenti d punti, dividendo entrambi i membri per 25.4 si ottengono $d/25.4$ punti per millimetro. Questo conclude la dimostrazione.

Per esempio 300, 600, 1200 e 2400 DPI corrispondono rispettivamente a circa 11.8, 23.6, 47.2 e 94.5 punti/mm.

Se un'immagine contiene $W \times H$ pixel e ogni pixel occupa b bit, la dimensione raw è

$$S = \frac{bWH}{8} \text{ byte.}$$

Dimostrazione. Il raster contiene WH pixel. Moltiplicando per b bit/pixel si ottengono bWH bit. Poiché un byte contiene 8 bit, la dimensione in byte è $bWH/8$. Questo conclude la dimostrazione.

Un'immagine 1024×768 a 24 bpp richiede quindi 2,359,296 byte, circa 2.36 MB in unità decimali oppure 2.25 MiB in unità binarie. Il valore è indipendente dalle dimensioni fisiche con cui l'immagine verrà mostrata: cambiare la densità di visualizzazione non cambia il numero di pixel del file.

2.2.5 Fattore di forma

Il fattore di forma (aspect ratio) è il rapporto fra larghezza e altezza dell'area visibile. 4:3 è tipico della televisione storica; 16:9 è il rapporto dominante nell'HD e nell'UHD; 16:10 e 5:4 sono comparsi frequentemente nei display informatici. Il rapporto può essere espresso come frazione oppure come numero reale.

È importante distinguere il display aspect ratio dal rapporto d'aspetto del pixel. In alcuni sistemi digitali storici i campioni non corrispondono a pixel geometricamente quadrati: il raster numerico e la forma finale dell'immagine non hanno allora lo stesso rapporto. Il player deve conoscere questa informazione per evitare immagini stirate o compresse orizzontalmente.

Il cinema ha utilizzato una varietà ancora maggiore di rapporti, legata a pellicole, ottiche anamorfiche e formati di proiezione. La conversione fra rapporti differenti richiede una scelta: crop, letterboxing/pillarboxing oppure deformazione geometrica. Solo le prime due preservano le proporzioni degli oggetti.

2.2.6 Monitor

Gli schermi dei calcolatori hanno tipicamente una risoluzione di visualizzazione prefissata in funzione delle loro caratteristiche costruttive. Questa risoluzione viene detta risoluzione massima. Risoluzioni inferiori sono possibili mediante strategie di interpolazione.

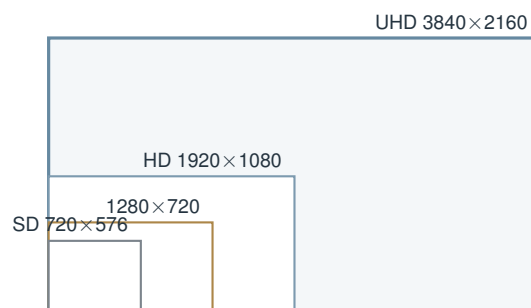


Figura 2.12. Confronto geometrico fra alcuni raster digitali. Le dimensioni sono espresse in campioni e non descrivono da sole dimensione fisica, densità o qualità percepita del display.

Le principali risoluzioni storicamente utilizzate sono riportate in figura 2.12. La risoluzione, tuttavia, è soltanto uno dei parametri di un display: frequenza di aggiorna-

mento, contrasto, luminanza, gamut, profondità di colore, tempo di risposta e tecnologia di emissione o modulazione della luce influenzano direttamente la qualità percepita. Il CRT viene mantenuto nel seguito perché spiega l'origine di scansione, persistenza, interlacciamento e gamma; LCD e OLED rappresentano invece le famiglie più importanti per comprendere i display moderni.

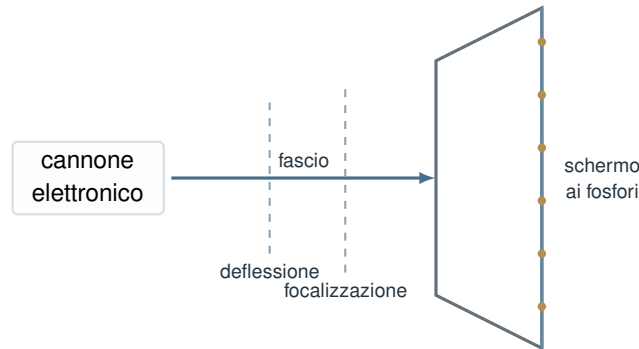


Figura 2.13. Schema funzionale di un CRT. Il fascio viene generato, focalizzato e deviato sulla superficie ai fosfori; la scansione temporale costruisce l'immagine riga dopo riga.

CRT

Nel Cathode Ray Tube un cannone elettronico genera un fascio di elettroni, lo accelera e lo focalizza verso la superficie interna dello schermo. I fosfori emettono luce quando vengono eccitati dal fascio e continuano a emetterla per un intervallo finito, detto persistenza. La persistenza deve essere sufficiente a mascherare il tempo fra due eccitazioni successive, ma non così lunga da lasciare scie visibili sugli oggetti in movimento.

Il cannone comprende un catodo riscaldato, elettrodi di controllo e anodi di accelerazione e focalizzazione. Campi magnetici o elettrici deviano il fascio orizzontalmente e verticalmente. Il sistema costruisce l'immagine per scansione: percorre una riga, effettua la ritraccia orizzontale, passa alla riga successiva e, dopo l'ultima riga, esegue la ritraccia verticale. La luminanza locale viene modulata variando l'intensità del fascio.

Nei CRT a colori tre fasci, oppure tre componenti dello stesso sistema di generazione, eccitano fosfori rossi, verdi e blu. Una shadow mask o una struttura equivalente fa sì che ciascun fascio colpisca principalmente i fosfori del proprio colore. La geometria introduce vincoli di convergenza e messa a fuoco che non esistono nello stesso modo nei pannelli a matrice fissa.

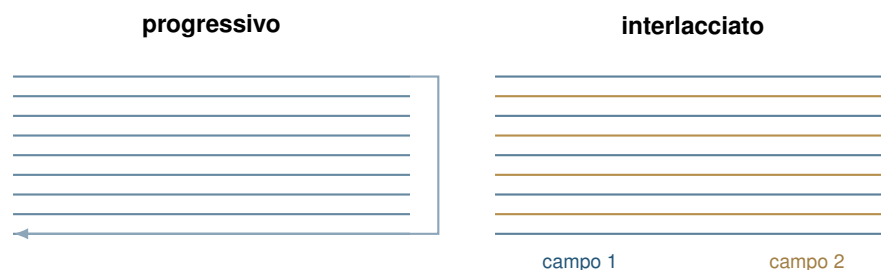


Figura 2.14. Scansione progressiva e interlacciata. Nell'interlacciato i due campi contengono righe alternate acquisite o mostrate in istanti differenti.

L'interlacciamento è nato per aumentare la frequenza di aggiornamento percepita senza raddoppiare la banda necessaria a trasmettere un quadro completo. Il primo campo contiene una famiglia di righe alternate, il secondo le righe mancanti. Il prezzo è temporale: i due

campi rappresentano istanti differenti, quindi un oggetto in moto può apparire con bordi “a pettine” quando i campi vengono combinati come se fossero simultanei.

La terminologia è importante. Nei sistemi 625/50 vi sono 625 linee nominali per quadro e 50 campi/s, quindi 25 quadri/s; nei sistemi 525/59.94 vi sono 525 linee nominali e circa 59.94 campi/s, quindi circa 29.97 quadri/s. Questi numeri descrivono la scansione analogica. Non definiscono una risoluzione orizzontale in pixel. Raster come 720×576 o 720×480 appartengono alla successiva rappresentazione digitale campionata, trattata nella sezione sul video.

La frequenza della rete elettrica ha influenzato storicamente la scelta delle frequenze di campo, perché una relazione stabile con 50 o 60 Hz riduceva la visibilità delle interferenze di alimentazione nei ricevitori analogici. Non è però corretto interpretare la rete elettrica come un vero canale di sincronizzazione del televisore: i sincronismi necessari alla scansione sono trasmessi nel segnale video stesso.

LCD

Un sistema alternativo per realizzare un monitor è quello offerto dai Liquid Crystal Display (LCD). In questo caso il dispositivo è piatto e sottile e contiene un certo numero di pixel predefiniti. Ogni pixel è composto da molecole di cristallo liquido sospese tra due elettrodi trasparenti e da due filtri di polarizzazione con assi di polarizzazione perpendicolari, come mostrato in figura 2.15. I cristalli liquidi presenti nel dispositivo inducono una rotazione di 90 gradi alla luce, per cui, in condizioni di riposo, la luce entra nel dispositivo, viene ruotata di 90 gradi dal primo polarizzatore, di ulteriori 90 gradi dal cristallo liquido e può uscire dal secondo polarizzatore, con un'attenuazione dovuta al passaggio per i polarizzatori, ma con una sostanziale trasparenza del dispositivo.

Quando viene applicato un campo elettrico agli elettrodi, i cristalli liquidi tendono ad orientarsi parallelamente al campo elettrico, limitando la rotazione della luce entrante e diminuendo conseguentemente la trasparenza. Quando i cristalli sono completamente orientati nella direzione del campo elettrico, la luce non subisce alcuna rotazione a causa dei cristalli liquidi ma è soggetta all'ortogonalità dei due polarizzatori, per cui il pixel diventa completamente opaco. Gli LCD sono opportunamente illuminati lateralmente, da luce artificiale, oppure presentano uno specchio posto sul fondo, per riflettere la luce naturale. Per realizzare schermi a colori il pixel viene composto da tre celle, filtrate rispettivamente in rosso, verde e blu.

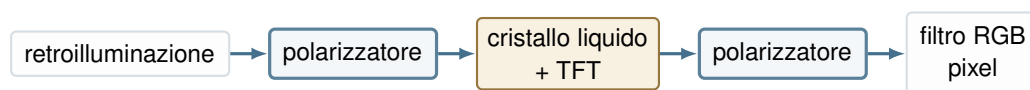


Figura 2.15. Catena ottica semplificata di un LCD a matrice attiva. Il pixel non emette direttamente luce: modula la retroilluminazione mediante polarizzatori e cristalli liquidi.

Negli ultimi anni i display LCD di tipo passivo, appena illustrati, sono stati migliorati con l'accoppiamento di un transistor MOS a ogni singolo pixel componente lo schermo. Questi nuovi display vengono chiamati a matrice attiva o (Thin Film Transistor) TFT e garantiscono una maggiore velocità di scansione dello schermo e contrasti nettamente superiori.

Display emissivi e tecnologie moderne

Il CRT è un display emissivo: il punto luminoso produce direttamente la luce osservata. L'LCD, invece, è prevalentemente un dispositivo di modulazione: una sorgente di retroilluminazione produce la luce e il pixel ne regola la trasmissione. Questa differenza resta fondamentale per comprendere le tecnologie moderne.

Nei display OLED ogni subpixel emette direttamente luce. Spegnendo il subpixel si può ottenere un livello di nero molto basso senza dover bloccare una retroilluminazione comune. Ne conseguono contrasto elevato e risposta rapida. La controparte è che i materiali emissivi possono avere efficienze e invecchiamenti differenti fra i colori; inoltre la luminanza sostenibile dipende dalla struttura del pannello e dalla modalità di pilotaggio.

I display MicroLED seguono un principio analogo ma utilizzano elementi LED inorganici di dimensione microscopica. L'obiettivo è combinare i vantaggi dell'emissione diretta con elevata luminanza ed efficienza. La difficoltà principale è tecnologica: realizzare e trasferire un numero enorme di elementi uniformi e funzionanti con resa produttiva elevata.

Gli LCD restano competitivi grazie alla retroilluminazione evoluta. La retroilluminazione può essere suddivisa in molte zone controllate indipendentemente; il local dimming riduce la luce dietro le regioni scure e la aumenta in quelle luminose. MiniLED non cambia il principio di funzionamento del pixel LCD: aumenta soprattutto il numero e la precisione delle sorgenti usate per la retroilluminazione.

I display al plasma hanno avuto un ruolo storico importante nei grandi schermi emissivi, ma non rappresentano più la direzione principale dello sviluppo. Vengono mantenuti come esempio dell'evoluzione che ha portato a ricercare pixel emissivi indipendenti.

Per i videoproiettori continuano a esistere architetture differenti. Una famiglia usa matrici LCD o dispositivi riflettenti; un'altra usa Digital Micromirror Device (DMD), nei quali una matrice di microspecchi controlla la quantità di luce inviata verso lo schermo. Il principio è ancora diverso da quello di un pannello: il pixel percepito nasce dalla modulazione di una sorgente luminosa esterna.

Dal punto di vista multimediale la tecnologia del display conta perché modifica il risultato visibile della stessa sequenza numerica. Un contenuto HDR, per esempio, non può essere valutato soltanto dal file: serve conoscere il gamut, il livello di nero, la luminanza di picco e la funzione di trasferimento effettivamente realizzabile dal display. Quando il dispositivo non riproduce l'intervallo richiesto si applica un tone mapping, cioè una trasformazione che comprime il contenuto nell'intervallo disponibile cercando di conservare le relazioni percettivamente più importanti.

2.2.7 Elaborazione grafica e GPU

La scheda video storica nasce come memoria di framebuffer letta periodicamente per generare il segnale destinato al monitor. Questo modello rimane concettualmente valido, ma oggi la parte dominante è la Graphics Processing Unit (GPU), un processore altamente parallelo che esegue trasformazioni geometriche, rasterizzazione, texture, filtri, compositing e calcolo generale.

Il cambiamento rispetto ai primi acceleratori grafici è importante: non si dispone più soltanto di primitive fisse come il riempimento di poligoni. Le GPU moderne eseguono programmi paralleli e vengono usate anche in molte fasi della catena multimediale, fra cui conversioni di colore, scaling, filtri, effetti, elaborazione di immagini e, quando sono presenti blocchi dedicati, codifica e decodifica video.

Questa evoluzione ha due conseguenze. La prima è prestazionale: molte operazioni indipendenti sui pixel o sui blocchi possono essere svolte in parallelo. La seconda è architettonica: acquisizione, elaborazione, codifica e visualizzazione possono condividere memoria e acceleratori, riducendo copie e latenza. Nei sistemi realtime questa organizzazione può essere importante quanto l'efficienza teorica del codec.

2.2.8 Formati raster non compressi

I formati raw sono formati di rappresentazione delle immagini digitali senza alcuna compressione.

PBM/PGM/PPM

Un formato veramente semplice di rappresentazione delle immagini è dato dai sistemi Portable Pixmap File Format (PPM), Portable Graymap File Format (PGM) e Portable Bitmap File Format (PBM). I tre formati si distinguono per la possibilità di rappresentare immagini in solo puro bianco e nero (PBM) con 1 bit per pixel, in scale di grigio (PGM) con 8 bit per pixel e a colori (PPM) con 24 bit per pixel.

Ogni formato supporta due metodologie di rappresentazione dei dati: tramite codice ASCII oppure tramite una rappresentazione binaria. In tutti i casi il file ha sempre un header ASCII nel quale viene specificato: la tipologia di file, secondo la notazione P1 PBM ASCII, P2 PGM ASCII, P3 PPM ASCII, P4 PBM binario, P5 PGM binario, P6 PPM binario; la dimensione dell'immagine, in termini di numero di pixel orizzontali e verticali; il numero massimo di valori che rappresentano la singola componente RGB o di scala di grigio, solo per PGM e PPM.

Dopo l'header seguono, in ASCII oppure in formato binario, le informazioni relative ad ogni singolo pixel. Nel caso PPM i colori sono memorizzati in formato RGB. Il formato binario per PGM e PPM prevede tipicamente la rappresentazione di ogni singolo valore tramite un unico byte e quindi 8 bit. Un'estensione successiva prevede l'utilizzo di 16 bit per ogni valore. Un esempio minimale di PBM ASCII è sufficiente a mostrare la struttura:

```
P1
8 8
0 0 1 1 1 1 0 0
0 1 0 0 0 0 1 0
1 0 1 0 0 1 0 1
1 0 0 0 0 0 0 1
1 0 1 0 0 1 0 1
1 0 0 1 1 0 0 1
0 1 0 0 0 0 1 0
0 0 1 1 1 1 0 0
```

La prima riga identifica il formato P1, la seconda specifica larghezza e altezza, le righe successive contengono direttamente i campioni binari. Il valore didattico del formato è proprio questa corrispondenza quasi immediata fra matrice numerica e immagine: rende visibile ciò che nei formati compressi sarà nascosto dietro strutture di codifica molto più complesse.

A cui corrisponde l'immagine di riferimento riportata in figura 2.16.

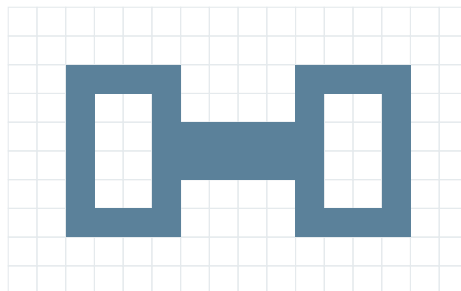


Figura 2.16. Piccola bitmap monocromatica originale usata come esempio di rappresentazione PBM: ogni cella della griglia corrisponde a un singolo campione binario.

XBM e XPM: formati storici

XBM e XPM sono formati storicamente importanti perché mostrano una idea diversa dal file immagine binario tradizionale: l'immagine viene rappresentata come testo direttamente incorporabile in un programma. Questo rende semplice compilare icone e piccole risorse grafiche insieme al codice sorgente, ma produce file poco efficienti rispetto ai formati compressi moderni.

XBM rappresenta essenzialmente una bitmap monocromatica mediante una sequenza di byte espressi nel sorgente C. Le dimensioni dell'immagine sono dichiarate separatamente e ogni bit identifica uno dei due colori possibili. Il vantaggio storico era la semplicità di integrazione con le applicazioni del sistema X Window; il limite è evidente: un solo bit per pixel e nessun meccanismo di compressione efficace.

XPM estende l'idea usando testo per descrivere una tavolozza e una matrice di simboli. Ogni simbolo viene associato a un colore e la matrice ricostruisce l'immagine. Il formato consente quindi immagini a colori e trasparenza, mantenendo la possibilità di incorporare la risorsa direttamente nel codice.

Il loro interesse nella nuova edizione è soprattutto concettuale: separano chiaramente tre elementi che ricompariranno in molti formati successivi, cioè dimensione dell'immagine, tavolozza e matrice degli indici. Per la distribuzione di immagini sul Web e nei sistemi multimediali moderni vengono invece usati formati compressi, che saranno analizzati nel capitolo dedicato alla compressione delle sorgenti statiche.

2.2.9 Grafica vettoriale e SVG

Le immagini raster descrivono direttamente una griglia di campioni; la grafica vettoriale descrive invece oggetti geometrici e le regole con cui devono essere disegnati. Segmenti, curve, forme, testo, riempimenti e trasformazioni possono quindi essere conservati senza fissare una risoluzione raster definitiva.

Scalare un'immagine raster richiede interpolazione fra campioni e può rendere visibile la griglia originaria. Una descrizione vettoriale può invece essere rasterizzata alla risoluzione richiesta dal dispositivo, per cui è particolarmente adatta a diagrammi, icone, mappe, loghi e interfacce grafiche. Non è però automaticamente più compatta: una scena fotografica descritta mediante migliaia di primitive può essere meno efficiente di una rappresentazione raster compressa.

Scalable Vector Graphics (SVG) è un formato vettoriale basato su XML e integrato nell'ecosistema Web. Può contenere forme geometriche, testo, gradienti, clipping, trasformazioni e riferimenti ad altre risorse; gli elementi possono inoltre essere manipolati tramite DOM e CSS. La stessa descrizione può quindi essere usata come immagine statica oppure come parte di una interfaccia interattiva e animata.

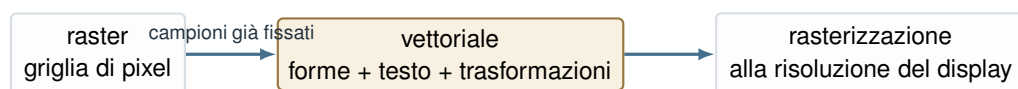


Figura 2.17. Raster e vettoriale descrivono l'immagine a livelli differenti. SVG conserva la struttura geometrica e demanda la rasterizzazione al sistema di rendering.

Il formato vettoriale non sostituisce quindi JPEG, PNG, WebP o AVIF: risolve un problema differente. Nei sistemi multimediali moderni le due famiglie convivono e vengono spesso composte nello stesso documento o nella stessa interfaccia.

2.3 Audio

2.3.1 Conversione A/D

Un segnale analogico è un segnale continuo nel tempo e continuo nei valori, mentre un segnale digitale è discreto nel tempo e discreto nei valori. In molti casi i valori possibili per un segnale digitale sono solo due e il segnale viene detto binario.

Analogico e digitale

Un segnale analogico varia continuamente nel tempo e nell'ampiezza. La conversione A/D costruisce invece una sequenza di numeri: prima si limita la banda del segnale, poi si campiona nel tempo, si quantizza l'ampiezza e infine si associa a ogni livello una parola binaria.



Figura 2.18. Catena concettuale di conversione analogico-digitale. Il filtraggio limita la banda, il campionamento discretizza il tempo, la quantizzazione discretizza l'ampiezza e la codifica produce bit.

Il vantaggio fondamentale del digitale non è che, a parità di un qualunque rapporto segnale-rumore, esso sia automaticamente “migliore” dell'analogico. Il vantaggio è la possibilità di rigenerare simboli discreti, usare codici di rivelazione e correzione, copiare senza accumulare necessariamente la stessa degradazione analogica ed elaborare numericamente il segnale. Il costo è che campionamento e quantizzazione devono essere progettati in modo coerente con la banda e con la qualità richiesta.

Anche un bit viene trasmesso attraverso una grandezza fisica continua. Il ricevitore osserva una tensione, una corrente, una potenza ottica o un altro segnale disturbato e decide quale simbolo sia stato inviato. Finché il rumore non porta l'osservazione oltre la regione decisionale, il simbolo può essere rigenerato senza propagare l'errore di ampiezza. Quando la soglia viene superata si produce invece un errore di simbolo o di bit. Questo comportamento non lineare è una delle ragioni per cui una lunga catena digitale può essere rigenerata nodo per nodo.

La qualità finale resta comunque limitata da ciò che è già avvenuto prima della decisione. Se il convertitore ha quantizzato con pochi bit, se il front-end ha saturato o se un errore di bit non viene corretto, l'informazione persa non può essere recuperata semplicemente perché il sistema è digitale.

Campionamento

La discretizzazione temporale sostituisce il segnale continuo $x(t)$ con la sequenza dei valori assunti agli istanti nT . Per definizione, se l'intervallo di campionamento è T , la frequenza di campionamento è

$$f_s = \frac{1}{T}.$$

Dimostrazione. In un intervallo di un secondo sono contenuti $1/T$ intervalli lunghi T secondi; viene quindi prelevato $1/T$ campione al secondo. Questo conclude la dimostrazione.

Sia ora $x(t)$ un segnale la cui trasformata di Fourier $X(f)$ sia nulla per $|f| > f_M$. I campioni sono

$$x[n] = x(nT).$$

La trasformata della sequenza, espressa rispetto alla frequenza fisica, è

$$X_s(f) = \sum_{n=-\infty}^{\infty} x(nT)e^{-j2\pi f nT}.$$

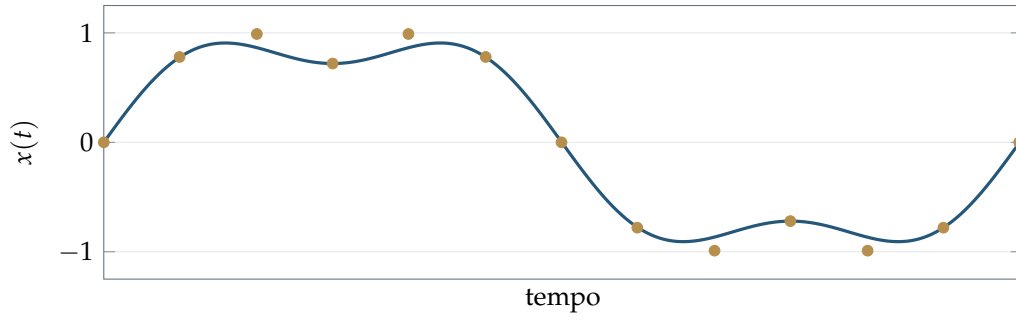


Figura 2.19. Campionamento temporale: i punti discreti conservano soltanto i valori del segnale agli istanti di campionamento. La curva continua è mostrata per rendere visibile ciò che il convertitore osserva prima della discretizzazione.

Usando la trasformata inversa di Fourier

$$x(nT) = \int_{-\infty}^{\infty} X(\phi) e^{j2\pi\phi nT} d\phi,$$

si ottiene la relazione fondamentale

$$X_s(f) = f_s \sum_{k=-\infty}^{\infty} X(f + kf_s).$$

Dimostrazione. Sostituendo la trasformata inversa nella definizione di $X_s(f)$,

$$X_s(f) = \int_{-\infty}^{\infty} X(\phi) \sum_{n=-\infty}^{\infty} e^{j2\pi(\phi-f)nT} d\phi.$$

La formula di sommazione di Poisson fornisce

$$\sum_{n=-\infty}^{\infty} e^{j2\pi(\phi-f)nT} = \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta\left(\phi - f - \frac{k}{T}\right).$$

Sostituendo e applicando la proprietà di campionamento dell'impulso,

$$\begin{aligned} X_s(f) &= \frac{1}{T} \sum_{k=-\infty}^{\infty} X\left(f + \frac{k}{T}\right) \\ &= f_s \sum_{k=-\infty}^{\infty} X(f + kf_s), \end{aligned}$$

poiché $f_s = 1/T$. Questo conclude la dimostrazione.

Il campionamento replica quindi periodicamente lo spettro analogico a intervalli f_s . Ogni replica occupa una banda larga $2f_M$; repliche adiacenti non si sovrappongono se

$$f_s > 2f_M.$$

Dimostrazione. La replica centrata in 0 occupa $[-f_M, f_M]$. La replica successiva, centrata in f_s , occupa $[f_s - f_M, f_s + f_M]$. Per non sovrapporsi deve valere $f_M < f_s - f_M$, cioè $f_s > 2f_M$. La stessa condizione vale per ogni coppia di repliche adiacenti. Questo conclude la dimostrazione.

Il valore $2f_M$ è il *tasso di Nyquist*; $f_s/2$ è invece la *frequenza di Nyquist* associata a una frequenza di campionamento data. L'uguaglianza $f_s = 2f_M$ è un caso limite ideale: nella pratica il filtro anti-aliasing richiede una banda di transizione e si usa una frequenza più alta.

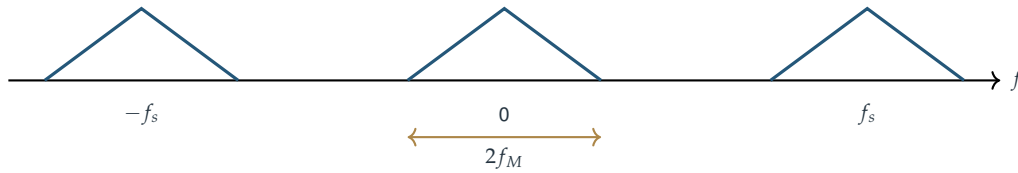


Figura 2.20. Repliche spettrali quando $f_s > 2f_M$: gli intervalli di supporto non si sovrappongono e la replica centrale può essere isolata mediante filtraggio.

Se $f_s < 2f_M$, parti di repliche adiacenti cadono sulle stesse frequenze. Il fenomeno è l'aliasing: componenti analogiche diverse producono la stessa sequenza di campioni e non possono più essere separate dal ricevitore.

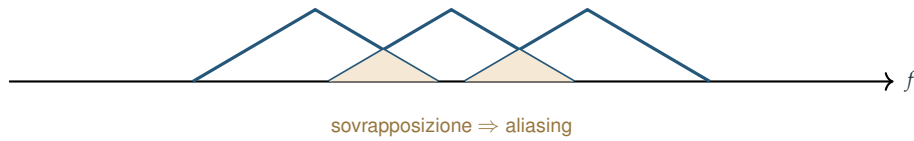


Figura 2.21. Quando la distanza fra le repliche spettrali è insufficiente, le componenti si sovrappongono. Dopo il campionamento l'informazione sulle frequenze originarie è persa.

In assenza di aliasing la ricostruzione ideale può essere espressa con la sinc normalizzata $\text{sinc}(u) = \sin(\pi u)/(\pi u)$:

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \text{sinc}\left(\frac{t - nT}{T}\right).$$

Dimostrazione. Dalla relazione spettrale appena dimostrata, nella banda della replica centrale vale $X_s(f) = X(f)/T$. Moltiplicando per il filtro ideale

$$H(f) = T \text{rect}(fT)$$

si recupera $X(f)$. La trasformata inversa di $H(f)$ è $h(t) = \text{sinc}(t/T)$. Nel dominio temporale filtrare la pettinatura dei campioni equivale a sommare copie traslate di $h(t)$, pesate con $x(nT)$; si ottiene quindi esattamente la somma precedente. Questo conclude la dimostrazione.

Quantizzazione

Il campionamento discretizza il tempo. La quantizzazione discretizza l'ampiezza e, salvo casi particolari, introduce perdita di informazione. Consideriamo un quantizzatore uniforme mid-rise sull'intervallo $[-1, 1)$ con 2^b livelli. La distanza fra due livelli adiacenti è

$$\Delta = \frac{2}{2^b} = 2^{1-b}.$$

Dimostrazione. L'intervallo ha lunghezza 2 e viene diviso in 2^b celle uguali. La larghezza di una cella è quindi $2/2^b = 2^{1-b}$. Questo conclude la dimostrazione.

Trascurando la saturazione, una possibile definizione del quantizzatore mid-rise è

$$Q(x) = \left\lfloor \frac{x}{\Delta} \right\rfloor \Delta + \frac{\Delta}{2}.$$

Questa è una definizione dell'algoritmo: la parte intera individua la cella e il termine $\Delta/2$ seleziona il suo punto medio.

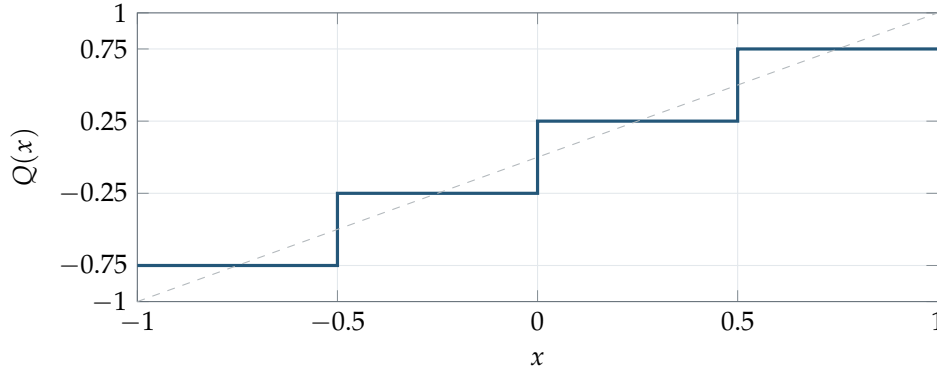


Figura 2.22. Esempio di quantizzatore uniforme mid-rise a quattro livelli. La diagonale tratteggiata rappresenta il caso ideale senza quantizzazione.

Se ogni campione è rappresentato con b bit e il campionamento produce f_s campioni/s, la bitrate PCM raw è

$$R_b = b f_s.$$

Dimostrazione. In un secondo vengono prodotti f_s campioni; ciascuno richiede b bit. Il numero di bit prodotti in un secondo è quindi $b f_s$. Questo conclude la dimostrazione.

Supponiamo ora che X sia uniforme in $[-1, 1)$, che non vi sia saturazione e che le celle siano allineate al dominio. L'errore $E = X - Q(X)$ è uniforme in $[-\Delta/2, \Delta/2)$ e la potenza di errore è

$$D = E[E^2] = \frac{\Delta^2}{12}.$$

Dimostrazione. La densità dell'errore è $1/\Delta$ nell'intervallo. Per simmetria la media è nulla e

$$\begin{aligned} E[E^2] &= \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} e^2 de \\ &= \frac{1}{\Delta} \left[\frac{e^3}{3} \right]_{-\Delta/2}^{\Delta/2} = \frac{\Delta^2}{12}. \end{aligned}$$

Questo conclude la dimostrazione.

Per la stessa sorgente uniforme,

$$P_X = E[X^2] = \frac{1}{3}.$$

Dimostrazione. La densità di X vale $1/2$ su $[-1, 1)$. Quindi

$$E[X^2] = \frac{1}{2} \int_{-1}^1 x^2 dx = \frac{1}{2} \left[\frac{x^3}{3} \right]_{-1}^1 = \frac{1}{3}.$$

Questo conclude la dimostrazione.

Il rapporto segnale-rumore di quantizzazione risulta

$$\rho = \frac{P_X}{D} = 2^{2b}, \quad \rho_{\text{dB}} \simeq 6.02b \text{ dB}.$$

Dimostrazione. Sostituendo $P_X = 1/3$, $D = \Delta^2/12$ e $\Delta = 2^{1-b}$,

$$\rho = \frac{1/3}{\Delta^2/12} = \frac{4}{2^{2-2b}} = 2^{2b}.$$

In decibel,

$$10 \log_{10}(2^{2b}) = 20b \log_{10} 2 \simeq 6.02b.$$

Questo conclude la dimostrazione.

Il valore numerico dipende dalla sorgente. Per una sinusoida a piena scala di picco unitario, con il consueto modello di errore uniforme e scorrelato dal segnale,

$$\rho_{\text{dB}} \simeq 6.02b + 1.76 \text{ dB}.$$

Dimostrazione. La potenza media di $x(t) = \sin(\omega t)$ è $1/2$. Con lo stesso $D = \Delta^2/12$,

$$\rho = \frac{1/2}{\Delta^2/12} = \frac{6}{2^{2-2b}} = \frac{3}{2} 2^{2b}.$$

Passando ai decibel, $10 \log_{10}(3/2) + 20b \log_{10} 2 \simeq 1.76 + 6.02b$. Questo conclude la dimostrazione. La relazione resta quindi legata alle ipotesi sul segnale e sul modello dell'errore.

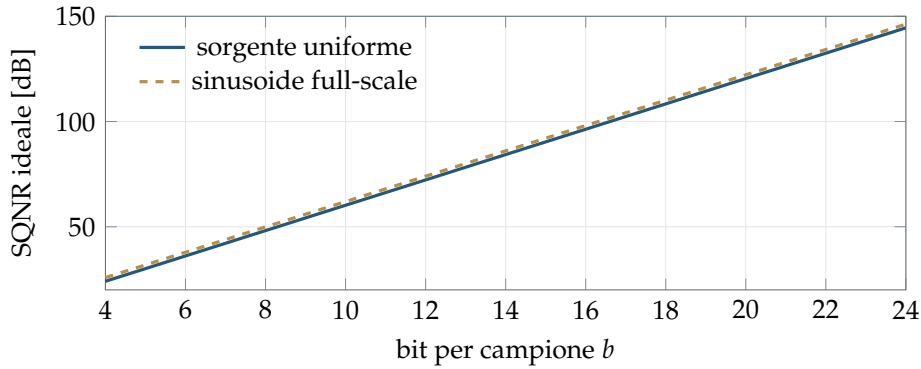


Figura 2.23. Crescita dell'SQNR nei due modelli ideali derivati. Ogni bit aggiunto produce circa 6 dB di incremento, ma l'intercetta dipende dalla statistica del segnale.

Quantizzazione non uniforme e Lloyd-Max. Sia X una variabile aleatoria con densità $p(x)$. Un quantizzatore scalare con soglie $t_0 < t_1 < \dots < t_N$ associa alla cella $[t_{i-1}, t_i)$ il valore di ricostruzione y_i . Per errore quadratico medio la distorsione è definita da

$$D = \sum_{i=1}^N \int_{t_{i-1}}^{t_i} (x - y_i)^2 p(x) dx.$$

Per soglie fissate, il livello ottimo è il centroide condizionato:

$$y_i = \frac{\int_{t_{i-1}}^{t_i} x p(x) dx}{\int_{t_{i-1}}^{t_i} p(x) dx}.$$

Dimostrazione. Solo l'integrale della cella i dipende da y_i . Derivando,

$$\frac{\partial D}{\partial y_i} = -2 \int_{t_{i-1}}^{t_i} (x - y_i) p(x) dx.$$

Nel punto stazionario la derivata è nulla; separando i termini,

$$y_i \int_{t_{i-1}}^{t_i} p(x) dx = \int_{t_{i-1}}^{t_i} x p(x) dx,$$

da cui segue la formula. Questo conclude la dimostrazione.

Per livelli adiacenti fissati, la soglia ottima è

$$t_i = \frac{y_i + y_{i+1}}{2}.$$

Dimostrazione. Sul confine fra due celle l'errore quadratico deve essere uguale: $(t_i - y_i)^2 = (t_i - y_{i+1})^2$. Poiché il confine cade fra $y_i < y_{i+1}$, la soluzione pertinente è il loro punto medio. Questo conclude la dimostrazione.

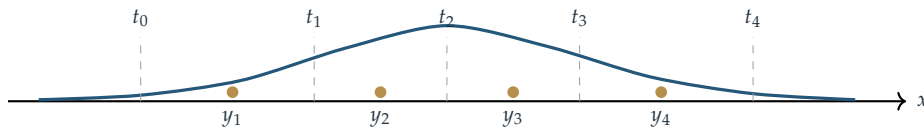


Figura 2.24. Interpretazione delle condizioni di Lloyd-Max: ogni valore di ricostruzione è il centroide probabilistico della propria cella e ogni soglia cade a metà fra due livelli adiacenti.

Alternando i due passi si ottiene l'algoritmo di Lloyd-Max. La distorsione non aumenta a ogni iterazione, ma la convergenza è in generale verso un minimo locale e dipende dall'inizializzazione.

La quantizzazione può essere anche vettoriale: un blocco di L campioni viene trattato come un punto in uno spazio L -dimensionale e associato a un codeword di ricostruzione. Ciò permette di sfruttare anche la geometria e la correlazione fra campioni, al prezzo di una ricerca e di un codebook più complessi.

2.3.2 Formati audio non compressi

PCM

La Pulse Code Modulation (PCM) rappresenta ciascun campione mediante una parola numerica. Nella telefonia numerica tradizionale il segnale vocale utile è limitato a circa 3.4 kHz e viene campionato a

$$f_s = 8 \text{ kHz},$$

lasciando una banda di transizione fra il limite del segnale e la frequenza di Nyquist $f_s/2 = 4$ kHz. La frequenza di campionamento non è quindi scelta uguale al doppio della massima frequenza utile, ma leggermente più alta per rendere realizzabile il filtro anti-aliasing.

Nella raccomandazione G.711 ogni campione codificato è rappresentato da 8 bit. Ne segue direttamente la bitrate di un canale PCM telefonico:

$$R_b = 8000 \frac{\text{campioni}}{\text{s}} 8 \frac{\text{bit}}{\text{campione}} = 64 \text{ kbit/s}.$$

Le due leggi di codifica previste sono A-law e μ -law. La prima è storicamente usata soprattutto nelle reti europee, la seconda in Nord America e Giappone. Eventuali canali storici a 56 kbit/s sono legati a modalità di trasporto e segnalazione e non vanno interpretati come una diversa definizione G.711 a 7 bit per campione.

Companing

Un quantizzatore uniforme assegna la stessa ampiezza a tutti gli intervalli di quantizzazione. Per la voce è vantaggioso avere una risoluzione relativa maggiore alle piccole ampiezze e minore alle grandi ampiezze. Una realizzazione concettualmente semplice consiste nel comprimere non linearmente l'ampiezza prima della quantizzazione uniforme e applicare al ricevitore la trasformazione inversa. La coppia compressore-espansore è detta compander.

Le due caratteristiche continue classiche sono la μ -law e la A-law. Le relazioni seguenti sono definizioni del modello continuo di companding, non risultati da dimostrare. Per un ingresso normalizzato $x \in [-1, 1]$, la legge μ può essere scritta come

$$F_{\mu}(x) = \text{sgn}(x) \frac{\log(1 + \mu|x|)}{\log(1 + \mu)},$$

con il valore tradizionale $\mu = 255$.

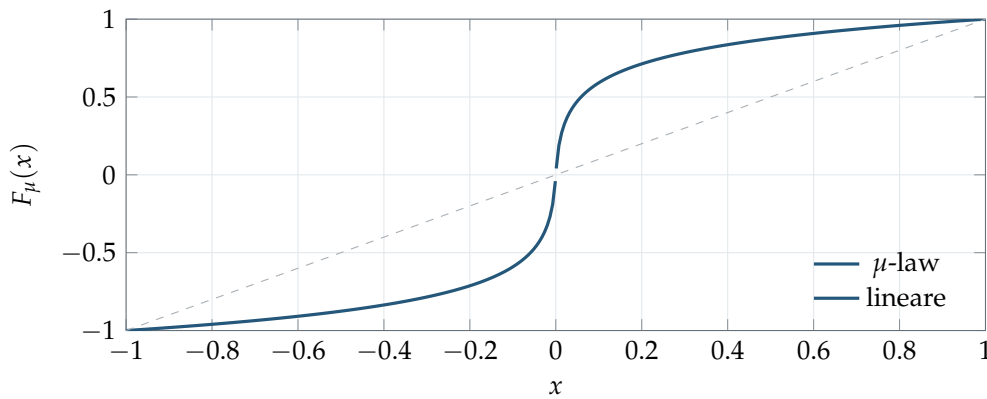


Figura 2.25. Caratteristica continua μ -law con $\mu = 255$. La pendenza elevata intorno all'origine dedica una risoluzione relativa maggiore alle piccole ampiezze.

La caratteristica continua A-law è

$$F_A(x) = \text{sgn}(x) \begin{cases} \frac{A|x|}{1 + \log A}, & \text{se } 0 \leq |x| < 1/A, \\ \frac{1 + \log(A|x|)}{1 + \log A}, & \text{se } 1/A \leq |x| \leq 1, \end{cases}$$

con il valore tradizionale $A = 87.6$.

Queste espressioni descrivono il principio matematico del companding. La codifica G.711 effettiva è una quantizzazione logaritmica a segmenti e produce direttamente parole di 8 bit. Non esiste quindi un unico miglioramento in dB equivalente a un numero fisso di bit: il vantaggio rispetto a un quantizzatore uniforme dipende dall'ampiezza e dalla statistica del segnale. La proprietà essenziale è che, dopo la compressione, più livelli effettivi vengono dedicati alle piccole ampiezze.

Compact Disc Digital Audio

Il Compact Disc Digital Audio (CD-DA) è un esempio importante perché mostra la differenza fra rappresentazione della sorgente e codifica fisica sul supporto. L'audio è PCM lineare stereo, campionato a 44.1 kHz con 16 bit per campione e per canale. Il supporto ottico non memorizza però direttamente questa sequenza di parole: aggiunge sincronizzazione, interleaving, codici di correzione e una codifica di canale adatta alla lettura ottica.

Il periodo di campionamento è

$$T = \frac{1}{44100} \simeq 22.68 \mu\text{s}.$$

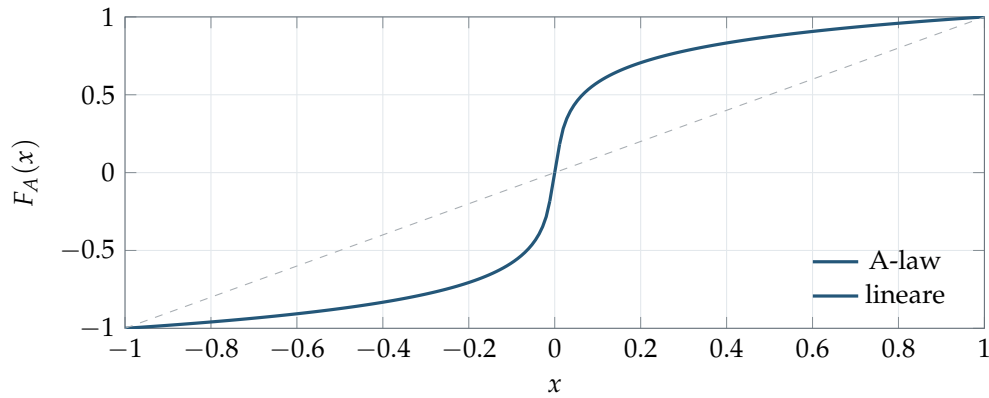


Figura 2.26. Caratteristica continua A-law con $A = 87.6$. Il tratto lineare vicino all'origine mantiene elevata la risoluzione per i piccoli segnali.

Dimostrazione. Per definizione $T = 1/f_s$. Sostituendo $f_s = 44100$ Hz si ottiene il valore numerico. Questo conclude la dimostrazione.

La frequenza di Nyquist è 22.05 kHz. Se si considera una banda audio utile fino a circa 20 kHz, restano poco più di 2 kHz di transizione per il filtraggio. Storicamente questo vincolo ha reso impegnativi i primi filtri anti-aliasing e di ricostruzione; tecniche successive di overdamping hanno spostato gran parte del filtraggio nel dominio digitale.

La bitrate PCM di un canale è

$$R_{b,1} = 44100 \cdot 16 = 705.6 \text{ kbit/s},$$

mentre per due canali

$$R_{b,\text{stereo}} = 2R_{b,1} = 1.4112 \text{ Mbit/s}.$$

Dimostrazione. Ogni canale genera 44100 campioni/s e ogni campione contiene 16 bit; quindi $44100 \cdot 16 = 705600$ bit/s. Due canali indipendenti raddoppiano il numero di bit al secondo. Questo conclude la dimostrazione.

Per una sinusoide a piena scala, il modello ideale di quantizzazione già derivato fornisce

$$\rho_{\text{dB}} \simeq 6.02 \cdot 16 + 1.76 \simeq 98.1 \text{ dB}.$$

Il valore non è la dinamica garantita di qualunque lettore CD: rumore analogico, jitter di conversione, non linearità e livello effettivo del segnale modificano le prestazioni reali.

Il disco da 120 mm è formato da un substrato di policarbonato, uno strato riflettente e protezioni superficiali. L'informazione segue una spirale di pit e land letta mediante un laser nel vicino infrarosso. Il dato non è associato semplicemente alla presenza di un pit o di una land, ma alle transizioni e alla lunghezza degli intervalli fra transizioni, secondo la codifica di canale Eight-to-Fourteen Modulation (EFM). EFM trasforma simboli di 8 bit in parole di 14 bit e aggiunge merging bit per imporre vincoli sulla distanza fra transizioni, facilitando recupero del clock e lettura ottica.

La protezione dagli errori usa il Cross-Interleaved Reed-Solomon Code (CIRC). L'interleaving distribuisce nel tempo errori fisicamente contigui, come quelli prodotti da un graffio, mentre i codici Reed-Solomon correggono simboli mancanti o errati. Se il danno supera la capacità di correzione, il lettore può ricorrere a tecniche di concealment, per esempio interpolando campioni vicini. Questa catena mostra una distinzione che ricorrerà nelle reti: la codifica di sorgente rappresenta l'audio, la codifica di canale aggiunge ridondanza per proteggerlo dal mezzo fisico.

I supporti CD-R usano uno strato organico la cui riflettività viene modificata irreversibilmente durante la scrittura; i CD-RW usano un materiale a cambiamento di fase che può passare reversibilmente fra stati con riflettività differente. Queste tecnologie mantengono la geometria e la semantica ottica necessarie a presentarsi al lettore come un supporto compatibile.

2.3.3 MIDI

Musical Instrument Digital Interface (MIDI) rappresenta eventi e controlli musicali, non campioni della forma d'onda. Questo punto è fondamentale: un file o una sequenza MIDI può essere estremamente compatta perché descrive *che cosa* deve essere suonato, mentre il suono concreto dipende dal sintetizzatore o dallo strumento che interpreta i messaggi.

MIDI 1.0 usa storicamente un collegamento seriale a 31.25 kbit/s. I byte di stato hanno il bit più significativo a 1; i byte dati lo hanno a 0 e contengono quindi 7 bit di valore. Quattro bit dello status identificano la classe del messaggio e quattro il canale, permettendo 16 canali logici.

Un messaggio Note On contiene tipicamente stato/canale, numero della nota e velocity. Per esempio i tre byte decimali 144 60 64 indicano Note On sul canale 1, nota MIDI 60 e velocity 64. Note Off, Control Change, Program Change, Pitch Bend e gli altri messaggi di canale seguono lo stesso principio. Messaggi di sistema gestiscono clock, trasporto, sincronizzazione e dati esclusivi del costruttore.

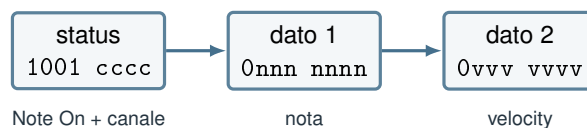


Figura 2.27. Esempio di struttura di un messaggio MIDI 1.0 Note On. I dati sono eventi musicali, non campioni audio.

MIDI 2.0 mantiene il principio di descrizione degli eventi ma introduce Universal MIDI Packet, maggiore risoluzione dei controlli, comunicazione bidirezionale più ricca e meccanismi di negoziazione delle capacità. La distinzione rispetto al PCM resta invariata: PCM codifica la forma d'onda; MIDI codifica intenzioni e parametri musicali che un motore di sintesi trasforma in audio.

2.4 Video

Il video aggiunge una dimensione temporale alla rappresentazione spaziale delle immagini. Una scena reale evolve in modo continuo; una telecamera la osserva mediante una sequenza di esposizioni e un sistema di riproduzione presenta una sequenza di quadri. Il campionamento spaziale determina quanti dettagli possono essere rappresentati in ogni immagine; il campionamento temporale determina come vengono rappresentati moto e variazioni nel tempo.

Questi due campionamenti non sono indipendenti dal sistema percettivo. Aumentare la risoluzione spaziale senza aumentare la frequenza di quadro può lasciare invariata la qualità del moto; aumentare il frame rate di un'immagine sfocata o fortemente compressa non recupera il dettaglio perso. Il video è quindi un problema congiunto di spazio, tempo, esposizione, colore e compressione.

2.4.1 Frequenza di quadro, esposizione e moto

La frequenza di quadro F indica il numero di frame completi prodotti o presentati in un secondo. Il periodo nominale fra due frame è

$$T_F = \frac{1}{F}.$$

Dimostrazione. Se in un secondo vengono prodotti F frame equidistanti nel tempo, l'intervallo fra due istanti consecutivi è $1/F$ secondi. Questo conclude la dimostrazione.

Il frequenza di quadro non coincide con una soglia fisiologica unica oltre la quale l'occhio “non vede più discontinuità”. La percezione dipende da velocità e contrasto del moto, luminanza, tempo di esposizione, persistenza del display, dimensione angolare e modalità di osservazione. 24 frame/s possono produrre un moto cinematografico convincente quando l'esposizione introduce un opportuno motion blur; 50, 60, 100 o più frame/s riducono la distanza temporale fra campioni e migliorano la rappresentazione del moto rapido e l'interazione.

Il tempo di esposizione T_E è un secondo parametro indipendente. Se un oggetto percorre sul sensore una velocità apparente v espressa in pixel/s, durante l'esposizione percorre una distanza

$$d = vT_E.$$

Dimostrazione. Per moto uniforme vale la definizione $v = d/T_E$; isolando d si ottiene $d = vT_E$. Questo conclude la dimostrazione. La relazione è un modello cinematico elementare: accelerazioni, rolling shutter e ricostruzioni temporali possono produrre effetti più complessi.

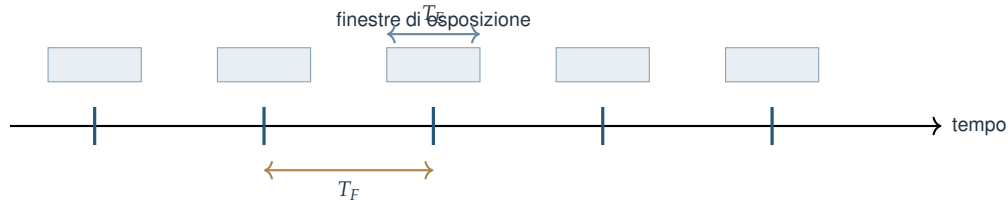


Figura 2.28. Frame period ed esposizione sono grandezze distinte. Il frequenza di quadro fissa la distanza fra campioni temporali; il tempo di esposizione controlla quanto moto viene integrato dentro ciascun campione.

2.4.2 Conversione fra cadenze cinematografiche e televisive

Il cinema su pellicola ha storicamente usato 24 fotogrammi/s, mentre la televisione analogica ha usato famiglie temporali legate a 50 e 60 Hz. La conversione fra cadenze non è quindi un dettaglio secondario: se le frequenze non sono multipli interi occorre modificare la velocità di riproduzione oppure distribuire in modo non uniforme i frame cinematografici sui campi o frame televisivi.

Nel mondo 625/50 una soluzione storica semplice consiste nel riprodurre il film a 25 frame/s. Il fattore di accelerazione è

$$g_{24 \rightarrow 25} = \frac{25}{24} \simeq 1.04167,$$

cioè circa il 4.17%.

Dimostrazione. Un secondo di film contiene 24 fotogrammi. Riproducendo gli stessi 24 fotogrammi a 25 fotogrammi/s, il tempo necessario diventa $24/25$ s. La velocità temporale relativa è l'inverso, $25/24$. L'incremento percentuale è $(25/24 - 1)100\% \simeq 4.17\%$. Questo conclude la dimostrazione.

Senza correzione audio, anche le frequenze del suono aumentano dello stesso fattore. Sistemi moderni possono effettuare time-stretch e pitch-shift separatamente, evitando di legare necessariamente durata e tonazione.

Nel mondo 525/59.94 la cadenza video è circa 29.97 frame/s. Una tecnica storica è il 3:2 pulldown: il film viene portato a circa 23.976 frame/s e quattro frame cinematografici generano dieci campi, cioè cinque frame interlacciati. La sequenza dei campi può essere schematizzata come A A B B C C D D D, con variazioni nella fase del pattern.

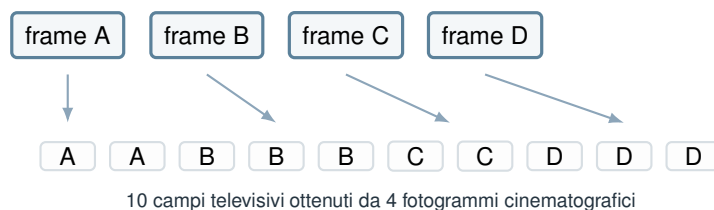


Figura 2.29. Schema concettuale del pulldown 3:2. La distribuzione non uniforme dei campi introduce una cadenza temporale periodica, detta judder, assente nella sorgente originale.

Le conversioni moderne possono usare frame interpolation, frame blending o inverse telecine. Nessuna tecnica è universalmente migliore: l'interpolazione può produrre artefatti su occlusioni e moto complesso; il blending riduce il judder ma sfoca; l'inverse telecine funziona solo quando il pattern originale è riconoscibile.

2.4.3 Scansione progressiva e interlacciata

Nella scansione progressiva tutte le righe di un frame appartengono allo stesso istante logico di campionamento. Nella scansione interlacciata il frame viene diviso in due campi di righe alternate, separati nel tempo di mezzo periodo di quadro. L'interlacciamento fu un compromesso brillante: a parità di numero di righe per quadro e di banda disponibile, raddoppia la frequenza con cui lo schermo riceve informazione verticale parziale, riducendo la percezione di flicker sui CRT.

Il prezzo è che le righe dei due campi non appartengono allo stesso istante. Un oggetto in moto occupa posizioni differenti nei due campi; ricombinarli senza elaborazione produce il tipico artefatto a pettine. Il deinterlacciamento deve stimare righe mancanti e moto, scegliendo fra preservazione della risoluzione verticale e continuità temporale.

La figura 2.14, introdotta nella sezione sui CRT, mostra la differenza geometrica fra i due modelli. Nei pannelli moderni la scansione nativa è progressiva; materiale interlacciato deve quindi essere convertito prima della visualizzazione.

2.4.4 Televisione analogica monocromatica

Studiare la televisione analogica resta utile perché molti concetti del video digitale derivano direttamente da essa: righe, blanking, sincronismi, luma/crominanza, interlacciamento e campionamento successivo BT.601. È però importante non descrivere un segnale analogico come se possedesse già una griglia orizzontale di pixel.

Struttura temporale 625/50 e 525/59.94

Nei sistemi europei a 625 linee e 25 frame/s, la frequenza di riga è

$$f_H = 625 \cdot 25 = 15,625 \text{ Hz.}$$

Dimostrazione. Ogni frame contiene nominalmente 625 periodi di riga e vengono trasmessi 25 frame/s. Il numero di righe per secondo è quindi il prodotto $625 \cdot 25 = 15625$. Questo conclude la dimostrazione.

Il sistema interlacciato trasmette 50 campi/s. Il mezzo periodo di riga fra i due campi produce l'offset verticale necessario affinché le righe del secondo campo cadano fra quelle del primo.

La famiglia 525/59.94 usa circa 29.97 frame/s. La frequenza di riga è circa

$$f_H = 525 \frac{30000}{1001} \simeq 15,734.27 \text{ Hz.}$$

Dimostrazione. Il frequenza di quadro è $30000/1001$ frame/s e ogni frame contiene 525 linee nominali. Moltiplicando si ottiene il valore indicato. Questo conclude la dimostrazione. Il valore è conseguenza della temporizzazione del sistema colore NTSC, non un arrotondamento arbitrario a 30 Hz.

Il numero di linee non coincide con le linee visibili. Una parte del quadro appartiene al vertical blanking interval e contiene i segnali necessari alla ritraccia e al sincronismo; nel tempo sono stati inseriti in questa zona anche dati ausiliari.

Riga composita, blanking e sincronismo

Durante ogni riga il ricevitore deve conoscere dove termina il video attivo e quando deve iniziare la riga successiva. Il segnale composito monocromatico unisce quindi luminanza, intervalli di blanking e impulsi di sincronismo in una singola forma d'onda.

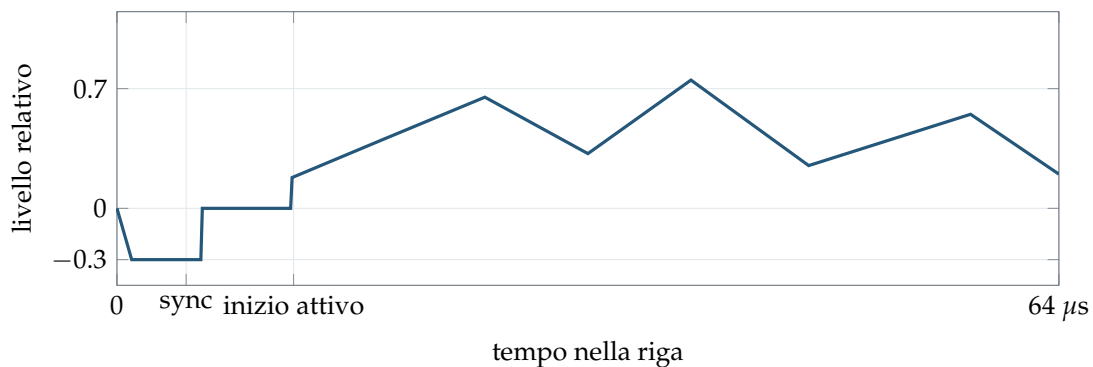


Figura 2.30. Riga composita monocromatica schematica: impulso di sincronismo, blanking e zona video attiva. I livelli sono normalizzati e non costituiscono una maschera normativa.

Nel sistema 625/50 il periodo di riga è $1/15625 \simeq 64 \mu s$. La parte attiva è più corta: il resto del tempo serve a front porch, sincronismo e back porch. Anche in orizzontale la parola “risoluzione” analogica va trattata con cautela: la quantità di dettaglio dipende dalla banda video, dalla risposta dei filtri, dal tubo e dalla catena di trasmissione, non da un numero intrinseco di pixel.

Il cosiddetto fattore di Kell è un coefficiente empirico usato storicamente per collegare numero nominale di linee e risoluzione verticale effettivamente percepibile in un sistema campionato e scansionato. Non è una legge matematica universale e dipende da apertura, scansione, fase e criterio di valutazione; nella nuova edizione viene quindi mantenuto come concetto storico, non come formula di progetto da applicare meccanicamente.

Trasmissione VSB

Un segnale video analogico di banda base occupa diversi megahertz. Una modulazione AM a doppia banda laterale raddoppierebbe quasi la banda necessaria; una SSB ideale

richiederebbe filtri molto selettivi vicino alla portante, proprio dove il video contiene componenti importanti. La Vestigial Side Band è il compromesso: si trasmette quasi interamente una banda laterale e soltanto un vestigio dell'altra.

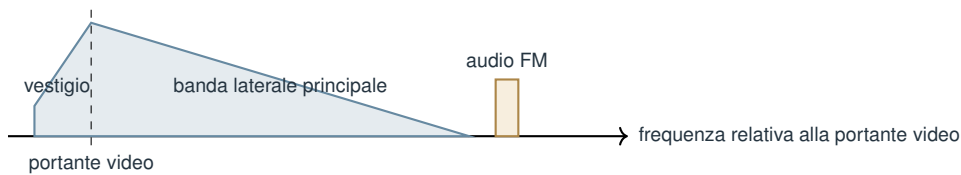


Figura 2.31. Principio della trasmissione televisiva VSB. Le larghezze sono schematiche: canali e offset effettivi dipendono dalla variante dello standard nazionale.

L'audio analogico viene normalmente trasmesso su una portante distinta, spesso in FM. Offset e larghezze di canale differiscono fra famiglie B/G, I, M e altre varianti: sono dati di specifica, non conseguenze di una formula generale.

2.4.5 Introduzione del colore: compatibilità e intercalamento

Il requisito fondamentale della televisione a colori fu la compatibilità con i ricevitori monocromatici. Il nuovo segnale doveva continuare a presentare una luminanza interpretabile dal vecchio ricevitore e aggiungere il colore senza raddoppiare la banda.

La scansione rende lo spettro della luminanza fortemente strutturato: l'energia tende a concentrarsi intorno alle armoniche della frequenza di riga e alle bande laterali generate dal moto e dalla struttura verticale. Rimangono quindi regioni relativamente meno occupate fra i cluster spettrali. La crominanza può essere modulata su una sottoportante scelta in modo da intercalare gran parte della propria energia in tali regioni.

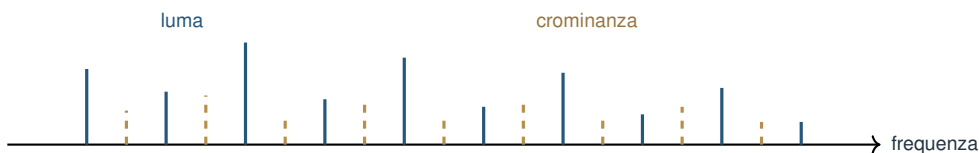


Figura 2.32. Intercalamento spettrale concettuale fra luminanza e crominanza. La struttura a righe nasce dalla periodicità della scansione.

PAL

PAL (Phase Alternating Line) è una famiglia di sistemi colore usata principalmente con la scansione 625/50. Le due informazioni di differenza di colore modulano in quadratura una sottoportante. Nelle varianti PAL B/G la frequenza della sottoportante è 4.43361875 MHz. Il valore è una specifica del sistema.

La caratteristica distintiva è l'inversione di fase, da una riga alla successiva, di una componente della crominanza. Un errore di fase che sposta la tinta in un senso su una riga tende a produrre lo spostamento opposto sulla successiva. Con una linea di ritardo il ricevitore può combinare le due informazioni e trasformare parte dell'errore di fase in errore di saturazione, generalmente meno disturbante.

Un breve color burst viene inserito nel back porch di ogni riga per fornire al ricevitore un riferimento di fase e frequenza della sottoportante. Il burst non contiene il colore dell'immagine: serve a ricostruire il riferimento necessario alla demodulazione coerente.

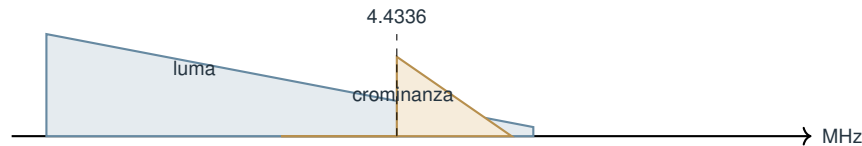


Figura 2.33. Spettro PAL qualitativo. L'area colorata mostra la sovrapposizione concettuale fra banda della luminanza e crominanza modulate.

NTSC

Il sistema NTSC a colori fu standardizzato prima di PAL. La famiglia 525 linee usa circa 59.94 campi/s e una sottoportante colore di circa 3.579545 MHz. Storicamente le informazioni cromatiche vengono descritte nel sistema YIQ, ottenuto come rotazione di componenti di differenza di colore per adattare la banda alle sensibilità visive e ai vincoli del canale.

A differenza di PAL, NTSC non alterna riga per riga la fase di una componente per compensare gli errori. Distorsioni di fase nella catena analogica si traducono quindi più direttamente in errori di tinta. I ricevitori storici disponevano per questo di un controllo di hue/tint.

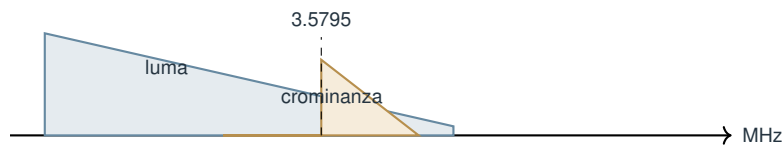


Figura 2.34. Spettro NTSC qualitativo con sottoportante colore intorno a 3.58 MHz.

SECAM

SECAM (Séquentiel couleur à mémoire) segue una strategia diversa. Le due componenti di differenza di colore vengono trasmesse alternativamente su righe successive e modulate in frequenza. Il ricevitore memorizza la crominanza della riga precedente per ricostruire, insieme a quella corrente, le due informazioni necessarie.

Il vantaggio storico della modulazione FM è una buona robustezza agli errori di fase; il prezzo è un'architettura incompatibile con PAL e NTSC pur condividendo, in molte varianti europee, la stessa struttura 625/50 della scansione monocromatica.

PAL, NTSC e SECAM mostrano tre filosofie differenti per soddisfare lo stesso vincolo: aggiungere colore senza rendere inutilizzabili i ricevitori monocromatici e senza aumentare drasticamente la banda.

2.4.6 Dal segnale analogico al video digitale

Digitalizzare una riga analogica richiede scegliere una frequenza di campionamento. A differenza della direzione verticale, dove il numero di linee è già imposto dalla scansione, la direzione orizzontale non ha un numero intrinseco di pixel: è la frequenza di campionamento che costruisce la griglia digitale.

BT.601 usa 13.5 MHz per la luma e 6.75 MHz per ciascuna crominanza nel 4:2:2. La parte attiva contiene 720 campioni di luma per riga e 360 per ciascuna crominanza. Sono comunemente rappresentate 576 linee attive per le sorgenti 625/50 e 480 per quelle 525/59.94.

2.4.7 Sottocampionamento della crominanza

La sensibilità del sistema visivo al dettaglio spaziale della crominanza è generalmente inferiore a quella per la luma. I sistemi video possono quindi usare meno campioni per C_b e C_r senza ridurre nella stessa proporzione la qualità percepita. La notazione 4:a:b descrive

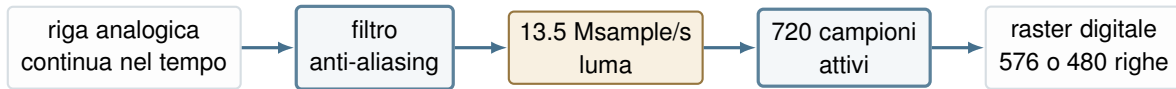


Figura 2.35. Passaggio concettuale dalla riga televisiva analogica alla griglia digitale BT.601. Il numero di campioni orizzontali nasce dalla scelta di campionamento, non era già contenuto nel segnale analogico come pixel.

la geometria relativa dei campioni; non è una bitrate e non identifica da sola risoluzione, profondità o frequenza di quadro.

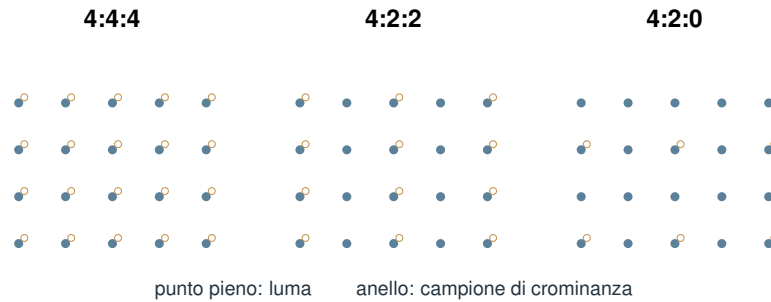


Figura 2.36. Densità relativa dei campioni nei formati 4:4:4, 4:2:2 e 4:2:0. Le convenzioni di siting possono collocare i campioni di crominanza in posizioni differenti rispetto alla luma.

Se la luma produce N_Y campioni/s e ciascuna crominanza una frazione q di tale numero, con b bit/campione, la bitrate grezza è

$$R = bN_Y(1 + 2q).$$

Dimostrazione. La luma produce N_Y campioni/s e le due crominanze insieme $2qN_Y$. Il totale è quindi $N_Y(1 + 2q)$ campioni/s; moltiplicando per b bit/campione si ottiene la relazione. Questo conclude la dimostrazione.

4:4:4

Nel 4:4:4 le tre componenti hanno la stessa densità. Se, soltanto per confronto, tutte fossero campionate a 13.5 MHz con 8 bit,

$$R = (13.5 + 13.5 + 13.5)10^6 \cdot 8 = 324 \text{ Mbit/s}.$$

Il valore è un esempio numerico ottenuto dal conteggio appena dimostrato; BT.601 standard usa invece 4:2:2.

4:2:2

Nel 4:2:2 le crominanze hanno metà della densità orizzontale e la stessa densità verticale della luma. Con BT.601 a 8 bit:

$$R = (13.5 + 6.75 + 6.75) 10^6 \cdot 8 = 216 \text{ Mbit/s}.$$

Nella parte attiva vi sono 720 campioni Y e 360 campioni per ciascuna crominanza su ogni riga.

4:1:1

Nel 4:1:1 ciascuna crominanza possiede un quarto della densità orizzontale della luma e la stessa densità verticale. Usando il medesimo riferimento di 13.5 MHz per la luma e 8 bit/campione, il conteggio fornisce 162 Mbit/s. Per una luma 720×576 le crominanze sono 180×576 .

4:2:0

Nel 4:2:0 ciascuna crominanza possiede metà della risoluzione orizzontale e metà di quella verticale. Il numero di campioni di ogni crominanza è quindi un quarto del numero di campioni di luma. Per un raster attivo $W \times H$, profondità b e frequenza di quadro F , la bitrate grezza è

$$R_{4:2:0} = \frac{3}{2}bWHF.$$

Dimostrazione. La luma contiene WH campioni. Ogni crominanza contiene $WH/4$ campioni; il totale è $WH + WH/4 + WH/4 = 3WH/2$. Moltiplicando per b bit/campione e per F frame/s si ottiene la formula. Questo conclude la dimostrazione.

Per 720×576 le crominanze sono 360×288 ; per 720×480 sono 360×240 .

4:1:0

Il meno comune 4:1:0 riduce la crominanza di quattro volte in orizzontale e due in verticale. Ogni piano di crominanza contiene quindi un ottavo dei campioni della luma. Per una luma 720×576 i due piani sono 180×288 . Il formato ha valore soprattutto storico e comparativo; il 4:2:0 è molto più comune nelle catene video moderne.

2.4.8 Formati video digitali storici

SIF, CIF e QCIF restano utili per comprendere MPEG-1, H.261 e H.263. Non rappresentano però le risoluzioni tipiche dei sistemi moderni.

SIF

SIF usa 352×240 campioni di luma nella famiglia derivata da 525/59.94 e 352×288 in quella 625/50. In 4:2:0 le crominanze sono rispettivamente 176×120 e 176×144 .

A 8 bit e 30000/1001 frame/s, il primo raster ha bitrate grezza attiva

$$R = \frac{3}{2} 352 240 8 \frac{30000}{1001} \simeq 30.38 \text{ Mbit/s.}$$

Dimostrazione. Si applica direttamente la relazione 4:2:0 già dimostrata con $W = 352$, $H = 240$, $b = 8$ e $F = 30000/1001$. Questo conclude la dimostrazione.

CIF e QCIF

CIF usa 352×288 campioni di luma e crominanze 4:2:0 da 176×144 . QCIF dimezza entrambe le dimensioni lineari: luma 176×144 e crominanze 88×72 . Alla stessa profondità e frequenza di quadro, QCIF possiede un quarto dei campioni di CIF e quindi un quarto della bitrate grezza.

Dimostrazione. Dimezzare larghezza e altezza trasforma WH in $(W/2)(H/2) = WH/4$. Nella formula $R = (3/2)bWHF$ tutti gli altri fattori restano invariati; anche la bitrate diventa quindi un quarto. Questo conclude la dimostrazione.

2.4.9 HD, UHD, HDR e frequenza di quadro elevati

BT.709 ha costituito il riferimento fondamentale per l'HDTV e usa comunemente il raster 1920×1080 con rapporto 16:9. Nel 4:2:2 le crominanze hanno 960×1080 campioni; nel 4:2:0 960×540 . UHD ha diffuso il raster 3840×2160 e, nelle catene più recenti, profondità di 10 bit, gamut più ampi e HDR.

A parità di profondità, frequenza di quadro e sottocampionamento, raddoppiare entrambe le dimensioni lineari quadruplica la bitrate grezza:

$$R' = 4R.$$

Dimostrazione. Per il 4:2:0, $R = (3/2)bWHF$. Il raster $2W \times 2H$ produce

$$R' = \frac{3}{2}b(2W)(2H)F = 4\frac{3}{2}bWHF = 4R.$$

Lo stesso fattore geometrico vale per qualunque formato il cui numero di campioni sia proporzionale all'area. Questo conclude la dimostrazione.

Passare contemporaneamente da 1080p a 2160p, da 8 a 10 bit e da 30 a 60 frame/s aumenta la quantità raw del fattore

$$4 \cdot \frac{10}{8} \cdot \frac{60}{30} = 10.$$

Dimostrazione. Il fattore 4 deriva dall'area, 10/8 dalla profondità e 60/30 dal numero di frame per secondo. Poiché i tre effetti sono indipendenti nel conteggio raw, i fattori si moltiplicano: $4 \cdot 1.25 \cdot 2 = 10$. Questo conclude la dimostrazione.

Questo semplice calcolo spiega perché l'evoluzione dei formati video è inseparabile dall'evoluzione dei codec. Senza compressione più efficiente, una crescita apparentemente moderata delle caratteristiche visive può moltiplicare di un ordine di grandezza il flusso da memorizzare o trasportare.

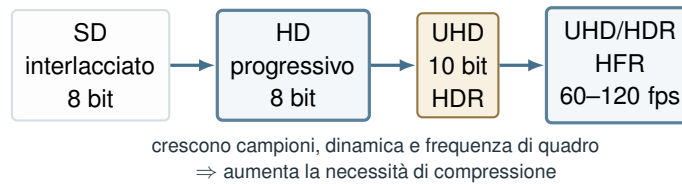


Figura 2.37. Evoluzione concettuale dei formati video. La qualità moderna deriva dall'aumento congiunto di risoluzione, profondità, dinamica e campionamento temporale, non da un solo parametro.

Compressione delle sorgenti statiche

3.1 Elementi di codifica di sorgente

In un sistema di comunicazione si può verificare la presenza contemporanea di tre sistemi di codifica di natura diversa. Andando dalla sorgente verso il canale di comunicazione si ha la codifica di sorgente, poi la codifica di canale ed infine la codifica di segnale. Il canale di comunicazione può essere un canale di trasmissione oppure di memorizzazione delle informazioni.

La codifica di sorgente ha lo scopo di ridurre la bitrate prodotta dalla sorgente eliminando la ridondanza presente. Tutte le sorgenti naturali sono infatti caratterizzate da ridondanza nella rappresentazione dell'informazione, utilizzata dal destinatario per aumentare il livello di comprensione o semplificarne l'acquisizione.

La codifica di canale inserisce ridondanza nella comunicazione, aumentando la bitrate. In questo caso, però, la ridondanza introdotta è controllata e nota e viene utilizzata in fase di ricezione per verificare la correttezza della comunicazione ed eventualmente correggere gli eventuali errori presenti. Quando è disponibile un canale di ritorno, in grado di mettere in comunicazione la destinazione con la sorgente, è possibile utilizzare uno schema di ritrasmissione delle informazioni errate di tipo Automatic Repeat-reQuest (ARQ), altrimenti si può procedere direttamente alla correzione mediante un meccanismo Forward Error Correction (FEC). Un meccanismo ARQ richiede alla codifica la sola capacità di rivelazione degli errori mentre un meccanismo FEC richiede anche la capacità di correzione. A parità di numero di bit errati, le esigenze in termini di ridondanza per la rivelazione degli errori sono circa la metà rispetto a quelle per la correzione degli errori.

La codifica di segnale consiste nell'adattamento o nella creazione di un segnale elettrico od ottico, compatibile con le caratteristiche del canale. Spesso questa codifica viene anche chiamata modulazione ed è legata alla specifica di opportune forme d'onda.

Nel seguito l'attenzione viene posta alla codifica di sorgente. Il processo di codifica viene effettuato alla sorgente e necessita di un processo inverso, detto di decodifica, che viene effettuato prima della riproduzione o dell'utilizzo del contenuto. A seconda dei requisiti di velocità di compressione e tempo di latenza nella compressione, il processo di codifica e decodifica di sorgente può essere effettuato via software o hardware.

3.1.1 Entropia

La codifica di sorgente cerca di ridurre la ridondanza statistica della sorgente. Per una codifica lossless non è possibile richiedere che ogni singola sequenza venga sempre accorciata: per un semplice argomento di conteggio, non esistono abbastanza parole più corte per rappresentare in modo iniettivo tutte le stringhe di una data lunghezza. La compressione deve quindi essere valutata in media oppure rispetto a una classe di sorgenti con regolarità sfruttabili.

Per un evento di probabilità $p > 0$ si definisce autoinformazione

$$I(p) = -\log_2 p.$$

Questa è una definizione. Il logaritmo è naturale perché trasforma il prodotto delle probabilit-

ità di eventi indipendenti nella somma delle informazioni associate.

Siano X e C due variabili aleatorie discrete. Per realizzazioni x e c con probabilità positiva si definisce informazione mutua puntuale

$$i(x; c) = \log_2 \frac{P(X = x, C = c)}{P(X = x)P(C = c)}.$$

Usando $P(x, c) = P(x | c)P(c)$ si ottiene equivalentemente

$$i(x; c) = \log_2 \frac{P(X = x | C = c)}{P(X = x)}.$$

Dimostrazione dell'equivalenza. Dalla definizione di probabilità condizionata, $P(x, c) = P(x | c)P(c)$. Sostituendo nella prima definizione,

$$\log_2 \frac{P(x | c)P(c)}{P(x)P(c)} = \log_2 \frac{P(x | c)}{P(x)}.$$

Questo conclude la dimostrazione.

La prima forma rende immediata la simmetria $i(x; c) = i(c; x)$. Se X e C sono indipendenti, $P(x, c) = P(x)P(c)$ e quindi $i(x; c) = 0$. Se conoscere c determina univocamente x , allora $P(x | c) = 1$ e l'informazione puntuale coincide con $-\log_2 P(x)$.

Mediando rispetto alla distribuzione congiunta si definisce l'informazione mutua

$$I(X; C) = \sum_x \sum_c P(x, c) \log_2 \frac{P(x, c)}{P(x)P(c)}.$$

L'entropia della sorgente è invece definita come autoinformazione media:

$$H(X) = E[-\log_2 P(X)] = -\sum_x P(x) \log_2 P(x),$$

con la convenzione $0 \log 0 = 0$.

Se la sorgente possiede n simboli equiprobabili, $p_j = 1/n$, allora

$$H(X) = \log_2 n.$$

Dimostrazione. Sostituendo $p_j = 1/n$ nella definizione,

$$H(X) = -\sum_{j=1}^n \frac{1}{n} \log_2 \frac{1}{n} = -n \frac{1}{n} (-\log_2 n) = \log_2 n.$$

Questo conclude la dimostrazione.

La distribuzione uniforme massimizza l'entropia fra tutte le distribuzioni su n simboli:

$$H(X) \leq \log_2 n,$$

con uguaglianza se e solo se $p_j = 1/n$ per ogni j .

Dimostrazione. Sia U la distribuzione uniforme, $u_j = 1/n$. Definiamo la divergenza relativa

$$D(P\|U) = \sum_j p_j \log_2 \frac{p_j}{u_j}.$$

La disuguaglianza $\ln z \leq z - 1$ implica

$$-\ln 2 D(P\|U) = \sum_j p_j \ln \frac{u_j}{p_j} \leq \sum_j p_j \left(\frac{u_j}{p_j} - 1 \right) = \sum_j u_j - \sum_j p_j = 0.$$

Quindi $D(P\|U) \geq 0$. Inoltre

$$\begin{aligned} D(P\|U) &= \sum_j p_j \log_2 (np_j) \\ &= \log_2 n + \sum_j p_j \log_2 p_j \\ &= \log_2 n - H(X). \end{aligned}$$

Da $D(P\|U) \geq 0$ segue $H(X) \leq \log_2 n$. L'uguaglianza nella Gibbs inequality si verifica soltanto quando $P = U$. Questo conclude la dimostrazione.

Per una sorgente binaria con $P(0) = q$ e $P(1) = 1 - q$ si ottiene

$$H_2(q) = -q \log_2 q - (1 - q) \log_2 (1 - q).$$

La funzione ha massimo unitario in $q = 1/2$.

Dimostrazione. Per $0 < q < 1$,

$$H'_2(q) = \log_2 \frac{1 - q}{q}.$$

L'unico punto stazionario è $q = 1/2$. La derivata seconda è

$$H''_2(q) = -\frac{1}{\ln 2} \left(\frac{1}{q} + \frac{1}{1 - q} \right) < 0,$$

quindi il punto è un massimo stretto. Sostituendo $q = 1/2$ nella definizione si ottiene $H_2(1/2) = 1$. Questo conclude la dimostrazione.

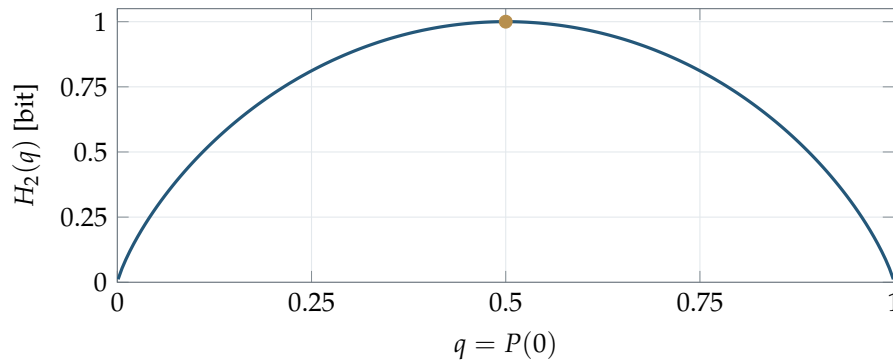


Figura 3.1. Entropia binaria. L'incertezza è nulla quando uno dei due simboli è certo e massima quando i simboli sono equiprobabili.

L'entropia misura quindi l'incertezza media del modello di sorgente, non la "complessità semantica" del contenuto. Una sorgente può produrre testo linguisticamente significativo ma avere entropia statistica bassa se la distribuzione e le dipendenze sono molto prevedibili. I codificatori moderni cercano proprio di costruire un modello condizionato che renda prevedibile il prossimo simbolo prima della codifica entropica.

3.1.2 Classificazione

Gli algoritmi di compressione si dividono in due grandi famiglie: senza perdita (lossless) e con perdita (lossy). Gli algoritmi lossless operano in modo reversibile, cioè riducono la quantità di ridondanza presente nella sorgente senza alterare minimamente l'informazione presente. È quindi possibile, se necessario, reintrodurre la ridondanza al ricevitore, sulla base della sola conoscenza dell'informazione trasmessa.

Nel caso degli algoritmi lossy, al ricevitore non si vuole ottenere esattamente la stessa informazione presente alla sorgente, ma una versione che venga percepita, da un essere umano, come tale. Gli algoritmi lossy sono solitamente parametrici, con la possibilità di definire il livello di degradazione rispetto all'originale sulla base di un parametro. Quanto più è elevata la degradazione della sorgente, tanto più si tende ad aumentare il livello di compressione. La parte di informazione mancante viene compensata dalla capacità di interpolazione del cervello umano. Le compressioni lossy sono solitamente legate a meccanismi di rappresentazione di sorgenti naturali e sono tarate sulle caratteristiche di un uomo medio. Questi sistemi di codifica possono mostrare ottime prestazioni rispetto ai sensi di un essere umano, ma non garantiscono alcuna fedeltà rispetto ad altre specie viventi.

Un'ulteriore classificazione prevede la distinzione tra codici a lunghezza fissa e codici a lunghezza variabile. I codici a lunghezza fissa hanno il vantaggio di una maggiore semplicità di realizzazione e elaborazione ma in generale sono ridondanti rispetto a quanto strettamente necessario. I codici a lunghezza variabile sono complessi da gestire, sia in fase di codifica che di decodifica, ma hanno il vantaggio di offrire un sistema in grado di avvicinarsi ai limiti entropici di rappresentazione dell'informazione.

Un esempio di codice a lunghezza variabile è il codice di Morse, riportato in tabella 3.1. Nel codice Morse una linea equivale a tre punti, lo spazio tra punti e/o linee nella stessa lettera equivale ad un punto, lo spazio tra due lettere equivale a tre punti, lo spazio tra due parole equivale a cinque punti.

A	.-	N	-.	1	.- - - -	.	.- .- .-
B	-...	O	- - -	2	.. - - -	,	- - - .. - -
C	- . - .	P	. - - .	3	... - -	?	.. - - ..
D	- ..	Q	- - . -	4	 -	(	- . - - .
E	.	R	. - .	5		)	- . - - . -
F	.. - .	S	...	6	-	-	- -
G	- - .	T	-	7	- - ...	"	.- .. - .
H		U	.. -	8	- - - ..	_	.. - - . -
I	..	V	... -	9	- - - - .	'	. - - - - .
J	. - - -	W	. - -	0	- - - - -	:	- - - ...
K	- . -	X	- .. -	/	- .. - .	;	- . - . - .
L	. - ..	Y	- . - -	+	. - . - .	\$	... - .. -
M	- -	Z	- - ..	=	- ... -		

Tabella 3.1. Codice Morse, codice a lunghezza variabile.

3.2 Testo

La compressione del testo è tipicamente una codifica di tipo lossless. Il testo, infatti, può essere indirizzato a due diverse tipologie di fruitori: esseri umani e macchine. Sebbene i primi fruitori siano in grado di comprendere il contenuto informativo anche in presenza di lievi errori, la seconda tipologia di fruitori è molto rigida nell'interpretare i messaggi.

Se estendiamo leggermente il concetto di testo ed invece di considerare solo i caratteri, ad esempio ASCII, includiamo anche simboli binari, allora il testo può rappresentare il codice macchina di un programma da eseguire all'interno di una CPU e un errore, anche solo in un bit, può cambiare completamente l'esecuzione del programma, trasformando, ad esempio, un'istruzione di salto in un'istruzione di assegnamento.

3.2.1 RLE

Il Run-Length Encoding (RLE) è una tecnica di compressione lossless nella quale vengono identificati e processati dei componenti chiamati run. Un run è una sequenza nella quale uno stesso valore, o una stessa stringa, è presente più volte in modo consecutivo.

Nella versione più semplice del RLE ogni run è costituito dalla ripetizione di un singolo carattere: tale run viene riconosciuto e sostituito con il simbolo che lo caratterizza, assieme all'indicazione della lunghezza del run stesso.

Esistono vari formati con i quali vengono rappresentate le informazioni nel RLE. A titolo di esempio consideriamo un semplice sistema di codifica in cui la stringa AAAABBAABBB-BAAAAAABB ha come codifica RLE 4A 2B 2A 4B 6A 2B, composta da un carattere che identifica la lunghezza del run e da un carattere che definisce l'elemento del run. In questo formato la lunghezza del run non è memorizzata in ASCII, ma direttamente come numero su 8 bit, con possibilità di rappresentare lunghezze da 1 a 256 elementi, anche se graficamente è riportata in ASCII per motivi di visualizzazione. Se il run risulta più lungo di 256 elementi si procede spezzandolo in più run successivi di dimensione massima pari a 256 elementi. Questa codifica per RLE è di tipo a lunghezza fissata.

Se la lunghezza dell' i -esimo run è f_i , con L run presenti, si ha che la lunghezza complessiva del file originale è $\sum_{i=1}^L f_i$ mentre la lunghezza del file dopo l'RLE è $2L$. L'RLE risulta conveniente a condizione che

$$\sum_{i=1}^L f_i > 2L$$

Passando a una descrizione statistica, se $p_j = P(F = j)$ è la distribuzione della lunghezza F dei run, la lunghezza media è

$$E[F] = \sum_j j p_j.$$

Per una sequenza lunga, quando la media empirica converge al valore atteso, la codifica risulta mediamente conveniente se $E[F] > 2$. Sul singolo file, invece, la condizione esatta resta $L^{-1} \sum_{i=1}^L f_i > 2$.

Se il file è composto da caratteri fortemente casuali e $p_1 = 1$, applicare un RLE con questa tipologia di codifica è fortemente sconsigliato.

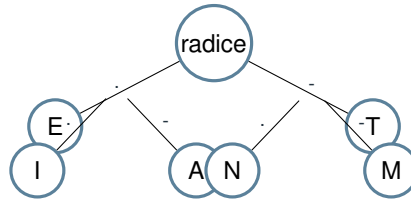
Una forma di RLE con maggiore complessità realizzativa prevede di utilizzare run composti da ripetizioni di stringhe generiche e non di soli singoli caratteri. Purtroppo le stringhe generiche hanno una lunghezza variabile per cui diventa complesso sia identificarle che rappresentarle.

Il RLE può essere utilizzato congiuntamente ad altre forme di compressione. Tipicamente viene eseguita una codifica di tipo RLE prima di effettuare altre tipologie di compressioni.

3.2.2 Huffman statico

La codifica di Huffman realizza un codice a lunghezza variabile ed è di tipo lossless. L'idea di base è di codificare i simboli più frequenti con parole di codice più corte e di generare una rappresentazione che consenta di effettuare la decodifica in modo semplice, tramite un albero binario in cui i simboli sono associati alle foglie. Il fatto che i simboli siano associati alle

foglie dell'albero, e non ad un nodo intermedio, viene detta proprietà di prefisso e consente la decodifica immediata del simbolo.



Nel Morse alcuni simboli sono nodi interni:
il separatore temporale fra lettere è parte della decodifica.

Figura 3.2. Albero concettuale del codice Morse. A differenza di un codice a prefisso, lettere come E e T corrispondono a nodi interni e richiedono una separazione temporale per essere riconosciute univocamente.

Un esempio di codice a lunghezza variabile, derivato dallo studio della lingua inglese e diretto a minimizzare il numero di battute su un tasto telegrafico, è il codice Morse riportato in figura 3.1. La rappresentazione ad albero della versione estesa del codice Morse è riportata in figura 3.2, dove si nota che non è valida la proprietà del prefisso. Per decodificare la lettera E, ad esempio, è necessario attendere le battute successive per verificare se c'è uno spazio di lettera, altrimenti il punto che caratterizza la E potrebbe essere parte della codifica di un'altra simbolo (una qualunque dell'albero di sinistra).

Bit per simbolo

Un parametro significativo per i codici a lunghezza variabile è il numero di bit medio necessario a rappresentare un simbolo. Sia x_j il generico simbolo di un alfabeto composto da n elementi ed m_j la lunghezza in bit della parola di codice corrispondente a x_j , il numero medio di bit per simbolo R , è dato da:

$$R = \sum_{j=1}^n m_j P(x_j)$$

La ricerca del codice ottimo a lunghezza variabile prevede la determinazione dell'attribuzione delle stringhe di bit, e quindi delle lunghezze m_j , che minimizzano R .

Rappresentando un codice binario a prefisso con un albero, le lunghezze m_j devono soddisfare la disuguaglianza di Kraft

$$\sum_{j=1}^n 2^{-m_j} \leq 1.$$

La condizione è necessaria e sufficiente per l'esistenza di un codice binario a prefisso con quelle lunghezze.

Per la necessità, sia $M = \max_j m_j$ e si prolunghi idealmente ogni parola di codice fino al livello M dell'albero. Una parola di lunghezza m_j occupa un sottoalbero contenente 2^{M-m_j} foglie. Poiché il codice è a prefisso, questi sottoalberi sono disgiunti; pertanto

$$\sum_j 2^{M-m_j} \leq 2^M.$$

Dividendo per 2^M si ottiene la disuguaglianza di Kraft.

Per la sufficienza si ordinino le lunghezze in modo non decrescente e si costruisca l'albero assegnando, per ogni m_j , un nodo libero a quel livello. L'assegnazione di tale nodo rende indisponibili 2^{M-m_j} foglie al livello M . La disuguaglianza di Kraft assicura che il numero totale di foglie richieste non supera 2^M ; scegliendo sempre un nodo non discendente da una parola già assegnata si ottiene quindi un insieme di foglie a prefisso. Questa è anche la base della costruzione dei codici canonici.

La relazione fra entropia e lunghezza media si può ora stabilire in modo preciso. Per qualunque codice a prefisso con probabilità $p_j = P(x_j)$, poniamo

$$K = \sum_j 2^{-m_j} \leq 1, \quad q_j = \frac{2^{-m_j}}{K}.$$

Allora

$$\begin{aligned} R - H(X) &= \sum_j p_j \log_2 \frac{p_j}{2^{-m_j}} \\ &= \sum_j p_j \log_2 \frac{p_j}{q_j} - \log_2 K \\ &= D(P\|Q) - \log_2 K \geq 0. \end{aligned}$$

Si è usata la non negatività della divergenza relativa e il fatto che $K \leq 1$. Ne segue il limite fondamentale

$$H(X) \leq R.$$

Per dimostrare che esiste sempre un codice a prefisso a meno di un bit dall'entropia, si scelgono le lunghezze di Shannon

$$m_j = \lceil -\log_2 p_j \rceil.$$

Da $m_j \geq -\log_2 p_j$ segue $2^{-m_j} \leq p_j$, quindi Kraft è soddisfatta. Inoltre

$$m_j < -\log_2 p_j + 1,$$

e mediando rispetto a p_j si ottiene

$$R_S < H(X) + 1.$$

L'algoritmo di Huffman produce, fra i codici binari a prefisso simbolo per simbolo, una lunghezza media minima. Di conseguenza

$$H(X) \leq R_H \leq R_S < H(X) + 1.$$

Il limite superiore non deriva quindi da una proprietà arbitraria delle lunghezze intere: viene prima costruito un codice di Shannon che lo soddisfa, e l'ottimalità di Huffman garantisce che Huffman non può essere peggiore.

Algoritmo di Huffman statico

L'algoritmo di Huffman assume che siano note le frequenze di ogni singolo elemento dell'alfabeto da codificare; tali frequenze sono equivalentemente considerabili come probabilità, qualora sia noto il numero totale degli elementi che compongono il messaggio da codificare e che questo numero sia sufficientemente elevato.

Per conoscere tali probabilità si possono utilizzare due meccanismi: il primo prevede la scansione del file per collezionare l'esatta statistica, il secondo prevede l'utilizzo di distribuzioni note, ottenute tramite analisi effettuate offline. Il primo metodo ha lo svantaggio

di richiedere un passaggio dell'intero file (o di una sua parte nel caso si proceda a generare una statistica su un campione del file in modo da accelerare la procedura) e richiede comunque la disponibilità dell'intero contenuto informativo da comprimere. Il secondo metodo ha il vantaggio di non richiedere tempo per raccogliere la statistica; di non richiedere la disponibilità del contenuto informativo e quindi di poter essere utilizzato anche con un testo non completamente disponibile. Il maggior problema di questo secondo metodo risiede nella potenziale discordanza tra la statistica assunta e quella reale, discrepanza che può portare a una diminuzione della capacità di compressione.

Supposte note le probabilità dei simboli, l'algoritmo di Huffman mantiene un insieme di nodi pesati. Inizialmente esiste un nodo per ciascun simbolo x_j , con peso $p_j = P(x_j)$. A ogni iterazione si scelgono i due nodi di peso minimo, con pesi a e b , e li si sostituisce con un nodo padre di peso

$$a + b.$$

Dopo ogni fusione il numero dei nodi diminuisce di uno; partendo da n foglie sono quindi necessarie esattamente $n - 1$ fusioni per ottenere la radice. Assegnando 0 e 1 ai due rami uscenti da ogni nodo interno si ottengono le parole di codice. L'assegnazione dei due bit ai figli può essere scambiata senza modificare le lunghezze né la prestazione del codice. In presenza di parità fra pesi, differenti regole di scelta possono produrre alberi diversi ma con la stessa lunghezza media ottima.

Il meccanismo descritto genera un albero che parte dalle foglie (i simboli) ed arriva alla radice con un unico simbolo a probabilità 1 che rappresenta la probabilità dell'intero alfabeto. L'attribuzione della codifica avviene dalla radice verso le foglie, ponendo convenzionalmente un bit ad 1 per ogni biforcazione (corrispondente alla fusione di due simboli) al sottoalbero con probabilità più elevata, 0 a quello con probabilità più bassa. Questa attribuzione può essere invertita ottenendo ancora un'attribuzione valida, con la regola del prefisso rispettata.

Considerando che l'ordinamento che avviene ad ogni passo può incappare in situazioni in cui due simboli, originali oppure ottenuti per fusione dei precedenti, risultano caratterizzati dalla stessa probabilità, si ha che il codice che si ottiene non è in generale unico, ma dipende dalle scelte effettuate in situazioni ad uguale probabilità. Tali scelte, comunque, non vanno a variare il numero di bit per simbolo che il codice è in grado di generare.

Ottimalità dell'algoritmo di Huffman. L'algoritmo greedy di fusione dei due pesi minimi produce un codice a prefisso di lunghezza media minima.

Dimostrazione. Consideriamo un qualunque albero binario a prefisso ottimo. Due foglie alla massima profondità possono essere scelte come sorelle: se una foglia profonda avesse probabilità maggiore di una foglia meno profonda, scambiandole la lunghezza media non aumenterebbe e, se la disuguaglianza fosse stretta, diminuirebbe. Possiamo quindi disporre i due simboli meno probabili, di probabilità p_a e p_b , come foglie sorelle alla massima profondità.

Sostituiamo ora le due foglie con il loro padre, al quale assegniamo probabilità $p_a + p_b$. Se le due foglie hanno profondità $d + 1$, il loro contributo alla lunghezza media è

$$p_a(d + 1) + p_b(d + 1) = (p_a + p_b)d + (p_a + p_b).$$

Il primo termine è esattamente il contributo del simbolo fuso nell'albero ridotto; il secondo, $p_a + p_b$, è una costante indipendente dalla struttura restante. Minimizzare la lunghezza media dell'albero originario equivale quindi a minimizzare quella dell'albero ridotto.

Per induzione sul numero di simboli, il problema ottimo con n simboli si riduce al problema ottimo con $n - 1$ simboli dopo avere fuso i due pesi minimi. Ripetendo il procedimento si ottiene precisamente l'algoritmo di Huffman. Questo conclude la dimostrazione.

Esempi

Consideriamo un alfabeto con $n = 7$ simboli caratterizzato dalle prime 7 lettere dell'alfabeto, per comodità prendiamo i primi 7 caratteri ASCII: $x_1 = A$, $x_2 = B$, $x_3 = C$, $x_4 = D$, $x_5 = E$, $x_6 = F$, $x_7 = G$. Le probabilità assegnate a queste 7 lettere sono rispettivamente $P(x_1) = 0.30$, $P(x_2) = 0.005$, $P(x_3) = 0.35$, $P(x_4) = 0.10$, $P(x_5) = 0.20$, $P(x_6) = 0.04$, $P(x_7) = 0.005$. In figura 3.3 viene riportato un esempio grafico di realizzazione dell'algoritmo con tutti i passi che lo caratterizzano. Si osserva che al passo $k = 5$, dopo l'ordinamento, vi sono due diverse possibilità, proprio dovute al fatto che l'algoritmo non porta ad un codice unico.

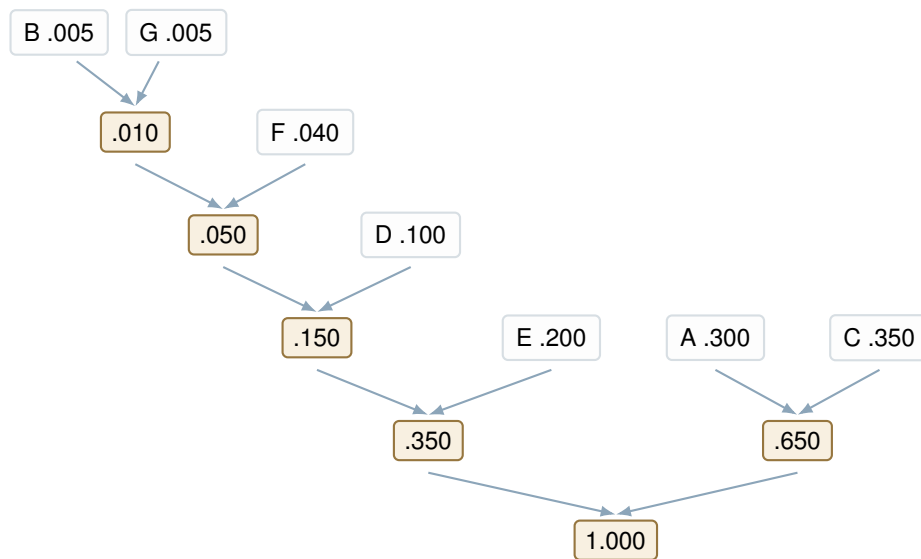


Figura 3.3. Esempio di fusioni successive di Huffman per le probabilità del testo. In presenza di pesi uguali alcune fusioni possono essere scelte in modi diversi, generando alberi differenti ma ugualmente ottimi.

L'albero relativo all'applicazione dell'algoritmo di Huffman (decisione parte superiore) di figura 3.3 è riportato in figura 3.4.

Da questo albero deriva la codifica: $c_1 = 10$, $m_1 = 2$, $c_2 = 110001$, $m_2 = 6$, $c_3 = 0$, $m_3 = 1$, $c_4 = 1101$, $m_4 = 4$, $c_5 = 111$, $m_5 = 3$, $c_6 = 11001$, $m_6 = 5$, $c_7 = 110000$, $m_7 = 6$. Calcolando il numero di bit per simbolo si ottiene $R = 2.21$. Valutando l'Entropia si ha $H(X) = 2.11$, ed in effetti risulta $[H(X) = 2.11] < [R = 2.21] < [H(X) + 1 = 3.11]$

L'albero relativo all'applicazione dell'algoritmo di Huffman (decisione parte inferiore) di figura 3.3 è riportato in figura 3.5.

Da questo albero deriva la codifica: $c_1 = 10$, $m_1 = 2$, $c_2 = 00001$, $m_2 = 5$, $c_3 = 11$, $m_3 = 2$, $c_4 = 001$, $m_4 = 3$, $c_5 = 01$, $m_5 = 2$, $c_6 = 0001$, $m_6 = 4$, $c_7 = 00000$, $m_7 = 5$, che risulta completamente diversa da quella ottenuta dall'albero di figura 3.4. Coerentemente a quanto discusso teoricamente, anche questa codifica, eroga lo stesso numero di bit per simbolo, pari a $R = 2.21$.

Rispetto alla codifica ASCII a lunghezza fissata, con 8 bit per simbolo, la codifica di Huffman per questo alfabeto porta ad un sistema caratterizzato da 2.21 bit per simbolo, con un rapporto di compressione di $\frac{8}{2.21} = 3.6$.

Consideriamo ora un alfabeto con $n = 4$, caratterizzato dalle prime 4 lettere dell'alfabeto, per comodità prendiamo i primi 4 caratteri ASCII: $x_1 = A$, $x_2 = B$, $x_3 = C$, $x_4 = D$. Le

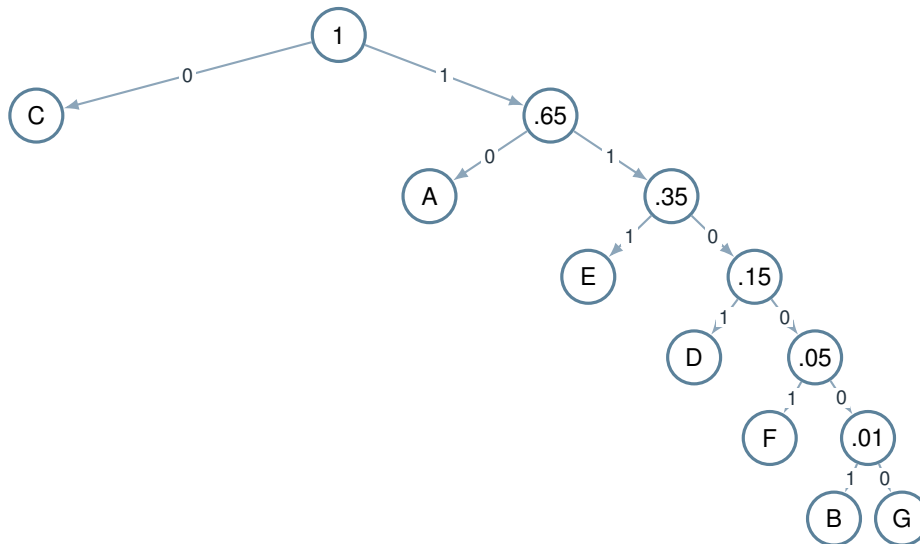


Figura 3.4. Uno degli alberi di Huffman compatibili con l'esempio a sette simboli. La parola di codice si legge dalla radice alla foglia.

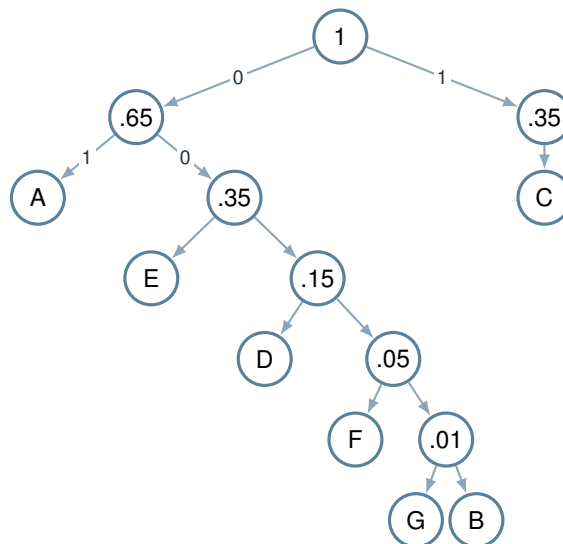


Figura 3.5. Albero alternativo ottenibile a causa dei pareggi nei pesi. Le parole cambiano, ma la lunghezza media resta $R = 2.21$ bit/simbolo.

probabilità assegnate a queste 4 lettere sono rispettivamente $P(x_1) = 0.25$, $P(x_2) = 0.5$, $P(x_3) = 0.125$, $P(x_4) = 0.125$. Si osserva che queste probabilità sono scrivibili nel formato 2^{-k} , per cui ci aspettiamo $R = H(X)$.

In figura 3.6 viene riportato un esempio grafico di realizzazione dell'algoritmo con tutti i passi che lo caratterizzano e l'albero da utilizzare per l'allocazione dei bit ai simboli. Si osserva che è possibile generare più di una codifica, ma per comodità se ne è riportata una sola.

Da questo albero deriva la codifica: $c_1 = 01$, $m_1 = 2$, $c_2 = 1$, $m_2 = 1$, $c_3 = 001$, $m_3 = 3$, $c_4 = 000$, $m_4 = 3$. Il numero di bit per simbolo risulta $R = 1.75$ e l'Entropia $H(X) = 1.75$. Risulta quindi, come previsto, $R = H(X)$.

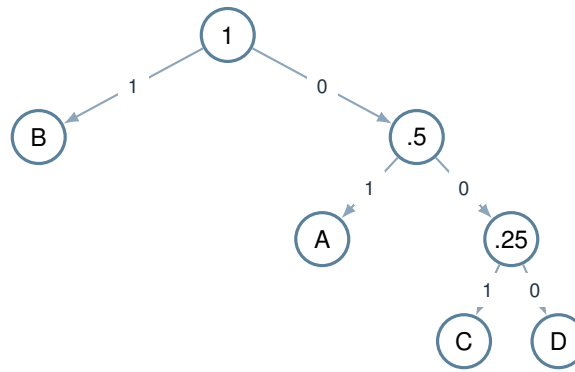


Figura 3.6. Huffman per probabilità diadiche: $P(B) = 1/2$, $P(A) = 1/4$, $P(C) = P(D) = 1/8$. Le lunghezze coincidono con $-\log_2 p_j$ e quindi $R = H(X) = 1.75$.

Memorizzazione

Essendo la codifica a lunghezza variabile è fondamentale avere delle funzioni efficienti per manipolare lo stream di bit. Tali funzioni devono consentire l'inserimento e l'estrazione di un numero arbitrario di bit, indipendentemente dalla gestione fisica della memoria.

Tipicamente la granularità effettiva di memorizzazione è un multiplo di 8 bit, dipendentemente dal sistema operativo e dall'architettura della macchina. Dato tale vincolo, il file composto da segmenti a lunghezza variabile viene terminato con dei bit di padding in modo da raggiungere la dimensione necessaria.

Particolare attenzione deve essere posta alla memorizzazione dell'albero di codifica, qualora non sia già noto al ricevitore. Tale memorizzazione rappresenta un overhead che può diminuire la capacità di compressione del sistema nel caso il file sia particolarmente breve. L'effetto di questo overhead tende a svanire con l'aumentare della dimensione del file.

In alternativa alla trasmissione dell'albero, è possibile trasmettere l'alfabeto dei simboli utilizzati dalla sorgente con le relative probabilità, lasciando al blocco di decompressione il calcolo dell'albero di Huffman per effettuare la decodifica, a condizione che vengano effettuate le stesse scelte nei casi ad uguale probabilità.

3.2.3 Huffman dinamico

La codifica di Huffman trattata nella sezione precedente prevede che vi sia una conoscenza sufficiente della sorgente per generare l'albero di codifica o per utilizzarne uno già esistente e noto sia al codificatore che al decodificatore. Qualora la sorgente sia incognita si pone il problema di non poter effettuare una codifica secondo Huffman sino a quando non si è collezionata una statistica sufficiente, il che prevede di ritardare la codifica, e quindi la trasmissione. Un simile approccio non può essere utilizzato in un sistema che necessita di tempi di consegna stringenti.

Un approccio alternativo, che risolve le problematiche esposte, prevede la creazione dinamica dell'albero: ogni simbolo originato dalla sorgente aggiorna la statistica mantenuta dall'encoder; al cambiare delle frequenze cambiano dinamicamente anche le corrispondenze fra simboli e parole di codice. Il decoder aggiorna una copia dello stesso stato applicando, nello stesso ordine, le medesime operazioni dell'encoder; ogni simbolo già presente nell'albero viene trasmesso utilizzando la sua attuale codifica, mentre un simbolo che non è presente nell'albero viene necessariamente trasmesso in chiaro.

Questo meccanismo prevede la creazione dell'albero di codifica contestualmente alla codifica, riuscendo ad effettuare l'intero processo in un solo passo. Viceversa, l'algoritmo di

Huffman statico può richiedere due passi, uno per la creazione della statistica della sorgente ed uno per la codifica.

Algoritmo di Huffman dinamico

Un meccanismo, semplice, per realizzare un albero di Huffman dinamico, consiste nell'accumulare le occorrenze dei simboli della sorgente alla generazione p -esima e ricalcolare l'albero di codifica utilizzando l'algoritmo di Huffman, sia all'encoder sia al decoder, per ogni simbolo generato.

Chiamiamo g_p l'indice nell'alfabeto del p -esimo simbolo generato dalla sorgente e $o_{j,p}$ il numero di occorrenze del simbolo j -esimo alla p -esima generazione. Il vettore delle occorrenze è inizializzato con la condizione $o_{j,0} = 0$. Alla generazione p -esima il vettore delle occorrenze viene aggiornato secondo

$$o_{j,p} = o_{j,p-1} + \mathbf{1}_{\{j=g_p\}},$$

dove $\mathbf{1}$ è la funzione indicatrice. L'albero di Huffman viene costruito sulla base del vettore delle occorrenze. Normalizzare tutti i conteggi per la stessa costante non cambia l'albero, per cui è equivalente usare le frequenze empiriche

$$\hat{p}_{j,p} = \frac{o_{j,p}}{\sum_i o_{i,p}}.$$

Queste sono stime delle probabilità della sorgente, non necessariamente le probabilità vere $P(x_j)$.

L'alfabeto dei simboli originali viene arricchito con un simbolo speciale, chiamato Not Yet Transmitted (NYT), che viene inserito nell'albero con occorrenza 1 (statica, cioè non si tiene conto dell'utilizzo in fase di costruzione dell'albero) ed è presente come condizione di inizializzazione dell'albero, con $p = 0$. In questo modo, a qualsiasi generazione p , è sempre disponibile il simbolo NYT, con lunghezza di codifica sempre più lunga. Il simbolo NYT viene utilizzato come codice di escape per indicare che gli 8 bit successivi sono la codifica ASCII (o qualsiasi altra) del nuovo simbolo della sorgente.

Un qualsiasi simbolo già presente nell'albero viene trasmesso con la sua codifica attuale. L'aggiornamento dell'albero al codificatore ed al decodificatore, funzione dell'aggiornamento dell'occorrenza del simbolo p -esimo, avviene successivamente alla trasmissione del simbolo stesso.

Sincronizzazione fra codificatore e decodificatore

Il meccanismo adattativo funziona soltanto se codificatore e decodificatore costruiscono esattamente lo stesso albero dopo ogni simbolo. Questo richiede che entrambi partano dallo stesso stato e applichino la stessa regola deterministica per risolvere i pareggi fra pesi uguali.

Dimostrazione. Procediamo per induzione sul numero p di simboli già decodificati. Per $p = 0$ i due lati partono per ipotesi dallo stesso stato iniziale, contenente soltanto NYT. Supponiamo che dopo $p - 1$ simboli i vettori delle occorrenze e l'albero coincidano. Il p -esimo simbolo viene interpretato usando proprio quell'albero comune; se è nuovo, la sequenza NYT più rappresentazione esplicita identifica il medesimo simbolo a entrambi i lati. Entrambi incrementano quindi la stessa componente $o_{j,p-1}$ e ottengono lo stesso vettore $o_{j,p}$. Poiché la costruzione di Huffman e la regola di tie-breaking sono deterministiche, ricostruiscono lo stesso albero. Per induzione la sincronizzazione vale per ogni p . Questo conclude la dimostrazione.

L'algoritmo descritto ricalcola l'albero completo dopo ogni simbolo ed è quindi pedagogicamente semplice ma non il più efficiente. Gli algoritmi classici di Faller–Gallager–Knuth e

Vitter aggiornano l'albero in modo incrementale mantenendo opportune proprietà di ordine e riducono il costo computazionale senza cambiare l'idea fondamentale di adattare il codice alle statistiche osservate.

Esempio completo

Consideriamo la sequenza di otto simboli

AA BBB C,

dove gli spazi sono simboli effettivi. La tabella 3.2 riporta tutti i passi della versione didattica appena descritta. Per un simbolo mai osservato si trasmette il codice corrente di NYT seguito da 8 bit che identificano il nuovo simbolo; per un simbolo già noto si trasmette soltanto la parola corrente di Huffman.

p	simbolo	nuovo?	azione di codifica	bit aggiunti	bit totali
1	A	sì	A esplicita, stato iniziale NYT	8	8
2	A	no	codice corrente di A	1	9
3	spazio	sì	NYT + simbolo esplicito	9	18
4	B	sì	NYT + simbolo esplicito	10	28
5	B	no	codice corrente di B	2	30
6	B	no	codice corrente di B	1	31
7	spazio	no	codice corrente dello spazio	3	34
8	C	sì	NYT + simbolo esplicito	11	45

Tabella 3.2. Evoluzione completa dell'esempio di Huffman dinamico. Le lunghezze dipendono dalla regola di costruzione e di tie-breaking dichiarata nel testo.

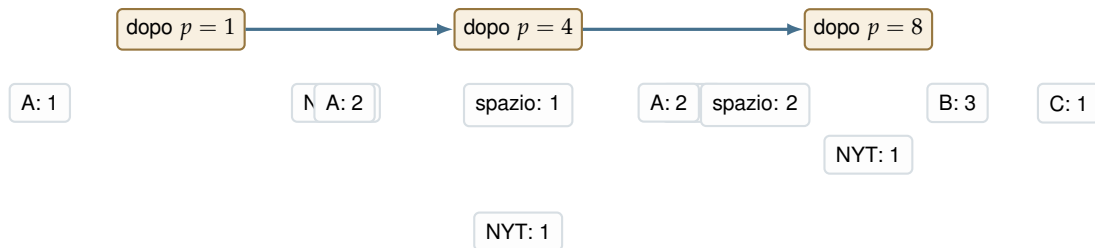


Figura 3.7. Tre snapshot dello stato statistico dell'esempio adattativo. A ogni passo il nuovo albero di Huffman è costruito a partire da questi conteggi più il simbolo NYT.

Al termine dell'esempio i conteggi sono $B=3$, $A=2$, $\text{spazio}=2$ e $C=1$. L'albero finale, se fosse noto in anticipo e usato come codice statico, richiederebbe 17 bit per rappresentare gli otto simboli con le lunghezze dell'esempio. Tuttavia il decoder dovrebbe conoscere anche alfabeto e statistiche. Una rappresentazione molto semplice delle quattro identità ASCII costa già $4 \cdot 8 = 32$ bit; aggiungendo almeno 2 bit per ciascuno dei quattro conteggi si ottengono altri 8 bit. Con questa codifica esplicita dell'header, il totale statico sarebbe dunque almeno

$$17 + 32 + 8 = 57 \text{ bit,}$$

contro i 45 bit del procedimento adattativo dell'esempio.

Dimostrazione. I 17 bit sono il costo dei dati quando si usa l'albero finale. La descrizione elementare dell'alfabeto usa quattro simboli da 8 bit, quindi 32 bit. Quattro conteggi rappresentabili con 2 bit ciascuno richiedono 8 bit. Sommando i tre contributi si ottengono 57 bit. Questo conclude la dimostrazione per la particolare rappresentazione di header assunta.

Il confronto non prova che un codice adattativo sia sempre migliore di uno statico: su file lunghi l'overhead dell'header statico viene ammortizzato, mentre l'adattativo paga la fase iniziale in cui le statistiche sono ancora poco accurate. L'esempio mostra invece perché la codifica online può essere vantaggiosa quando la sorgente non è nota in anticipo o cambia nel tempo.

3.2.4 Codifica aritmetica

La codifica aritmetica non assegna una parola separata a ciascun simbolo. Rappresenta invece l'intero messaggio mediante un intervallo sempre più piccolo contenuto in $[0, 1)$. A ogni simbolo della sorgente corrisponde una frazione dell'intervallo corrente proporzionale alla sua probabilità. Il messaggio completo è identificato da qualunque numero appartenente all'intervallo finale, purché il decoder conosca lo stesso modello probabilistico e il criterio di terminazione.

Questo meccanismo supera una limitazione dei codici simbolo-per-simbolo: una probabilità come 0.9 corrisponde a un'informazione di circa 0.152 bit, ma una singola parola binaria non può avere lunghezza frazionaria. La codifica aritmetica accumula invece l'informazione di molti simboli prima di emettere bit e può avvicinarsi molto più finemente al limite entropico.

Algoritmo di codifica aritmetica

Consideriamo una sorgente di n simboli x_j con probabilità $p_j = P(X = x_j)$ e definiamo le cumulative

$$u_0 = 0, \quad u_j = \sum_{h=1}^j p_h, \quad j = 1, \dots, n.$$

Questa è una definizione del modello: l'intervallo unitario è diviso nei sottointervalli $[u_{j-1}, u_j)$, ciascuno lungo p_j .

Se prima del simbolo g_p l'intervallo corrente è $[A_{p-1}, B_{p-1})$, il nuovo intervallo è definito da

$$\begin{aligned} A_p &= A_{p-1} + (B_{p-1} - A_{p-1})u_{g_p-1}, \\ B_p &= A_{p-1} + (B_{p-1} - A_{p-1})u_{g_p}. \end{aligned}$$

Le due relazioni sono la definizione operativa del rescaling affine. Con $A_0 = 0$ e $B_0 = 1$, iterarle sui simboli del messaggio produce una catena di intervalli annidati.

La lunghezza dell'intervallo dopo p simboli è

$$B_p - A_p = \prod_{r=1}^p p_{g_r}.$$

Dimostrazione. Sottraendo le due definizioni,

$$\begin{aligned} B_p - A_p &= (B_{p-1} - A_{p-1})(u_{g_p} - u_{g_p-1}) \\ &= (B_{p-1} - A_{p-1})p_{g_p}. \end{aligned}$$

Poiché $B_0 - A_0 = 1$, applicando ricorsivamente la relazione si ottiene il prodotto. Questo conclude la dimostrazione.

La decodifica procede nel verso opposto. Dato un numero $C \in [A_p, B_p)$, al primo passo si individua l'unico simbolo j per cui $C \in [u_{j-1}, u_j)$; quindi si ricostruisce l'intervallo relativo al simbolo e si ripete la stessa operazione all'interno di esso. Il decoder deve sapere quando fermarsi: si può trasmettere la lunghezza del messaggio fuori banda oppure aggiungere un simbolo speciale di terminazione, indicato nel seguito con TER.

Quanti bit servono?

Sia

$$s = B_p - A_p$$

la lunghezza dell'intervallo finale. Questa è una definizione. La quantità

$$-\log_2 s$$

misura l'informazione ideale associata alla probabilità del messaggio. Per ottenere una rappresentazione binaria finita dobbiamo scegliere un numero diadico interno all'intervallo.

Se

$$L = \lceil -\log_2 s \rceil + 1,$$

allora esiste almeno un numero diadico con L cifre frazionarie contenuto nell'intervallo finale.

Dimostrazione. Dalla definizione del soffitto, $L - 1 \geq -\log_2 s$ e quindi

$$2^{-L} \leq \frac{s}{2} < s.$$

La griglia dei numeri $k2^{-L}$ ha passo 2^{-L} , strettamente inferiore alla lunghezza s dell'intervallo. Qualunque intervallo di lunghezza maggiore del passo di una griglia uniforme contiene almeno un punto della griglia; nel caso limite in cui un punto coincida con A_p , esso appartiene comunque all'intervallo semiaperto $[A_p, B_p)$. Questo conclude la dimostrazione.

Il numero di bit richiesto differisce quindi da $-\log_2 s$ di un termine limitato, indipendente dalla lunghezza del messaggio.

Per una sorgente senza memoria correttamente modellata,

$$-\log_2 s = -\sum_{r=1}^p \log_2 p_{g_r}.$$

Dimostrazione. Dalla relazione già dimostrata $s = \prod_{r=1}^p p_{g_r}$. Applicando il logaritmo e usando $\log(ab) = \log a + \log b$,

$$\log_2 s = \sum_{r=1}^p \log_2 p_{g_r}.$$

Moltiplicando per -1 si ottiene la relazione. Questo conclude la dimostrazione.

Se o_j indica il numero di occorrenze del simbolo x_j nel blocco, allora

$$-\frac{1}{p} \log_2 s = -\sum_{j=1}^n \frac{o_j}{p} \log_2 p_j.$$

Dimostrazione. Nel membro destro della somma sui simboli, il termine $-\log_2 p_j$ compare esattamente o_j volte. Raggruppando quindi i termini della somma sui p simboli e dividendo per p si ottiene la relazione. Questo conclude la dimostrazione.

Per una sorgente i.i.d., la legge dei grandi numeri implica $o_j/p \rightarrow p_j$ quasi certamente. Di conseguenza

$$-\frac{1}{p} \log_2 s \rightarrow -\sum_j p_j \log_2 p_j = H(X).$$

Poiché anche il sovraccosto di terminazione dell'intervallo resta limitato, il suo contributo per simbolo tende a zero. La codifica aritmetica raggiunge quindi asintoticamente l'entropia del modello corretto, ma non produce necessariamente $H(X)$ bit/simbolo su ogni messaggio finito.

Esempio: la stringa LUNA

Consideriamo l'alfabeto

$$\{A, U, N, L, TER\}$$

con probabilità rispettivamente

$$\{0.40, 0.20, 0.15, 0.15, 0.10\}.$$

Questi insiemi sono dati dell'esempio. Le cumulative sono quindi 0, 0.40, 0.60, 0.75, 0.90, 1.

Codificando la sequenza L, U, N, A, TER si ottengono gli intervalli successivi:

passo	intervallo
<i>L</i>	[0.75, 0.90)
<i>LU</i>	[0.81, 0.84)
<i>LUN</i>	[0.828, 0.8325)
<i>LUNA</i>	[0.828, 0.8298)
<i>LUNA TER</i>	[0.82962, 0.8298)

Dimostrazione. Al primo passo *L* occupa l'intervallo cumulativo [0.75, 0.90). La sua larghezza è 0.15. Il simbolo *U* occupa la frazione [0.40, 0.60) del nuovo intervallo, quindi $0.75 + 0.15 \cdot 0.40 = 0.81$ e $0.75 + 0.15 \cdot 0.60 = 0.84$. Ripetendo il medesimo rescaling con *N*, *A* e *TER* si ottengono gli intervalli della tabella e infine [0.82962, 0.8298). Questo conclude la dimostrazione.

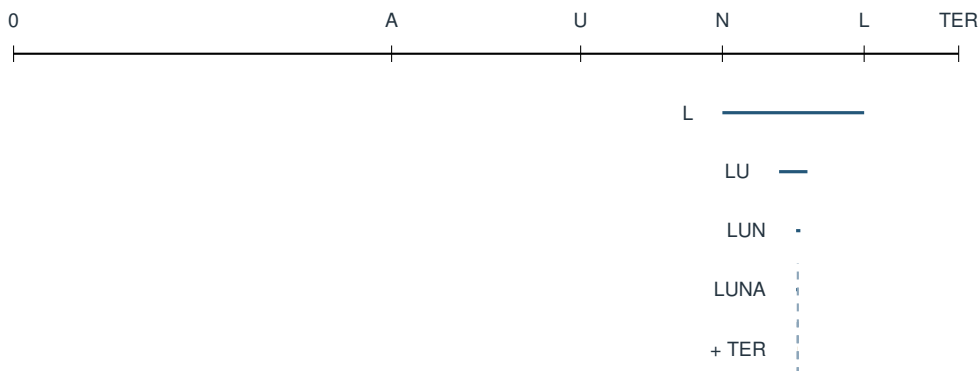


Figura 3.8. Restringimento progressivo dell'intervallo nell'esempio LUNA. La scala orizzontale è comune, quindi gli ultimi intervalli appaiono molto piccoli: è proprio questa contrazione che accumula informazione.

Le implementazioni reali non manipolano numeri reali a precisione illimitata. Usano registri interi, rinormalizzazione dell'intervallo e produzione progressiva dei bit più significativi che non possono più cambiare. Codificatore e decodificatore eseguono operazioni aritmetiche speculari; in questo modo la precisione necessaria resta limitata pur codificando sequenze arbitrariamente lunghe.

3.2.5 Famiglia Lempel–Ziv

I codici di Huffman e la codifica aritmetica sfruttano un modello probabilistico dei simboli. La famiglia Lempel–Ziv affronta il problema da un'altra direzione: costruisce dinamicamente un alfabeto di *stringhe*. Una sequenza già osservata può essere sostituita da un riferimento, evitando di trasmettere nuovamente tutti i suoi simboli.

Le due famiglie classiche sono LZ77 e LZ78. LZ77 cerca le ripetizioni nel testo già decodificato mediante una finestra scorrevole; LZ78 crea un dizionario esplicito di frasi. LZSS

modifica LZ77 scegliendo fra letterale e riferimento in base al costo effettivo; LZW modifica LZ78 precaricando il dizionario con l'alfabeto di base e trasmettendo solo indici.

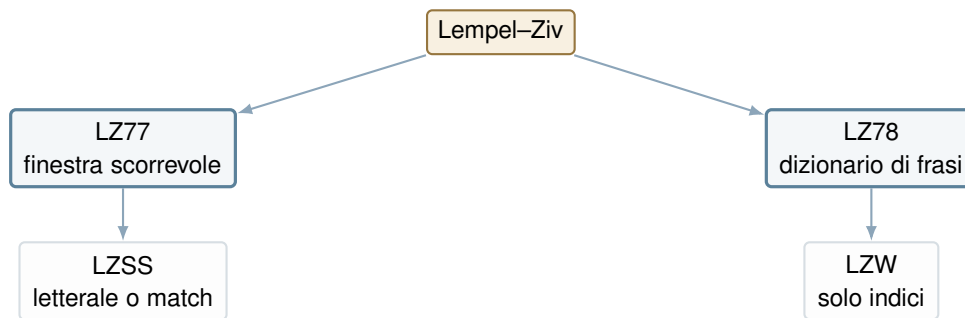


Figura 3.9. Le due linee principali della famiglia Lempel-Ziv e due evoluzioni storicamente importanti.

LZ77: finestra scorrevole

LZ77 mantiene una finestra composta da una zona già decodificata (search buffer) e da una zona ancora da codificare (look-ahead buffer). Il codificatore cerca, nella parte passata, la stringa più lunga che coincide con l'inizio del look-ahead. Un riferimento può essere descritto mediante distanza all'indietro B e lunghezza L . Nella forma didattica originale si aggiunge anche il primo simbolo S che segue il match e si emette la terna

$$(B, L, S).$$

Questa è una definizione del formato dell'esempio, non una formula da dimostrare.

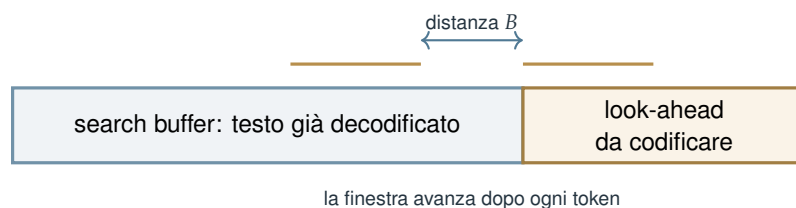


Figura 3.10. Principio di LZ77. Il match nel passato sostituisce una sequenza futura con una distanza e una lunghezza.

Una realizzazione pratica limita sia la dimensione della finestra sia la lunghezza massima del match. Ciò rende limitati i campi B e L e permette implementazioni con strutture di ricerca efficienti. Alcuni formati consentono match sovrapposti: durante la copia, i byte appena ricostruiti possono diventare a loro volta sorgente della copia. È il meccanismo che permette di rappresentare in modo molto compatto run lunghi anche con una finestra finita.

Esempio LZ77. Consideriamo la stringa AABCBBABC. Con la convenzione della terna (B, L, S) descritta sopra si ottiene:

Il rapporto di compressione effettivo non può essere dedotto dal solo numero di token: dipende da quanti bit vengono assegnati ai campi B , L e S , dalla dimensione della finestra e dalla distribuzione dei match.

LZSS: riferimenti solo quando convengono

LZSS osserva che un riferimento corto può occupare più bit dei letterali che sostituisce. L'uscita contiene quindi due tipi di token: letterali e riferimenti (B, L) , distinti da un flag o da una sintassi equivalente.

Posizione	testo considerato	interpretazione	output
1	A	nessun match	(0, 0, A)
2–3	AB	A coincide con la posizione 1, poi B	(1, 1, B)
4	C	prima occorrenza di C	(0, 0, C)
5–6	BB	B coincide con la posizione 3, poi B	(2, 1, B)
7–9	ABC	match con le posizioni 2–4, fine input	(5, 3, –)

Tabella 3.3. Parsing LZ77 della stringa AABCBBABC secondo la convenzione didattica (B, L, S) .

Supponiamo che un simbolo letterale costi b bit e che un riferimento (B, L) costi c_P bit, senza contare un flag comune ai due tipi. Un match di lunghezza L conviene se

$$c_P < bL.$$

Dimostrazione. Trasmettere i L simboli come letterali costa bL bit; il riferimento costa c_P bit. Il riferimento riduce la lunghezza esattamente quando il suo costo è minore, cioè quando $c_P < bL$. Questo conclude la dimostrazione.

Se definiamo la soglia equivalente in simboli

$$L_P = \frac{c_P}{b},$$

la condizione diventa $L > L_P$.

Dimostrazione. Poiché $b > 0$, dividere $c_P < bL$ per b conserva il verso della disuguaglianza e produce $c_P/b < L$. Per definizione $L_P = c_P/b$. Questo conclude la dimostrazione.

Questo corregge un possibile equivoco: i match *più lunghi* della soglia sono quelli da rappresentare con il puntatore, non quelli più corti.

Esempio LZSS. Usiamo ancora AABCBBABC e assumiamo una soglia equivalente $L_P = 2$. I match di un solo simbolo vengono quindi inviati come letterali; il match finale ABC, lungo 3, viene sostituito da un riferimento:

Posizione	match massimo	scelta	output
1	–	letterale	A
2	A	troppo corto	A
3	–	letterale	B
4	–	letterale	C
5	B	troppo corto	B
6	B	troppo corto	B
7–9	ABC	match lungo 3	(5, 3)

Tabella 3.4. Esempio LZSS: la soglia evita riferimenti che costerebbero più dei letterali sostituiti.

Questa idea è ancora centrale in algoritmi di compressione moderni come DEFLATE: prima si producono letterali e coppie lunghezza/distanza, poi tali simboli vengono a loro volta codificati entropicamente.

LZ78: dizionario esplicito

LZ78 non cerca il match direttamente nel testo precedente. Costruisce un dizionario di frasi. La posizione 0 rappresenta convenzionalmente la stringa vuota. Si mantiene un prefisso P già presente nel dizionario e si legge il prossimo simbolo C . Se $P + C$ è già presente, il prefisso

viene esteso; altrimenti si emette la coppia (W, C) , dove W è l'indice di P , e si inserisce $P + C$ come nuova voce.

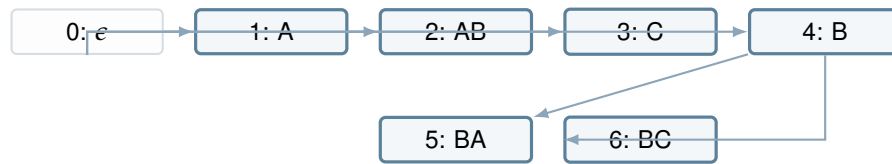


Figura 3.11. Dizionario LZ78 dell'esempio: ogni nuova frase è ottenuta estendendo una frase già nota con un simbolo.

Per AABCBBABC il parsing è:

Posizioni	frase aggiunta	output (W, C)
1	A	$(0, A)$
2-3	AB	$(1, B)$
4	C	$(0, C)$
5	B	$(0, B)$
6-7	BA	$(4, A)$
8-9	BC	$(4, C)$

Tabella 3.5. Codifica LZ78 della stringa AABCBBABC.

Il dizionario risultante contiene A, AB, C, B, BA e BC, come mostrato nella figura 3.11. Encoder e decoder costruiscono il dizionario nello stesso ordine a partire dalle coppie ricevute.

Se il dizionario contiene M possibili indici, un campo a lunghezza fissa capace di rappresentarli richiede

$$b_I = \lceil \log_2 M \rceil$$

bit.

Dimostrazione. Con b bit esistono 2^b configurazioni. Serve quindi il minimo intero b tale che $2^b \geq M$. Applicando il logaritmo, $b \geq \log_2 M$; il minimo intero che soddisfa la relazione è il soffitto $\lceil \log_2 M \rceil$. Questo conclude la dimostrazione.

In implementazioni reali la larghezza dell'indice può crescere durante la codifica oppure il dizionario può essere azzerato quando raggiunge una dimensione prefissata.

LZW: dizionario inizializzato con l'alfabeto

LZW parte da un dizionario che contiene già tutti i simboli elementari. Per questo l'uscita non deve trasmettere il nuovo carattere insieme all'indice: quando $P + C$ non è presente, il codificatore emette l'indice di P , inserisce $P + C$ e riparte con $P = C$. La stessa regola permette al decoder di costruire il nuovo dizionario usando soltanto la sequenza degli indici.

Per l'esempio con alfabeto $\{A, B, C\}$ il dizionario iniziale contiene A=1, B=2 e C=3. Codificando AABCBBABC si producono gli indici

$$1, 1, 2, 3, 2, 2, 5, 3.$$

Verifica dell'esempio. La prima A viene emessa quando la coppia AA, non ancora presente, viene aggiunta come voce 4. Il secondo indice 1 viene emesso quando AB viene aggiunta come voce 5. Seguono B=2 con aggiunta di BC, C=3 con aggiunta di CB, B=2 con aggiunta di BB e B=2 con aggiunta di BA. A questo punto la sequenza successiva inizia con AB, già

Indice	stringa	stato
1	A	iniziale
2	B	iniziale
3	C	iniziale
4	AA	aggiunta
5	AB	aggiunta
6	BC	aggiunta
7	CB	aggiunta
8	BB	aggiunta
9	BA	aggiunta
10	ABC	aggiunta

Tabella 3.6. Dizionario LZW generato nell'esempio.

presente come voce 5, quindi il prefisso si estende e viene emesso 5 quando ABC deve essere aggiunta. Infine resta C=3. La sequenza degli indici è dunque quella indicata.

LZW è diventato particolarmente noto attraverso GIF. DEFLATE segue invece la linea LZ77/LZSS e combina riferimenti a stringhe con codici di Huffman. Il principio generale resta lo stesso: la prima fase trasforma ripetizioni lunghe in simboli di ordine superiore, la seconda assegna codici brevi agli eventi più probabili.

3.3 Compressione delle immagini

Un'immagine digitale possiede più tipi di ridondanza. Pixel vicini sono spesso correlati; alcune configurazioni di colore sono più probabili di altre; il sistema visivo non è ugualmente sensibile a tutte le frequenze spaziali e a tutte le componenti cromatiche. Un formato lossless può sfruttare soltanto ridondanze reversibili; un formato lossy può anche modificare i campioni, purché la distorsione resti coerente con l'obiettivo applicativo.

È utile distinguere tre passaggi concettuali:

- **modellazione o predizione**, che rende i dati più prevedibili senza necessariamente perdere informazione;
- **quantizzazione**, presente nei codec lossy, che riduce il numero di stati rappresentati e introduce distorsione;
- **codifica entropica**, che rappresenta in modo compatto i simboli risultanti senza ulteriore perdita.

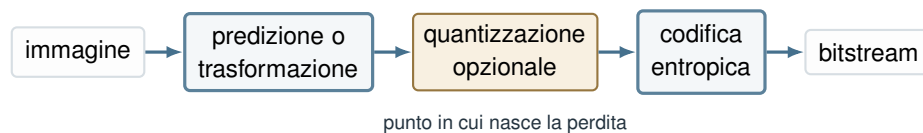


Figura 3.12. Schema generale della compressione di immagini. Nei formati lossless la quantizzazione distruttiva è assente; nei formati lossy è il principale punto di controllo del compromesso rate-distortion.

3.3.1 Misure di distorsione e qualità

Per confrontare un'immagine originale X con una ricostruita C di dimensione $W \times H$ si definisce l'errore quadratico medio

$$MSE = \frac{1}{WH} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} [X(i,j) - C(i,j)]^2.$$

Questa è una definizione di metrica. Per immagini multicomponente si può calcolare la metrica separatamente sui piani oppure estendere la media a tutti i campioni considerati. È quindi necessario dichiarare la convenzione usata quando si confrontano risultati pubblicati.

Il Peak Signal-to-Noise Ratio è definito da

$$PSNR = 10 \log_{10} \frac{MAX^2}{MSE},$$

dove MAX è il massimo valore rappresentabile del campione. Per un canale unsigned a b bit,

$$MAX = 2^b - 1.$$

Dimostrazione. Un intero unsigned rappresentato con b bit possiede 2^b configurazioni, corrispondenti ai valori da 0 a $2^b - 1$. Il massimo è quindi $2^b - 1$. Questo conclude la dimostrazione.

Indicando con

$$RMSE = \sqrt{MSE}$$

la radice dell'errore quadratico medio, il PSNR può essere scritto anche come

$$PSNR = 20 \log_{10} \frac{MAX}{RMSE}.$$

Dimostrazione. Dalla definizione $MSE = RMSE^2$. Sostituendo,

$$\begin{aligned} PSNR &= 10 \log_{10} \frac{MAX^2}{RMSE^2} \\ &= 10 \log_{10} \left(\frac{MAX}{RMSE} \right)^2 \\ &= 20 \log_{10} \frac{MAX}{RMSE}. \end{aligned}$$

Questo conclude la dimostrazione.

Se $MSE = 0$, le due immagini coincidono campione per campione e il PSNR è formalmente infinito. Un PSNR elevato indica quindi piccola energia dell'errore, ma non garantisce che l'errore sia percettivamente poco visibile.

Per questo le valutazioni oggettive vengono spesso affiancate da metriche percettive o da test soggettivi. Il Mean Opinion Score (MOS) riassume valutazioni umane su una scala ordinata, spesso da 1 a 5. Il valore medio è semplice da calcolare, ma un esperimento credibile deve specificare ambiente di visione, display, distanza, contenuti, procedura di randomizzazione e popolazione dei valutatori. Il numero MOS non è quindi una proprietà assoluta del codec indipendente dal protocollo sperimentale.

Metriche strutturali come SSIM confrontano luminanza, contrasto e struttura locale invece di ridurre tutto alla sola energia dell'errore. MS-SSIM ripete il confronto a più scale, cercando di rappresentare il fatto che la percezione di struttura e dettaglio cambia con la

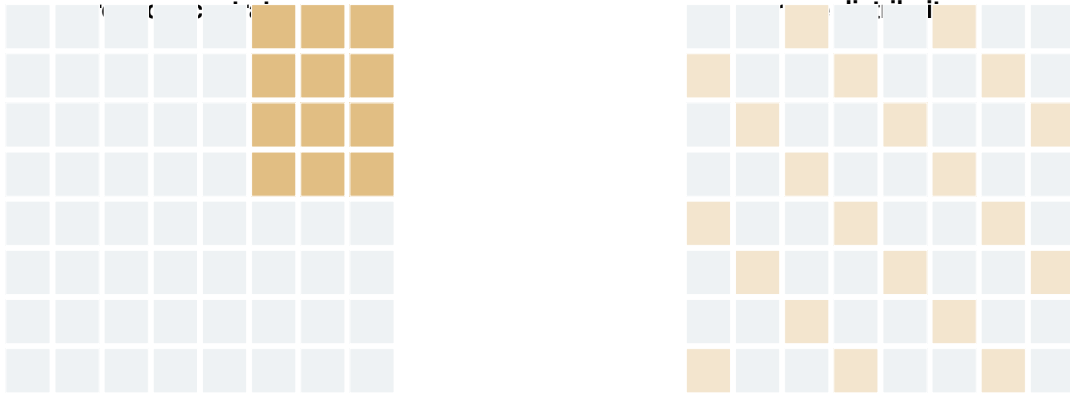


Figura 3.13. Due distribuzioni spaziali dell'errore possono avere MSE simile ma impatto visivo molto differente. Le metriche medie non descrivono da sole la struttura dell'artefatto.

risoluzione spaziale. Sono metriche molto utili, ma restano modelli: una stessa variazione di SSIM non corrisponde necessariamente alla stessa differenza percettiva per contenuti e artefatti differenti.

Per il video moderno vengono inoltre usati modelli che combinano più feature. VMAF, trattato nel capitolo successivo nel contesto del confronto fra codec video, è un esempio importante di metrica ottenuta combinando indicatori oggettivi e dati provenienti da valutazioni soggettive. Anche in questo caso versione del modello, preprocessing e condizioni di prova devono essere dichiarati insieme al punteggio.

3.3.2 Curve rate–distortion

Un codec lossy non produce un unico punto di funzionamento. Variando la quantizzazione o altri parametri si ottiene una famiglia di coppie bitrate–qualità. Il confronto corretto fra codec usa quindi curve rate–distortion e non una singola percentuale di compressione.

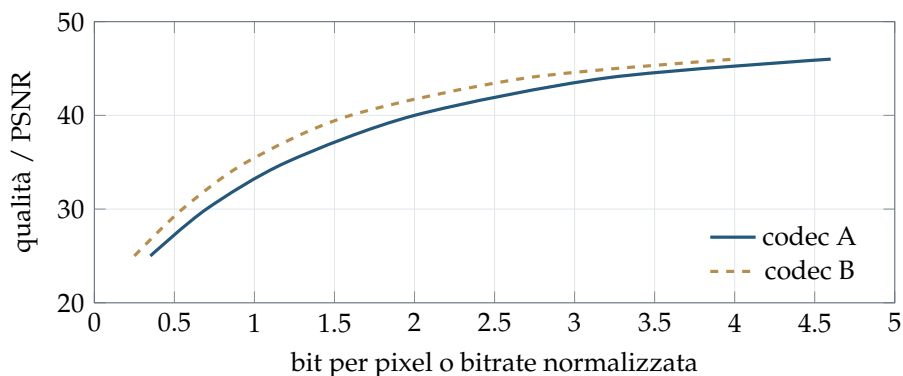


Figura 3.14. Esempio concettuale di curve rate–distortion. A parità di qualità il codec B richiede meno bit; a parità di bitrate produce qualità maggiore. I valori sono illustrativi.

3.3.3 GIF

Graphics Interchange Format (GIF) è nato nel 1987 ed è rimasto a lungo un formato fondamentale del Web. Il principio è separare i colori dell'immagine dagli indici che li richiamano: una Color Table contiene al massimo 256 colori RGB e ogni pixel memorizza un indice nella tavolozza. La sequenza degli indici viene poi compressa con LZW.

La compressione LZW è lossless rispetto agli indici. L'eventuale perdita nasce prima,

quando un'immagine true-color contenente più colori di quelli disponibili deve essere ridotta a una palette. Se invece tutti i colori dell'immagine sono già rappresentabili nella tavolozza, la codifica può essere completamente lossless.

GIF può contenere una Global Color Table e, per ciascuna image description, una Local Color Table. Non esiste però una suddivisione obbligatoria dell'immagine in blocchi quadrati: ogni image descriptor può definire una regione rettangolare, meccanismo sfruttato anche nelle animazioni.

Costruzione della palette

Lo standard non impone l'algoritmo con cui scegliere la palette. Una strategia ingenua consiste nel prendere i colori più frequenti, ma può sprecare molte entry per colori quasi indistinguibili fra loro. Un approccio migliore considera contemporaneamente frequenza e distanza nello spazio colore.

Il median cut è un esempio classico. Si considerano tutti i colori come punti in uno spazio tridimensionale; si trova il parallelepipedo che li contiene, lo si divide lungo la dimensione di maggiore estensione in modo che le due parti contengano popolazioni comparabili e si ripete il procedimento. Ogni regione finale viene sostituita da un colore rappresentativo.

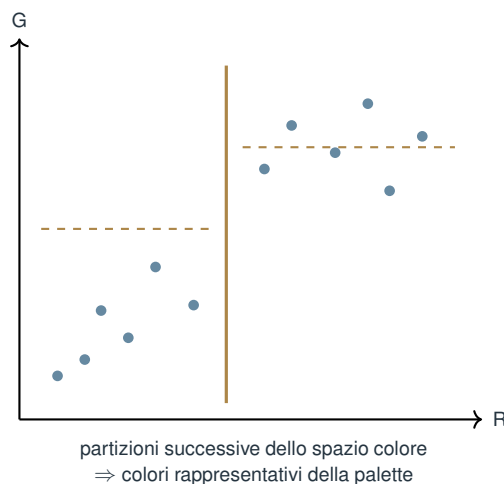


Figura 3.15. Principio bidimensionale semplificato del median cut. Nell'algoritmo reale i colori vivono in uno spazio tridimensionale e i punti possono essere pesati con le occorrenze.

Un altro strumento è il dithering: invece di rappresentare localmente ogni pixel con il solo colore più vicino, si distribuisce spazialmente l'errore fra pixel vicini. La risoluzione cromatica locale viene in parte scambiata con una trama ad alta frequenza che l'occhio integra a distanza. Il risultato può ridurre banding e falsi contorni, ma può anche peggiorare la compressione LZW perché introduce variazioni più rapide negli indici.

Struttura del file

La struttura GIF può essere letta come una sequenza di blocchi:

Il Logical Screen Descriptor contiene le dimensioni logiche e i flag relativi alla tavolozza globale. Una Color Table contiene triple RGB da 8 bit per componente; con g bit di indice una tavolozza completa contiene

$$N_c = 2^g$$

entry.

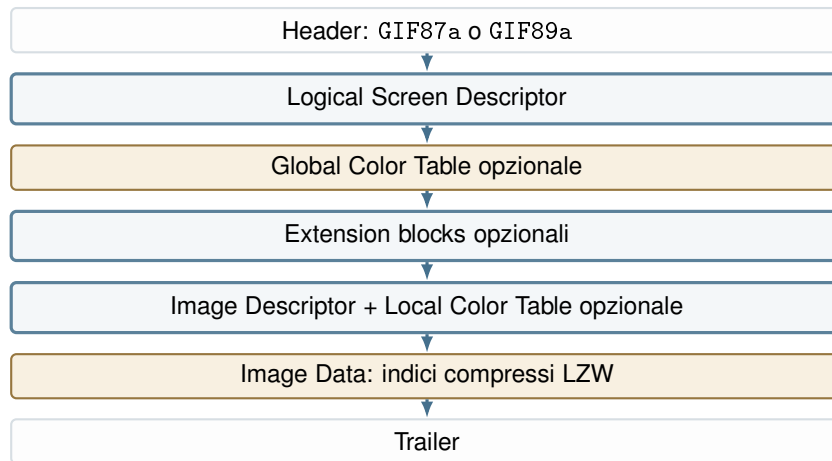


Figura 3.16. Struttura concettuale di un file GIF. Image Descriptor e Image Data possono ripetersi, permettendo anche animazioni e aggiornamenti di regioni.

Dimostrazione. Un indice di g bit possiede 2^g configurazioni distinte, ciascuna delle quali può selezionare un'entry. Questo conclude la dimostrazione.

GIF87a definisce il formato di base; GIF89a aggiunge estensioni fra cui Graphics Control Extension, trasparenza a un singolo indice e tempi di visualizzazione utili per le animazioni. L'interlacciamento GIF non è quello televisivo: cambia l'ordine con cui vengono trasmesse le righe in quattro passaggi per mostrare rapidamente una versione grossolana dell'intera immagine durante un download lento.

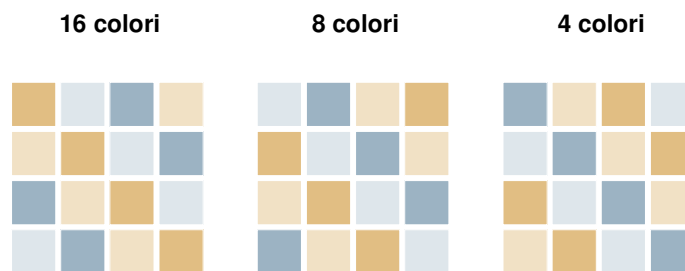


Figura 3.17. Illustrazione editoriale della riduzione di palette. Diminuendo i colori disponibili aumentano quantizzazione cromatica e banding; l'effetto reale dipende dal contenuto e dall'algoritmo di palette.

Non esiste un rapporto universale fra numero di colori e PSNR. I risultati dipendono dalla distribuzione cromatica dell'immagine e dalla palette scelta. Per questo i valori numerici ricavati da una singola fotografia non devono essere generalizzati come proprietà del formato.

3.3.4 PNG

Portable Network Graphics (PNG) è un formato lossless progettato per immagini raster. Può rappresentare palette, livelli di grigio, true-color e canale alpha. A differenza di GIF, il true-color non deve essere ridotto a 256 colori; a differenza di JPEG, nessuna quantizzazione distruttiva è necessaria.

La compressione PNG combina due stadi. Prima di DEFLATE ogni scanline può essere trasformata da uno dei cinque filtri reversibili: None, Sub, Up, Average e Paeth. Sia x il byte corrente, a quello a sinistra, b quello sopra e c quello sopra-sinistra. Nel filtro Sub, per esempio, si memorizza il residuo

$$r = x - a \pmod{256}.$$

Questa è una definizione del filtro. Il decoder ricostruisce

$$x = r + a \pmod{256}.$$

Dimostrazione. Nell'aritmetica modulo 256, sostituendo $r = x - a$ nell'inversa si ottiene $r + a = (x - a) + a = x$. Questo conclude la dimostrazione della reversibilità del filtro Sub. Gli altri filtri seguono la stessa struttura: il predittore è calcolato da campioni già noti al decoder e viene sommato al residuo modulo 256.

Il filtro Paeth costruisce invece una previsione scegliendo fra a , b e c quello più vicino alla stima lineare $p = a + b - c$. L'obiettivo non è migliorare direttamente l'immagine ma produrre residui piccoli e ripetitivi che DEFLATE comprime meglio.



Figura 3.18. Pipeline di compressione PNG. Il filtraggio non elimina informazione: cambia la statistica dei byte prima di DEFLATE.

Il file è organizzato in chunk. Ogni chunk contiene una lunghezza, un tipo di quattro byte, i dati e un CRC. IHDR descrive l'immagine, IDAT trasporta lo stream compresso, IEND conclude il file; PLTE contiene la palette quando necessaria. Chunk ancillary possono trasportare gamma, profili colore, testo e altre informazioni.

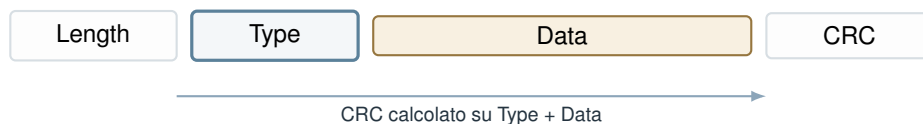


Figura 3.19. Struttura generale di un chunk PNG. La separazione in chunk rende il formato estensibile e permette di distinguere dati critici e metadati opzionali.

PNG è particolarmente efficace per schermate, diagrammi, testo rasterizzato, grafica con alpha e immagini in cui ogni campione deve essere preservato. Per fotografie naturali, l'obbligo di ricostruzione esatta impedisce di ottenere i rapporti tipici dei codec lossy a qualità visivamente equivalente.

3.3.5 JPEG baseline

JPEG è il riferimento storico della compressione fotografica lossy. La sua importanza didattica va oltre il formato stesso: molte idee che ricorreranno nei codec video sono già presenti qui – conversione di colore, riduzione della cromaticanza, trasformazione, quantizzazione, serializzazione, run-length e codifica entropica.

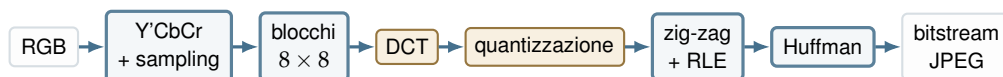
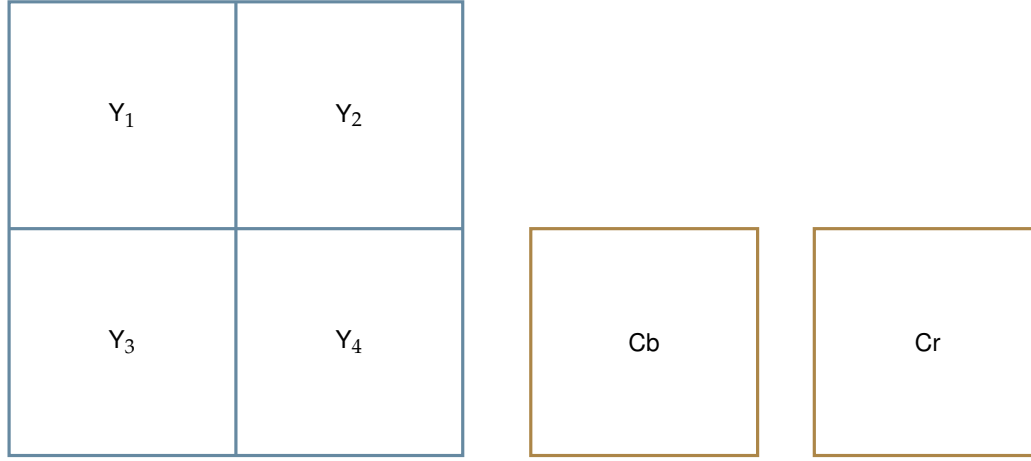


Figura 3.20. Pipeline concettuale del JPEG DCT-based. La quantizzazione è il passaggio irreversibile principale; gli stadi successivi sono lossless rispetto ai coefficienti quantizzati.

Preparazione, componenti e MCU

L'immagine viene normalmente convertita da RGB a Y'CbCr. La separazione fra luma e cromaticanze consente di sottocampionare Cb e Cr, per esempio in 4:4:4, 4:2:2 o 4:2:0. JPEG non impone un unico rapporto: i fattori di campionamento delle componenti sono descritti nel frame header.

Ogni componente viene suddivisa in blocchi 8×8 . Quando le componenti hanno fattori di campionamento differenti, i blocchi che coprono la stessa regione dell'immagine sono raggruppati in Minimum Coded Unit (MCU). Nel comune caso 4:2:0, un MCU può contenere quattro blocchi Y e un blocco per ciascuna crominanza.



esempio concettuale di MCU 4:2:0:
4 blocchi Y + 1 Cb + 1 Cr

Figura 3.21. Composizione di un MCU nel caso 4:2:0. Le dimensioni grafiche sono schematiche: ogni rettangolo etichettato rappresenta un blocco 8×8 della rispettiva componente.

Per campioni a 8 bit, prima della DCT si effettua tipicamente il level shift

$$y_{i,j} = x_{i,j} - 128,$$

che trasforma $[0, 255]$ in $[-128, 127]$. Questa è una definizione dello stadio di preparazione e centra numericamente il segnale intorno a zero.

DCT bidimensionale

Per un blocco 8×8 definiamo

$$c_u(i) = \frac{\Lambda(u)}{2} \cos \left[\frac{\pi}{8} \left(i + \frac{1}{2} \right) u \right],$$

con

$$\Lambda(0) = \frac{1}{\sqrt{2}}, \quad \Lambda(u) = 1 \quad \text{per } u > 0.$$

Queste sono definizioni delle funzioni di base. La DCT-II 2D è

$$Y_{u,v} = \sum_{i=0}^7 \sum_{j=0}^7 y_{i,j} c_u(i) c_v(j), \quad 0 \leq u, v \leq 7.$$

Il coefficiente $Y_{0,0}$ è la componente DC e dipende dal valore medio del blocco; gli altri 63 coefficienti sono detti AC. Aumentando u cresce la frequenza orizzontale della base; aumentando v cresce quella verticale.

Le funzioni di base sono ortonormali:

$$\sum_{i=0}^7 c_u(i) c_v(i) = \delta_{u,v}.$$

Dimostrazione. Usiamo

$$2 \cos a \cos b = \cos(a - b) + \cos(a + b)$$

e la somma finita

$$\sum_{i=0}^7 \cos \left[\left(i + \frac{1}{2} \right) \theta \right] = \frac{\sin(8\theta)}{2 \sin(\theta/2)}$$

quando il denominatore è non nullo, con il valore limite 8 per $\theta = 0$. Per $u \neq v$ le frequenze $u - v$ e $u + v$ portano a numeratori $\sin[\pi(u - v)]$ e $\sin[\pi(u + v)]$, entrambi nulli; la somma è quindi zero. Per $u = v > 0$ la somma non normalizzata dei prodotti vale 4 e il fattore $(1/2)^2$ la rende 1. Per $u = v = 0$ la somma vale 8 e il fattore $(1/(2\sqrt{2}))^2 = 1/8$ la rende ancora 1. Questo conclude la dimostrazione.

Dall'ortogonalità segue l'inversa

$$y_{i,j} = \sum_{u=0}^7 \sum_{v=0}^7 Y_{u,v} c_u(i) c_v(j).$$

Dimostrazione. Sostituendo la trasformata diretta nell'inversa,

$$\sum_{u,v} \sum_{h,k} y_{h,k} c_u(h) c_v(k) c_u(i) c_v(j).$$

Scambiando l'ordine delle somme, le due ortogonalità producono $\delta_{h,i}$ e $\delta_{k,j}$; tutti i termini si annullano tranne $y_{i,j}$. Questo conclude la dimostrazione.

La trasformazione è separabile. Indicando con A la matrice $A_{u,i} = c_u(i)$,

$$Y = A, y, A^T.$$

Dimostrazione. L'elemento (u, v) del prodotto matriciale è

$$(AyA^T)_{u,v} = \sum_i \sum_j A_{u,i} y_{i,j} A_{j,v}^T = \sum_i \sum_j c_u(i) y_{i,j} c_v(j),$$

che coincide con la definizione della DCT 2D. Questo conclude la dimostrazione. Operativamente si possono quindi trasformare prima le righe e poi le colonne, invece di implementare direttamente ogni doppia somma.

La DCT ideale è invertibile e non introduce perdita. La perdita JPEG nasce principalmente dalla quantizzazione; eventuali arrotondamenti delle implementazioni possono aggiungere differenze numeriche molto più piccole.

Quantizzazione

Per ogni frequenza JPEG usa un passo positivo $Q_{u,v}$ e definisce il coefficiente quantizzato

$$O_{u,v} = \text{round} \left(\frac{Y_{u,v}}{Q_{u,v}} \right).$$

In decodifica si ricostruisce

$$\hat{Y}_{u,v} = O_{u,v} Q_{u,v}.$$

Le due relazioni definiscono la quantizzazione scalare del coefficiente. Con arrotondamento al più vicino, l'errore soddisfa

$$|Y_{u,v} - \hat{Y}_{u,v}| \leq \frac{Q_{u,v}}{2},$$

a eccezione della convenzione usata esattamente sui punti a metà.

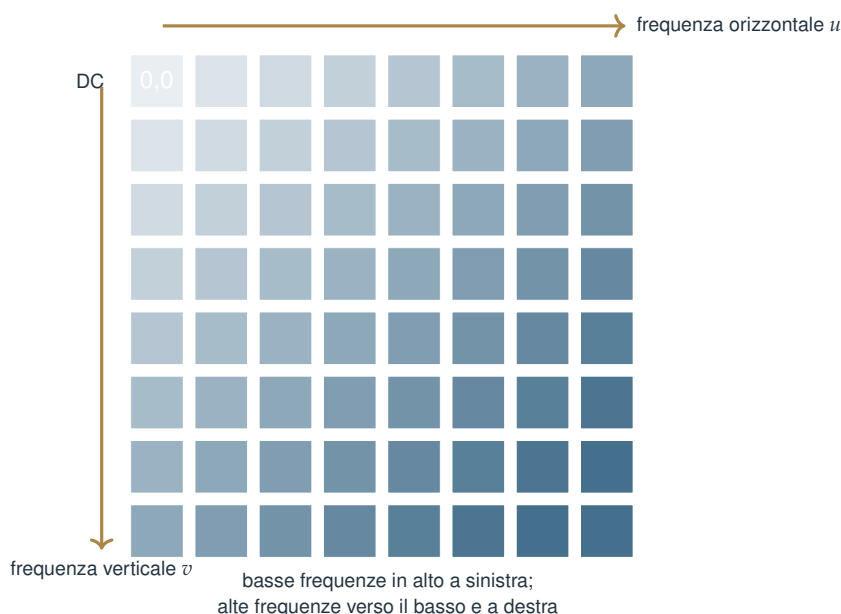


Figura 3.22. Organizzazione delle 64 componenti DCT di un blocco JPEG. La figura mostra la disposizione in frequenza, non il valore dei coefficienti di una particolare immagine.

Dimostrazione. Il numero $Y_{u,v}/Q_{u,v}$ viene sostituito dall'intero più vicino. La distanza dall'intero scelto è quindi al massimo $1/2$. Moltiplicando per $Q_{u,v} > 0$ si ottiene il limite sull'errore nel dominio DCT. Questo conclude la dimostrazione.

Passi piccoli conservano meglio il coefficiente ma producono valori quantizzati più numerosi; passi grandi aumentano il numero di zeri e facilitano RLE e Huffman. Le tabelle possono attribuire passi differenti alle frequenze in funzione della sensibilità visiva e delle statistiche dell'immagine.

Le tabelle seguenti sono le note tabelle di esempio storicamente associate al JPEG per luminanza e crominanza. Sono valori di riferimento molto diffusi, non l'unica scelta ammessa da un file JPEG.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	36	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Tabella 3.7. Tabella di quantizzazione di luminanza comunemente usata come riferimento JPEG.

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Tabella 3.8. Tabella di quantizzazione di crominanza comunemente usata come riferimento JPEG.

Il parametro “quality” esposto da molti programmi non fa parte della sintassi normativa

JPEG come grandezza universale. È una scelta dell'encoder che scala o modifica le tabelle di quantizzazione secondo una propria politica. Due encoder impostati a "quality 80" possono quindi produrre tabelle e risultati differenti.

DC, AC e scansione a zig-zag

Il coefficiente DC è fortemente correlato fra blocchi adiacenti. Se D_j è il DC del blocco j , si definisce la differenza

$$M_j = D_j - D_{j-1}, \quad D_0 = 0.$$

Questa è la definizione della predizione differenziale usata per il DC.

Per un valore non nullo M , il numero di bit di ampiezza della sua categoria è

$$L = \lfloor \log_2 |M| \rfloor + 1.$$

Per $M = 0$ si usa $L = 0$.

Dimostrazione. Se $M \neq 0$ e $L = \lfloor \log_2 |M| \rfloor + 1$, dalla definizione di parte intera segue

$$L - 1 \leq \log_2 |M| < L.$$

Esponenziando in base 2,

$$2^{L-1} \leq |M| < 2^L.$$

Sono quindi necessari esattamente L bit per rappresentare l'ampiezza nella categoria. Questo conclude la dimostrazione.

Gli AC vengono serializzati dal basso verso l'alto in frequenza con l'ordine zig-zag. Una numerazione utile per studiare il percorso è:

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figura 3.23. Ordine zig-zag JPEG numerato da 0 a 63. L'ordinamento visita prima le basse frequenze e tende a spostare le lunghe code di zeri verso la fine della sequenza.

Dopo la quantizzazione molte alte frequenze valgono zero. Il JPEG rappresenta gli AC mediante coppie RUN/SIZE: RUN indica quanti zeri precedono il prossimo coefficiente non nullo e SIZE la categoria della sua ampiezza. EOB segnala che tutti i coefficienti rimanenti del blocco sono zero; ZRL rappresenta una corsa di 16 zeri quando necessario.

Per esempio la sequenza AC

$$3, 4, 0, 0, 2, 0, 0, 0, 1, 0, 0, \dots, 0$$

produce concettualmente gli eventi

$$(0, 3), (0, 4), (2, 2), (3, 1), \text{EOB}.$$

Le coppie qui mostrano il run e il valore per chiarezza didattica; nel bitstream reale l'evento Huffman codifica RUN/SIZE e i bit successivi codificano l'ampiezza.

Codifica di Huffman nel JPEG

JPEG baseline usa tabelle di Huffman separate per DC e AC e, in genere, per luminanza e crominanza. Esistono tabelle di default molto note, ma il file può trasportare tabelle diverse mediante segmenti DHT. Non è quindi corretto considerare un unico albero “predefinito” come parte immutabile del formato.

Per la tabella DC di luminanza di default, le categorie da 0 a 11 sono associate, per esempio, alle parole seguenti:

SIZE	0	1	2	3	4	5	6	7	8	9	10	11
code	00	010	011	100	101	110	1110	11110	111110	1111110	11111110	111111110

Tabella 3.9. Tabella Huffman DC luminanza di default del JPEG baseline. Al codice segue un numero di bit di ampiezza pari a SIZE.

Le tabelle AC sono molto più estese perché il simbolo Huffman combina RUN e SIZE. Il principio è però lo stesso: eventi comuni come EOB o piccoli coefficienti dopo run brevi ricevono parole corte; eventi rari ricevono parole più lunghe. Una tabella completa non aggiunge un nuovo principio algoritmico rispetto a Huffman già studiato: ciò che conta è capire che la statistica è ora quella degli eventi RUN/SIZE prodotti dalla quantizzazione e dallo zig-zag.

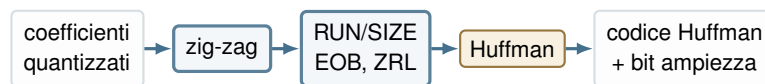


Figura 3.24. Catena di serializzazione e codifica entropica dei coefficienti AC JPEG.

Marker e struttura del file

Un JPEG interchange file è organizzato mediante marker. I byte FFD8 (SOI) iniziano l’immagine e FFD9 (EOI) la terminano. Fra essi possono comparire segmenti APP per metadati, DQT per le tabelle di quantizzazione, SOF per la struttura del frame, DHT per le tabelle Huffman, DRI per l’intervallo di restart e SOS per l’inizio di uno scan.

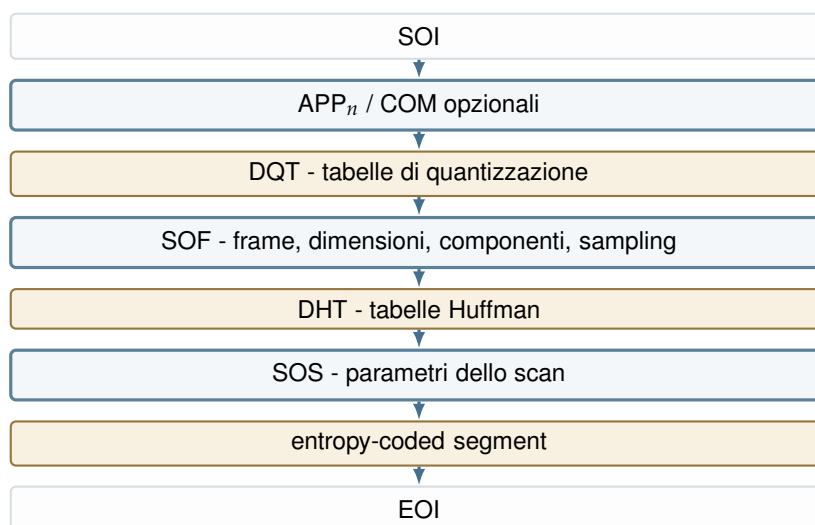


Figura 3.25. Struttura concettuale di un JPEG baseline. L’ordine reale può includere più tabelle, più scan, restart marker e segmenti applicativi.

I restart marker possono essere inseriti periodicamente nel flusso entropico. Forniscono

punti di risincronizzazione e limitano la propagazione di alcuni errori, oltre a facilitare certe forme di elaborazione parallela.

Modalità JPEG

La famiglia JPEG originale comprende più processi. Il baseline sequential DCT è la modalità più interoperabile e codifica i coefficienti di ogni componente in un singolo scan sequenziale. Il progressive DCT distribuisce l'informazione su più scan, per esempio inviando prima bande di frequenza basse e poi alte oppure raffinando successivamente la precisione dei coefficienti. Il risultato finale può essere equivalente a quello di una codifica sequenziale con gli stessi coefficienti, ma una versione parziale dell'immagine diventa visibile prima del completamento del file.

Il JPEG originale definisce anche processi lossless basati su predizione, che non usano la DCT, e modalità gerarchiche multirisoluzione. Queste opzioni sono storicamente e concettualmente importanti ma molto meno diffuse del baseline DCT e del progressive JPEG nelle applicazioni ordinarie.

Artefatti e comportamento al variare della quantizzazione

Aumentare i passi di quantizzazione tende a eliminare coefficienti AC e a rendere più grossolana la ricostruzione. Gli artefatti tipici hanno una struttura coerente con l'architettura del codec:

- **blocking**, perché blocchi 8×8 vengono quantizzati indipendentemente;
- **ringing**, vicino a bordi netti quando le alte frequenze vengono fortemente attenuate;
- **perdita di texture**, quando coefficienti piccoli ma visivamente importanti vengono portati a zero;
- **color bleeding**, accentuato dal sottocampionamento della cromaticità e da quantizzazione forte.



Figura 3.26. Illustrazione concettuale dell'effetto della quantizzazione nel dominio DCT: aumentando il passo, una regione crescente di coefficienti tende a diventare zero. Non è il risultato di una fotografia specifica.

Non esiste una corrispondenza universale fra un parametro “quality”, rapporto di compressione e PSNR. L'immagine, le tabelle, il sottocampionamento e le scelte dell'encoder modificano la curva rate-distortion. Un confronto scientifico deve quindi specificare encoder, parametri, contenuti e metrica.

3.3.6 JPEG 2000 e HTJ2K

JPEG 2000 affronta la compressione fotografica con un'architettura molto diversa dal JPEG DCT-based. L'immagine viene trasformata mediante wavelet su regioni estese, non mediante

blocchi indipendenti 8×8 . Questo riduce la tendenza al blocking e produce una rappresentazione naturalmente multirisoluzione.

Una decomposizione wavelet 2D separa, a ogni livello, una banda di approssimazione LL e tre bande di dettaglio LH, HL e HH. La banda LL può essere nuovamente decomposta, costruendo una piramide di risoluzioni.

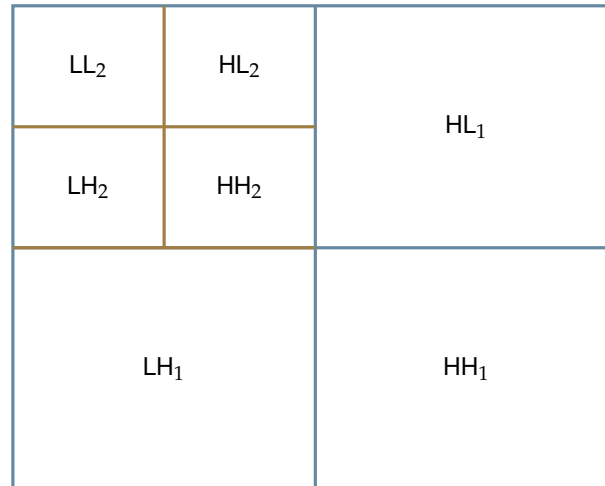


Figura 3.27. Decomposizione wavelet 2D a due livelli. La banda LL conserva l'approssimazione a bassa risoluzione e viene decomposta nuovamente; le altre bande descrivono dettagli orientati.

JPEG 2000 supporta sia trasformazioni reversibili per la modalità lossless sia trasformazioni irreversibili per la modalità lossy. Dopo la trasformata, i coefficienti vengono organizzati in code-block e codificati bit-plane per bit-plane. La struttura del codestream permette di organizzare i dati secondo progressioni differenti: per qualità, risoluzione, componente o posizione spaziale.

Questa proprietà ha una conseguenza importante: da un unico codestream si possono estrarre porzioni con qualità o risoluzione differenti senza ricodificare integralmente l'immagine. Tile, precinct e code-block introducono inoltre livelli diversi di località e accesso regionale. È un'architettura particolarmente adatta a immagini molto grandi, digital cinema, archivi professionali e applicazioni in cui serve accesso progressivo.

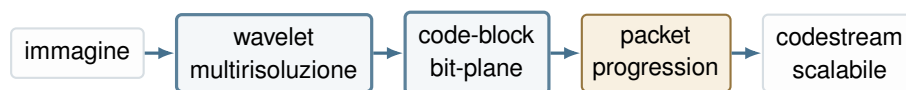


Figura 3.28. Catena concettuale JPEG 2000: la trasformata multirisoluzione e la codifica per bit-plane permettono progressività e accesso selettivo.

High-Throughput JPEG 2000 (HTJ2K), definito nella Part 15 della famiglia JPEG 2000, mantiene gran parte della struttura e delle proprietà del sistema ma sostituisce il block coder con un algoritmo progettato per architetture moderne e molto maggiore throughput. L'obiettivo non è creare un formato semanticamente nuovo: è rimuovere un collo di bottiglia computazionale preservando interoperabilità e possibilità di transcodifica matematicamente lossless con JPEG 2000 tradizionale.

Questa evoluzione mostra un principio generale importante: uno standard può rimanere valido concettualmente ma richiedere una nuova implementazione interna quando cambiano CPU, SIMD, GPU e requisiti di throughput.

3.3.7 JPEG XS: compressione a latenza estremamente bassa

JPEG XS, famiglia ISO/IEC 21122, occupa un punto di progetto ancora diverso. Non cerca il minimo numero di bit possibile: mira a qualità visivamente lossless, bassa complessità, memoria ridotta e latenza di poche linee. È destinato a collegamenti professionali, produzione, broadcast, Pro-AV, sensori e pipeline nelle quali il video non compresso sarebbe troppo costoso ma una codifica temporale complessa introdurrebbe una latenza inaccettabile.

Il vantaggio si comprende confrontando due obiettivi. Un codec di archiviazione può analizzare un'immagine intera o molte immagini per ottenere più compressione; un codec di contribution deve spesso iniziare a produrre l'uscita prima di avere ricevuto il frame completo. La latenza algoritmica diventa quindi una grandezza di progetto, non un effetto secondario.

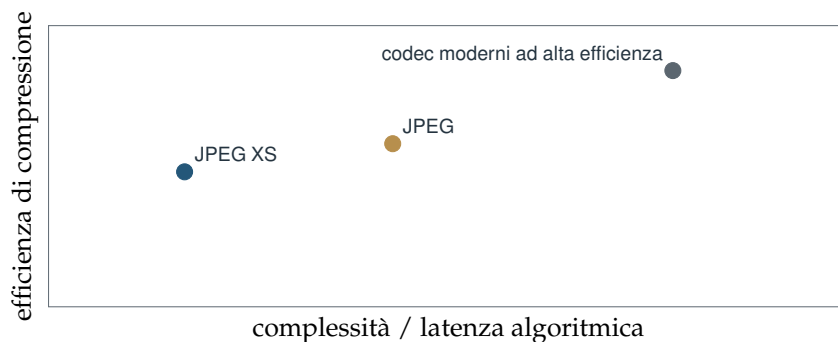


Figura 3.29. Mappa qualitativa dei diversi obiettivi di progetto. JPEG XS sacrifica parte dell'efficienza massima per ridurre drasticamente memoria, complessità e latenza. Le coordinate sono illustrative.

Nel 2026 JPEG XS è una tecnologia matura e continua a evolvere con profili, livelli, metadati e supporto per sorgenti raw Bayer. Il punto didattico è essenziale: il rapporto di compressione non è l'unica metrica con cui giudicare un codec professionale.

3.3.8 WebP

WebP riunisce in un formato orientato al Web modalità lossy, lossless, trasparenza e animazione. La modalità lossy deriva dalla codifica intra di VP8 e introduce una differenza concettuale importante rispetto al JPEG baseline: prima della trasformazione si costruisce una predizione spaziale del blocco usando campioni vicini già ricostruiti. Si codifica quindi il residuo

$$e(i, j) = x(i, j) - \hat{x}(i, j).$$

Questa è la definizione del residuo di predizione.

Se la predizione è efficace, il residuo concentra valori intorno a zero e richiede meno informazione del blocco originale. La stessa idea sarà centrale nei codec video, con la differenza che la predizione potrà provenire anche da altri istanti temporali.

La modalità lossless usa trasformazioni reversibili, predizione e codifica entropica senza quantizzazione distruttiva. WebP può quindi coprire grafica lossless, fotografie lossy, alpha e animazione con un singolo ecosistema di file.

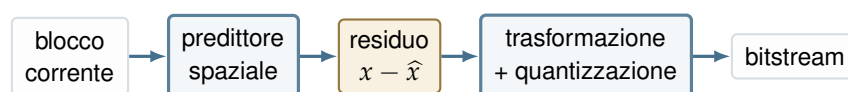


Figura 3.30. Idea della modalità WebP lossy: predire il blocco e trasformare il residuo invece dei campioni originali.

Non esiste un fattore universale di vantaggio rispetto a JPEG: dipende da contenuto, qualità, encoder, preset e metrica. I confronti seri devono essere effettuati su curve rate-distortion.

3.3.9 HEIF e HEIC: separare contenitore e codec

High Efficiency Image File Format (HEIF) è un formato contenitore basato sulla famiglia ISO Base Media File Format. Può organizzare immagini singole, sequenze, immagini derivate, miniature, alpha, metadati e relazioni fra elementi. HEIF non identifica da solo un algoritmo di compressione specifico.

HEIC è il nome comunemente associato a file HEIF che usano HEVC per codificare le immagini. Questa distinzione è importante: il contenitore definisce come immagini e proprietà vengono organizzate; il codec definisce come i campioni dell'immagine vengono compressi.

Il modello a item di HEIF permette, per esempio, di memorizzare una immagine principale, una thumbnail e un piano alpha come oggetti correlati senza duplicare necessariamente tutte le informazioni di struttura. La stessa architettura è stata riutilizzata da AVIF.

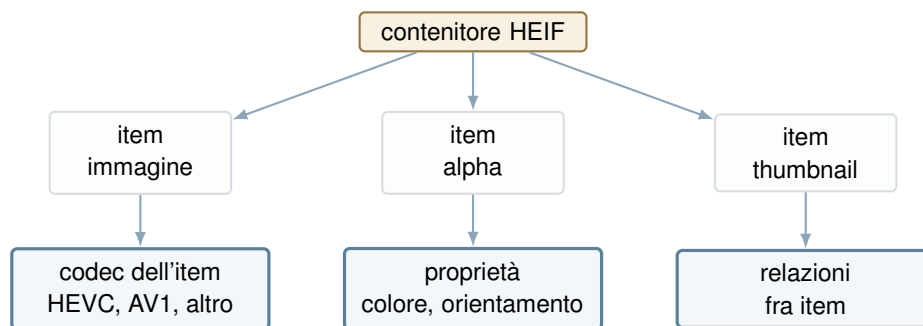


Figura 3.31. Separazione fra contenitore HEIF, dati codificati e proprietà. HEIC è un caso d'uso HEIF con immagini codificate HEVC, non un sinonimo concettuale del contenitore.

3.3.10 AVIF

AVIF usa la codifica intra di AV1 all'interno di un file conforme a HEIF. La versione 1.2.0 della specifica AOMedia, pubblicata nel 2025, formalizza anche strumenti per immagini stratificate e trasformazioni derivate, oltre alle funzioni consolidate di HDR, wide color gamut, alpha e sequenze animate.

Rispetto a JPEG il cambiamento algoritmico è profondo. AV1 dispone di partizionamenti più flessibili, numerosi modi di predizione intra, trasformazioni adattabili e codifica entropica contestuale. Rispetto a un codec video completo, AVIF usa la parte necessaria a descrivere una immagine o una sequenza di immagini nel modello HEIF.

La maggiore efficienza potenziale richiede però più elaborazione. Un encoder può esplorare molte decisioni di partizionamento e predizione; il punto di lavoro dipende quindi fortemente dal preset. Anche qui "AVIF è il 50% più piccolo" non è una legge: è una misura possibile in uno specifico esperimento, non una prestazione universale.

3.3.11 JPEG XL

JPEG XL, famiglia ISO/IEC 18181, è progettato specificamente per immagini. Il core coding system è nella seconda edizione ISO/IEC 18181-1:2024; nel 2026 è stata pubblicata la terza edizione del file format ISO/IEC 18181-2.

JPEG XL combina due famiglie di strumenti. La modalità VarDCT usa trasformazioni adattive per contenuti fotografici; la modalità Modular usa trasformazioni e predizione reversibili particolarmente adatte al lossless e a contenuti non fotografici. Il formato supporta

alta profondità, HDR, wide gamut, alpha, animazione e progressive decoding.

Una caratteristica peculiare è la ricompressione lossless di immagini JPEG esistenti: il JPEG originario può essere rappresentato in forma più compatta preservando informazione sufficiente a ricostruire esattamente il bitstream JPEG di partenza. Questo rende JPEG XL interessante anche come strumento di migrazione di archivi, perché non richiede una nuova generazione lossy.

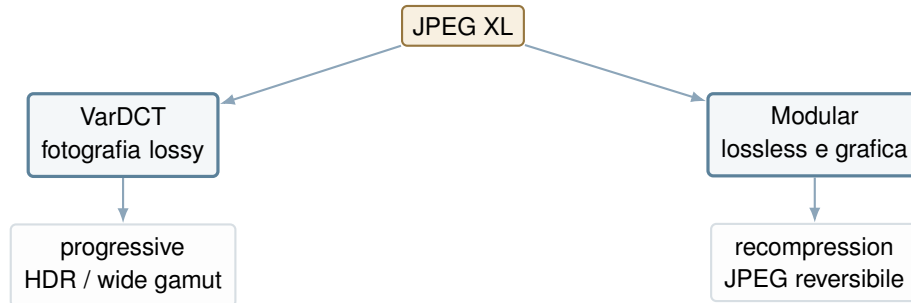


Figura 3.32. Due percorsi concettuali di JPEG XL: strumenti trasformati per fotografia e modalit  modulari per ricostruzione lossless e altre classi di immagine.

3.3.12 JPEG AI: trasformazioni apprese

JPEG AI, famiglia ISO/IEC 6048 e ITU-T T.840, rappresenta un cambio di paradigma. Il core coding system   stato pubblicato come ISO/IEC 6048-1:2025; nel 2026 sono stati pubblicati ISO/IEC 6048-2:2026 per profiling e ISO/IEC 6048-5:2026 per il file format.

Nei codec tradizionali il progettista sceglie esplicitamente predittore, trasformata, quantizzatore e modello entropico. In un sistema learning-based una parte sostanziale della trasformazione fra immagine e rappresentazione latente viene appresa da dati di training. Un encoder neurale produce variabili latenti; un modello probabilistico ne stima la distribuzione per la codifica entropica; un decoder neurale ricostruisce l'immagine.

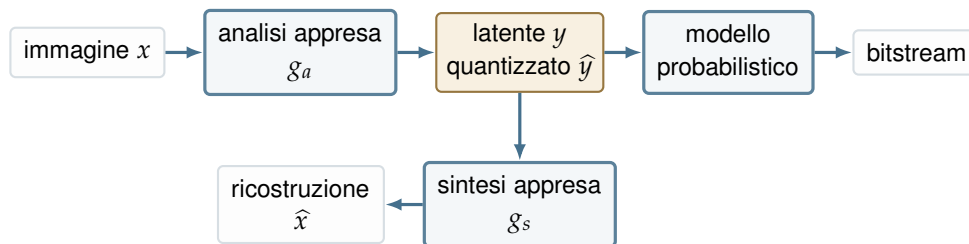


Figura 3.33. Architettura concettuale di un codec learning-based: trasformazioni e modello della distribuzione latente vengono appresi, mentre il bitstream deve restare definito in modo interoperabile.

L'addestramento pu  ottimizzare una funzione rate–distortion del tipo

$$J = R + \lambda D.$$

Questa equazione definisce una comune funzione obiettivo, non una legge universale: R stima il costo di codifica, D misura la distorsione e $\lambda > 0$ sceglie il compromesso.

Se due soluzioni candidate hanno costi (R_1, D_1) e (R_2, D_2) , la prima   preferita da questa funzione quando

$$R_1 + \lambda D_1 < R_2 + \lambda D_2.$$

Dimostrazione. La funzione obiettivo assegna per definizione a ogni soluzione il valore $J = R + \lambda D$. Minimizzare J significa scegliere il candidato con valore numerico minore;

sostituendo i due candidati si ottiene la disuguaglianza. Questo conclude la dimostrazione.

La novità non è soltanto usare una rete neurale. Lo standard deve rendere interoperabile un processo di ricostruzione basato su modelli appresi e deve controllare complessità, precisione numerica, profili e conformità. Inoltre una rappresentazione latente può diventare utile non solo per la visione umana ma anche per elaborazioni successive di machine vision, avvicinando compressione e analisi automatica.

3.3.13 Confronto fra le principali famiglie

Le tecnologie moderne non formano una semplice sequenza in cui ogni formato rende inutile il precedente. Ottimizzano obiettivi differenti.

Formato	Lossless	Lossy	HDR/WCG	Alpha	Obiettivo distintivo
GIF	sì sugli indici	palette	no nativo	1 bit	semplicità / animazione
PNG	sì	no	limitato dal workflow	sì	ricostruzione esatta
JPEG	processi dedicati	sì	non naturale nel baseline	no	fotografia semplice
JPEG 2000	sì	sì	sì	possibile	scalabilità / accesso
JPEG XS	opzionale / profili	sì	sì	workflow	latenza minima
WebP	sì	sì	SDR tipico	sì	Web general purpose
HEIC	dipende da HEVC	sì	sì	sì	ecosistema HEIF/HEVC
AVIF	sì	sì	sì	sì	alta efficienza / AV1
JPEG XL	sì	sì	sì	sì	immagini / progressive / JPEG
JPEG AI	profilo dipendente	sì	sì	profili	coding appreso

Tabella 3.10. Confronto qualitativo delle famiglie trattate. “Sì” indica una capacità del sistema, non garantisce che ogni profilo o implementazione la utilizzi.

3.3.14 Evoluzione delle idee

L’evoluzione dei formati è più chiara se la si legge come evoluzione dei meccanismi:

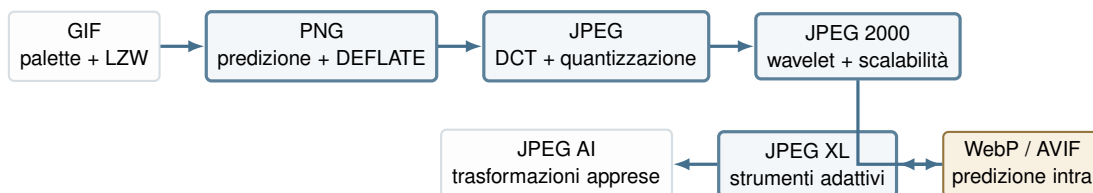


Figura 3.34. Evoluzione concettuale delle tecniche di compressione delle immagini. Le famiglie co-esistono perché ottimizzano obiettivi diversi di efficienza, complessità, latenza, scalabilità e interoperabilità.

Le idee fondamentali restano poche e ricorrenti: rimuovere ridondanza, predire ciò che è prevedibile, trasformare per concentrare energia, quantizzare quando la perdita è ammissibile e rappresentare entropicamente gli eventi risultanti. Nel video il capitolo successivo aggiungerà la più potente sorgente di ridondanza ancora non sfruttata: la correlazione temporale fra frame.

Compressione delle sorgenti dinamiche

4.1 Audio

La compressione audio sfrutta tre classi di struttura, spesso combinate nello stesso codec. La prima è la correlazione temporale: campioni vicini di voce e musica non sono indipendenti e possono essere predetti. La seconda è la struttura della sorgente vocale: invece di trasmettere la forma d'onda completa si possono stimare parametri del modello di produzione. La terza è percettiva: il sistema uditivo non è ugualmente sensibile a tutte le componenti spettrali e temporali, per cui una quantizzazione lossy può distribuire l'errore dove è meno udibile.

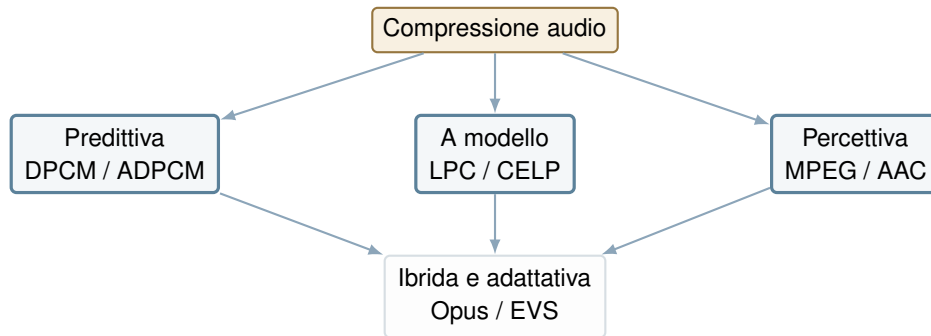


Figura 4.1. Tre idee fondamentali della compressione audio e loro convergenza nei codec moderni.

La qualità non può essere riassunta da una sola bitrate. Dipende da banda audio, latenza, robustezza alle perdite, contenuto e criterio di ascolto. I test soggettivi possono essere riassunti mediante MOS, ma il valore ha senso soltanto insieme al protocollo di prova. Numeri MOS ricavati da campagne differenti non vanno confrontati come se fossero misure assolute.

G.711 è il riferimento della telefonia PCM tradizionale a 64 kbit/s e usa companding A-law oppure μ -law. Non esiste una terza modalità G.711 lineare a 8 bit da contrapporre alle due leggi: il PCM lineare è un concetto generale, mentre la raccomandazione G.711 specifica proprio le due codifiche logaritmiche a 8 bit già discusse nel capitolo 2.

4.1.1 DPCM

Differential Pulse Code Modulation (DPCM) sfrutta la correlazione temporale del segnale: invece di quantizzare direttamente il campione x_n , costruisce una previsione $\hat{x}_n$ dai campioni passati e quantizza il residuo. Un predittore lineare di ordine p è definito da

$$\hat{x}_n = \sum_{k=1}^p \alpha_k x_{n-k}.$$

La relazione è la definizione del predittore; i coefficienti α_k devono essere scelti secondo un criterio di ottimalità.

Per un processo a media nulla, stazionario in senso lato, definiamo

$$\phi(\ell) = E[x_n x_{n-\ell}]$$

come funzione di autocorrelazione. I coefficienti che minimizzano l'errore quadratico medio

$$E_p = E \left[\left(x_n - \sum_{k=1}^p \alpha_k x_{n-k} \right)^2 \right]$$

soddisfano le equazioni di Wiener-Hopf

$$\sum_{k=1}^p \alpha_k \phi(j-k) = \phi(j), \quad j = 1, \dots, p.$$

Dimostrazione. Sviluppando il quadrato,

$$E_p = \phi(0) - 2 \sum_{k=1}^p \alpha_k \phi(k) + \sum_{k=1}^p \sum_{h=1}^p \alpha_k \alpha_h \phi(k-h).$$

Per un processo reale stazionario $\phi(-r) = \phi(r)$. Derivando rispetto a α_j ,

$$\frac{\partial E_p}{\partial \alpha_j} = -2\phi(j) + 2 \sum_{k=1}^p \alpha_k \phi(j-k).$$

Nel minimo la derivata è nulla. Dividendo per 2 e isolando $\phi(j)$ si ottengono le relazioni dichiarate. Questo conclude la dimostrazione. Nel modello autoregressivo esse assumono la forma delle equazioni di Yule-Walker.

Se l'autocorrelazione non è nota, deve essere stimata. Una possibile scelta per un blocco di N campioni è lo stimatore *biased*

$$\hat{\phi}(a) = \frac{1}{N} \sum_{n=0}^{N-1-a} x_n x_{n+a}, \quad 0 \leq a < N.$$

Questa è una definizione di stimatore. La normalizzazione con $N - a$ produce invece la forma comunemente detta *unbiased* per un processo a media nota nulla; le due scelte hanno proprietà statistiche differenti e possono influenzare stabilità e varianza della stima.

Il residuo ideale è

$$e_n = x_n - \hat{x}_n.$$

Se il decoder ricostruisce la predizione usando campioni diversi da quelli usati dall'encoder, gli stati dei due lati divergerebbero. DPCM pratico usa quindi un anello chiuso: il predittore dell'encoder è alimentato dagli stessi campioni ricostruiti $\tilde{x}_n$ che saranno disponibili al decoder,

$$\hat{x}_n = \sum_{k=1}^p \alpha_k \tilde{x}_{n-k}, \quad e_n = x_n - \hat{x}_n.$$

Il quantizzatore produce un residuo ricostruito $\tilde{e}_n$ e il campione ricostruito è

$$\tilde{x}_n = \hat{x}_n + \tilde{e}_n.$$

Queste relazioni definiscono il loop DPCM.

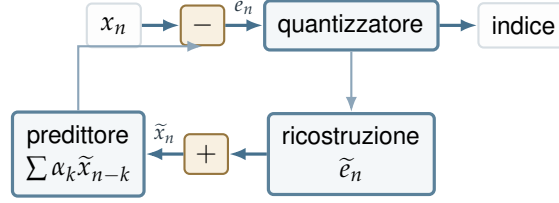


Figura 4.2. DPCM in anello chiuso. Encoder e decoder predicono dagli stessi campioni ricostruiti, evitando deriva dovuta alla quantizzazione.

Verifica della ricostruzione senza perdita. Se la quantizzazione fosse esatta, $\tilde{e}_n = e_n$. Allora

$$\tilde{x}_n = \hat{x}_n + e_n = \hat{x}_n + x_n - \hat{x}_n = x_n.$$

Questo conclude la verifica. Nel caso lossy l'errore residuo è dovuto alla differenza fra e_n e $\tilde{e}_n$.

La compressione non deriva automaticamente dall'ordine del predittore. Dipende da quanto la predizione concentra il residuo vicino a zero e da come tale residuo viene quantizzato e codificato. Per esempio, una realizzazione che produca 7 bit per campione a 8 ksample/s genera

$$R_b = 7 \cdot 8000 = 56 \text{ kbit/s}.$$

Dimostrazione. In un secondo vengono prodotti 8000 indici, ciascuno di 7 bit. Il prodotto è 56,000 bit/s. Questo conclude la dimostrazione. Il valore è una scelta del quantizzatore, non una conseguenza matematica di $p = 1$ o di un particolare predittore.

Un modello più generale può includere anche residui passati,

$$\hat{x}_n = \sum_{k=1}^p \alpha_k \tilde{x}_{n-k} + \sum_{k=1}^m \beta_k \tilde{e}_{n-k}.$$

Questa relazione definisce una struttura predittiva con una parte autoregressiva e una parte legata agli errori ricostruiti.

Delta modulation

La Delta Modulation può essere vista come un caso estremo di codifica differenziale: a ogni passo trasmette un solo bit che indica se la ricostruzione deve salire o scendere di un gradino prefissato. Non rappresenta quindi direttamente l'ampiezza dell'errore.

L'uso di un solo bit per aggiornamento richiede una frequenza di aggiornamento sufficientemente elevata rispetto alla dinamica del segnale. Se il gradino è troppo piccolo rispetto alla pendenza del segnale si produce *slope overload*; se è troppo grande in regioni quasi costanti domina il *granular noise*. Non esiste un fattore di oversampling universale valido per ogni segnale: passo e frequenza di aggiornamento devono essere progettati insieme.

4.1.2 APCM e ADPCM

I segnali reali, in particolare la voce, cambiano statistiche nel tempo. Un quantizzatore e un predittore fissati una volta per tutte non sono quindi sempre efficienti. Adaptive PCM modifica il passo di quantizzazione in funzione della dinamica osservata; Adaptive DPCM può adattare quantizzatore, predittore o entrambi.

L'adattamento può essere *forward*: l'encoder stima parametri su un blocco e li trasmette al decoder. Oppure può essere *backward*: encoder e decoder aggiornano lo stato usando soltanto campioni già ricostruiti e quindi restano sincronizzati senza inviare esplicitamente

tutti i parametri. La seconda soluzione risparmia side information ma vincola l'algoritmo di adattamento e può reagire più lentamente ai cambiamenti.

G.721 è un importante predecessore storico ADPCM a 32 kbit/s. Le sue funzioni sono state successivamente incorporate nella famiglia G.726, che definisce modalità ADPCM a 16, 24, 32 e 40 kbit/s. Questi valori sono proprietà della specifica di codifica; eventuali valori MOS dipendono invece dal protocollo di ascolto e non vengono trattati come costanti del codec.

4.1.3 LPC

Il Linear Predictive Coding è un codificatore per modello.

Per poter costruire un modello occorre analizzare come i suoni umani vengono prodotti. I polmoni emettono un segnale paragonabile al rumore bianco con una pressione dell'aria caratterizzata da un certo guadagno G . Le corde vocali possono essere aperte, e non modulano il segnale (e in tal caso vengono emessi suoni non vocalizzati), o chiuse, ed emettono un'oscillazione periodica con una certa frequenza fondamentale (suoni vocalizzati). Il tratto vocale si comporta come un filtro tempo variante: la sua funzione di trasferimento dipende dal fonema pronunciato e presenta alcune frequenze di risonanza chiamate formanti.

Nel LPC si assume che un segnale vocale sia prodotto da un ronzatore posto alla fine di un tubo (suoni vocalizzati) al quale vengono aggiunti occasionalmente suoni sibilanti e schioccanti (suoni non vocalizzati). Per i suoni vocalizzati il ronzatore costituisce la funzione di eccitazione mentre il tubo effettua la modulazione del suono. Questo modello rappresenta un'approssimazione semplificata di come un essere umano produce suoni: la glottide produce il ronzio caratterizzato da un'intensità e una frequenza; la gola e la lingua formano il tubo in cui il suono si propaga e modifica, caratterizzato da risonanze prodotte dai formanti. Per i suoni non vocalizzati i sibili e gli schiocchi sono realizzati dall'insieme: gola, lingua, labbra. In questo caso le corde vocali non vibrano e il tubo viene utilizzato come risuonatore.

Il sistema LPC analizza il segnale vocale e cerca di stimare i formanti, rimuovendo il loro effetto dal segnale per determinare la frequenza e l'intensità del ronzio di eccitazione. Considerando che vi sono due diverse tipologie di suoni da riprodurre (vocalizzati e non) si considerano due diversi generatori: un oscillatore con una determinata frequenza e un generatore di rumore bianco. Uno schema del codificatore è riportato in figura 4.3.

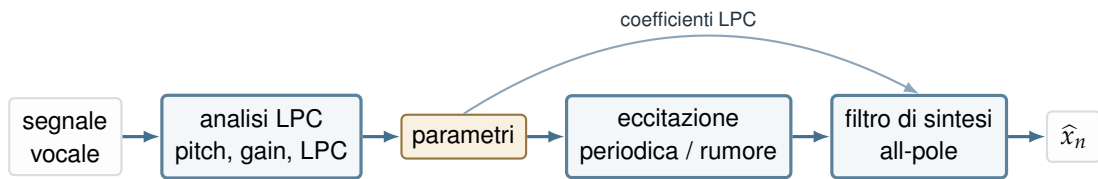


Figura 4.3. Modello LPC di analisi e sintesi: l'eccitazione rappresenta la sorgente, il filtro all-pole modella il tratto vocale.

Un codificatore LPC non trasmette direttamente i campioni. Per ogni frame deve rappresentare almeno i coefficienti del filtro, il guadagno e i parametri dell'eccitazione. Il numero di bit assegnato a ciascun parametro dipende dal codec specifico: non esiste una quantizzazione universale dei coefficienti LPC.

Consideriamo il segnale di eccitazione campionato v_n . Nel modello all-pole definiamo il filtro di sintesi

$$H(z) = \frac{G}{1 - \sum_{k=1}^p \alpha_k z^{-k}}.$$

Questa è una definizione del modello LPC. Ne segue la relazione temporale

$$x_n = \sum_{k=1}^p \alpha_k x_{n-k} + Gv_n = \tilde{x}_n + Gv_n.$$

Dimostrazione. Dalla relazione $X(z) = H(z)V(z)$,

$$X(z) \left(1 - \sum_{k=1}^p \alpha_k z^{-k} \right) = GV(z).$$

Sviluppando,

$$X(z) - \sum_{k=1}^p \alpha_k z^{-k} X(z) = GV(z).$$

La trasformata Z inversa usa la proprietà $z^{-k}X(z) \leftrightarrow x_{n-k}$ e produce

$$x_n - \sum_{k=1}^p \alpha_k x_{n-k} = Gv_n.$$

Isolando x_n si ottiene la relazione dichiarata. Questo conclude la dimostrazione.

L'uscita è quindi la somma della predizione di ordine p ,

$$\tilde{x}_n = \sum_{k=1}^p \alpha_k x_{n-k},$$

e dell'eccitazione corrente. Quest'ultima può essere modellata in modo differente per tratti voiced e unvoiced: il modello classico usa una sorgente quasi periodica per il parlato sonoro e una sorgente simile al rumore per componenti non sonore.



Figura 4.4. Catena concettuale di un vocoder LPC. Il decoder riceve parametri del modello, non campioni della forma d'onda.

LPC-10/FS-1015 è un esempio storico di vocoder lineare a circa 2.4 kbit/s e ordine di predizione 10. La bitrate molto bassa deriva dal fatto che il sistema trasmette pochi parametri per frame invece di una forma d'onda dettagliata. Il prezzo è una qualità fortemente legata al modello vocale: musica, rumori complessi e transitori non sono descritti bene da una sorgente periodica/rumorosa filtrata da un modello all-pole. Il valore storico di LPC-10 è quindi mostrare quanto si possa comprimere quando si accetta di codificare il *modello della sorgente* invece del segnale campione per campione.

4.1.4 CELP

Il Code Excited Linear Predictor (CELP) è un codificatore per modello basato su 4 idee: il modello della sorgente è più complesso rispetto a quello utilizzato in LPC; si utilizzano congiuntamente sia funzioni di eccitazione fisse e note che funzioni adattative ottenute sulla base delle caratteristiche della sorgente; si effettua un closed-loop per la selezione della forma d'onda di eccitazione; la quantizzazione è di tipo vettoriale.

Il modello della produzione vocale considera le corde vocali come generatori a spettro piatto ed assume che il tratto vocale agisca come filtro in grado di colorare lo spettro. I diversi fenomeni vocali sono distinti sulla base di differenti eccitazioni e differenti colorazioni

dello spettro. Le vocalizzazioni hanno un'eccitazione periodica, approssimata con un treno di impulsi nel dominio del tempo, cioè con armoniche equispaziate in frequenza. Suoni non vocalizzati hanno una eccitazione simile a quella del rumore gaussiano bianco. Inoltre sono presenti eccitazioni composte congiuntamente da una parte armonica e da una parte di rumore.

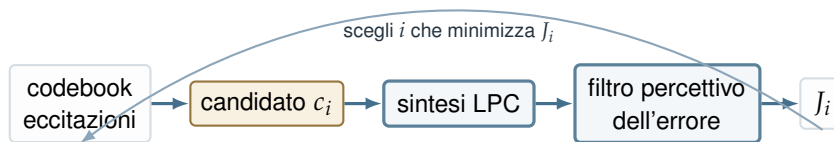


Figura 4.5. CELP analysis-by-synthesis: l'encoder prova eccitazioni candidate, simula la ricostruzione e sceglie quella che minimizza un errore pesato percettivamente.

Il closed-loop è diretto alla scelta della funzione di eccitazione. Una forma d'onda di eccitazione viene utilizzata per stimolare il codificatore e localmente viene effettuata la decodifica, ottimizzando tutti i parametri come nel caso LPC. La forma d'onda ottenuta dal processo di codifica e decodifica viene confrontata con la funzione da rappresentare e ne viene calcolato l'errore. Sulla base dell'errore prodotto, opportunamente pesato, si effettua una nuova scelta di una forma d'onda di eccitazione, andando a minimizzare l'errore all'aumentare delle prove. Il loop è tra forma d'onda, codificatore, decodificatore, filtraggio sull'errore e minimizzazione per selezionare la nuova forma d'onda. In figura 4.5 è riportato l'esempio del closed-loop descritto.

La famiglia CELP ha avuto un ruolo importante nei codec vocali ITU-T. G.728 usa Low-Delay CELP (LD-CELP) a 16 kbit/s. La struttura elabora vettori di cinque campioni a 8 ksample/s e la raccomandazione specifica un ritardo algoritmico di 0.625 ms. È un buon esempio di progetto nel quale la latenza viene ridotta rinunciando a frame lunghi e look-ahead esteso.

G.729 usa Conjugate-Structure Algebraic CELP (CS-ACELP) a 8 kbit/s. La struttura base opera su frame di 10 ms con 5 ms di look-ahead, per un ritardo algoritmico di 15 ms prima di considerare pacchettizzazione, rete, jitter buffer e tempo di esecuzione. Varianti della famiglia introducono Voice Activity Detection, discontinuous transmission e Comfort Noise Generation, utili per non trasmettere a bitrate piena durante le pause senza produrre al ricevitore un silenzio innaturale.

G.723.1, progettato per comunicazioni multimediali a bitrate molto bassa, definisce due modalità obbligatorie: 6.3 kbit/s e 5.3 kbit/s. Usa frame di 30 ms e 7.5 ms di look-ahead, quindi un ritardo algoritmico di 37.5 ms. La modalità a 6.3 kbit/s usa un'eccitazione MP-MLQ, mentre quella a 5.3 kbit/s usa ACELP.

I valori di complessità espressi in MIPS dipendono fortemente da architettura, implementazione e ottimizzazioni, mentre i punteggi MOS dipendono dal protocollo di ascolto, dal materiale e dalle condizioni di prova. Per questo non vengono trattati come proprietà assolute dei codec; bitrate, frame size e look-ahead appartengono invece alla struttura definita dalle rispettive raccomandazioni.

4.1.5 FLAC: compressione audio lossless

La compressione audio non deve essere necessariamente percettiva e lossy. Free Lossless Audio Codec (FLAC) rappresenta un riferimento importante per archiviazione e distribuzione quando i campioni PCM devono essere ricostruiti esattamente.

FLAC lavora per blocchi. Per ciascun blocco il codificatore può scegliere fra campioni verbatim, valori costanti o una previsione lineare. Nel caso predittivo non viene memorizzato

direttamente il segnale, ma il residuo fra il campione e la previsione; il residuo, spesso concentrato intorno a zero, viene poi rappresentato con codici entropici della famiglia Rice. Il decoder ricostruisce il predittore e somma il residuo, senza introdurre quantizzazione.

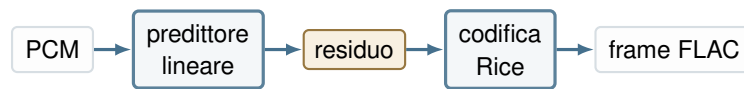


Figura 4.6. Principio della compressione FLAC: la previsione riduce la dinamica del residuo, che viene codificato senza quantizzazione e quindi senza perdita.

Il principio richiama DPCM e LPC, ma con un obiettivo diverso: non si accetta alcun errore di ricostruzione. Il rapporto di compressione dipende quindi dalla prevedibilità del segnale; rumore quasi bianco o dati già compressi offrono poco margine, mentre musica e voce PCM contengono correlazioni sfruttabili. CRC e framing permettono inoltre di rilevare corruzioni e di mantenere punti di sincronizzazione nel bitstream.

FLAC completa il quadro dei codec audio del capitolo: PCM rappresenta il segnale senza compressione, FLAC comprime senza perdita, AAC/Opus/EVS e le altre famiglie lossy accettano invece una degradazione controllata per ottenere riduzioni di bitrate molto maggiori.

4.1.6 MPEG-1

Nei codificatori LPC e CELP viene utilizzato un modello in grado di simulare l'apparato vocale e quindi un trasduttore del suono. Un'altra categoria di codificatore a modello prevede di utilizzare il modello di percezione dell'orecchio umano e quindi del sensore. Questa tipologia di codificatore utilizza congiuntamente la codifica a modello, il dominio del tempo e il dominio della frequenza e viene chiamato codificatore percettivo. Un esempio di codifica percettiva è stato sviluppato dal Moving Picture Experts Group (MPEG).

L'orecchio umano ha caratteristiche molto particolari, definite nel modello psicoacustico. Il modello psicoacustico consente di eliminare una serie di informazioni dalla sorgente che l'orecchio non è in grado di sentire, diminuendo la bitrate e quindi effettuando una compressione.

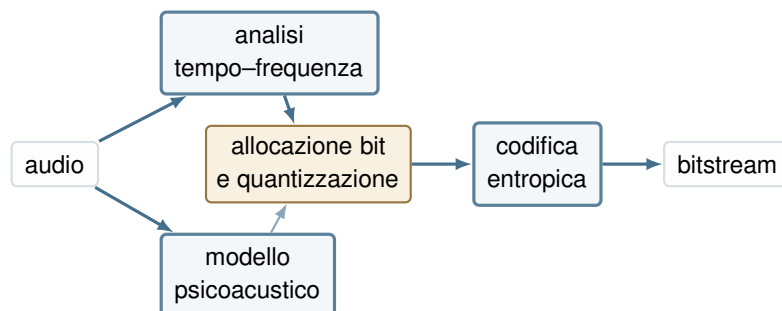


Figura 4.7. Architettura generale di un codificatore audio percettivo: la rappresentazione spettrale e il modello dell'udito guidano l'allocazione della precisione.

Uno schema di massima di un codificatore percettivo è riportato in figura 4.7. Il segnale di ingresso viene sottoposto a due diverse tipologie di analisi: tempo/frequenza e modello psicoacustico. Il modello psicoacustico viene utilizzato per definire quanti bit utilizzare nella quantizzazione dei parametri in uscita dall'analisi tempo/frequenza. Il numero dei bit utilizzati e le informazioni quantizzate vengono compresse mediante un sistema lossless ed opportunamente impacchettate per formare lo stream di uscita.

Modello psicoacustico

Il Sound Pressure Level (SPL) misura il livello della pressione sonora efficace rispetto a una pressione di riferimento:

$$L_p = 20 \log_{10} \frac{p_{\text{rms}}}{p_0}, \quad p_0 = 20 \mu\text{Pa}$$

per il riferimento convenzionale in aria. Questa relazione è la definizione di SPL.

Il fattore 20, invece di 10, deriva dal fatto che in condizioni di impedenza acustica fissata l'intensità è proporzionale al quadrato della pressione efficace.

Dimostrazione. Se $I \propto p_{\text{rms}}^2$ e $I_0 \propto p_0^2$, il livello di potenza è

$$10 \log_{10} \frac{I}{I_0} = 10 \log_{10} \left(\frac{p_{\text{rms}}}{p_0} \right)^2 = 20 \log_{10} \frac{p_{\text{rms}}}{p_0}.$$

Questo conclude la dimostrazione del fattore 20 nelle condizioni assunte.

L'intervallo udibile dipende da frequenza, individuo, età, ambiente e criterio sperimentale; i valori prossimi a 120–130 dB fra soglia di udibilità e livelli dolorosi sono quindi ordini di grandezza, non costanti fisiologiche universali.

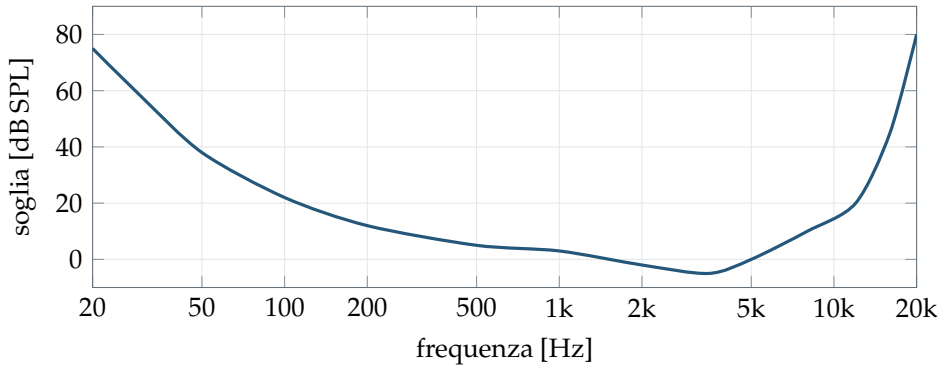


Figura 4.8. Andamento qualitativo della soglia assoluta di udibilità. La massima sensibilità si trova nell'ordine di pochi kHz; la curva dipende da condizioni di misura e popolazione.

La soglia di udibilità caratterizza l'ammontare di energia di un tono sinusoidale ascoltabile da un orecchio umano in un ambiente rumoroso. Questa soglia è stata elaborata da Fletcher nel 1940 sulla base di una campagna di misure su giovani ascoltatori e viene riportata in figura 4.8. I segnali udibili sono quelli superiori alla soglia di udibilità. Un'approssimazione empirica frequentemente usata per la soglia assoluta di udibilità, con f espresso in Hz e T in dB SPL, è la seguente. È un fit sperimentale, non una relazione teorica da dimostrare:

$$T(f) = 3.64 \left(\frac{f}{1000} \right)^{-0.8} - 6.5e^{-0.6 \left(\frac{f}{1000} - 3.3 \right)^2} + 10^{-3} \left(\frac{f}{1000} \right)^4$$

Come si verifica immediatamente sia dal grafico che dalla funzione $T(f)$, il comportamento dell'orecchio è fortemente non-lineare. La maggiore sensibilità dell'orecchio si ha per frequenze nel range da 2 kHz a 5 kHz. Un suono al di sotto della soglia di udibilità può essere eliminato senza degradare la percezione sonora.

L'orecchio realizza un'analisi spettrale comportandosi come un insieme di filtri sovrapposti, con banda variabile in funzione della frequenza. La banda di ogni filtro viene chiamata

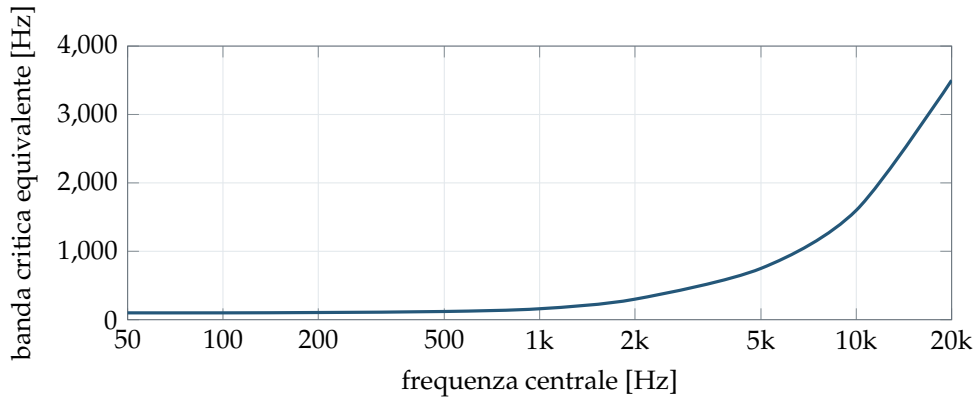


Figura 4.9. Crescita qualitativa della larghezza di banda critica con la frequenza. Le formule usate nei modelli psicoacustici sono approssimazioni empiriche.

banda critica. Un grafico qualitativo della banda critica è riportato in figura 4.9. Una comune approssimazione empirica della larghezza equivalente è:

$$BW(f) = 25 + 75 \left[1 + 1.4 \left(\frac{f}{1000} \right)^2 \right]^{0.69}$$

La regola mnemonica di circa 100 Hz alle basse frequenze e circa $f/5$ alle frequenze superiori è soltanto una semplificazione didattica del comportamento medio, non una legge esatta dell'udito.

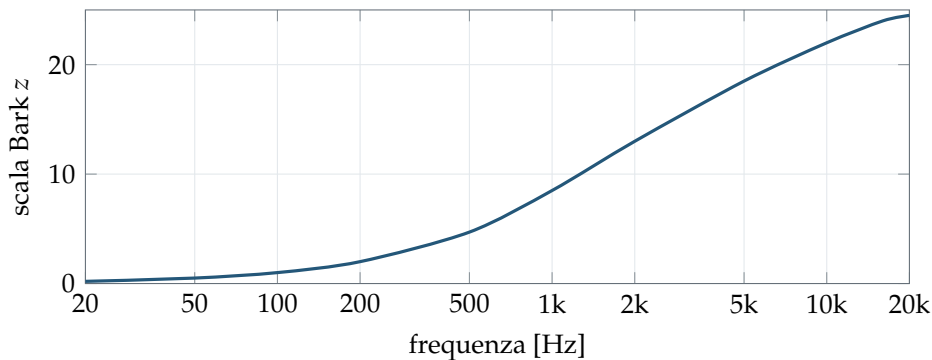


Figura 4.10. Relazione qualitativa fra frequenza e scala Bark, usata per esprimere distanza percettiva fra componenti spettrali.

Volendo trattare l'orecchio come un insieme discreto e finito di filtri, è importante definire la spaziatura tra i filtri e quindi la successione delle bande. Tale spaziatura viene spesso riferita come spaziatura di Bark. La curva in figura 4.10 può essere descritta da una delle approssimazioni empiriche comunemente usate; anche questa è una parametrizzazione percettiva, non un teorema:

$$z(f) = 13 \tan^{-1}(0.00076f) + 3.5 \tan^{-1} \left[\left(\frac{f}{7500} \right)^2 \right]$$

Un fenomeno interessante si ha quando la presenza di un suono ne rende un altro non udibile. Questo fenomeno viene detto di mascheramento ed è fondamentale per ridurre le informazioni, da rappresentare, della sorgente. Sono possibili due diverse tipologie di mascheramento: quello frequenziale e quello temporale. La spiegazione fisica di questo

fenomeno è che una sollecitazione forte crea uno stress alla membrana uditiva, relativamente alla banda critica, tale da non consentire l'ascolto di segnali più deboli, nell'ambito della stessa banda.

Analizziamo alcuni scenari di mascheramento. Il primo viene detto Noise-Masking-Tone (NMT), dove del rumore a banda stretta, con estensione di 1 Bark, maschera un tono all'interno della stessa banda critica, facendo in modo che l'intensità del tono mascherato sia al di sotto della soglia di udibilità. Si definisce Signal-to-Mask Ratio (SMR) il rapporto tra le intensità del segnale di mascheramento e del segnale mascherato. In questo caso il mascheramento è fornito dal rumore e il mascherato è il tono. L'SMR viene utilizzato per decidere se il tono è mascherato. Valori tipici di SMR, per questo contesto, sono inferiori ai 5dB.

Lo scenario Tone-Masking-Noise (TMN) prevede che vi sia un tono al centro di una banda critica in grado di mascherare il rumore, facendo in modo che sia al di sotto della soglia di udibilità. In questo caso l'SMR è il rapporto tra il tono mascherante e il rumore mascherato. Valori tipici di SMR per questo contesto sono nel range da 21dB a 28dB.

Lo scenario Noise-Masking-Noise (NMN) è quello più complesso da caratterizzare. In questo caso l'SMR è relativo a rumore sia per il segnale mascherato che per il mascherante. Il valore tipico di SMR per questo contesto è 26dB.

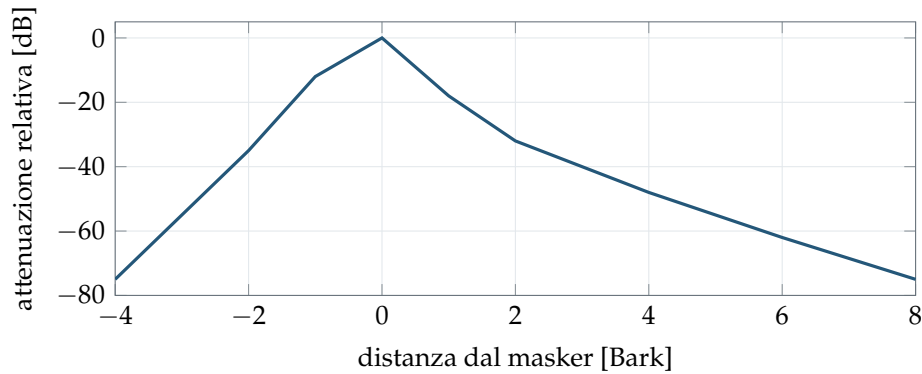


Figura 4.11. Spreading function qualitativa: un masker influenza soprattutto regioni vicine sulla scala percettiva, con asimmetria fra frequenze inferiori e superiori.

I fenomeni di tipo NMT, TMN e NMN non sono strettamente circoscritti all'interno di una sola banda critica. Infatti, è possibile che si verifichi un fenomeno di mascheramento tra bande adiacenti. Questo effetto è noto come allargamento del mascheramento ed è spesso approssimato con una funzione triangolare con pendenza +25dB per bande inferiori e +10dB per bande superiori. Una rappresentazione del fenomeno di allargamento interbanda, relativo alle unità di Bark, è data in figura 4.11 e può essere approssimato empiricamente, in dB, dalla funzione seguente, che è parte del modello percettivo e non un teorema:

$$SF(x) = 15.81 + 7.5(x + 0.474) - 17.5\sqrt{1 + (x + 0.474)^2}$$

Un ulteriore fenomeno è il mascheramento temporale. In figura 4.12 è riportato l'andamento della soglia di udibilità nel tempo in presenza di un impulso sonoro. Si osserva che il mascheramento inizia qualche millisecondo prima della sua percezione ma soprattutto che la soglia rimane più elevata per parecchio tempo dopo il termine del segnale di mascheramento. Valori tipici sono di 2 ms prima del mascheramento e da 50 a 300 ms dopo. La durata dipende dall'intensità del segnale di mascheramento.

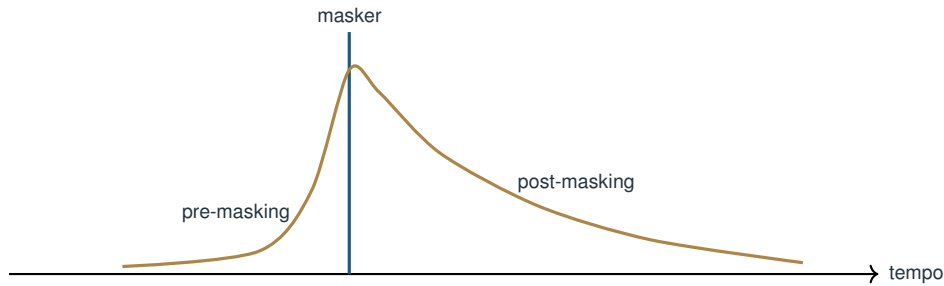


Figura 4.12. Mascheramento temporale qualitativo: uno stimolo intenso modifica la soglia di percezione anche immediatamente prima e, soprattutto, dopo il proprio istante.

Banco di sottobande e modello psicoacustico

MPEG-1 Audio separa due analisi con scopi differenti. Il percorso che produce i dati codificati usa un banco polifase di 32 sottobande uniformemente spaziate. In parallelo, il modello psicoacustico può usare una FFT più fine per stimare con maggiore risoluzione tonali, rumore e soglie di mascheramento. Le due rappresentazioni non devono quindi essere confuse.

Se la frequenza di campionamento è f_s , la banda audio positiva rappresentabile arriva idealmente a

$$f_N = \frac{f_s}{2}.$$

Dividendo tale intervallo in 32 sottobande di uguale larghezza si ottiene

$$B = \frac{f_s}{64}.$$

Dimostrazione. Il teorema del campionamento limita la banda positiva a $f_s/2$. Suddividere l'intervallo $[0, f_s/2]$ in 32 parti uguali produce

$$B = \frac{f_s/2}{32} = \frac{f_s}{64}.$$

Questo conclude la dimostrazione.

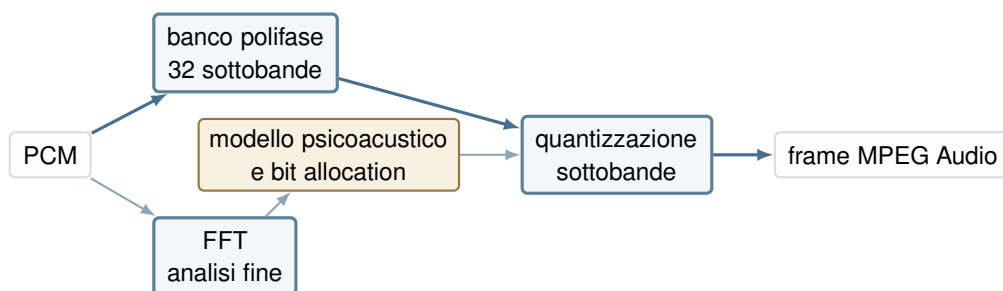


Figura 4.13. MPEG Audio: il banco di 32 sottobande produce i campioni da quantizzare; l'analisi FFT parallela serve al modello psicoacustico e non sostituisce il banco polifase.

Il modello psicoacustico non è una parte che il decoder debba riprodurre. È uno strumento dell'encoder per decidere dove spendere i bit. Lo standard definisce il bitstream e fornisce modelli di riferimento; encoder differenti possono usare modelli più sofisticati purché producano un flusso conforme.

Le formule che seguono descrivono una forma classica del modello psicoacustico MPEG. Sono definizioni e approssimazioni empiriche del modello, non teoremi sulla fisiologia dell'udito.

1. Analisi spettrale. Con N campioni x_n , finestra w_n e una costante di calibrazione P_N coerente con la scala numerica, una stima logaritmica dell'energia alla linea k può essere definita come

$$P(k) = P_N + 10 \log_{10} \left| \sum_{n=0}^{N-1} w_n x_n e^{-j2\pi kn/N} \right|^2.$$

La finestra di Hann, per esempio, è definita da

$$w_n = \frac{1}{2} \left[1 - \cos \left(\frac{2\pi n}{N-1} \right) \right], \quad 0 \leq n < N.$$

Il valore di P_N dipende dalla normalizzazione scelta; non è una costante universale del formato MPEG.

2. Componenti tonali e non tonali. I massimi locali sufficientemente prominenti rispetto alle linee circostanti vengono classificati come masker tonali. Le linee restanti vengono aggregate per banda critica in masker di rumore. Se $\mathcal{N}_\beta$ è l'insieme delle linee assegnate al rumore nella banda critica β , la loro energia complessiva espressa in dB è

$$P_{NM}(\beta) = 10 \log_{10} \sum_{j \in \mathcal{N}_\beta} 10^{P(j)/10}.$$

Dimostrazione della somma energetica. Una grandezza $P(j)$ espressa in decibel rispetto a un riferimento ha rapporto lineare $10^{P(j)/10}$. Energie indipendenti o componenti trattate energeticamente si sommano nel dominio lineare; tornando ai decibel si applica $10 \log_{10}$ alla somma. Si ottiene la relazione precedente. Questo conclude la dimostrazione.

Per un masker tonale centrato sulla linea k , una stima locale può analogamente aggregare il picco e le linee adiacenti:

$$P_{TM}(k) = 10 \log_{10} \sum_{j=-1}^1 10^{P(k+j)/10}.$$

La relazione è la stessa conversione energia-decibel applicata a un insieme diverso di linee.

3. Decimazione percettiva. Masker al di sotto della soglia assoluta vengono eliminati; masker molto vicini sulla scala Bark possono essere fusi o sostituiti dal più forte. Questa fase riduce il numero di componenti che devono essere considerate nel calcolo delle soglie individuali.

4. Soglie individuali. Per ogni masker si costruisce una spreading function sulla scala Bark. Il livello della soglia dipende dal tipo tonale/non tonale, dal livello del masker e dalla distanza percettiva $\Delta z = z(i) - z(j)$. Le funzioni a tratti impiegate nei modelli storici sono fit empirici; non descrivono un filtro fisico unico dell'orecchio.

5. Soglia globale. Sia $T_q(i)$ la soglia assoluta e siano $T_m(i, j)$ le soglie individuali dei masker, tutte espresse in dB. La soglia complessiva si ottiene sommando le potenze nel dominio lineare:

$$T_g(i) = 10 \log_{10} \left[10^{T_q(i)/10} + \sum_j 10^{T_m(i, j)/10} \right].$$

Dimostrazione. Ogni livello in dB viene riportato al dominio lineare mediante $10^{T/10}$. Le potenze contribuiscono additivamente alla soglia energetica complessiva; applicando nuovamente $10 \log_{10}$ si ritorna ai decibel. Questo conclude la dimostrazione per il modello additivo di potenza adottato.

6. Allocazione dei bit. Per ciascuna sottobanda il codificatore confronta il livello del segnale con la soglia globale e stima quanta potenza di rumore di quantizzazione può essere introdotta senza superare il margine percettivo. I bit disponibili vengono distribuiti iterativamente alle sottobande più critiche fino al raggiungimento della bitrate target.

Il decoder non deve ricostruire la soglia di mascheramento: riceve l'informazione necessaria a decodificare allocazioni, scalefactor e campioni quantizzati. Questo permette agli encoder di migliorare il modello psicoacustico senza modificare il decoder o la sintassi del bitstream.

Layer I, II e III

I tre layer MPEG-1 Audio condividono parte dell'architettura ma introducono strumenti progressivamente più complessi.

Layer	campioni/frame	struttura spettrale	bitrate MPEG-1	uso storico
I	384	32 sottobande	fino a 448 kbit/s	DCC / sistemi semplici
II	1152	32 sottobande	fino a 384 kbit/s	broadcast / Video CD
III	1152	sottobande + MDCT	fino a 320 kbit/s	musica / distribuzione

Tabella 4.1. Caratteristiche principali dei tre layer MPEG-1 Audio. I limiti di bitrate si riferiscono alla sintassi MPEG-1 dei rispettivi layer.

Layer I raggruppa 12 campioni per ciascuna delle 32 sottobande:

$$12 \cdot 32 = 384$$

campioni PCM per frame.

Dimostrazione. Ogni istante di uscita del banco produce 32 campioni di sottobanda, uno per banda. Raggruppando 12 istanti si rappresentano $12 \cdot 32 = 384$ campioni equivalenti della sorgente PCM. Questo conclude la dimostrazione.

Layer II usa gruppi più lunghi e una codifica più flessibile degli scalefactor e dell'allocazione. Un frame rappresenta 1152 campioni PCM. Layer III mantiene 1152 campioni per frame ma divide il lavoro in due granule da 576 campioni per canale e aggiunge una struttura ibrida: all'uscita del banco di 32 sottobande applica MDCT per aumentare la risoluzione spettrale.

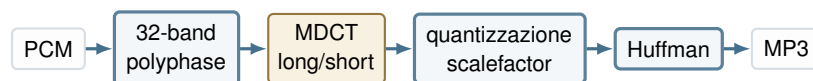


Figura 4.14. Catena ibrida di MPEG-1 Layer III: il banco di sottobande è seguito da MDCT e codifica entropica.

La MDCT usa finestre lunghe sui segnali stazionari e può passare a finestre corte in presenza di transitori, riducendo il pre-echo. Il joint stereo sfrutta ridondanza fra i canali mediante strumenti come mid/side stereo e intensity stereo quando appropriato.

Layer III introduce inoltre un *bit reservoir*: una parte dei bit non utilizzati da un frame semplice può essere resa disponibile a frame successivi più complessi. Il frame conserva una

bitrate nominale, ma la quantità di dati spettrali effettivamente associata a esso può quindi attingere a dati collocati fisicamente prima nel bitstream. Ciò migliora la qualità a bitrate costante ma rende più complesso l'accesso casuale e la gestione di errori locali.

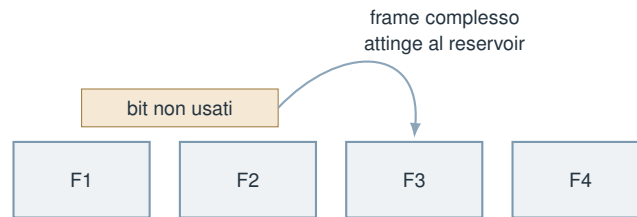


Figura 4.15. Principio del bit reservoir di Layer III: la capacità non utilizzata immediatamente può aiutare frame successivi più complessi.

Header e sincronizzazione

Ogni frame MPEG Audio inizia con una parola di sincronismo e contiene campi che identificano versione, layer, protezione CRC opzionale, indice di bitrate, frequenza di campionamento, padding e modalità dei canali. Le dimensioni dei campi sono definite dalla sintassi; il payload successivo dipende dal layer.

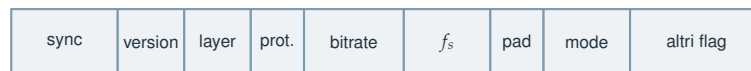


Figura 4.16. Campi concettuali dell'header di un frame MPEG Audio. Le larghezze grafiche non sono proporzionali al numero di bit.

La ripetizione degli header facilita l'aggancio a un flusso iniziato a metà e limita l'effetto di una perdita di sincronizzazione. La protezione CRC non rende però il codec un sistema di correzione degli errori: può rilevare determinate corruzioni, mentre il recupero dipende dal trasporto e dalle strategie del decoder.

Prestazioni e implementazione

Lo standard definisce il decoder conforme e la sintassi, ma lascia ampio spazio al codificatore: modello psicoacustico, ricerca dei parametri, strategie di joint stereo e allocazione dei bit possono essere migliorati senza cambiare il formato. Due encoder MP3 alla stessa bitrate possono quindi produrre qualità differenti.

Non è corretto associare a ogni layer un singolo MOS o una singola "complessità relativa" valida in generale. Tali valori dipendono da implementazione, contenuto, piattaforma e metodologia di ascolto. Il confronto corretto usa test controllati e, quando possibile, curve di qualità rispetto alla bitrate per più classi di contenuto.

4.1.7 MPEG-2

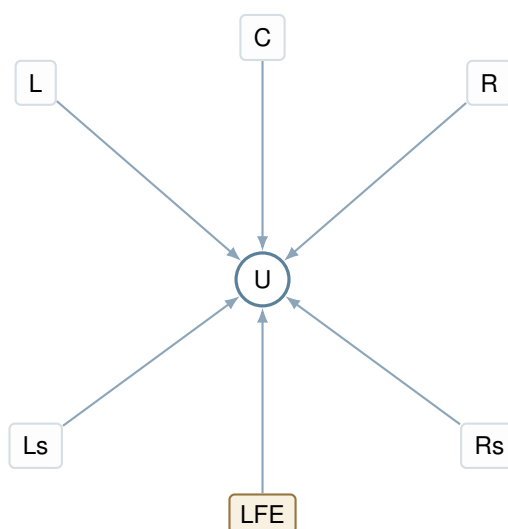
MPEG-2 Audio estende il quadro MPEG-1 in più direzioni che è utile separare. Le estensioni backward-compatible della famiglia Layer I/II introducono, fra l'altro, multicanale e frequenze di campionamento più basse mantenendo la possibilità per un decoder MPEG-1 di ricavare una parte compatibile del programma. MPEG-2 definisce inoltre Advanced Audio Coding (AAC), architettura non semplicemente retrocompatibile con i Layer I/II/III e progettata con strumenti differenti.

AAC viene quindi trattato nella sezione successiva come una nuova famiglia di codifica, non come una modalità accessoria del vecchio bitstream MPEG-1.

Audio multicanale 5.1

La notazione 5.1 indica cinque canali principali a banda piena e un canale Low-Frequency Effects (LFE) a banda limitata. Una disposizione tipica comprende i canali Left, Center e Right nella parte frontale e due canali surround laterali/posteriori. Il “.1” non indica un sesto canale equivalente agli altri: il canale LFE è destinato a effetti a bassa frequenza e richiede una frazione minore della banda complessiva.

È importante distinguere il canale LFE dalla gestione dei bassi del sistema di riproduzione. Il crossover che inoltra al subwoofer le basse frequenze provenienti dai cinque canali principali è una funzione di *bass management*; non definisce la banda del canale LFE. Nei workflow cinematografici e home-theater il contenuto LFE è normalmente limitato alla regione inferiore dello spettro, tipicamente fino all'ordine di 120 Hz, mentre la frequenza di crossover del sistema può essere scelta diversamente, spesso intorno a 80 Hz.



il canale LFE è il “.1” della configurazione

Figura 4.17. Disposizione concettuale 5.1: tre canali frontali, due surround e un canale LFE a banda limitata. La geometria reale dipende dalla raccomandazione e dall'ambiente di ascolto.

MPEG-2 ha introdotto estensioni multicanale rispetto all'audio MPEG-1. AAC appartiene alla successiva evoluzione MPEG-2 Advanced Audio Coding e viene trattato separatamente perché la sua architettura non è una semplice estensione retrocompatibile dei Layer I/II/III.

4.1.8 AAC

Advanced Audio Coding (AAC) nasce per superare i limiti dei codec MPEG-1 audio, in particolare quando si richiedono buona qualità a bitrate più basse, supporto multicanale e maggiore flessibilità. Il principio resta quello della codifica percettiva: si cerca una rappresentazione nel dominio trasformato nella quale il rumore di quantizzazione possa essere controllato in funzione della percezione.

La differenza rispetto a MP3 non è quindi l'abbandono del modello psicoacustico, ma il raffinamento dell'intera catena. AAC usa una trasformazione a coseno modificata, finestre di diversa lunghezza per adattarsi a segnali stazionari o transitori, strumenti di predizione e tecniche di codifica stereo più flessibili. La scelta della finestra è importante: blocchi lunghi aumentano l'efficienza in frequenza, mentre blocchi corti limitano la diffusione temporale degli errori di quantizzazione in presenza di attacchi rapidi.

Questa è la grande novità concettuale rispetto ai primi codec percettivi: il codificatore non

usa un'unica struttura rigida, ma seleziona strumenti diversi in funzione della natura locale del segnale. Le estensioni successive della famiglia AAC aggiungono altri strumenti, come la ricostruzione parametrica di porzioni dello spettro o della spazialità, ma il principio di base resta la combinazione fra modello percettivo, trasformazione adattiva e quantizzazione.

AAC è particolarmente adatto a distribuzione, broadcasting e streaming dove l'efficienza e la qualità hanno priorità maggiore rispetto alla latenza minima assoluta. Per applicazioni fortemente interattive il ritardo introdotto dalla struttura a blocchi diventa invece un parametro progettuale centrale.

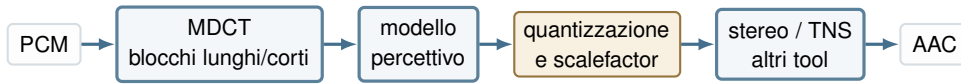


Figura 4.18. Catena concettuale AAC. Il passaggio fra finestre lunghe e corte bilancia risoluzione spettrale e controllo del pre-echo sui transitori.

4.1.9 Opus

Opus affronta un problema diverso: utilizzare un unico codec per segnali che vanno dalla voce alla musica, mantenendo una latenza sufficientemente bassa per la comunicazione interattiva. L'idea è combinare due famiglie di strumenti. La parte SILK deriva dalla codifica predittiva della voce; la parte CELT utilizza una codifica a trasformata adatta a segnali musicali e a banda larga. Il codec può operare in modalità prevalentemente predittiva, prevalentemente a trasformata o ibrida.

Il passaggio rispetto ai codec precedenti è importante. Un codec telefonico classico ottimizza quasi esclusivamente la voce; un codec musicale privilegia efficienza spettrale e qualità ma può accettare ritardi maggiori. Opus cerca invece di coprire entrambi gli estremi e di passare da uno all'altro senza cambiare formato di trasporto.

La durata del frame ha un effetto diretto sulla latenza minima di accumulo. Se un frame contiene N campioni di un segnale campionato a frequenza f_s , la sua durata è

$$T_f = \frac{N}{f_s}.$$

Dimostrazione. Una frequenza di campionamento f_s produce f_s campioni in un secondo. Il tempo associato a un singolo intervallo di campionamento è quindi $1/f_s$ secondi. Un blocco di N campioni rappresenta N intervalli di campionamento e quindi una durata

$$T_f = N \frac{1}{f_s} = \frac{N}{f_s}.$$

Questo conclude la dimostrazione.

La relazione mostra il compromesso: ridurre N riduce il tempo che il codificatore deve rappresentare in un singolo frame e può abbassare la latenza di pacchettizzazione; blocchi più lunghi offrono invece più contesto e possono migliorare l'efficienza di compressione. A questa latenza vanno poi aggiunti look-ahead, elaborazione, rete, buffering e decodifica.

Opus integra inoltre strumenti utili quando il canale è instabile: bitrate variabile, adattamento della banda audio, packet-loss concealment e modalità di ridondanza/FEC utilizzabili in funzione dell'applicazione. Per questo il codec è particolarmente adatto a VoIP, videoconferenza e applicazioni WebRTC, pur essendo utilizzabile anche per streaming musicale.

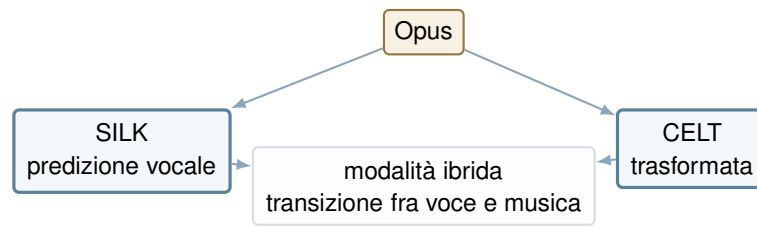


Figura 4.19. Architettura concettuale di Opus: strumenti predittivi e a trasformata possono essere usati separatamente o insieme in funzione di banda, bitrate e contenuto.

4.1.10 EVS e LC3: altri obiettivi moderni

Enhanced Voice Services (EVS), sviluppato in ambito 3GPP, mostra come la distinzione storica fra “codec vocale” e “codec musicale” sia diventata meno netta. EVS copre voce a banda stretta, wideband e super-wideband, ma include anche modalità capaci di trattare contenuto misto e musica. Combina strumenti predittivi e trasformati, adattando la strategia al segnale e alle condizioni del servizio mobile.

Low Complexity Communication Codec (LC3) per Bluetooth LE Audio ottimizza invece un altro insieme di vincoli: bassa complessità, consumo energetico contenuto, frame brevi e robustezza sufficiente per dispositivi alimentati a batteria. Il confronto con AAC o Opus mostra ancora una volta che un codec non si giudica soltanto dalla curva rate–distortion: energia, latenza, memoria e piattaforma di implementazione fanno parte delle specifiche di progetto.

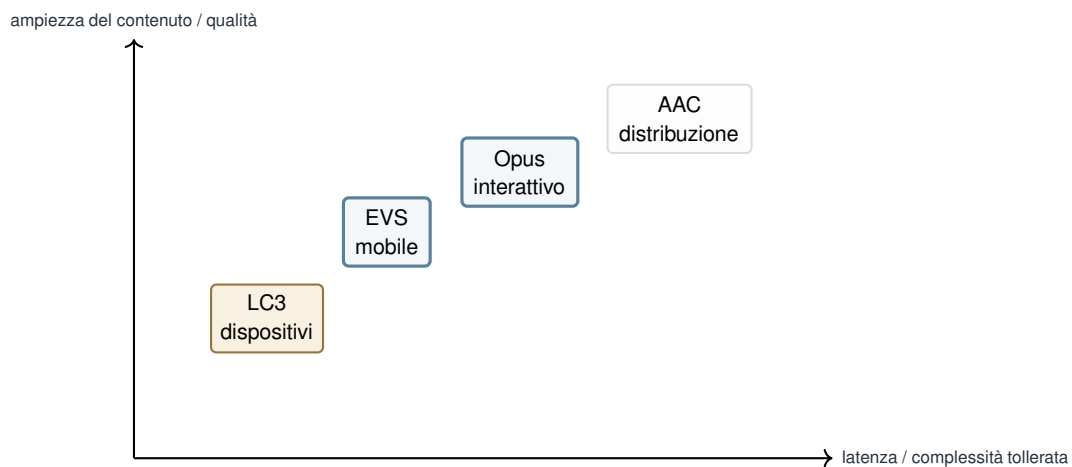


Figura 4.20. Mappa qualitativa degli obiettivi di alcuni codec moderni. Le coordinate non sono misure normative: servono a evidenziare che i vincoli applicativi sono differenti.

4.1.11 Dolby

AC-1 e AC-2 vengono mantenuti come passaggi storici utili a comprendere come sia evoluta l’allocazione percettiva dei bit; AC-3 resta invece un riferimento importante per l’audio multicanale distribuito.

I modelli psicoacustici di vari codificatori MPEG controllano l’accuratezza di quantizzazione di ciascuna sottobanda allocando un corrispondente numero di bit. Tale informazione deve essere disponibile al decoder. In questi schemi, come riportato in figura 4.21, il modello psicoacustico può restare una scelta dell’encoder, mentre il bitstream trasporta le informazioni necessarie a ricostruire l’allocazione dei bit.

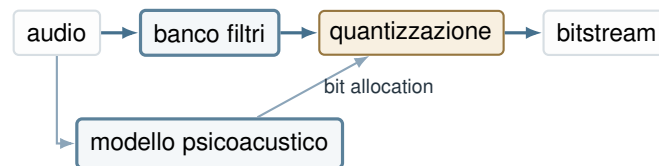


Figura 4.21. Codifica percettiva con modello psicoacustico presente soltanto nell'encoder: il bitstream deve trasportare le decisioni necessarie al decoder.

AC-1

Una variante rispetto a questo approccio è data dal codificatore Dolby denominato Acoustic Coder 1 (AC-1) in cui si utilizza una strategia di allocazione di bit fissa per ogni sottobanda. Tipicamente l'allocazione dei bit viene scelta sulla base di parametri medi; il degrado dovuto all'utilizzo di una quantizzazione non ottimale può essere controbilanciato dal non dover trasmettere le allocazioni dei bit. Questa soluzione è più semplice, ma meno adattiva, rispetto a una bit allocation guidata dinamicamente dal contenuto.



Figura 4.22. Principio storico di AC-1: allocazione di precisione prefissata fra sottobande, semplice ma poco adattiva al contenuto corrente.

In figura 4.22 è riportato uno schema del codificatore AC-1. Storicamente questa famiglia è stata usata in applicazioni di broadcast e distribuzione audio. Il punto didatticamente importante è la semplicità dell'allocazione prefissata: il decoder non deve ricevere o ricostruire una decisione di bit allocation variabile per ogni intervallo, ma l'encoder dispone di minore libertà nel seguire la struttura istantanea del segnale.

AC-2

Un codificatore Dolby che utilizza un'allocazione dinamica dei bit è l'AC-2. Nell'AC-2, a differenza che nell'MPEG, non viene immesso nel file di uscita l'informazione relativa al numero dei bit utilizzati per la quantizzazione, ma tale informazione viene ricavata al decodificatore applicando nuovamente il modello psicoacustico.

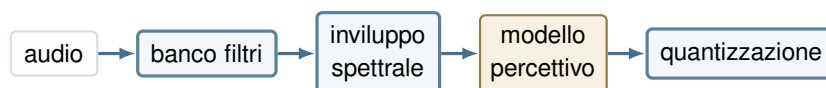


Figura 4.23. Principio storico di AC-2: il decoder può ricavare l'allocazione da una rappresentazione dell'inviluppo invece di ricevere esplicitamente ogni decisione.

Con questa strategia il decoder assume una parte maggiore della logica di ricostruzione. Il modello psicoacustico deve poter operare su campioni a valle dell'analisi dei filtri. Invece di utilizzare direttamente tali campioni, si interpone un blocco in grado di effettuare un'analisi dell'inviluppo spettrale.

Uno schema di questo codec è riportato in figura 4.23. L'idea riduce la side information esplicita, ma rende il decoder dipendente dalle regole con cui l'allocazione viene ricostruita. Se tali regole fanno parte della semantica necessaria alla decodifica, non possono essere cambiate unilateralmente dall'encoder senza perdere interoperabilità. È un compromesso opposto a quello dei sistemi in cui il modello percettivo resta interamente una scelta dell'encoder e il bitstream trasporta direttamente le decisioni che il decoder deve applicare.

AC-3

L'analisi dell'involuppo spettrale, presente in AC-2, produce una uscita con un primo livello di quantizzazione. Questo significa che il modello psicoacustico lavora su campioni già affetti da un rumore di quantizzazione. Per eliminare questo fenomeno è possibile utilizzare due diversi modelli psicoacustici: uno a valle dell'uscita dei filtri di analisi (psicoacustico F) ed uno a valle dell'uscita quantizzata del blocco di analisi dell'involuppo (psicoacustico B), come era già presente nel codificatore AC-2. In figura 4.24 viene riportato lo schema di questo sistema.

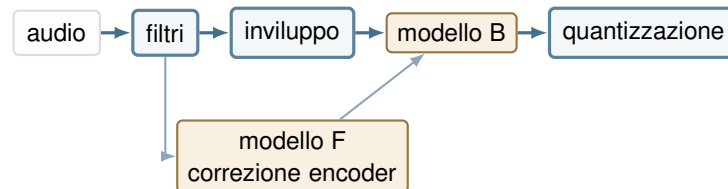


Figura 4.24. Idea storica descritta per AC-3: separare una parte del modello percettivo aggiornata nell'encoder dalla logica condivisa con il decoder.

Il blocco di elaborazione psicoacustica F riceve campioni in ingresso anche dall'analizzatore di involuppo ed è in grado di fornire delle informazioni aggiuntive al modello psicoacustico B, informazioni legate al degrado introdotto dalla quantizzazione presente nell'analisi dell'involuppo. Tali informazioni sono immesse nel bitstream e vengono utilizzate dal decoder.

Con questo schema è possibile modificare l'encoder per la parte di correzione introdotta dal blocco di modello psicoacustico F, senza dover cambiare il decoder. Si ottiene quindi un sistema parzialmente aggiornabile alla codifica, senza necessità di modifiche alla decodifica. Un esempio storico di questa evoluzione è Dolby AC-3.

4.1.12 Audio spaziale: canali, oggetti e scene

La configurazione 5.1 descrive una scena mediante canali associati a posizioni di riproduzione prefissate. Le architetture moderne possono separare maggiormente il contenuto dalla geometria finale del sistema di altoparlanti.

Tre modelli concettuali sono particolarmente importanti:

- **channel-based:** il contenuto è già organizzato in canali destinati a posizioni note, come stereo, 5.1 o 7.1;
- **object-based:** una sorgente sonora viene accompagnata da metadati di posizione, traiettoria, dimensione o priorità; il renderer decide come distribuirla sugli altoparlanti disponibili;
- **scene-based:** la scena acustica viene rappresentata come campo sonoro e successivamente renderizzata; Ambisonics e le rappresentazioni Higher-Order Ambisonics appartengono a questa famiglia.

Dolby Atmos è un esempio noto di ecosistema che combina bed multicanale e oggetti audio. MPEG-H 3D Audio definisce invece strumenti standard per contenuti channel-based, object-based e scene-based, includendo possibilità di rendering e personalizzazione. Il principio comune è che la scena non viene più descritta soltanto come un vettore fisso di canali.

Questa evoluzione è particolarmente importante per cuffie, dispositivi mobili, televisori, home theater e sistemi immersivi: lo stesso contenuto può essere renderizzato su layout differenti, eventualmente applicando binauralizzazione o virtualizzazione. La qualità dipende quindi non solo dal codec, ma anche dal renderer, dai metadati e dal modello acustico del

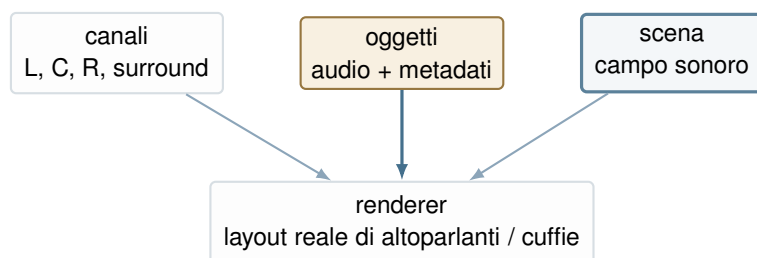


Figura 4.25. Tre modi di descrivere audio spaziale. Nel modello object-based e scene-based il rendering finale può adattarsi al dispositivo di riproduzione.

dispositivo finale.

4.2 Video

4.2.1 MJPEG: il caso senza predizione temporale

Motion JPEG comprime ogni frame indipendentemente con una tecnica di tipo JPEG. È il caso limite più semplice della compressione video: si sfrutta la ridondanza spaziale all'interno del frame ma non la correlazione temporale fra frame consecutivi.

La scelta ha vantaggi reali. Ogni frame è random-access, una perdita non propaga il proprio effetto a frame futuri e la complessità del decoder è contenuta. Per contro, una zona dello sfondo identica in centinaia di frame viene ricodificata centinaia di volte. Non esiste un rapporto di compressione “tipico” universale: dipende da qualità, contenuto, campionamento e implementazione JPEG.

MJPEG resta quindi utile in acquisizione, editing, sistemi embedded e applicazioni nelle quali semplicità, accesso indipendente e latenza valgono più dell'efficienza massima.

4.2.2 Ridondanza temporale e predizione

In molte sequenze video una grande frazione dell'immagine cambia poco fra frame vicini. Una telecamera ferma osserva sfondi quasi invariati; una panoramica sposta coerentemente vaste regioni; un oggetto rigido produce movimenti localmente simili. Il codec cerca quindi di costruire una predizione del frame corrente a partire da frame già ricostruiti e trasmette soltanto ciò che serve a descrivere la predizione e il residuo.

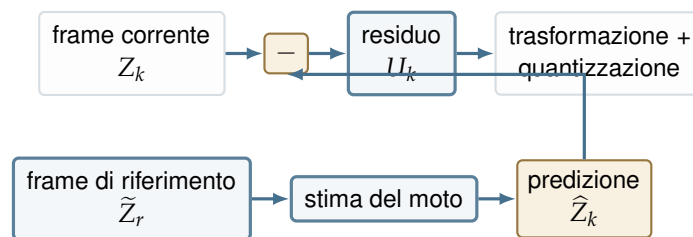


Figura 4.26. Principio della codifica video ibrida: il frame corrente viene predetto da riferimenti ricostruiti e si codifica il residuo fra originale e predizione.

È essenziale usare come riferimento la stessa immagine ricostruita che sarà disponibile al decoder. Se l'encoder predicesse dai campioni originali mentre il decoder disponesse soltanto dei campioni quantizzati, i due stati divergerebbero e l'errore si accumulerebbe nel tempo. Per questo il loop di ricostruzione è presente anche nell'encoder.

Frame I, P e B

La terminologia storica distingue tre famiglie di immagini:

- un frame **I** è intra-coded e può essere decodificato senza dipendere da altri frame;
- un frame **P** usa uno o più riferimenti già disponibili nell'ordine di decodifica;
- un frame **B** usa predizioni bidirezionali o, nei codec moderni, combinazioni di riferimenti appartenenti a più direzioni temporali.

Queste categorie sono storicamente molto utili, ma i codec moderni rendono più generale il concetto di riferimento: un frame può usare liste di riferimenti multiple, e la relazione fra “prima/dopo nel tempo” e “prima/dopo nell'ordine di decodifica” non è sempre la semplice struttura MPEG-1.

Un insieme di frame organizzato attorno a punti di accesso intra viene spesso chiamato Group of Pictures (GOP). La distanza fra punti intra controlla un compromesso: GOP lunghi possono sfruttare più ridondanza temporale, ma aumentano dipendenza dal passato, tempo

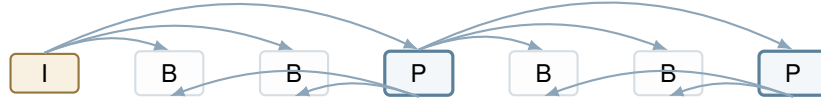


Figura 4.27. Esempio storico di struttura IBBPBBP. Le frecce rappresentano dipendenze di predizione, non l'ordine con cui i frame devono necessariamente essere trasmessi.

di accesso casuale e durata potenziale della propagazione di un errore.

Supponiamo che tutti i frame raw abbiano dimensione S e che i rapporti di compressione medi delle tre classi siano ρ_I , ρ_P e ρ_B . Un GOP con un frame I, n_P frame P e n_B frame B contiene

$$N = 1 + n_P + n_B$$

frame e ha rapporto complessivo

$$\rho_{\text{GOP}} = \frac{N}{\frac{1}{\rho_I} + \frac{n_P}{\rho_P} + \frac{n_B}{\rho_B}}.$$

Dimostrazione. Per definizione, se un frame di tipo t ha rapporto $\rho_t = S/C_t$, la sua dimensione compressa è $C_t = S/\rho_t$. La dimensione compressa del GOP è quindi

$$C_{\text{GOP}} = S \left(\frac{1}{\rho_I} + \frac{n_P}{\rho_P} + \frac{n_B}{\rho_B} \right).$$

La dimensione raw dei N frame è NS . Il rapporto raw/compresso è NS/C_{GOP} ; semplificando S si ottiene la formula dichiarata. Questo conclude la dimostrazione. Il risultato è una media armonica pesata, non la media aritmetica dei rapporti individuali.

Se, a puro titolo illustrativo, si assumono $\rho_I = 15$, $\rho_P = 25$ e $\rho_B = 40$, una sequenza IPPP ha

$$\rho_{\text{IPPP}} = \frac{4}{1/15 + 3/25} \simeq 21.43,$$

mentre una sequenza IBBPBB ha

$$\rho_{\text{IBBPBB}} = \frac{6}{1/15 + 1/25 + 4/40} \simeq 29.03.$$

Dimostrazione. Le due relazioni si ottengono sostituendo nella formula generale, rispettivamente, $(n_P, n_B) = (3, 0)$ e $(1, 4)$. I valori numerici dipendono dalle ipotesi sui tre rapporti e non sono caratteristiche normative dei tipi di frame. Questo conclude la dimostrazione.

Ordine di presentazione e ordine di decodifica

Un frame B che usa un riferimento futuro nel tempo non può essere decodificato finché quel riferimento non è stato ricevuto. Per questo l'ordine del bitstream può differire dall'ordine di presentazione. Per una sequenza di presentazione

$$I_0 \ B_1 \ B_2 \ P_3$$

un ordine di decodifica possibile è

$$I_0 \ P_3 \ B_1 \ B_2.$$

Il riordinamento introduce memoria e ritardo. In applicazioni a latenza molto bassa il codificatore può quindi ridurre o evitare riferimenti futuri, accettando una possibile perdita di efficienza.

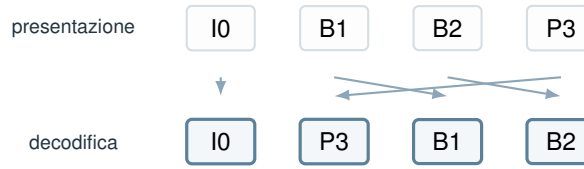


Figura 4.28. Esempio di riordinamento necessario quando i B-frame dipendono da un riferimento futuro nell'ordine di presentazione.

Stima del moto a blocchi

La predizione inter-frame più classica divide il frame corrente in blocchi e cerca, in un frame di riferimento, una regione simile. Se il blocco corrente ha origine (x, y) e dimensione $X \times Y$, un vettore candidato (i, j) può essere valutato mediante Mean Absolute Difference:

$$MAD(i, j; x, y) = \frac{1}{XY} \sum_{m=0}^{X-1} \sum_{n=0}^{Y-1} |Z_r(x + m + i, y + n + j) - Z_k(x + m, y + n)|.$$

Questa equazione definisce la metrica MAD. Il vettore scelto da una ricerca puramente distortiva è

$$(i^*, j^*) = \arg \min_{(i,j) \in S} MAD(i, j; x, y),$$

dove S è l'insieme delle posizioni candidate.

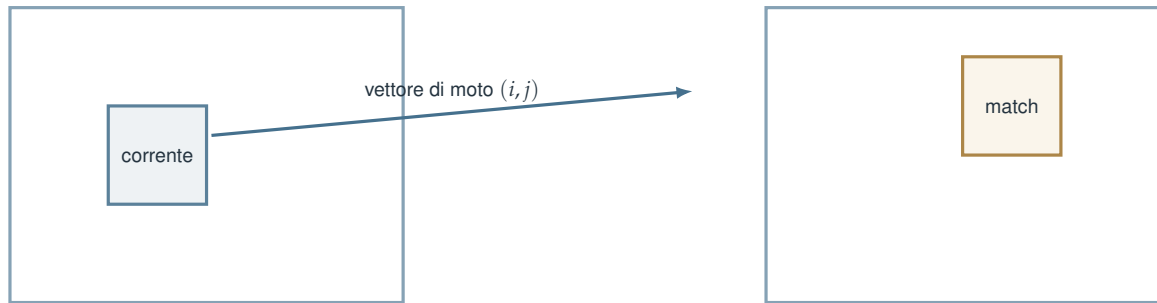


Figura 4.29. Block matching: il vettore di moto collega il blocco corrente alla regione scelta nel frame di riferimento.

La metrica non misura un movimento fisico dell'oggetto: identifica soltanto la posizione che fornisce la migliore predizione secondo il criterio scelto. Occlusioni, variazioni di illuminazione, deformazioni, rotazioni e zoom rendono il modello traslazionale solo una approssimazione.

Se la ricerca è limitata a spostamenti interi $-p \leq i, j \leq p$, il numero di candidati del full search è

$$N_{FS} = (2p + 1)^2.$$

Dimostrazione. Sono disponibili $2p + 1$ valori interi indipendenti per la coordinata orizzontale e altrettanti per quella verticale. Per il principio di moltiplicazione il numero di coppie è $(2p + 1)^2$. Questo conclude la dimostrazione.

Ricerche veloci

Il full search trova il minimo della metrica sull'intera finestra ma è costoso. Le ricerche veloci campionano solo parte delle posizioni e sfruttano l'ipotesi che la superficie di costo sia localmente regolare. Tale ipotesi non è sempre vera: un pattern ripetitivo può produrre molti minimi locali.

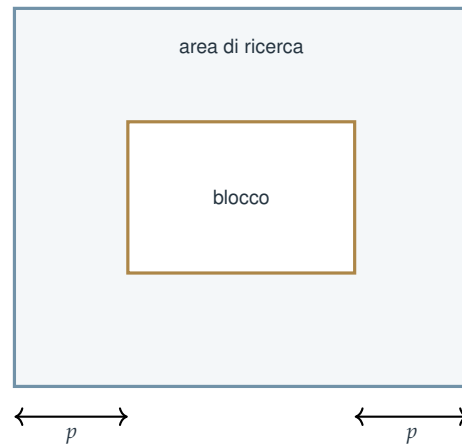


Figura 4.30. Area di ricerca per block matching: il blocco candidato può spostarsi entro un margine p in entrambe le direzioni.

Nel Three-Step Search si parte con un passo grande, si valutano il centro e otto punti circostanti, si sposta il centro sul candidato migliore e si dimezza il passo fino a raggiungere la granularità unitaria.

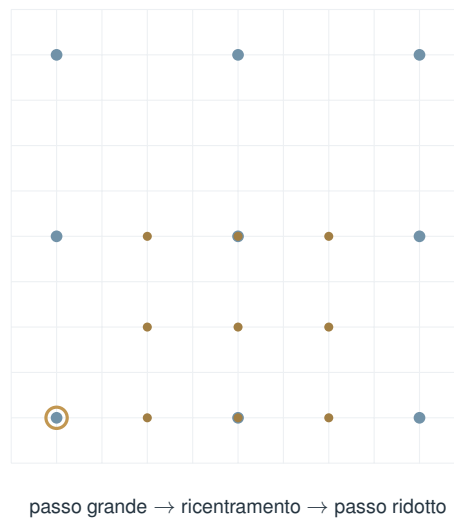


Figura 4.31. Principio di una ricerca multistep: si esplora una griglia rada e si raffina progressivamente intorno al candidato migliore.

Una ricerca logaritmica usa tipicamente una croce di candidati e riduce il passo quando il centro rimane il migliore. Una ricerca ortogonale separa invece raffinamento orizzontale e verticale. Nei codec e encoder moderni vengono usati pattern a diamante, esagono, ricerche gerarchiche multirisoluzione e molte euristiche basate sui vettori dei blocchi vicini.

La riduzione dei candidati non garantisce lo stesso vettore del full search. Per questo la qualità dell'encoder dipende anche dalla qualità della stima del moto, pur producendo un bitstream perfettamente conforme allo stesso standard.

Sub-pixel, partizioni e costo rate-distortion

Limitarsi a vettori di moto interi introduce un errore quando lo spostamento apparente è fra pixel. I codec moderni interpolano il frame di riferimento e valutano posizioni frazionarie, per esempio half-pel o quarter-pel. La maggiore precisione riduce spesso il residuo ma richiede filtri di interpolazione e bit aggiuntivi per descrivere il vettore.

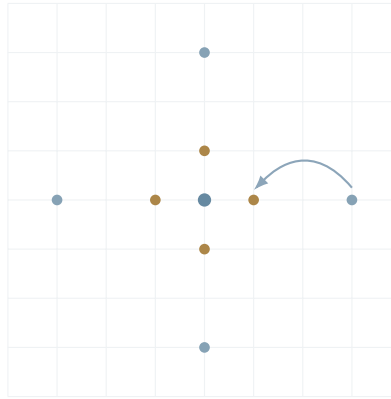


Figura 4.32. Schema di ricerca logaritmica/cross-search: il pattern viene ricentrato e il passo progressivamente ridotto.

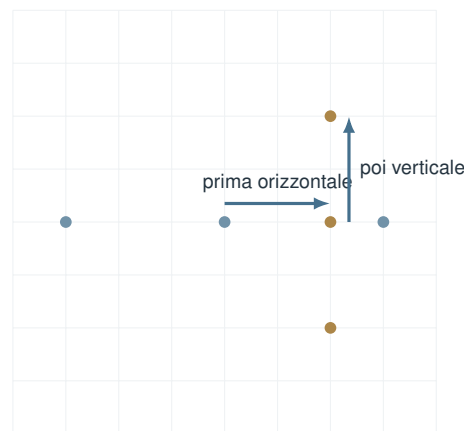


Figura 4.33. Ricerca ortogonale: l'esplorazione delle due coordinate è separata, riducendo il numero di candidati rispetto al full search.

Inoltre il moto non segue necessariamente una griglia di blocchi uniformi. Regioni omogenee possono essere predette con blocchi grandi; bordi e moto complesso richiedono partizioni più piccole. H.264, HEVC, VP9, AV1 e VVC aumentano progressivamente la flessibilità di partizionamento.

Un encoder moderno non sceglie in genere il vettore che minimizza la sola distorsione. Valuta un costo del tipo

$$J = D + \lambda R,$$

dove D è una misura di errore della predizione, R una stima dei bit necessari per vettore, modo e residuo, e $\lambda > 0$ pesa il costo della bitrate. Questa è una definizione di una comune funzione di costo rate-distortion, non una proprietà obbligatoria del bitstream.

Fra due candidati 1 e 2, il primo viene scelto quando

$$D_1 + \lambda R_1 < D_2 + \lambda R_2.$$

Dimostrazione. Per definizione l'encoder minimizza J . Sostituendo i due candidati nella funzione $J = D + \lambda R$, il candidato 1 ha costo inferiore se e solo se vale la disuguaglianza indicata. Questo conclude la dimostrazione.

Predizione bidirezionale

Un blocco B può usare due riferimenti. Indichiamo con $P_b(x, y)$ e $P_f(x, y)$ le predizioni ottenute da un riferimento precedente e da uno successivo nell'ordine temporale. Una predi-

zione composta semplice è

$$P_m(x, y) = \frac{P_b(x, y) + P_f(x, y)}{2}.$$

Questa è la definizione della media dei due predittori; codec moderni possono usare pesi differenti o modelli più complessi.

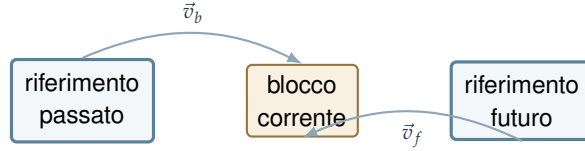


Figura 4.34. Predizione bidirezionale di un blocco mediante due vettori di moto e due immagini di riferimento.

Compensazione del moto e residuo

Scelto il vettore (i^*, j^*) , la predizione da un singolo riferimento è

$$\hat{Z}_k(x, y) = Z_r(x + i^*, y + j^*),$$

e il residuo è definito da

$$U_k(x, y) = Z_k(x, y) - \hat{Z}_k(x, y).$$

Il decoder ricostruisce

$$\tilde{Z}_k(x, y) = \hat{Z}_k(x, y) + \tilde{U}_k(x, y),$$

dove $\tilde{U}_k$ è il residuo dopo trasformazione, quantizzazione e ricostruzione.

Dimostrazione nel caso senza perdita. Se il residuo è ricostruito esattamente, $\tilde{U}_k = U_k = Z_k - \hat{Z}_k$. Sostituendo nella ricostruzione,

$$\tilde{Z}_k = \hat{Z}_k + Z_k - \hat{Z}_k = Z_k.$$

Questo conclude la dimostrazione. Nei codec lossy la differenza fra U_k e $\tilde{U}_k$ costituisce una componente dell'errore di ricostruzione.

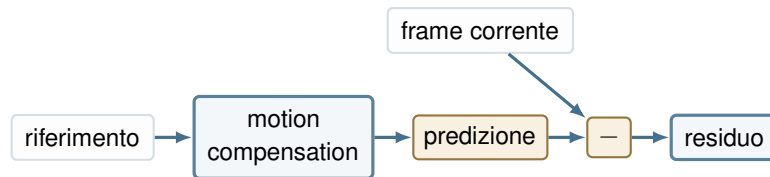


Figura 4.35. Stima e compensazione del moto: il vettore genera una predizione; ciò che resta da codificare è il residuo.

La trasformazione e la quantizzazione del residuo seguono lo stesso principio studiato per JPEG, ma con blocchi, trasformate e matrici di quantizzazione dipendenti dallo standard. La codifica video ibrida è proprio l'unione di predizione e codifica trasformata del residuo.

4.2.3 H.261: la struttura ibrida diventa standard

H.261 è uno dei primi standard internazionali nei quali la codifica video ibrida assume una forma completa e interoperabile. Fu progettato per videotelefonia e videoconferenza su canali digitali a capacità multipla di 64 kbit/s. La famiglia di bitrate nominali è quindi

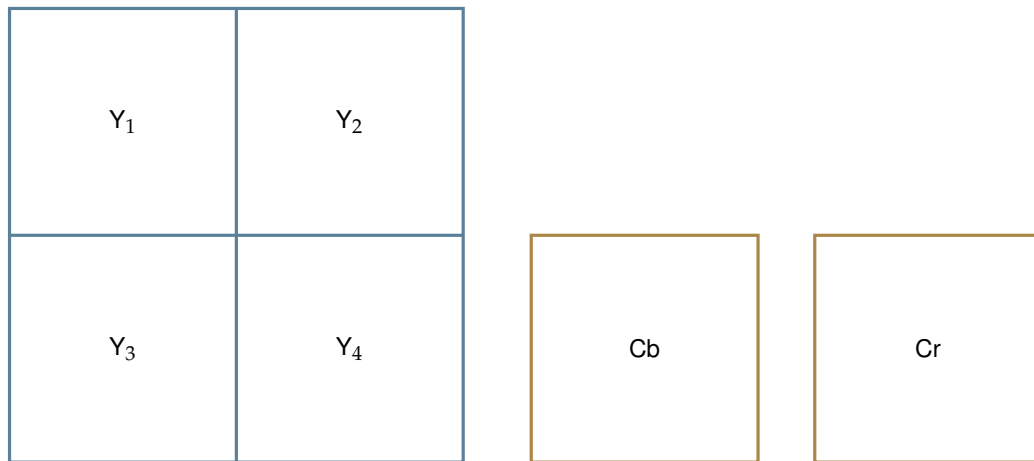
$$R = 64p \text{ kbit/s}, \quad p = 1, \dots, 30.$$

Questa relazione è un vincolo storico della specifica e non una legge della compressione: esprime il legame con i canali ISDN dell'epoca.

H.261 usa i formati QCIF e CIF, campionamento 4:2:0, predizione inter-frame, trasformazione DCT 8×8 , quantizzazione scalare e codici a lunghezza variabile. I frame possono essere intra oppure predetti; non esistono i B-frame bidirezionali della successiva famiglia MPEG.

Macroblocco e blocchi trasformati

Il macroblocco copre una regione di luma 16×16 e, nel 4:2:0, due regioni di crominanza 8×8 . La luma viene quindi suddivisa in quattro blocchi 8×8 , ai quali si aggiungono un blocco Cb e uno Cr: in totale sei blocchi trasformati.



macroblocco H.261: 4 blocchi Y + 1 Cb + 1 Cr

Figura 4.36. Organizzazione spaziale del macroblocco H.261 in 4:2:0. Ogni rettangolo etichettato rappresenta un blocco DCT 8×8 .

Per un macroblocco inter il codificatore può trasmettere un vettore di moto e il residuo trasformato; per un macroblocco intra i campioni sono codificati senza riferimento temporale. Il coded block pattern indica quali dei sei blocchi contengono coefficienti da trasmettere. Questa scelta evita di spendere bit per blocchi di residuo completamente nulli o non significativi.

La quantizzazione H.261 è più semplice di quella dei codec moderni e il parametro di quantizzazione può essere modificato per controllare la bitrate. I coefficienti quantizzati vengono serializzati e codificati mediante VLC definite dallo standard. H.261 contiene quindi già la catena concettuale completa

$$\text{predizione} \rightarrow \text{residuo} \rightarrow \text{DCT} \rightarrow \text{quantizzazione} \rightarrow \text{VLC}.$$

Questa espressione è uno schema funzionale, non una formula matematica.

Gerarchia del bitstream

La sintassi è gerarchica. Una Picture contiene Group of Blocks (GOB), ogni GOB contiene macroblocchi e ogni macroblocco contiene fino a sei blocchi trasformati.

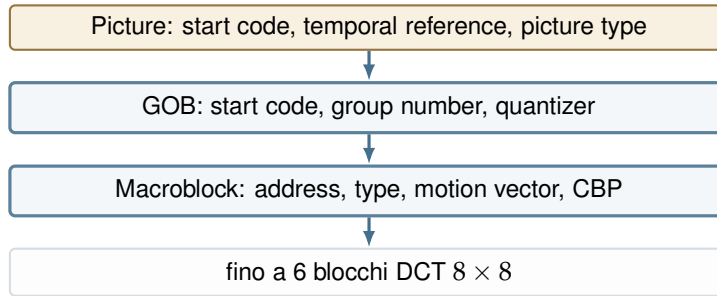


Figura 4.37. Gerarchia concettuale del bitstream H.261. I codici di start forniscono punti di risincronizzazione a granularità superiore al macroblocco.

QCIF usa luma 176×144 e crominanze 88×72 ; CIF usa luma 352×288 e crominanze 176×144 . Queste dimensioni sono definizioni dei formati. Poiché un macroblocco copre 16×16 campioni di luma, una riga di QCIF contiene 11 macroblocchi e una di CIF 22.

Dimostrazione. Per QCIF, $176/16 = 11$ e $144/16 = 9$, quindi la luma contiene $11 \cdot 9 = 99$ macroblocchi. Per CIF, $352/16 = 22$ e $288/16 = 18$, quindi i macroblocchi sono $22 \cdot 18 = 396$. Questo conclude la verifica geometrica.

Buffer e controllo della bitrate

Un encoder a bitrate vincolata deve adattare la quantizzazione alla complessità istantanea del video. Un modello utile considera un buffer di uscita. Sia O_k l'occupazione immediatamente prima di codificare il frame k , B_k il numero di bit prodotti da quel frame, R_T il rate con cui il canale svuota il buffer e T_k il tempo fino al frame successivo. Ignorando saturazione ai bordi, la dinamica è

$$O_{k+1} = O_k + B_k - R_T T_k.$$

Dimostrazione. Durante il ciclo k il codificatore aggiunge B_k bit al buffer. Nello stesso intervallo il canale ne rimuove $R_T T_k$. Il bilancio fra ingresso e uscita produce la relazione. Questo conclude la dimostrazione.

In regime stazionario medio si desidera

$$E[B_k] = R_T E[T_k].$$

Dimostrazione. Prendendo il valore atteso della dinamica e imponendo occupazione media costante, $E[O_{k+1} - O_k] = 0$. Ne segue $E[B_k] - R_T E[T_k] = 0$, da cui la relazione. Questo conclude la dimostrazione.

Se l'occupazione cresce troppo, l'encoder può aumentare il passo di quantizzazione, riducendo in media B_k ; se il buffer si svuota può ridurre il passo e spendere più bit. La legge precisa di controllo non è prescritta in modo universale dal principio della codifica: appartiene all'encoder e influenza qualità e stabilità della bitrate.

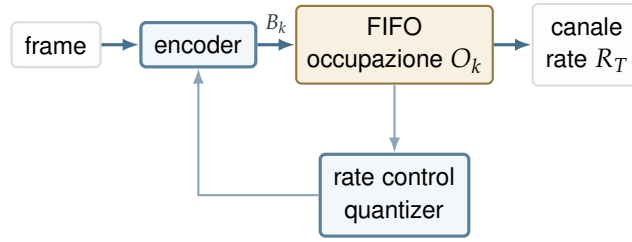


Figura 4.38. Rate control a retroazione: l'occupazione del buffer guida il parametro di quantizzazione per evitare underflow e overflow.

PSNR di sequenza e media dei PSNR

Se i frame hanno la stessa dimensione e MSE_k è il loro errore quadratico medio, l'MSE aggregato su T frame è

$$MSE_{seq} = \frac{1}{T} \sum_{k=1}^T MSE_k.$$

Il PSNR della sequenza è quindi

$$PSNR_{seq} = 10 \log_{10} \frac{MAX^2}{MSE_{seq}}.$$

Una misura differente è la media aritmetica dei PSNR dei singoli frame:

$$\overline{PSNR} = \frac{1}{T} \sum_{k=1}^T 10 \log_{10} \frac{MAX^2}{MSE_k}.$$

In generale

$$\overline{PSNR} \geq PSNR_{seq},$$

con uguaglianza se tutti gli MSE_k sono uguali.

Dimostrazione. Separando il termine costante,

$$\overline{PSNR} = 10 \log_{10} MAX^2 - \frac{10}{T} \sum_k \log_{10} MSE_k.$$

Poiché il logaritmo è concavo, la disuguaglianza di Jensen fornisce

$$\frac{1}{T} \sum_k \log_{10} MSE_k \leq \log_{10} \left(\frac{1}{T} \sum_k MSE_k \right).$$

Moltiplicando per -10 il verso si inverte; aggiungendo il termine $10 \log_{10} MAX^2$ si ottiene $\overline{PSNR} \geq PSNR_{seq}$. L'uguaglianza nella Jensen vale, per argomenti positivi e logaritmo strettamente concavo, quando tutti gli MSE_k coincidono. Questo conclude la dimostrazione.

4.2.4 H.263: più efficienza e più strumenti

H.263 nasce per videocomunicazione a bassa bitrate e mantiene l'architettura ibrida di H.261, migliorandone predizione, motion compensation e flessibilità. I formati di riferimento comprendono SQCIF 128×96 , QCIF 176×144 , CIF 352×288 , 4CIF 704×576 e 16CIF 1408×1152 per la luma. In 4:2:0 ciascuna crominanza dimezza entrambe le dimensioni.

Le prime versioni definiscono un core semplice e numerose modalità opzionali; le revisioni H.263+ e H.263++ aggiungono ulteriori annex. Questa architettura a strumenti opzionali anticipa la filosofia di profili e tool che diventerà comune nelle generazioni successive.

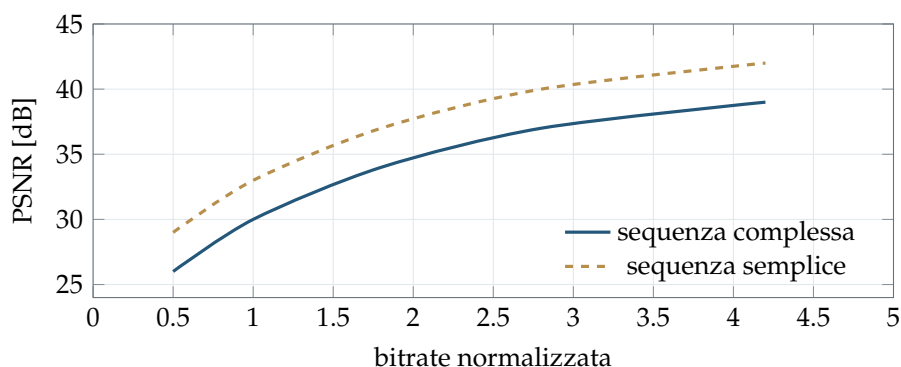


Figura 4.39. Curve rate–distortion illustrative per due contenuti differenti codificati dallo stesso sistema. Non esiste una curva di prestazione indipendente dalla sorgente e dall’encoder.

Miglioramenti della predizione

H.263 introduce vettori di moto a precisione sub-pixel e opzioni che estendono la ricerca oltre i limiti tradizionali del frame o del macroblocco, con regole di estrapolazione ai bordi. Advanced Prediction può usare vettori più granulari e sovrapposizione delle predizioni, riducendo errori lungo i confini dei blocchi. PB-frames permettono di rappresentare insieme informazione di tipo P e B con ritardo e struttura adatti alle applicazioni dell’epoca.

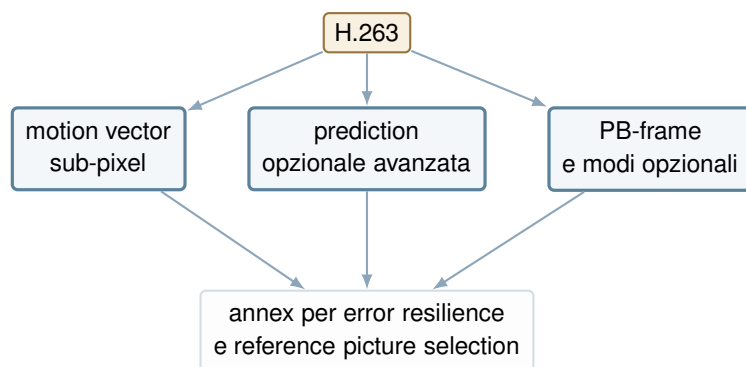


Figura 4.40. Principali direzioni di evoluzione da H.261 a H.263. Le funzionalità effettivamente disponibili dipendono dalla versione e dagli annex negoziati.

Resistenza agli errori e feedback

La predizione temporale rende economica la codifica ma crea dipendenza: un errore nel riferimento può contaminare frame futuri. H.263 sviluppa meccanismi di risincronizzazione, segmentazione indipendente e reference picture selection. Quando esiste un canale di ritorno, il decoder può segnalare regioni o immagini ricevute correttamente oppure corrotte; l’encoder può evitare riferimenti compromessi o forzare regioni intra.

Nel modello ACK si usano come riferimento soltanto immagini confermate; nel modello NACK si assumono valide le immagini non esplicitamente segnalate come errate. Il primo è più conservativo e può aumentare il ritardo prima di sfruttare un riferimento; il secondo riduce il feedback ma può usare un riferimento di cui l’encoder non ha ancora certezza positiva.

Il principio è ancora attuale: nei sistemi interattivi moderni feedback di rete e decoder possono richiedere refresh intra, reference frame specifici o riduzione temporanea della dipendenza predittiva.

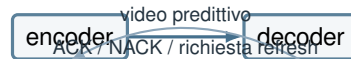


Figura 4.41. Feedback per limitare la propagazione degli errori: il decoder segnala lo stato dei riferimenti e l'encoder adatta la struttura predittiva.

Prestazioni: come confrontare H.261 e H.263

H.263 è stato progettato per migliorare l'efficienza soprattutto alle basse bitrate. Non esiste però un numero universale di kbit/s "risparmiati" rispetto a H.261. Il vantaggio dipende da sequenza, formato, frame rate, encoder, modalità opzionali e metrica.

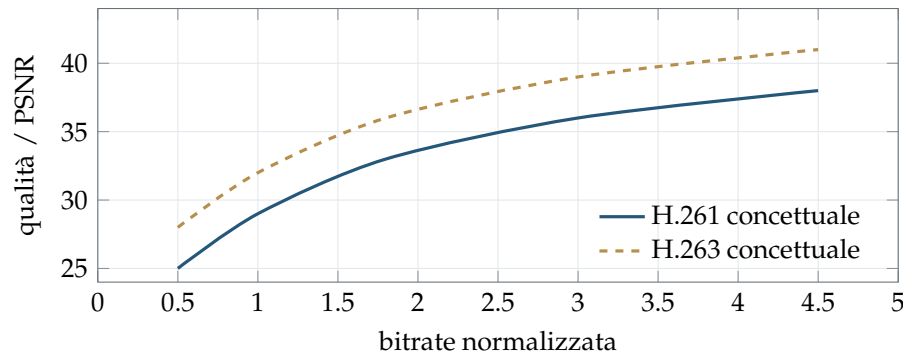


Figura 4.42. Confronto rate-distortion qualitativo: il codec più efficiente sposta la curva verso sinistra/alto. I punti non sono misure normative.

4.2.5 MPEG-1 e MPEG-2 Video

MPEG porta la codifica ibrida dalla videoconferenza alla distribuzione e alla memorizzazione. L'obiettivo cambia: diventa accettabile introdurre riordinamento e maggiore latenza per sfruttare B-frame, random access a GOP e strutture adatte a file, broadcast e supporti ottici.

Gerarchia MPEG Video

MPEG-1/2 Video usa una gerarchia di sintassi che separa parametri globali e dati locali:

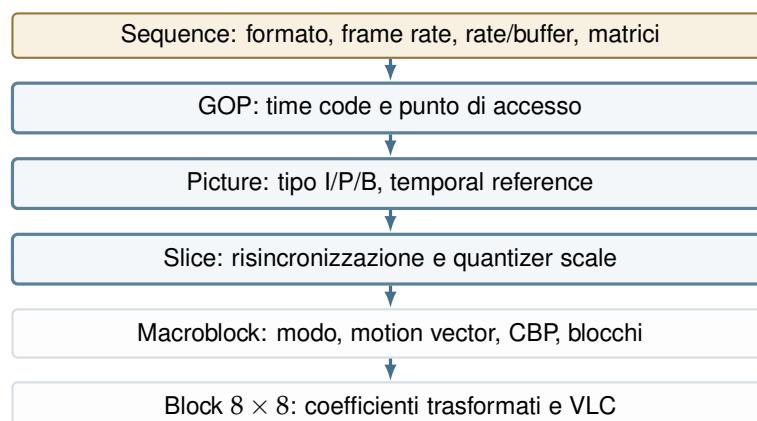


Figura 4.43. Gerarchia concettuale del bitstream MPEG-1/2 Video. Slice e start code offrono punti di accesso e risincronizzazione a granularità differenti.

A livello di macroblocco vengono segnalati modo intra/inter, quantizzazione, motion vector e coded block pattern. L'indirizzo del macroblocco è implicito nella scansione raster e

codificato mediante incrementi quando vengono saltate posizioni: non dipende dal colore dei pixel.

MPEG-1 Video è pensato storicamente per memorizzazione digitale a bitrate dell'ordine di quella di un CD-ROM 1x e usa comunemente formati SIF. Introduce I/P/B-frame, compensazione del moto half-pel, GOP e riordinamento di decodifica.

MPEG-2 Video estende l'architettura verso televisione digitale, interlacciamento, risoluzioni e bitrate maggiori, formati 4:2:2 e scalabilità. Profili e livelli separano gli strumenti ammessi dai limiti su risoluzione, frame rate, bitrate e buffer. Il concetto è più importante della memorizzazione di una tabella storica completa: un *profilo* seleziona un sottoinsieme di tool, un *livello* fissa vincoli quantitativi alla complessità del bitstream.

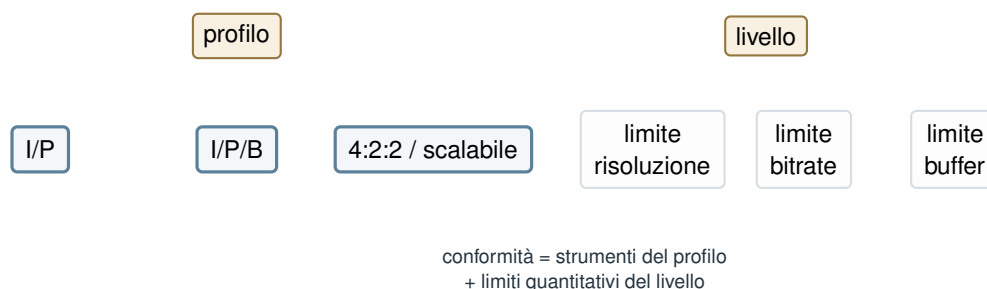


Figura 4.44. Separazione concettuale fra profili e livelli in MPEG-2. La combinazione permette decoder di classi diverse pur mantenendo interoperabilità dichiarata.

Interlacciamento in MPEG-2

MPEG-2 deve rappresentare efficientemente sia sorgenti progressive sia interlacciate. Può descrivere frame picture e field picture e adattare la predizione alla struttura dei campi. Il movimento fra due campi adiacenti, separati temporalmente, non coincide necessariamente con quello fra frame completi. Per questo la predizione field-based può essere più efficace su materiale realmente interlacciato.

Questa distinzione è un esempio di progettazione coerente con la sorgente: un codec non deve trattare come immagine progressiva una sequenza che contiene due istanti temporali intrecciati nelle righe.

4.2.6 MPEG-4 Visual e l'idea degli oggetti

MPEG-4 non è un singolo codec. La famiglia ISO/IEC 14496 comprende sistemi, contenitori, descrizione di scene, audio e più generazioni di video. La parte storica MPEG-4 Visual (Part 2) estende la codifica a blocchi e introduce anche strumenti orientati agli oggetti visivi. DivX e Xvid sono implementazioni note di profili MPEG-4 Part 2; H.264 è invece MPEG-4 Part 10/AVC e appartiene a una generazione differente.

Nel modello a oggetti una scena può essere descritta come composizione di Video Object Plane (VOP), ciascuno con forma, texture e moto. Una persona in primo piano può essere separata dallo sfondo e codificata come oggetto distinto; sprite e global motion compensation possono rappresentare sfondi o trasformazioni geometriche più ampie di un semplice vettore di blocco.

L'approccio è potente ma complesso: segmentare semanticamente gli oggetti non è necessario per la maggior parte della compressione video generica e non è diventato il modello dominante dei codec successivi. L'eredità più duratura è la separazione fra strumenti, profili, bitstream e composizione multimediale.

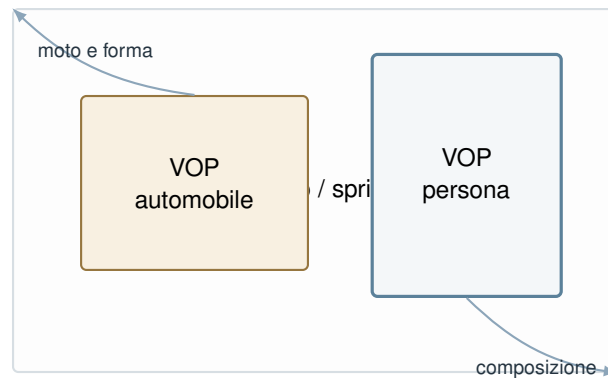


Figura 4.45. Idea object-based di MPEG-4 Visual: texture, forma e moto degli oggetti possono essere descritti separatamente e ricomposti nella scena.

Error resilience e RVLC

MPEG-4 Visual include strumenti opzionali di resilienza agli errori, fra cui resynchronization marker, data partitioning e Reversible Variable Length Codes (RVLC) in determinati profili e configurazioni. Non è corretto affermare che ogni bitstream MPEG-4 usi codici Huffman reversibili: si tratta di un tool opzionale.

Un RVLC è progettato in modo che la sequenza di parole possa essere analizzata anche a ritroso da un punto di risincronizzazione. Se un errore rende impossibile la decodifica in avanti, il decoder può cercare un marker successivo e recuperare all'indietro parte dei dati che altrimenti andrebbero persi.

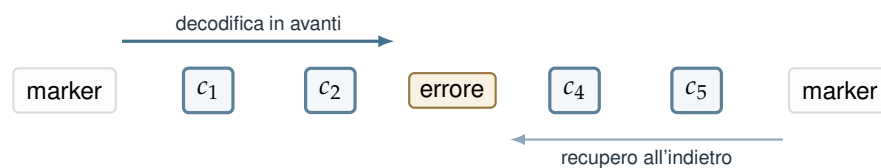


Figura 4.46. Principio di recupero con codici reversibili fra due marker di risincronizzazione. L'errore può limitare la perdita a una regione più piccola del packet video.

4.2.7 Confrontare le generazioni storiche

Le generazioni H.261, H.263, MPEG-1/2 e MPEG-4 Visual non possono essere ordinate mediante una singola curva universale. Encoder differenti, profili, contenuti e obiettivi possono cambiare il risultato. Il trend storico è però chiaro: cresce la flessibilità della predizione, aumenta la granularità degli strumenti e migliorano resilience e adattamento.

Il passaggio successivo, H.264/AVC, consolida questa tendenza e sposta il progetto verso un'architettura molto più adattiva, che diventerà il punto di partenza delle generazioni HEVC, VP9, AV1 e VVC.

4.2.8 Dall'AVC ai codec contemporanei

Dopo MPEG-4 Visual l'evoluzione dominante non segue più la segmentazione semantica in oggetti, ma rende sempre più flessibile il codec ibrido a blocchi. Crescono il numero di partizioni, riferimenti, predittori, trasformazioni e filtri; la codifica entropica diventa contestuale e il codificatore esplora uno spazio decisionale sempre più grande.

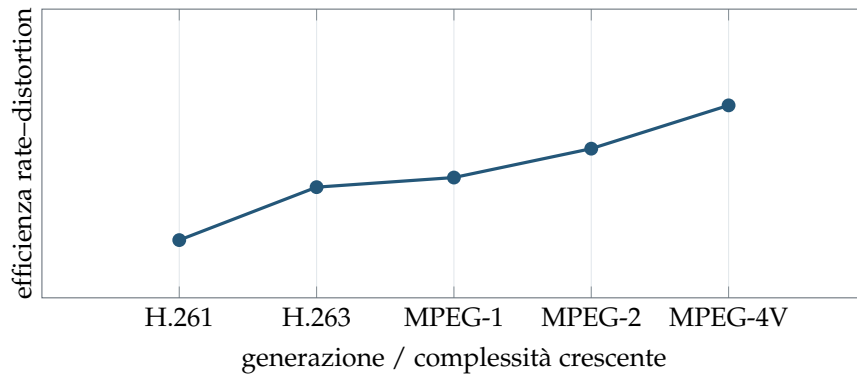


Figura 4.47. Evoluzione qualitativa delle generazioni storiche. La curva non rappresenta misure sperimentali: visualizza l'aumento generale di strumenti e flessibilità, che richiede anche maggiore complessità.

H.264/AVC

H.264/Advanced Video Coding, standardizzato congiuntamente da ITU-T e MPEG come H.264 e MPEG-4 Part 10, rappresenta un salto importante nella flessibilità del modello ibrido. Il macroblocco di luma è ancora 16×16 , ma può essere suddiviso in partizioni più piccole per la predizione inter; la predizione intra usa campioni già ricostruiti nello stesso frame; il moto può avere precisione frazionaria e può fare riferimento a più immagini.

macroblocco 16×16

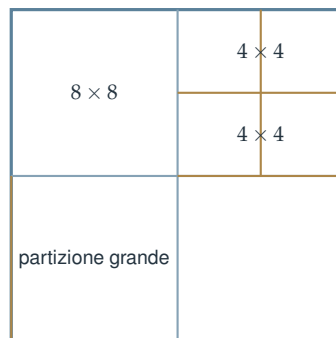


Figura 4.48. Esempio concettuale della maggiore flessibilità di partizione introdotta da H.264. Le partizioni di predizione e le trasformazioni non coincidono necessariamente con un'unica griglia rigida.

La predizione intra è una novità concettualmente importante rispetto ai codec precedenti: un blocco può essere stimato usando bordi e campioni dei blocchi vicini dello stesso frame. Il codec sfrutta quindi correlazione spaziale anche senza ricorrere a una DCT completamente indipendente come nel JPEG.

H.264 usa trasformazioni intere derivate da trasformate coseno-simili. L'uso di aritmetica intera evita discrepanze di inversione fra encoder e decoder dovute a implementazioni floating-point differenti. Dopo la ricostruzione, un filtro deblocking in-loop riduce le discontinuità ai bordi dei blocchi. Poiché il frame filtrato può essere usato come riferimento futuro, il filtro migliora non soltanto l'immagine mostrata ma anche la qualità delle predizioni successive.

La codifica entropica può usare CAVLC oppure, nei profili che lo prevedono, CABAC. CAVLC adatta tabelle di codici a lunghezza variabile alle statistiche locali dei coefficienti; CABAC binarizza gli eventi, seleziona modelli di contesto e usa codifica aritmetica binaria. CABAC è più complesso ma sfrutta meglio dipendenze statistiche locali.

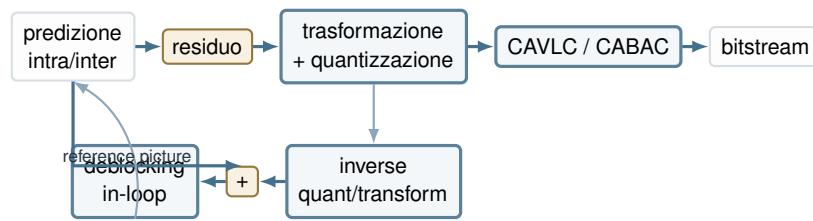


Figura 4.49. Loop di ricostruzione H.264: il riferimento usato per frame futuri è quello ricostruito e filtrato, identico a quello disponibile al decoder.

H.264 separa Video Coding Layer (VCL) e Network Abstraction Layer (NAL). Il VCL definisce come vengono rappresentati i campioni; la NAL organizza i dati in NAL unit utilizzabili in file e protocolli di trasporto. Parametri globali sono trasportati in parameter set, mentre slice e altri NAL VCL contengono i dati delle immagini.

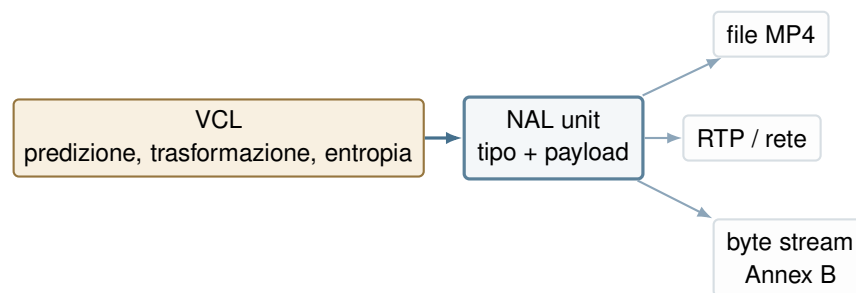


Figura 4.50. Separazione H.264 fra codifica video e astrazione di rete. Lo stesso contenuto VCL può essere impacchettato in sistemi di trasporto differenti.

H.265/HEVC

High Efficiency Video Coding (HEVC), H.265, è stato progettato per rendere sostenibile la crescita verso HD, UHD, profondità superiori e frame rate elevati. La novità strutturale più evidente è la Coding Tree Unit (CTU), che sostituisce il macroblocco rigido come unità principale di partizionamento.

Una CTU può essere suddivisa ricorsivamente in Coding Unit. Predizione e trasformazione sono ulteriormente descritte tramite strutture che consentono blocchi grandi nelle regioni uniformi e blocchi piccoli nei bordi o nei dettagli. Il principio è economico: la descrizione della partizione deve essere abbastanza flessibile da seguire la complessità locale senza spendere più segnalazione del vantaggio ottenuto.

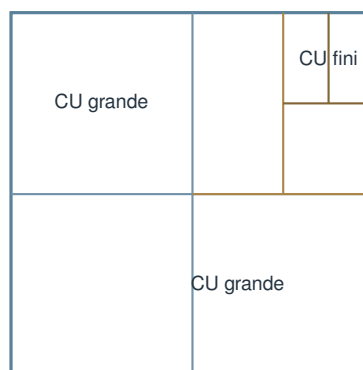


Figura 4.51. Partizionamento gerarchico concettuale di una CTU HEVC: regioni semplici restano grandi, regioni complesse vengono suddivise.

HEVC amplia i modi di predizione intra, i riferimenti e le partizioni inter, usa trasformazioni di più dimensioni e mantiene CABAC come sistema entropico. Alla catena in-loop aggiunge Sample Adaptive Offset (SAO), che classifica campioni e applica offset segnalati per ridurre errori sistematici dopo il deblocking.

Tile e Wavefront Parallel Processing dividono parte del lavoro per favorire implementazioni parallele. Anche questa scelta mostra un nuovo vincolo progettuale: l'efficienza del bitstream non basta; un codec per UHD deve essere implementabile su architetture multicore e hardware massivamente parallelo.

Per confrontare due codec A e B alla stessa qualità definiamo il risparmio relativo di bitrate di B rispetto ad A come

$$S_{B/A} = 1 - \frac{R_B}{R_A}.$$

Dimostrazione. La riduzione assoluta è $R_A - R_B$. Dividendo per la bitrate di riferimento $R_A > 0$,

$$\frac{R_A - R_B}{R_A} = 1 - \frac{R_B}{R_A}.$$

Questo conclude la dimostrazione. La formula definisce come calcolare un risparmio una volta misurate le bitrate; non assegna un valore universale a HEVC o ad alcun altro codec.

VP9

VP9 sviluppa in ambito Web molte idee parallele a HEVC: superblock partizionabili, numerosi predittori intra, più frame di riferimento, predizione inter con precisione frazionaria, trasformazioni adattive e codifica entropica contestuale. È stato fortemente associato al contenitore WebM e alla distribuzione adattativa su Internet.

VP9 permette un partizionamento quadtree del superblock e separa alcune probabilità del modello entropico dal semplice concetto di tabella VLC. Il decoder mantiene contesti che modificano le probabilità degli eventi in funzione della sintassi già osservata. Come in CABAC, l'idea è che la probabilità di un simbolo dipenda dal contesto locale e non sia una costante globale.

AV1

AV1, sviluppato da Alliance for Open Media, amplia ulteriormente la flessibilità. Il superblock può essere partizionato con forme più numerose; la predizione intra include modi direzionali e strumenti per texture; la predizione inter usa più riferimenti e combinazioni; trasformazioni e modelli entropici sono adattati al contesto.

La catena in-loop può includere deblocking, CDEF e loop restoration. CDEF riduce artefatti seguendo direzioni locali; restoration applica filtri per recuperare struttura dopo quantizzazione e predizione. Il film grain synthesis separa inoltre una componente stocastica dalla struttura visiva: invece di spendere molti bit per riprodurre esattamente il rumore/grana, l'encoder può trasmettere parametri di un modello che il decoder rigenera.

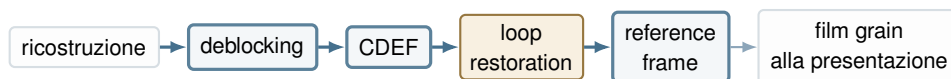


Figura 4.52. Catena concettuale dei filtri AV1. Il film grain può essere sintetizzato separatamente dalla reference picture usata per la predizione.

AVIF, trattato nel capitolo precedente, usa proprio strumenti intra di AV1 all'interno di HEIF. Questo collegamento mostra come un moderno codec video possa fornire anche una potente modalità di compressione per immagini statiche.

AV2

AOMedia Video 2 (AV2) è la generazione successiva di AOMedia. La specifica pubblica v1.0.0 è stata resa disponibile come Final Deliverable il 28 maggio 2026. È quindi corretto trattarla come nuova specifica, ma non attribuirle ancora la stessa maturità di ecosistema, diffusione e accelerazione hardware raggiunta da AV1.

AV2 prosegue la direzione generale delle generazioni precedenti: aumentare la capacità di modellare struttura spaziale e temporale, ampliare le opzioni di predizione/trasformazione e migliorare l'efficienza mantenendo un bitstream interoperabile. Per un testo del 2026 è importante conoscerne l'esistenza e lo stato normativo; sarebbe invece prematuro presentare rapporti di compressione o vantaggi energetici come proprietà universali prima di una base ampia di implementazioni e confronti indipendenti.

H.266/VVC

Versatile Video Coding (VVC), ITU-T H.266 e ISO/IEC 23090-3, è la generazione congiunta successiva a HEVC. Il 13 gennaio 2026 è stata approvata la Versione 4, attualmente in vigore. Gli obiettivi includono UHD, HDR/WCG, contenuti 360 gradi e applicazioni immersive, oltre al video tradizionale.

La struttura di partizionamento abbandona il solo quadtree e usa un Multi-Type Tree capace di split quaternari, binari e ternari. Questa flessibilità permette di seguire strutture allungate o dettagli locali senza dover creare necessariamente quattro blocchi quadrati a ogni passo.

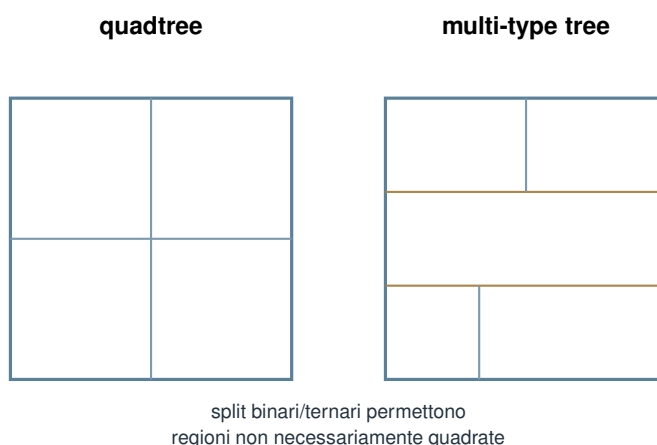


Figura 4.53. Confronto concettuale fra un partizionamento solo quadtree e il Multi-Type Tree di VVC.

VVC aggiunge e combina molti strumenti: predizione intra più ricca, affine e geometric partitioning per il moto, merge e motion vector prediction avanzati, trasformazioni adattive e filtri in-loop più sofisticati. Il principio fondamentale resta però quello già studiato: la maggiore efficienza nasce da un modello locale più espressivo e da una scelta encoder più complessa.

La Versione 4 del 2026 amplia inoltre il quadro di Supplemental Enhancement Information, includendo nuove informazioni di processo, metadati e segnali relativi anche a workflow con strumenti AI. Questi SEI non trasformano il core VVC in un codec neurale: trasportano informazioni supplementari associate al bitstream e alla pipeline.

Dal macroblocco al modello adattivo

La progressione storica può essere visualizzata attraverso la libertà di partizionamento:

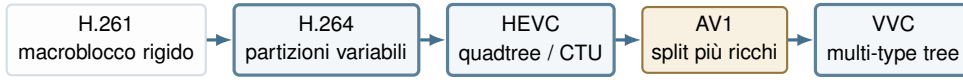


Figura 4.54. Evoluzione qualitativa della granularità di partizionamento. Maggiore flessibilità riduce la distorsione o la bitrate, ma aumenta il numero di decisioni dell'encoder.

Scalabilità: dal FGS ai layer moderni

Fine Granular Scalability (FGS) di MPEG-4 Visual è un passaggio storico utile perché rende esplicita l'idea di separare un base layer indispensabile da informazione di enhancement progressiva. Il base layer produce una qualità minima decodificabile; l'enhancement layer raffina la ricostruzione e può essere troncato in funzione della capacità disponibile.

Supponiamo che la capacità di rete soddisfi

$$B_{min} \leq B(t) \leq B_{max}$$

e che il base layer richieda

$$0 < B_{BL} \leq B_{min}.$$

La quota massima disponibile per l'enhancement è

$$B_{EL,max} = B_{max} - B_{BL}.$$

Una politica che non supera la capacità istantanea è

$$B_{EL}(t) = \max\{0, \min[B(t) - B_{BL}, B_{EL,max}]\}.$$

Ne segue

$$B_{BL} + B_{EL}(t) \leq B(t).$$

Dimostrazione. Poiché $B_{BL} \leq B_{min} \leq B(t)$, la quantità $B(t) - B_{BL}$ è non negativa. Quindi

$$B_{EL}(t) = \min\{B(t) - B_{BL}, B_{EL,max}\} \leq B(t) - B_{BL}.$$

Sommando B_{BL} ai due membri si ottiene la relazione. Questo conclude la dimostrazione.

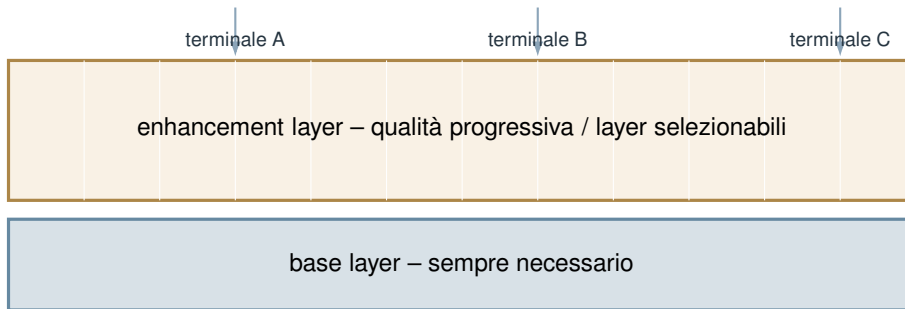


Figura 4.55. Principio di scalabilità: tutti ricevono il base layer, mentre quantità differenti di enhancement producono qualità o risoluzioni differenti.

Nei sistemi contemporanei la scalabilità può essere temporale, spaziale o qualitativa e può essere realizzata internamente al codec oppure mediante simulcast di rappresentazioni complete. Nella videoconferenza con SFU, layer scalabili o simulcast permettono al server di inoltrare a ogni ricevitore soltanto ciò che banda, layout e dispositivo possono utilizzare.

4.2.9 Metriche di qualità video moderne

PSNR resta utile perché è semplice, riproducibile e direttamente legato all'errore quadratico, ma non rappresenta da solo la qualità percepita. SSIM e MS-SSIM cercano di confrontare struttura, contrasto e luminanza, mentre VMAF combina più feature oggettive mediante un modello addestrato su giudizi soggettivi.

Una metrica appresa non va interpretata come misura fisica assoluta. Il punteggio dipende dalla versione del modello, dalla risoluzione, dal preprocessing e dalla classe di distorsioni per cui il modello è stato validato. Confrontare due codec richiede quindi di dichiarare sia il protocollo di codifica sia la metrica e la sua versione.

Per contenuti HDR, frame rate elevati, scaling e artefatti temporali è particolarmente importante usare più indicatori e, quando il confronto è critico, confermare i risultati con test soggettivi. Nessuna singola metrica sostituisce completamente la valutazione percettiva.

4.2.10 Come confrontare i codec video

Un confronto scientifico deve fissare almeno:

- sequenze e risoluzioni di prova;
- metrica di qualità e intervallo di bitrate;
- encoder, versione, preset e tool abilitati;
- latenza e struttura GOP consentite;
- numero di thread, hardware ed energia quando si misura la complessità;
- eventuale accelerazione hardware disponibile sul decoder.

Famiglia	Partizione	Strumenti principali	Focus / stato 2026
H.261	macroblocco rigido	VLC, DCT, moto intero	videoconferenza; storico
H.264	macroblocco e sotto-partizioni	intra/inter, CAVLC/CABAC, deblocking	general purpose; diffusissimo
HEVC VP9	CTU e quadtree superblock quadtree	CABAC, DBF, SAO, tile/WPP e predizione contestuale, loop filter	UHD/HDR; maturo Web; maturo
AV1	superblock e split ricchi	CDEF, restoration, multi-reference, grain synthesis	streaming/Web; maturo
VVC	quadtree + multi-type tree	CABAC evoluto, moto affine, filtri avanzati	UHD/immersive; in vigore
AV2	struttura di nuova generazione	strumenti predittivi e trasformativi evoluti	AOMedia; specifica 1.0 del 2026

Tabella 4.2. Confronto qualitativo delle generazioni. La tabella sintetizza l'architettura e non costituisce una graduatoria di efficienza.

L'evoluzione non cambia il problema fondamentale: predire il frame, rappresentare il residuo, quantizzare ciò che può essere perso e codificare efficientemente il risultato. Cambia la capacità del modello di adattarsi al contenuto e quindi la quantità di calcolo necessaria a scegliere fra le alternative. Il capitolo successivo sposta l'attenzione dal codec alla rete: anche il miglior bitstream perde valore se ritardo, jitter, perdita o congestione non vengono gestiti in modo coerente con il servizio.

5.1 Canale

Un sistema multimediale trasporta sorgenti con requisiti molto diversi. Testo e immagini statiche richiedono in genere ricostruzione esatta ma sono relativamente tolleranti alla latenza; audio e video interattivi accettano spesso una piccola perdita residua, ma sono molto più sensibili a ritardo, jitter e regolarità temporale. Lo stesso canale deve quindi essere valutato con più metriche contemporaneamente.

Per un pacchetto che attraversa H hop, il ritardo end-to-end può essere modellato come

$$D_{e2e} = D_{src} + D_{dst} + \sum_{h=1}^H (D_{q,h} + D_{tx,h} + D_{prop,h} + D_{proc,h}).$$

Questa è una decomposizione contabile: distingue elaborazione, accodamento, serializzazione e propagazione. Non implica che i termini siano statisticamente indipendenti.

Il termine di trasmissione di un pacchetto di L bit su una linea a rate R è

$$D_{tx} = \frac{L}{R}.$$

Dimostrazione. Per definizione il rate è il numero di bit trasmessi nell'unità di tempo, $R = L/D_{tx}$. Isolando D_{tx} si ottiene L/R . Questo conclude la dimostrazione.

Il ritardo di propagazione su un mezzo lungo d percorso a velocità v è

$$D_{prop} = \frac{d}{v}.$$

Dimostrazione. Per moto di propagazione a velocità media costante vale $v = d/D_{prop}$. Poiché $v > 0$, isolando il tempo si ottiene $D_{prop} = d/v$. Questo conclude la dimostrazione.

Il jitter non è un nuovo ritardo fisico: è la variabilità del ritardo da pacchetto a pacchetto. Se il ritardo totale è somma di variabili casuali D_h e i termini fossero indipendenti,

$$\text{Var}\left(\sum_h D_h\right) = \sum_h \text{Var}(D_h).$$

Dimostrazione. Per variabili con varianza finita,

$$\text{Var}\left(\sum_h D_h\right) = \sum_h \text{Var}(D_h) + 2 \sum_{i < j} \text{Cov}(D_i, D_j).$$

Se i termini sono indipendenti, tutte le covarianze sono nulle e resta la somma delle varianze. Questo conclude la dimostrazione.

Nelle reti reali l'indipendenza è solo un'approssimazione: flussi che condividono colli di bottiglia, burst comuni e meccanismi di controllo della congestione possono introdurre correlazione. Il modello va quindi usato come strumento analitico, non come proprietà universale di Internet.

La perdita di pacchetti può derivare da errori fisici, congestione, policy di coda, cambi di percorso o scadenza della deadline applicativa. Nelle reti radio molte corruzioni vengono intercettate o recuperate al link layer; ciò non elimina però le perdite viste dall'applicazione, che possono derivare anche da ritrasmissioni fallite o troppo tardive.

I pacchetti possono inoltre arrivare fuori ordine. TCP ricostruisce un flusso ordinato di byte e nasconde il riordino all'applicazione, ma può introdurre head-of-line blocking fra dati appartenenti allo stesso flusso. UDP lascia invece all'applicazione la gestione di sequenza, perdita e timing. QUIC mantiene affidabilità e ordinamento per stream indipendenti, riducendo il blocco fra stream ma non eliminando la congestione condivisa sulla connessione.

5.1.1 Buffering e playout

Un ricevitore multimediale può ritardare volontariamente la riproduzione e accumulare pacchetti in un playout buffer. Il buffer non “rimuove” il jitter dalla rete: trasforma variabilità di arrivo in latenza aggiuntiva e, se la deadline viene superata, in perdita applicativa.

Siano s_i l'istante di generazione del pacchetto i , r_i quello di ricezione e

$$Z_i = r_i - s_i$$

il ritardo di rete osservato. Una stima smussata del ritardo può essere aggiornata con

$$d_i = (1 - u)d_{i-1} + uZ_i, \quad 0 < u < 1.$$

Una stima smussata della deviazione assoluta può essere definita da

$$v_i = (1 - u)v_{i-1} + u|Z_i - d_i|.$$

Queste sono definizioni di medie mobili esponenziali. v_i non è una deviazione standard: è un indicatore robusto della variabilità.

Una semplice regola di playout è

$$p_i = s_i + d_i + Kv_i,$$

dove $K > 0$ controlla il margine. Il valore di K è un parametro di progetto e dipende dal compromesso desiderato fra latenza e probabilità di pacchetto tardivo.

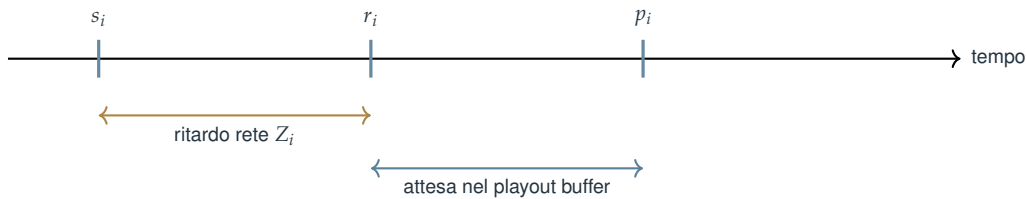


Figura 5.1. Il playout buffer aggiunge attesa dopo la ricezione per assorbire parte della variabilità del ritardo di rete.

Modello uniforme

Per studiare il compromesso in forma chiusa consideriamo un ritardo

$$Z \sim U[m - \Delta, m + \Delta]$$

e una deadline di playout D . Se un pacchetto che arriva prima della deadline attende fino a D , mentre uno tardivo viene riprodotto appena arriva, il ritardo effettivo è

$$Y = \max\{Z, D\}.$$

Questa è la definizione del modello.

Per la distribuzione uniforme,

$$E[Z] = m, \quad \text{Var}(Z) = \frac{\Delta^2}{3}.$$

Dimostrazione. La densità vale $1/(2\Delta)$ su $[m - \Delta, m + \Delta]$. Ponendo $u = z - m$,

$$E[Z] = m + \frac{1}{2\Delta} \int_{-\Delta}^{\Delta} u \, du = m.$$

Inoltre

$$\text{Var}(Z) = \frac{1}{2\Delta} \int_{-\Delta}^{\Delta} u^2 \, du = \frac{\Delta^2}{3}.$$

Questo conclude la dimostrazione.

Se si sceglie

$$D = m + \Delta,$$

tutti i pacchetti arrivano entro la deadline; pertanto

$$Y = D, \quad E[Y] = m + \Delta, \quad \text{Var}(Y) = 0.$$

Dimostrazione. Per ipotesi $Z \leq m + \Delta = D$ con probabilità uno. Quindi $\max\{Z, D\} = D$ sempre. Una variabile costante ha media pari alla costante e varianza nulla. Questo conclude la dimostrazione.

Il risultato dipende dal supporto limitato della distribuzione. Nelle reti reali una coda lunga di ritardi rende impossibile eliminare il jitter con una deadline finita senza accettare una probabilità di pacchetto tardivo.

Modello gaussiano

Consideriamo

$$Z \sim \mathcal{N}(m, \sigma^2), \quad D = m + \Delta, \quad t = \frac{\Delta}{\sigma}.$$

Definiamo la densità e la funzione di ripartizione della normale standard:

$$\varphi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}, \quad \Phi(t) = \int_{-\infty}^t \varphi(u) \, du,$$

e $Q(t) = 1 - \Phi(t)$.

La probabilità che il pacchetto superi la deadline è

$$P(Z > D) = Q(t).$$

Dimostrazione. Standardizzando,

$$P(Z > D) = P\left(\frac{Z - m}{\sigma} > \frac{D - m}{\sigma}\right) = P(U > t) = 1 - \Phi(t) = Q(t).$$

Questo conclude la dimostrazione.

Poiché $Y = \max\{Z, D\}$, la distribuzione di uscita contiene una massa puntuale in D di probabilità $\Phi(t)$ e conserva la coda gaussiana oltre D :

$$f_Y(y) = \Phi(t) \delta(y - D) + f_Z(y) \mathbf{1}_{\{y > D\}}.$$

Questa espressione descrive una distribuzione mista: non è una normale.

Scrivendo $Z = m + \sigma U$ con $U \sim \mathcal{N}(0, 1)$,

$$Y = m + \sigma \max\{U, t\}.$$

Usiamo le identità

$$\begin{aligned} \int_t^\infty u \varphi(u) du &= \varphi(t), \\ \int_t^\infty u^2 \varphi(u) du &= t\varphi(t) + Q(t). \end{aligned}$$

Dimostrazione delle identità. Poiché $\varphi'(u) = -u\varphi(u)$,

$$\int_t^\infty u \varphi(u) du = [-\varphi(u)]_t^\infty = \varphi(t).$$

Per il secondo integrale, integriamo per parti usando $u^2 \varphi(u) = -u\varphi'(u)$:

$$\int_t^\infty u^2 \varphi(u) du = [-u\varphi(u)]_t^\infty + \int_t^\infty \varphi(u) du = t\varphi(t) + Q(t).$$

Questo conclude la dimostrazione.

La media è

$$E[Y] = m + \sigma [t\Phi(t) + \varphi(t)].$$

Dimostrazione. La variabile $\max\{U, t\}$ vale t per $U \leq t$ e U per $U > t$:

$$\begin{aligned} E[\max\{U, t\}] &= t\Phi(t) + \int_t^\infty u \varphi(u) du \\ &= t\Phi(t) + \varphi(t). \end{aligned}$$

Moltiplicando per σ e aggiungendo m si ottiene la relazione. Questo conclude la dimostrazione.

Il secondo momento normalizzato è

$$E[\max\{U, t\}^2] = t^2\Phi(t) + t\varphi(t) + Q(t).$$

Da cui

$$\text{Var}(Y) = \sigma^2 [t^2\Phi(t) + t\varphi(t) + Q(t) - \{t\Phi(t) + \varphi(t)\}^2].$$

Dimostrazione. Il secondo momento si ottiene separando i due eventi $U \leq t$ e $U > t$ e usando la seconda identità integrale. La varianza di $m + \sigma V$ è $\sigma^2 \text{Var}(V)$; sottraendo il quadrato della media normalizzata si ottiene la relazione. Questo conclude la dimostrazione.

Se i pacchetti tardivi vengono scartati, $Q(t)$ non rappresenta più jitter residuo ma una componente della probabilità di perdita applicativa. Il buffering realizza quindi un compromesso a tre vie fra latenza, jitter e perdita.

5.1.2 Perdita, ritrasmissione e FEC

Una perdita applicativa può essere affrontata in tre modi principali: ritrasmettere l'informazione, aggiungere ridondanza preventiva oppure nascondere l'errore usando informazione già disponibile al decoder. Le tre strategie non sono alternative assolute e nei sistemi reali possono essere combinate.

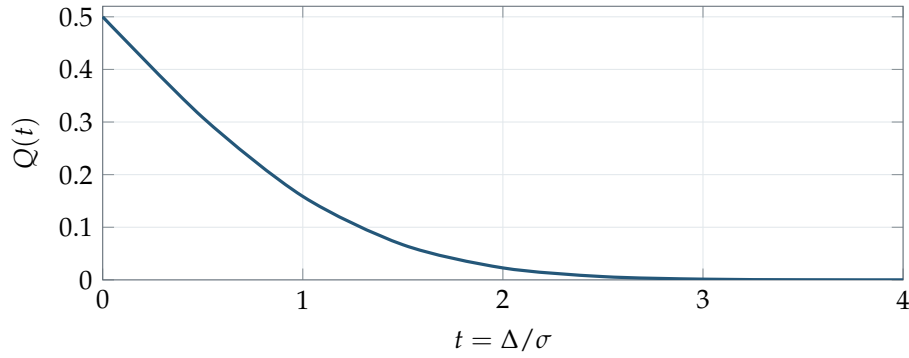


Figura 5.2. Probabilità di pacchetto oltre deadline nel modello gaussiano. Aumentare il margine di playout riduce rapidamente la coda tardiva, al prezzo di maggiore latenza.

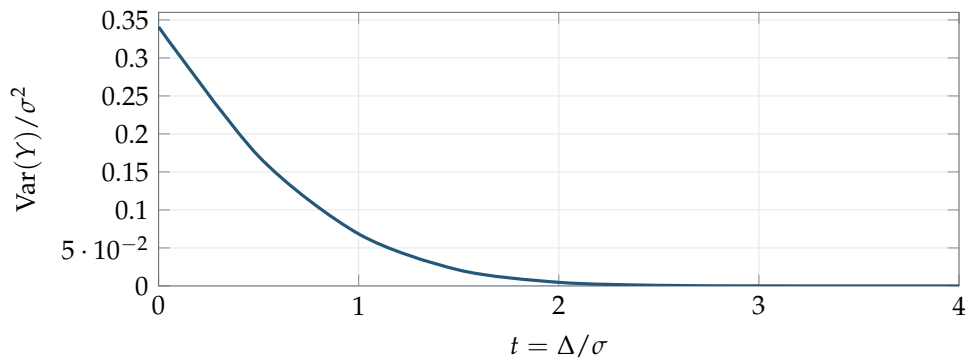


Figura 5.3. Varianza residua normalizzata del ritardo di uscita nel modello gaussiano.

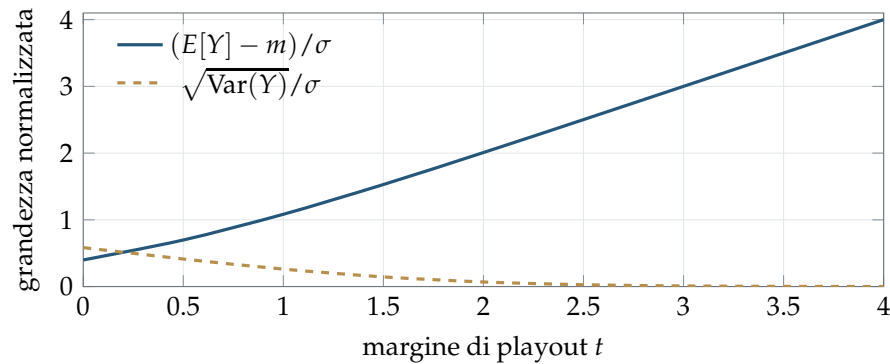


Figura 5.4. Compromesso fra ritardo medio aggiunto e jitter residuo nel modello gaussiano.

Automatic Repeat reQuest (ARQ) è efficace quando la ritrasmissione può arrivare prima della deadline. Non è quindi corretto affermare che le applicazioni multimediali non usino mai ritrasmissioni: in streaming con buffer ampio sono normali, mentre nella conversazione a bassa latenza la stessa ritrasmissione può diventare inutile se il round-trip time è paragonabile al tempo residuo prima del playout.

Forward Error Correction (FEC) trasmette invece informazione ridondante prima di sapere se si verificherà una perdita. Il vantaggio è evitare l'attesa del feedback; il costo è una bitrate maggiore anche quando il canale non perde pacchetti.

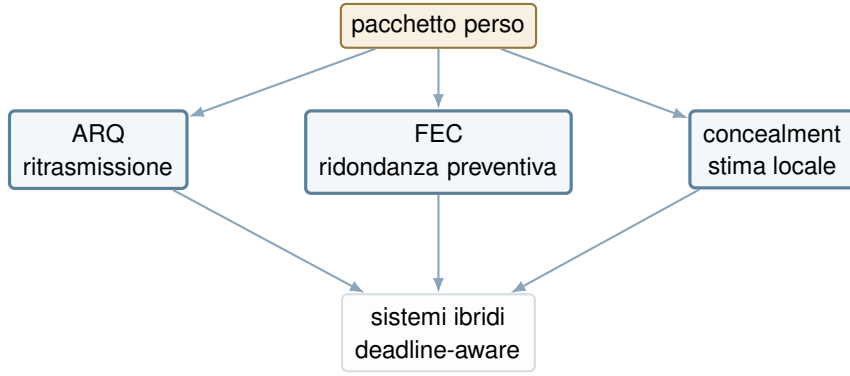


Figura 5.5. Tre famiglie di risposta alla perdita. La scelta dipende soprattutto dalla deadline, dalla statistica delle perdite e dalla possibilità di feedback.

n-way XOR

Consideriamo gruppi di n pacchetti informativi di uguale lunghezza L bit. Per il gruppo g definiamo il pacchetto di parità

$$P_{g,j} = \bigoplus_{k=0}^{n-1} B_{gn+k,j}, \quad j = 0, \dots, L-1.$$

La relazione è la definizione del codice XOR sistematico: si trasmettono i n pacchetti originali e un pacchetto di parità.

Se nel gruppo manca un solo pacchetto informativo $B_{v,j}$, con

$$g = \left\lfloor \frac{v}{n} \right\rfloor, \quad q = v - gn,$$

si ricostruisce

$$B_{v,j} = P_{g,j} \oplus \bigoplus_{\substack{k=0 \\ k \neq q}}^{n-1} B_{gn+k,j}.$$

Dimostrazione. Sostituendo la definizione di $P_{g,j}$, ogni pacchetto ricevuto compare due volte nello XOR. Poiché $a \oplus a = 0$ e $a \oplus 0 = a$, tutti i termini noti si cancellano e resta esattamente $B_{v,j}$. Questo conclude la dimostrazione.

Il codice recupera qualunque singola perdita fra gli $n+1$ pacchetti del gruppo. Se si perde soltanto la parità, i dati sono già completi; se si perde un dato e la parità arriva, il dato viene ricostruito. Due o più perdite nello stesso gruppo non sono in generale correggibili con una sola parità.

Assumendo perdite indipendenti con probabilità p uguale per ciascuno degli $n+1$ pacchetti, il numero Z di perdite è binomiale:

$$P(Z = z) = \binom{n+1}{z} p^z (1-p)^{n+1-z}.$$

La probabilità che l'intero contenuto informativo sia recuperabile è

$$\begin{aligned} P_{ok} &= P(Z = 0) + P(Z = 1) \\ &= (1-p)^{n+1} + (n+1)p(1-p)^n \\ &= (1-p)^n(1+np). \end{aligned}$$

Di conseguenza

$$P_s = 1 - (1-p)^n(1+np).$$

Dimostrazione. Il gruppo è recuperabile per zero o una perdita. Sommando i due termini della distribuzione binomiale e raccogliendo $(1 - p)^n$ si ottiene la prima relazione; il fallimento è l'evento complementare. Questo conclude la dimostrazione.

La ridondanza, ignorando gli header, è

$$\phi = \frac{n+1}{n} = 1 + \frac{1}{n}.$$

Dimostrazione. Ogni nL bit informativi vengono accompagnati da altri L bit di parità. Il rapporto fra bit trasmessi e bit sorgente è $(nL + L)/(nL) = (n + 1)/n$. Questo conclude la dimostrazione.

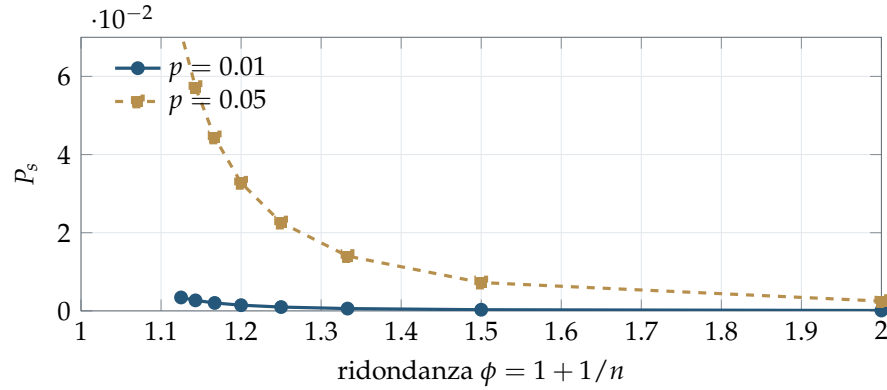


Figura 5.6. Compromesso qualitativo fra ridondanza e probabilità di fallimento del gruppo per XOR a singola parità. I punti sono calcolati dal modello di perdite indipendenti.

Le perdite reali sono spesso bursty e quindi correlate. In quel caso il modello binomiale può sottostimare la probabilità di più perdite nello stesso gruppo; interleaving e codici capaci di correggere più erasures sono strumenti più appropriati.

Ridondanza multi-rate

Un'altra strategia consiste nel trasmettere una rappresentazione primaria ad alta qualità insieme a una copia ritardata e più compatta di intervalli temporali precedenti. Se il blocco principale contiene L bit e la rappresentazione ridondante usa una bitrate inferiore di un fattore $\rho > 1$, la sua dimensione ideale è L/ρ bit. Con lunghezze intere si può usare

$$L_r = \left\lceil \frac{L}{\rho} \right\rceil.$$

La ridondanza relativa è

$$\phi = \frac{L + L_r}{L} = 1 + \frac{\lceil L/\rho \rceil}{L}.$$

Per blocchi grandi,

$$\phi \approx 1 + \frac{1}{\rho}.$$

Dimostrazione dell'approssimazione. Per la proprietà del soffitto,

$$\frac{L}{\rho} \leq \left\lceil \frac{L}{\rho} \right\rceil < \frac{L}{\rho} + 1.$$

Dividendo per L e sommando 1,

$$1 + \frac{1}{\rho} \leq \phi < 1 + \frac{1}{\rho} + \frac{1}{L}.$$

Quindi, al crescere di L , la differenza dall'approssimazione tende a zero. Questo conclude la dimostrazione.

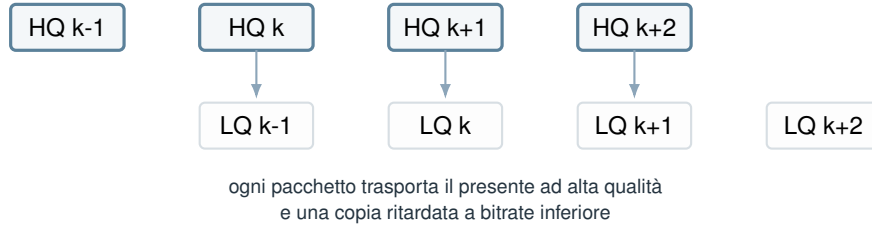


Figura 5.7. Protezione multi-rate: la perdita del blocco principale può essere nascosta usando la copia a qualità inferiore trasportata successivamente.

La tecnica introduce almeno un intervallo di ritardo prima che la copia ridondante diventi disponibile. È quindi adatta soltanto quando la deadline lascia tale margine. Nella telefonia moderna idee analoghe appaiono come in-band FEC o packet-loss concealment assistito dal codec, anche se la sintassi concreta dipende dallo standard.

Interleaving

L'interleaving non aggiunge ridondanza: cambia l'ordine temporale dei dati per trasformare un burst di perdita in errori più distanziati dopo il deinterleaving. Sia f una permutazione di $\{0, \dots, s-1\}$. Il trasmettitore invia

$$O_i = P_{f(i)},$$

e il ricevitore, quando dispone del blocco, applica

$$\hat{P}_i = O_{f^{-1}(i)}.$$

Queste relazioni definiscono un block interleaver ideale.

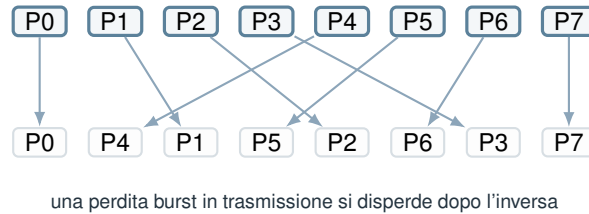


Figura 5.8. Esempio di block interleaving come semplice permutazione di pacchetti.

Se il ricevitore deve attendere tutti gli s pacchetti di un blocco prima di emettere il primo elemento nell'ordine originale, e ogni pacchetto richiede un tempo di serializzazione $T_p = L/R$, il ritardo di riempimento del blocco è dell'ordine di

$$D_I \leq sT_p = s\frac{L}{R}.$$

Dimostrazione. Un pacchetto di L bit serializzato a rate R richiede $T_p = L/R$. Nel caso più conservativo il primo elemento utile non può essere emesso finché non sono stati ricevuti tutti gli s pacchetti del blocco; il tempo di riempimento non supera quindi $sT_p = sL/R$. Questo conclude la dimostrazione del limite. Il ritardo esatto può essere minore se il deinterleaver può iniziare a produrre output prima del completamento del blocco.

Protezione differenziata (UEP)

Non tutti i bit hanno la stessa importanza. Header, informazioni di sincronizzazione, parameter set e frame di riferimento possono avere un impatto maggiore rispetto a dati locali o enhancement layer. La Unequal Error Protection (UEP) assegna quindi ridondanze differenti a classi differenti.

Supponiamo che ogni intervallo sorgente contenga C classi di lunghezza L_i e che la classe i riceva una parola di parità ogni n_i pacchetti. Trascurando header aggiuntivi, la ridondanza media è

$$\phi = \frac{\sum_{i=1}^C L_i + \sum_{i=1}^C L_i/n_i}{\sum_{i=1}^C L_i}.$$

Dimostrazione. In media, per ogni pacchetto sorgente la classe i contribuisce con L_i bit informativi e con L_i/n_i bit di parità. Sommando su tutte le classi e dividendo per i bit informativi totali si ottiene la formula. Questo conclude la dimostrazione.

Se per ciascuna classe si usa il codice XOR a singola parità e si assumono perdite indipendenti con probabilità p , la probabilità che un gruppo della classe i non sia recuperabile è

$$P_{s,i} = 1 - (1 - p)^{n_i}(1 + n_i p).$$

Questa è la formula già dimostrata per n -way XOR applicata al valore n_i .

Per due classi con $n_2 = A n_1$, $A > 1$, si ottiene

$$\begin{aligned} P_{s,1} &= 1 - (1 - p)^{n_1}(1 + n_1 p), \\ P_{s,2} &= 1 - (1 - p)^{A n_1}(1 + A n_1 p). \end{aligned}$$

La classe 1 è più protetta perché usa gruppi più piccoli e quindi più parità per unità di dati.

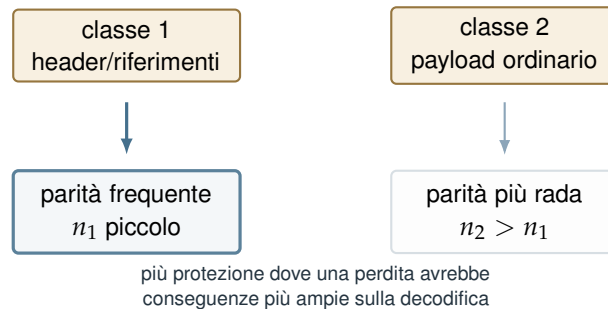


Figura 5.9. Principio della protezione differenziata. Le classi più importanti ricevono una quota maggiore della ridondanza disponibile.

La UEP moderna può operare anche a livello di layer scalabili, priorità di pacchetto o simboli fountain/Raptor. Il principio resta invariato: la ridondanza va allocata in funzione del danno atteso della perdita, non semplicemente in modo uniforme.

5.2 Congestione

Una rete è congestionata quando la domanda di servizio supera, almeno per un intervallo significativo, la capacità disponibile in uno o più colli di bottiglia. Un burst può inizialmente essere assorbito dalla coda; se persiste, crescono ritardo e occupazione fino al drop o al marking dei pacchetti.

Il problema non è quindi semplicemente “avere poca banda”. Anche una linea veloce può sviluppare code molto profonde se il traffico in ingresso arriva a burst e il buffer è dimensionato per assorbirli. Il fenomeno noto come *bufferbloat* mostra che aumentare indiscriminatamente la memoria di coda può ridurre le perdite ma peggiorare drasticamente la latenza.

Il controllo della congestione coinvolge più livelli:

- gli endpoint regolano il rate inviato mediante algoritmi di congestion control;
- i router effettuano scheduling, queue management e, se previsto, marking ECN;
- l'applicazione multimediale può adattare bitrate, risoluzione, frame rate o robustezza in funzione delle condizioni osservate.

5.2.1 FIFO e classificazione

La disciplina FIFO trasmette i pacchetti nell'ordine di arrivo. Se il buffer ha capacità finita e non esistono meccanismi di early marking, la politica più semplice è tail drop: il nuovo pacchetto viene scartato quando la coda è piena.

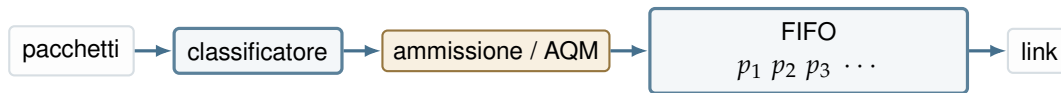


Figura 5.10. Coda FIFO con classificazione e funzione di ammissione/queue management. La politica di drop o marking può usare classe, occupazione e ritardo della coda.

La classificazione può usare indirizzi e porte, protocollo di trasporto, identificatori di tunnel, DSCP, informazioni applicative rese disponibili al nodo o una combinazione di questi elementi. Nelle reti moderne il campo Differentiated Services dell'header IP contiene DSCP per la classificazione per-hop e due bit ECN per la segnalazione di congestione senza necessariamente scartare il pacchetto.

È importante non confondere classificazione e garanzia. Marcare un pacchetto con un determinato DSCP non crea da solo capacità: il dominio di rete deve associare a quella classe code, scheduler e policy coerenti.

5.2.2 Fair Queuing e GPS

La Fair Queuing separa il traffico in flussi e cerca di impedire che un singolo flusso aggressivo monopolizzi il link. Il modello ideale è Generalized Processor Sharing (GPS), un server fluido che può servire simultaneamente più flussi.

Con pesi uguali, se in un certo intervallo sono attivi N flussi e il link ha capacità C , ciascuno riceve idealmente

$$r_i = \frac{C}{N}.$$

Dimostrazione. GPS con pesi uguali divide la capacità in parti uguali fra i N flussi backlogged. La somma dei rate deve essere C ; quindi $Nr_i = C$ e $r_i = C/N$. Questo conclude la dimostrazione.

Un server reale trasmette pacchetti indivisibili. Una semplice approssimazione didattica mantiene per ogni flusso il servizio cumulativo S_i e seleziona una coda non vuota con servizio minimo:

$$k \in \arg \min_{i: q_i > 0} S_i,$$

poi aggiorna

$$S_k \leftarrow S_k + L_k.$$

Questa costruzione rende visibile il principio di fairness, ma il WFQ classico usa tempi virtuali di inizio e fine per approssimare in modo più preciso il server GPS.

5.2.3 Weighted Fair Queuing

Nel Weighted Fair Queuing a ogni flusso i è associato un peso $w_i > 0$. Nel modello GPS, se $\mathcal{A}(t)$ è l'insieme dei flussi attivi,

$$r_i(t) = C \frac{w_i}{\sum_{j \in \mathcal{A}(t)} w_j}.$$

Questa è la definizione della condivisione pesata del server fluido.

Per due flussi contemporaneamente backlogged,

$$\frac{r_i(t)}{r_j(t)} = \frac{w_i}{w_j}.$$

Dimostrazione. Dividendo le due espressioni GPS, i fattori C e la somma dei pesi attivi si cancellano e resta il rapporto w_i/w_j . Questo conclude la dimostrazione.

Nella versione semplificata basata sul servizio cumulativo si confronta

$$\frac{S_i}{w_i},$$

non $w_i S_i$: un peso maggiore deve permettere più servizio a parità di valore normalizzato.

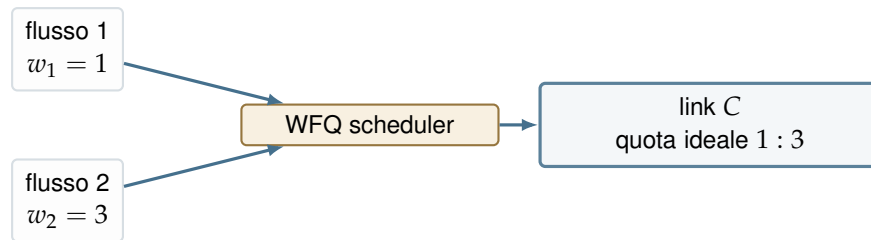


Figura 5.11. Condivisione pesata: se entrambi i flussi restano attivi, il secondo riceve idealmente tre volte il rate del primo.

5.2.4 Drop, marking e Active Queue Management

Una coda può reagire alla congestione con drop o marking. Tail drop interviene soltanto a buffer pieno; head drop elimina il pacchetto più vecchio; politiche casuali o class-aware possono distribuire il costo in modo differente fra flussi.

Active Queue Management (AQM) interviene prima della saturazione per mantenere il ritardo sotto controllo e fornire un segnale di congestione agli endpoint. Il segnale può essere un drop oppure un mark ECN quando i protocolli coinvolti lo supportano.

RED

Random Early Detection usa una media mobile dell'occupazione:

$$\bar{q}_i = (1 - w_q) \bar{q}_{i-1} + w_q q_i, \quad 0 < w_q < 1.$$

Questa è la definizione dell'average queue estimator.

Con soglie $T_m < T_M$ e parametro P_{max} , una forma classica della probabilità base è

$$p_b = \begin{cases} 0, & \bar{q}_i \leq T_m, \\ P_{max} \frac{\bar{q}_i - T_m}{T_M - T_m}, & T_m < \bar{q}_i < T_M, \\ 1, & \bar{q}_i \geq T_M. \end{cases}$$

La regione centrale è un'interpolazione lineare fra 0 e P_{max} .

Dimostrazione. La retta che passa per $(T_m, 0)$ e (T_M, P_{max}) ha pendenza $P_{max}/(T_M - T_m)$; valutandola in $\bar{q}_i$ si ottiene $P_{max}(\bar{q}_i - T_m)/(T_M - T_m)$. Questo conclude la dimostrazione della parte lineare.

Nella formulazione classica un contatore c dei pacchetti accettati dall'ultimo evento può correggere la probabilità mediante

$$p_a = \min \left\{ 1, \frac{p_b}{1 - cp_b} \right\},$$

quando il denominatore è positivo. I parametri sono scelte di configurazione e non costanti universali.

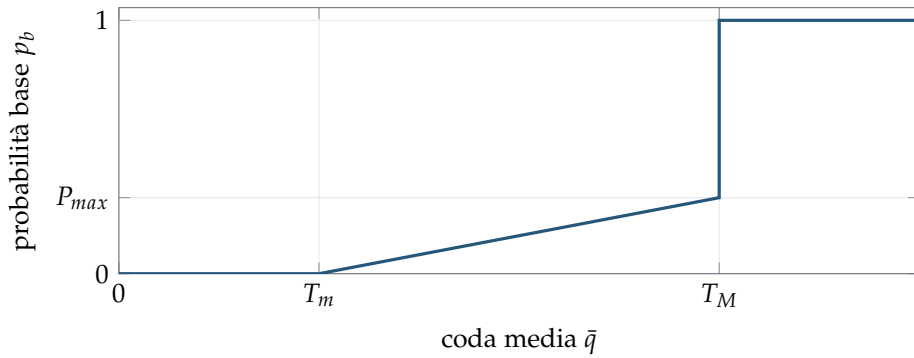


Figura 5.12. Curva RED schematica. La probabilità cresce nella regione intermedia e diventa unitaria oltre la soglia massima.

CoDel, PIE e FQ-CoDel

RED usa principalmente la lunghezza media della coda come indicatore. Gli AQM più recenti cercano invece di controllare direttamente il ritardo o una sua stima. CoDel osserva il sojourn time dei pacchetti e reagisce quando il ritardo minimo rimane sopra un target per un intervallo sufficientemente lungo. PIE usa un controllore che adatta la probabilità di drop/mark in funzione della stima del delay e della sua variazione.

FQ-CoDel combina una schedulazione fair per flusso con CoDel sulle code interne. L'effetto è duplice: isola meglio i flussi aggressivi e limita il ritardo persistente. Questi algoritmi mostrano il passaggio da "evitare il buffer pieno" a "controllare la latenza del buffer", concetto particolarmente importante per voce, gioco e video interattivo.



Figura 5.13. Idea di FQ-CoDel: separare i flussi e controllare il ritardo delle code invece di affidarsi soltanto alla dimensione del buffer.

5.3 Qualità di servizio

Quality of Service (QoS) indica l'insieme dei meccanismi con cui una rete differenzia il trattamento del traffico oppure cerca di rispettare vincoli quantitativi su banda, ritardo, jitter e perdita. Non esiste un unico numero di QoS: l'importanza relativa delle metriche dipende dall'applicazione.

Internet rimane fondamentalmente un sistema best-effort end-to-end, ma all'interno di domini amministrativi possono essere applicati classificazione, policing, shaping, scheduling, admission control e Active Queue Management. L'applicazione moderna combina spesso questi meccanismi di rete con adattamento del codec e del rate, invece di affidarsi a una garanzia rigida lungo l'intero percorso Internet.

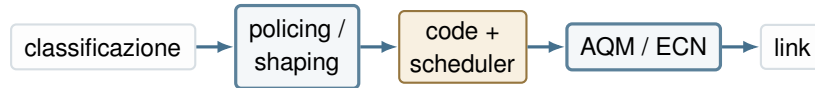


Figura 5.14. Blocchi funzionali tipici di una catena QoS in un router o in un edge node.

5.3.1 Code a priorità

Consideriamo un sistema $M/G/1$ non preemptive con K classi di priorità, numerate da 1, la più alta, a K , la più bassa. Gli arrivi della classe i sono Poisson con intensità λ_i e tutte le classi condividono la stessa distribuzione del tempo di servizio S . Definiamo

$$\rho_i = \lambda_i E[S], \quad \rho = \sum_{i=1}^K \rho_i < 1.$$

La condizione $\rho < 1$ è necessaria per la stabilità media del server.

Poiché il servizio è non preemptive, anche un pacchetto della classe più alta deve attendere il completamento di un pacchetto già in servizio. Il lavoro residuo medio visto da un arrivo Poisson è

$$W_0 = \frac{1}{2} \sum_{j=1}^K \lambda_j E[S^2].$$

Sia W_i il tempo medio di attesa in coda della classe i . Un pacchetto della classe i attende il lavoro residuo, i pacchetti delle classi $1, \dots, i$ già in coda e i nuovi arrivi delle classi $1, \dots, i-1$ durante l'attesa. Applicando la legge di Little,

$$W_i = W_0 + E[S] \sum_{j=1}^i \lambda_j W_j + E[S] \sum_{j=1}^{i-1} \lambda_j W_i.$$

Isolando W_i ,

$$W_i = \frac{W_0 + \sum_{j=1}^{i-1} \rho_j W_j}{1 - \sum_{j=1}^i \rho_j}.$$

Poniamo

$$R_i = 1 - \sum_{j=1}^i \rho_j, \quad R_0 = 1.$$

La soluzione è

$$W_i = \frac{W_0}{R_i R_{i-1}}.$$

Dimostrazione. Per $i = 1$ la ricorrenza fornisce $W_1 = W_0/R_1$, che coincide con la formula perché $R_0 = 1$. Supponiamola valida fino a $i - 1$. Poiché $\rho_j = R_{j-1} - R_j$,

$$\frac{\rho_j}{R_j R_{j-1}} = \frac{1}{R_j} - \frac{1}{R_{j-1}}.$$

La somma diventa telescopica:

$$\begin{aligned} W_0 + \sum_{j=1}^{i-1} \rho_j W_j &= W_0 \left[1 + \sum_{j=1}^{i-1} \frac{\rho_j}{R_j R_{j-1}} \right] \\ &= \frac{W_0}{R_{i-1}}. \end{aligned}$$

Dividendo per R_i si ottiene $W_i = W_0/(R_i R_{i-1})$. Questo conclude la dimostrazione.

Il tempo medio nel sistema è

$$E[T_i] = W_i + E[S].$$

Per servizio esponenziale, $E[S^2] = 2E[S]^2$ e quindi

$$W_0 = \rho E[S].$$

Nel caso $K = 2$ si ottiene

$$\begin{aligned} E[T_1] &= \frac{1 + \rho_2}{1 - \rho_1} E[S], \\ E[T_2] &= \frac{1 - \rho_1(1 - \rho_1 - \rho_2)}{(1 - \rho_1)(1 - \rho_1 - \rho_2)} E[S]. \end{aligned}$$

Dimostrazione. Per servizio esponenziale $W_0 = \rho E[S]$. Per $i = 1$,

$$E[T_1] = E[S] + \frac{\rho E[S]}{1 - \rho_1} = \frac{1 + \rho_2}{1 - \rho_1} E[S].$$

Per $i = 2$,

$$E[T_2] = E[S] + \frac{\rho E[S]}{(1 - \rho_1 - \rho_2)(1 - \rho_1)}.$$

Portando i termini allo stesso denominatore e usando $\rho = \rho_1 + \rho_2$, il numeratore diventa $1 - \rho_1(1 - \rho_1 - \rho_2)$. Si ottiene quindi la seconda relazione. Questo conclude la dimostrazione.

Verifica del caso a singola classe. Ponendo $\rho_2 = 0$ nella prima relazione oppure $\rho_1 = 0$ nella seconda si ottiene $E[T] = E[S]/(1 - \rho)$, la nota formula $M/M/1$. Questo conclude la verifica.

La priorità stretta può ridurre molto il ritardo della classe alta, ma può affamare le classi basse. In pratica si combinano spesso priorità, rate limit e scheduler pesati per evitare starvation.

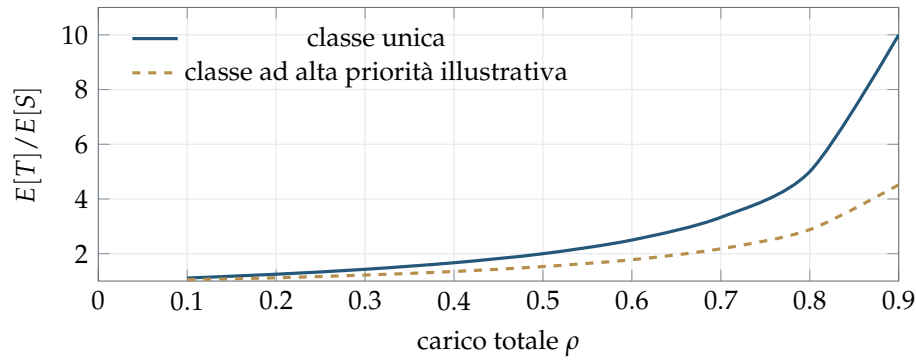


Figura 5.15. Effetto qualitativo della priorità sul ritardo medio. La curva della classe alta è illustrativa: il risultato esatto dipende dalla ripartizione del carico fra classi.

5.3.2 Leaky Bucket e Token Bucket

Il Leaky Bucket produce un'uscita regolarizzata: i burst vengono assorbiti da una coda e il traffico esce a un rate configurato. È adatto quando si vuole imporre una cadenza quasi costante, ma elimina anche burst brevi che l'applicazione potrebbe tollerare.

Il Token Bucket separa rate medio e burst massimo. I token sono generati a rate r e accumulati fino a una capacità b . Un byte può essere ammesso soltanto se esiste un token corrispondente. In una formulazione cumulativa, il traffico conforme soddisfa

$$A(t) - A(s) \leq b + r(t - s), \quad t \geq s.$$

Dimostrazione. All'istante s possono essere disponibili al massimo b token accumulati. Nell'intervallo $(s, t]$ ne vengono generati altri $r(t - s)$. Non è quindi possibile ammettere più di $b + r(t - s)$ byte. Questo conclude la dimostrazione.

Per intervalli molto lunghi,

$$\frac{A(t) - A(s)}{t - s} \leq r + \frac{b}{t - s} \rightarrow r.$$

Dimostrazione. Dividendo la disuguaglianza del token bucket per $t - s > 0$ si ottiene la prima relazione. Quando $t - s \rightarrow \infty$, il termine $b/(t - s)$ tende a zero. Questo conclude la dimostrazione del limite sul rate medio.

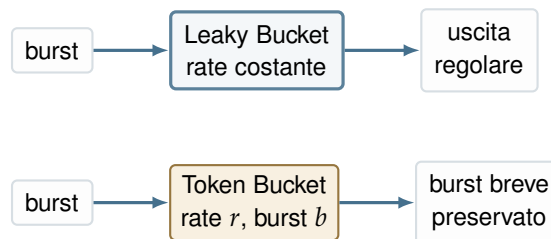


Figura 5.16. Differenza concettuale fra Leaky Bucket e Token Bucket. Il secondo permette burst fino alla riserva di token accumulata.

Nel policing, un pacchetto non conforme può essere scartato o marcato; nel shaping viene normalmente ritardato finché torna conforme. I due meccanismi usano quindi criteri simili ma producono conseguenze applicative differenti.

5.3.3 IntServ e RSVP

Integrated Services (IntServ) rappresenta l'approccio per-flusso alla QoS IP. Un flusso dichiara il proprio traffico mediante una TSPEC e richiede un servizio; i nodi lungo il percorso mantengono stato e verificano se le risorse possono essere ammesse senza violare gli impegni esistenti.

Le due classi storiche fondamentali sono Guaranteed Service, progettato per fornire bound quantitativi sul ritardo di accodamento sotto le ipotesi del modello di traffico, e Controlled-Load Service, che mira a un comportamento simile a quello di una rete poco caricata. È più corretto parlare di bound e modello di servizio che di "jitter prefissato a ogni router".

RSVP è un protocollo di segnalazione a soft state. Il mittente invia messaggi PATH lungo il percorso verso uno o più ricevitori; i ricevitori inviano messaggi RESV nel verso opposto per richiedere la riserva. Il protocollo supporta sia unicast sia multicast e mantiene lo stato mediante refresh periodici e timeout.



Figura 5.17. Segnalazione RSVP semplificata: PATH segue il percorso dati, RESV torna verso il mittente e installa lo stato di riserva.

Il limite principale dell'approccio è la scalabilità dello stato per-flusso nei core router e la necessità di coordinamento lungo il percorso. Per questo IntServ è rimasto importante come modello e in domini specifici, mentre Internet su larga scala usa più spesso aggregazione per classe e adattamento end-to-end.

5.3.4 DiffServ

Differentiated Services aggrega i pacchetti in classi e associa a ogni classe un Per-Hop Behavior (PHB). I router core non devono mantenere lo stato di ogni singola sessione; la complessità maggiore può essere spostata ai bordi del dominio, dove si effettuano classificazione, marking, policing e shaping.

Il Differentiated Services field dell'header IP usa sei bit DSCP per la classificazione e due bit ECN per la segnalazione di congestione. Il DSCP non identifica un contratto end-to-end: seleziona il PHB che il dominio decide di applicare.

Fra i PHB più importanti:

- **BE** rappresenta il comportamento best-effort di default;
- **EF** è progettato per traffico che richiede basso ritardo, basso jitter e bassa perdita entro un profilo opportunamente controllato;
- **AF** definisce classi Assured Forwarding con differenti precedenze di drop, lasciando al dominio la scelta di scheduler e queue management coerenti.



Figura 5.18. Architettura DiffServ: classificazione e conditioning ai bordi, trattamento aggregato per PHB nel core.

DiffServ migliora la scalabilità rispetto allo stato per-flusso, ma una classe privilegiata resta efficace soltanto se capacità e policy del dominio sono dimensionate coerentemente. Un DSCP attraversato da un dominio che non applica il PHB desiderato non crea una garanzia

magica.

5.3.5 L4S: latenza bassa e segnalazione ECN frequente

Low Latency, Low Loss, Scalable Throughput (L4S) porta il controllo della congestione verso code molto corte senza richiedere di riservare banda per ogni flusso. Il principio è usare congestion control scalabile, marking ECN molto più frequente e un AQM capace di separare il traffico L4S dal traffico classico mantenendo equità fra i due regimi.

Nel modello DualQ Coupled AQM due code logiche, una classic e una L4S, condividono lo stesso link. La coda L4S può mantenere un target di ritardo molto più basso perché i sender reagiscono a marking frequente senza interpretare ogni mark come un evento equivalente a una perdita classica.

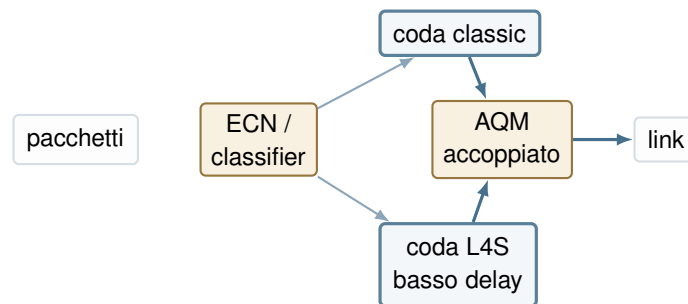


Figura 5.19. Idea di DualQ L4S: code logiche separate con AQM accoppiato consentono bassa latenza al traffico scalabile senza isolare completamente il traffico classico.

Per audio, gaming, cloud rendering e videoconferenza, L4S è concettualmente importante perché sposta l'obiettivo dalla sola massimizzazione del throughput alla riduzione simultanea di coda, perdita e latenza sotto carico.

5.4 RTP e RTCP

Real-time Transport Protocol (RTP) fornisce le informazioni necessarie a interpretare temporalmente un flusso realtime: numero di sequenza, timestamp, identificazione della sorgente e tipo di payload. RTP non garantisce consegna, ritardo o banda, non ritrasmette autonomamente i pacchetti e non decide come adattare il codec. Queste decisioni appartengono al trasporto, all'applicazione e ai meccanismi di feedback.

RTP è frequentemente usato sopra UDP perché la semantica datagram e l'assenza di ritrasmissione obbligatoria sono adatte ai media realtime, ma il protocollo RTP non è concettualmente vincolato a UDP. Nelle architetture WebRTC, per esempio, RTP/SRTP viene usato insieme a ICE, DTLS e meccanismi di congestion control che saranno trattati nel capitolo successivo.

La distribuzione può essere unicast o multicast. In IPv4 il multicast usa il prefisso 224.0.0.0/4, storicamente chiamato spazio "classe D"; la terminologia classful è oggi obsoleta. IPv6 usa indirizzi multicast nel prefisso ff00::/8. Il multicast di rete è molto efficiente quando disponibile nel dominio, ma Internet pubblico e le CDN su larga scala usano prevalentemente distribuzione unicast e replica applicativa.

5.4.1 Sorgenti e mixer

Ogni sorgente RTP è identificata da un Synchronization Source identifier (SSRC) a 32 bit. Il valore è scelto in modo da minimizzare collisioni e non è un identificatore permanente dell'utente. Un mixer può combinare più sorgenti e indicare gli SSRC contributori mediante una lista di Contributing Source identifiers (CSRC).

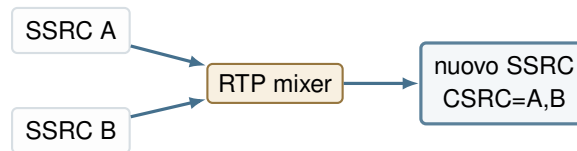
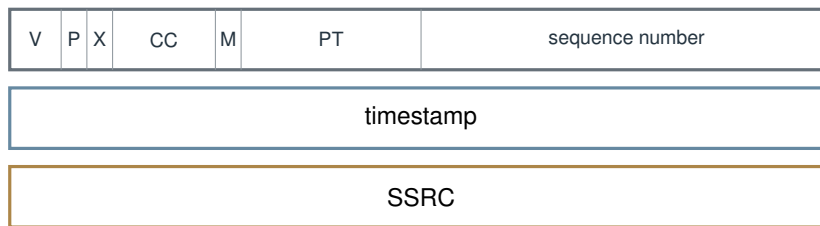


Figura 5.20. Un mixer RTP diventa una nuova sorgente sincronizzata e può dichiarare le sorgenti che hanno contribuito al payload mediante la lista CSRC.

5.4.2 Header RTP

L'header base RTP è lungo 12 byte. La struttura comprende versione, padding, extension flag, CSRC count, marker, payload type, sequence number, timestamp e SSRC.



seguono 0–15 CSRC se CC > 0, quindi eventuale header extension e payload

Figura 5.21. Header base RTP di 12 byte. La figura mostra i campi fondamentali; estensioni e CSRC rendono l'header più lungo.

I campi principali hanno le seguenti funzioni:

- **sequence number**, 16 bit, aumenta di uno per ogni pacchetto RTP e consente di rilevare perdita e riordinamento;
- **timestamp**, 32 bit, rappresenta l'istante di campionamento del primo campione o evento rappresentato nel payload, secondo il clock rate definito dal payload format;
- **payload type (PT)**, 7 bit, seleziona il formato secondo il profilo e la negoziazione della sessione;
- **marker (M)**, 1 bit, ha semantica definita dal particolare payload format e non significa universalmente "inizio GOP";
- **SSRC**, 32 bit, identifica la sorgente di sincronizzazione.

Il timestamp RTP non è normalmente un timestamp assoluto di wall clock. Per esempio un payload audio può incrementarlo in unità di campioni, mentre un payload video usa un clock rate specificato dal proprio payload format. Questo permette di mantenere una base temporale regolare anche quando i pacchetti vengono emessi con jitter.

Se un pacchetto contiene N campioni audio di un flusso il cui clock RTP è f_c tick/s, l'incremento di timestamp è

$$\Delta T = N$$

quando un tick corrisponde a un campione, e l'intervallo temporale rappresentato è

$$\Delta t = \frac{N}{f_c}.$$

Dimostrazione. Con clock f_c vengono contati f_c tick in un secondo. N tick corrispondono quindi a N/f_c secondi. Questo conclude la dimostrazione.

5.4.3 RTCP: feedback e sincronizzazione

RTCP accompagna RTP con messaggi di controllo. I tipi fondamentali comprendono Sender Report (SR), Receiver Report (RR), Source Description (SDS), BYE e APP; estensioni successive aggiungono feedback più tempestivo, per esempio informazioni su perdite, richieste di keyframe e altri segnali utili al controllo del media.

Un Sender Report associa il clock RTP della sorgente a un riferimento NTP. Questa coppia permette di sincronizzare stream con clock RTP differenti, per esempio audio e video: non si confrontano direttamente i loro timestamp numerici, ma si mappano entrambi sullo stesso tempo di riferimento.

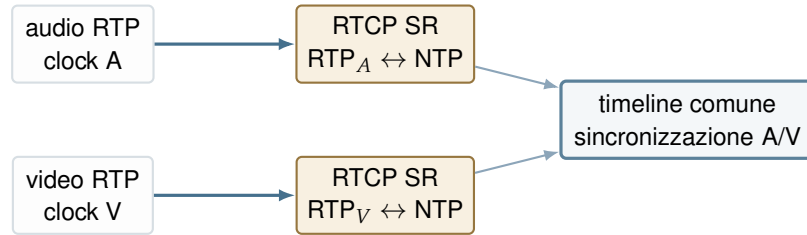


Figura 5.22. La sincronizzazione inter-media usa la corrispondenza fra clock RTP e riferimento NTP trasmessa nei Sender Report RTCP.

Receiver Report e blocchi di report possono contenere fraction lost, conteggi di perdita cumulativa, sequence number esteso, interarrival jitter e dati per stimare il round-trip time. Sono misure: RTCP non ordina automaticamente al codec di cambiare bitrate. Un congestion controller o l'applicazione interpreta il feedback e decide se modificare rate, risoluzione, FEC o struttura predittiva.

Storicamente RTP e RTCP usavano spesso due porte UDP adiacenti. Il multiplexing RTP/RTCP sulla stessa porta è però supportato da profili e specifiche moderne ed è comune in WebRTC; non va quindi trattata la separazione di porta come proprietà necessaria di ogni sessione.

5.4.4 Overhead e pacchettizzazione

Con IPv4 senza opzioni, UDP e RTP base gli header sono rispettivamente 20, 8 e 12 byte, per un totale

$$H_{v4} = 40 \text{ byte.}$$

Con IPv6 base, UDP e RTP,

$$H_{v6} = 60 \text{ byte.}$$

Sono valori di specifica per gli header base; estensioni IPv6, RTP header extension, SRTP, tunneling e layer 2 possono aumentare l'overhead.

Se il payload utile è P byte e l'overhead per pacchetto è H byte, l'efficienza di pacchettizzazione è

$$\eta = \frac{P}{P + H}.$$

Dimostrazione. Ogni pacchetto trasporta P byte utili su $P + H$ byte totali. Il rapporto fra i due è l'efficienza. Questo conclude la dimostrazione.

Se ogni pacchetto rappresenta T_p secondi di media, la bitrate di rete dovuta a payload e header è

$$R_{net} = \frac{8(P + H)}{T_p}.$$

Dimostrazione. Ogni T_p secondi vengono inviati $P + H$ byte, cioè $8(P + H)$ bit. Dividendo per il periodo si ottengono bit/s. Questo conclude la dimostrazione.

Consideriamo G.729 a 8 kbit/s con frame di 10 ms. Ogni frame contiene

$$P = 8000 \cdot 0.010 / 8 = 10 \text{ byte.}$$

Con IPv4/UDP/RTP base, un frame per pacchetto richiede

$$R_{net} = \frac{8(10 + 40)}{0.010} = 40 \text{ kbit/s,}$$

escludendo layer 2.

Aggregando due frame in un pacchetto, $P = 20$ byte e $T_p = 20$ ms:

$$R_{net} = \frac{8(20 + 40)}{0.020} = 24 \text{ kbit/s.}$$

Dimostrazione. A 8 kbit/s, 10 ms di sorgente contengono $8000 \cdot 0.010 = 80$ bit, cioè 10 byte. Con 40 byte di header base, un pacchetto ogni 10 ms contiene 50 byte e richiede $8 \cdot 50 / 0.010 = 40$ kbit/s. Aggregando due frame, il payload diventa 20 byte e il periodo 20 ms; quindi $8 \cdot 60 / 0.020 = 24$ kbit/s. Questo conclude la dimostrazione.

L'overhead diminuisce, ma la pacchettizzazione aggiunge attesa e una singola perdita coinvolge un intervallo di media più lungo. Anche qui emerge un compromesso fra efficienza, latenza e robustezza.

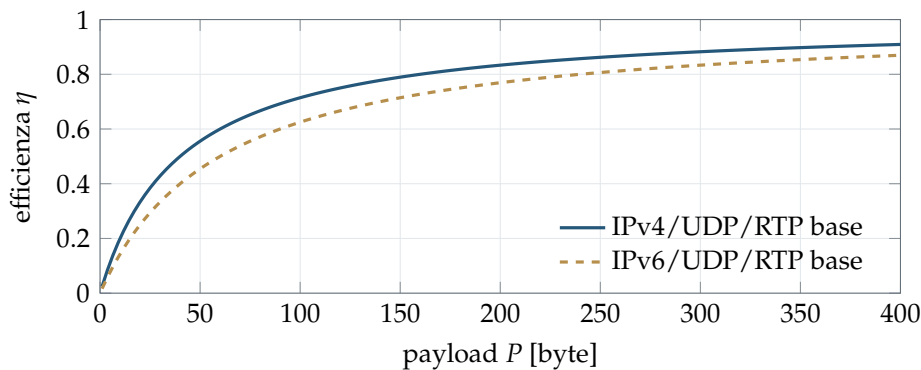


Figura 5.23. Efficienza di pacchettizzazione in funzione del payload. Per pacchetti piccoli l'overhead RTP/UDP/IP può dominare la bitrate.

Tecniche di header compression, come ROHC in collegamenti dove è appropriato, sfruttano la forte prevedibilità dei campi RTP/UDP/IP per ridurre l'overhead su pacchetti piccoli. Il beneficio dipende dal link, dal profilo e dalla robustezza richiesta agli errori.

RTP non cifra né autentica il media. Secure RTP (SRTP) aggiunge confidenzialità, autenticazione e protezione anti-replay mantenendo la semantica temporale RTP.

5.5 Streaming multimediale su HTTP

Una famiglia di sistemi multimediali molto diversa da RTP tratta il contenuto come una sequenza di oggetti HTTP. Il vantaggio è poter usare server Web, cache, CDN, autenticazione e infrastrutture di distribuzione già diffuse. Il prezzo è che la latenza dipende non solo dal codec ma anche da segmentazione, manifest, richieste HTTP e buffer del player.

Il contenuto viene suddiviso in intervalli temporali consecutivi. Se lo stesso intervallo è disponibile in più rappresentazioni, il client può scegliere dinamicamente quella più adatta alle condizioni correnti. Questo modello scala bene verso grandi popolazioni perché trasforma lo stream in oggetti cacheabili e replicabili.

5.5.1 HLS e MPEG-DASH

HTTP Live Streaming (HLS) e MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) condividono l'idea di base ma usano sintassi di manifest e modelli temporali differenti. HLS usa playlist; DASH usa un Media Presentation Description (MPD). Entrambi possono descrivere più rappresentazioni, tracce audio alternative, sottotitoli e live window.



Figura 5.24. Catena tipica dello streaming HTTP adattativo: il client sceglie la rappresentazione, mentre CDN e cache distribuiscono gli stessi oggetti a molti utenti.

La durata del segmento controlla un compromesso. Segmenti lunghi riducono il numero di richieste e migliorano l'efficienza delle cache; segmenti corti riducono il tempo di reazione e possono diminuire la latenza live, ma aumentano overhead e sensibilità alla variabilità del trasporto.

5.5.2 Adaptive Bitrate Streaming

Consideriamo un segmento di durata media T_s secondi e dimensione S bit. Se durante il trasferimento il throughput utile è costante e pari a R bit/s,

$$T_d = \frac{S}{R}.$$

Dimostrazione. Per definizione $R = S/T_d$. Poiché $R > 0$, isolando T_d si ottiene $T_d = S/R$. Questo conclude la dimostrazione.

Sia B_k il buffer, misurato in secondi di media, all'inizio del download del segmento k . Al termine del download,

$$B_{k+1} = \max\{0, B_k - T_d\} + T_s.$$

Dimostrazione. Se $B_k \geq T_d$, durante il download vengono consumati T_d secondi e restano $B_k - T_d$. Se $B_k < T_d$, il buffer si svuota e non può diventare negativo. L'arrivo del nuovo segmento aggiunge T_s in entrambi i casi. Questo conclude la dimostrazione.

In assenza di stall il buffer non diminuisce se

$$\frac{S}{T_s} \leq R.$$

Dimostrazione. Senza stall, $B_{k+1} - B_k = T_s - T_d$. La differenza è non negativa se $T_s \geq T_d = S/R$, equivalente a $S/T_s \leq R$. Questo conclude la dimostrazione.

Un algoritmo ABR reale non conosce R futuro. Deve stimare throughput, osservare il buffer e bilanciare almeno quattro obiettivi: qualità media, numero e ampiezza degli switch, rebuffering e latenza rispetto al live edge. Per questo esistono algoritmi throughput-based, buffer-based e ibridi.

5.5.3 Low-Latency HLS e Low-Latency DASH

Lo streaming HTTP classico attende spesso che un intero segmento sia completo prima di renderlo disponibile. La latenza può essere ridotta rendendo consumabili parti più piccole del segmento mentre il segmento stesso è ancora in produzione.

Low-Latency HLS usa partial segments e meccanismi di playlist pensati per ridurre il ritardo di polling e aggiornamento: delta update, blocking playlist reload, preload hint e rendition report sono esempi di strumenti che riducono lavoro e attesa del client.

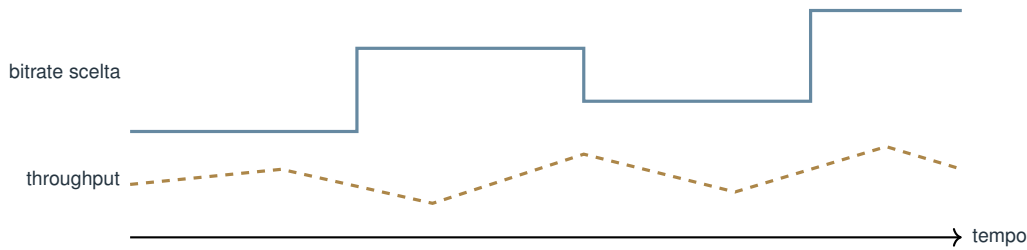


Figura 5.25. ABR concettuale: il player adatta la rappresentazione usando una stima imperfetta del throughput e lo stato del buffer.

Nel mondo DASH, la bassa latenza è frequentemente combinata con CMAF chunked media. Un segmento fragmented MP4 può contenere unità progressivamente disponibili; il client può iniziare a ricevere e processare media senza attendere la chiusura dell'intero segmento.

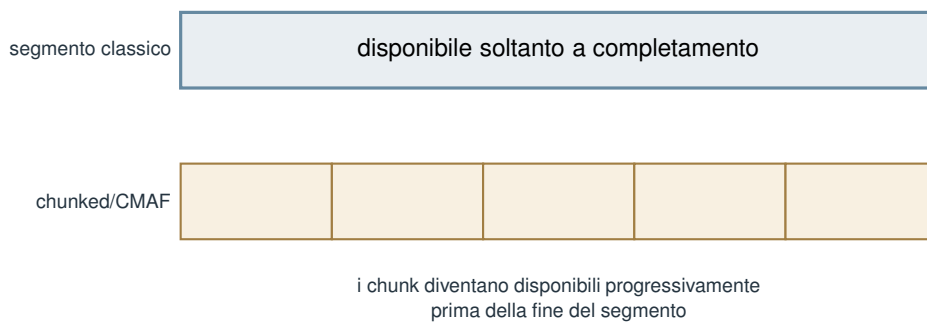


Figura 5.26. Segmento completo e consegna chunked. La bassa latenza deriva dal poter iniziare il trasferimento prima che l'intero segmento sia terminato.

Se un segmento di durata T_s è suddiviso in m chunk temporali di durata uniforme,

$$T_c = \frac{T_s}{m}.$$

Dimostrazione. La somma delle durate dei m chunk deve essere T_s . Se le durate sono uguali, $mT_c = T_s$, quindi $T_c = T_s/m$. Questo conclude la dimostrazione.

Ridurre T_c non rende però arbitrariamente piccola la latenza totale: restano encoding delay, tempo di ingest, propagazione CDN, buffer del player, sincronizzazione live e margine necessario a tollerare jitter e variazioni di throughput.

5.5.4 CDN, cache e live edge

Una CDN replica i segmenti verso nodi edge vicini agli utenti. Nel video on-demand un oggetto può essere servito molte volte dalla cache; nel live l'edge deve ricevere rapidamente il nuovo media e mantenersi vicino al live edge.

La distribuzione HTTP separa quindi due scale del problema. Il CDN ottimizza il trasporto da origine a popolazioni di edge; il player ottimizza la propria selezione ABR sull'ultimo tratto. Due utenti dello stesso edge possono scegliere rappresentazioni diverse e avere buffer molto differenti.

5.6 Distribuzione P2P e ibrida

La distribuzione peer-to-peer usa le risorse dei ricevitori come parte della rete di distribuzione. Un peer scarica chunk del contenuto e, se possibile, li inoltra ad altri peer. Il

sistema crea così un *overlay* applicativo nel quale disponibilità e capacità dipendono sia dall'infrastruttura sia dalla popolazione attiva.

Per un file statico il problema principale è completare il download. Per video on-demand e live compare un vincolo ulteriore: i chunk hanno un ordine temporale e una deadline di riproduzione. Una parte rara ma molto lontana dal playhead può essere importante per la salute dello swarm; un chunk comune ma necessario fra pochi secondi può essere più urgente per la continuità del singolo player. La progettazione deve quindi bilanciare disponibilità globale e deadline locale.

5.6.1 Discovery, overlay e ruoli

Un sistema P2P deve innanzitutto permettere ai partecipanti di trovarsi. Il discovery può essere centralizzato mediante tracker, distribuito mediante Distributed Hash Table (DHT), ottenuto per gossip fra peer o fornito dal piano di controllo del servizio. La scelta del peer con cui scambiare dati è distinta dalla scoperta: fra decine o centinaia di candidate un client può preferire nodi con RTT basso, buon upload, chunk complementari o maggiore stabilità.

Un *seed* possiede già il contenuto necessario e lo distribuisce; un *leecher* o peer incompleto sta ancora acquisendo parti del contenuto ma può contemporaneamente inoltrare ciò che possiede. Nel video live nessun peer possiede il futuro: i nuovi chunk devono essere immessi continuamente dalla sorgente o da un nodo di ingest e poi replicati nell'overlay.

5.6.2 Albero, mesh e swarming

Una distribuzione ad albero costruisce percorsi relativamente stabili: la sorgente alimenta alcuni figli, ciascuno dei quali replica verso altri nodi. L'overhead di scheduling è ridotto e non vengono richieste molte copie dello stesso chunk allo stesso livello. Il limite è il churn: l'uscita di un peer interno interrompe temporaneamente tutti i suoi discendenti.

Una mesh non assegna un unico genitore permanente. Ogni peer mantiene più vicini e richiede chunk differenti a nodi differenti. La ridondanza dei percorsi aumenta robustezza e sfrutta meglio upload eterogenei, ma richiede più signaling, buffer map e decisioni di scheduling.

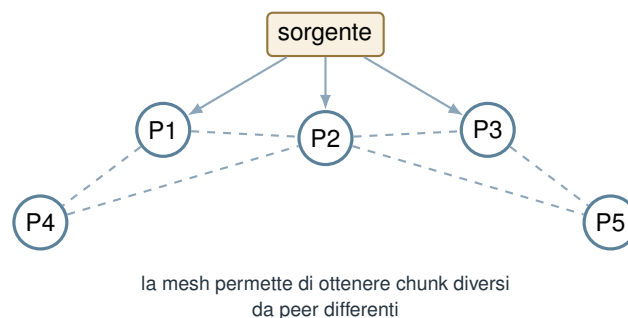


Figura 5.27. Swarm P2P a mesh. Le relazioni logiche possono cambiare nel tempo in funzione di disponibilità, upload e churn.

5.6.3 Push, pull e selezione dei chunk

Nel modello *push* chi possiede un chunk decide a quali vicini inviarlo. Il vantaggio è ridurre il round-trip di richiesta; lo svantaggio è che il mittente deve conoscere abbastanza bene lo stato dei ricevitori per non inviare copie inutili.

Nel modello *pull* il ricevitore osserva le buffer map dei vicini e richiede esplicitamente ciò che gli manca. La decisione è più informata e si adatta bene a una mesh, ma ogni richiesta introduce signaling e almeno una parte del RTT prima del trasferimento. Molti sistemi usano

forme ibride.

Nella distribuzione di file una politica celebre è *rarest first*: si preferiscono i chunk con meno copie note nello swarm, in modo da non lasciare che una parte del file diventi un collo di bottiglia. Nel multimedia questa politica deve essere combinata con la deadline. Una strategia ragionevole è privilegiare i chunk prossimi al playhead e, all'interno della finestra che possiede sufficiente slack temporale, favorire quelli più rari.

Il chunk size introduce un compromesso. Chunk piccoli permettono scheduling fine, parallelismo e recupero rapido da un peer lento; chunk grandi riducono hash, richieste e meta-dati per byte utile. Nel live, chunk troppo grandi aumentano inoltre il tempo store-and-forward prima che un peer possa diventare sorgente per altri.

5.6.4 Capacità aggregata e carico sull'infrastruttura

Consideriamo N utenti che consumano contemporaneamente lo stesso flusso alla bitrate r . La domanda aggregata ai ricevitori è

$$D = Nr.$$

Dimostrazione. Ciascuno dei N ricevitori deve ottenere r bit/s. Sommando i rate dei N flussi si ottiene Nr . Questo conclude la dimostrazione.

Sia u_i la capacità di upload fisica del peer i e sia $0 \leq \alpha_i \leq 1$ la frazione effettivamente utilizzabile per quel contenuto, dopo avere considerato disponibilità, topologia, protocollo, deadline e capacità dei destinatari. Definiamo

$$U_P = \sum_{i=1}^N \alpha_i u_i$$

come capacità P2P utile aggregata. Questa è una definizione di modello: α_i non è una costante del protocollo, ma riassume tutte le inefficienze che impediscono di usare integralmente l'upload nominale.

Sia R_I il rate aggregato erogato dall'infrastruttura verso ricevitori o overlay: comprende quindi consegna diretta da origin/CDN, relay e, nel live, l'eventuale iniezione della copia originaria. Nel modello trascuriamo retransmission e traffico di controllo. La conservazione della capacità impone

$$R_I \geq Nr - U_P.$$

Nel live, se la sorgente/ingest che genera il contenuto viene contabilizzata nell'infrastruttura, esiste inoltre contenuto nuovo a rate r che deve entrare almeno una volta nell'overlay; pertanto

$$R_I \geq r.$$

Combinando i due vincoli,

$$\boxed{R_I \geq \max\{r, Nr - U_P\}}.$$

Dimostrazione. I ricevitori consumano complessivamente Nr bit/s. I peer possono fornire al massimo U_P bit/s di tale domanda; la parte rimanente deve quindi provenire dall'infrastruttura, da cui $R_I \geq Nr - U_P$. In un flusso live i bit appena prodotti non esistono ancora nei peer e, nel modello in cui sorgente/ingest appartiene all'infrastruttura, almeno una copia del nuovo flusso di rate r deve essere immessa nell'overlay; quindi $R_I \geq r$. Entrambi i vincoli devono valere contemporaneamente, per cui R_I deve essere almeno il loro massimo. Questo conclude la dimostrazione.

La relazione è un *lower bound*: capacità aggregata sufficiente non garantisce che i chunk corretti si trovino nel peer corretto prima della deadline. Topologia, churn e scheduling possono richiedere molto più traffico infrastrutturale.

Se tutti i peer hanno upload medio u e la stessa efficienza media α , allora

$$U_P = \alpha Nu$$

e il bound diventa

$$R_I \geq \max\{r, N(r - \alpha u)\}.$$

Dividendo per N ,

$$\frac{R_I}{N} \geq \max\left\{\frac{r}{N}, r - \alpha u\right\}.$$

Dimostrazione. Sostituendo $U_P = \alpha Nu$ nel bound generale si ottiene $R_I \geq \max\{r, Nr - \alpha Nu\}$. Raccogliendo N nel secondo termine e poi dividendo entrambi i membri per $N > 0$ si ottengono le due relazioni. Questo conclude la dimostrazione.

Il risultato spiega la scalabilità ideale del P2P. Se $\alpha u \geq r$, la capacità utile che ogni nuovo peer porta con sé è almeno pari alla domanda che introduce: il bound totale sull'infrastruttura può allora restare dell'ordine di r invece di crescere come Nr . Se $\alpha u < r$, resta invece un deficit medio $r - \alpha u$ per utente e il carico infrastrutturale continua a crescere linearmente.

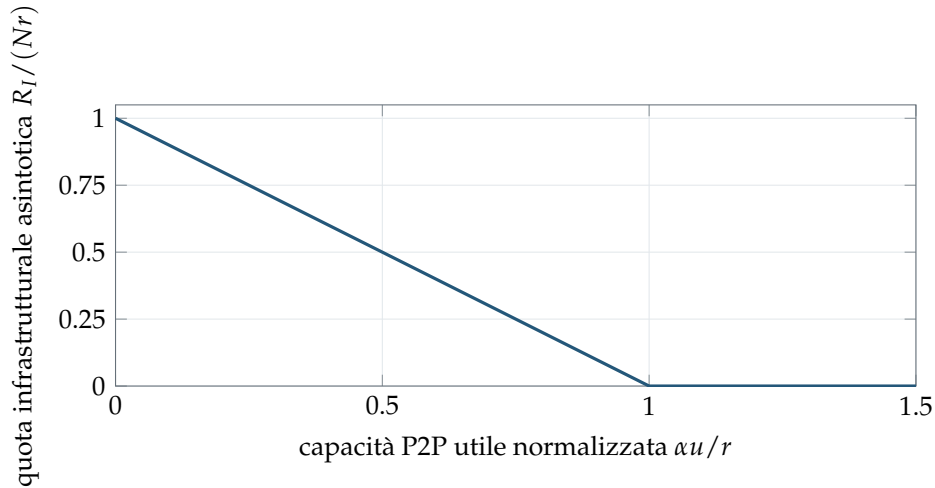


Figura 5.28. Bound aggregato per una popolazione grande e peer omogenei. Finché $\alpha u < r$, l'infrastruttura deve coprire il deficit; oltre tale punto il limite dominante nel live è l'iniezione della copia originaria.

5.6.5 Offload P2P

Definiamo l'*offload infrastrutturale* normalizzato rispetto al caso client-server, il cui rate di riferimento è Nr :

$$\Omega = 1 - \frac{R_I}{Nr}.$$

Nel regime operativo in cui l'ibrido non usa più infrastruttura del baseline, $0 \leq \Omega \leq 1$; se $R_I > Nr$, la stessa quantità diventa negativa e segnala un offload controproducente. Qui R_I mantiene la definizione precedente e non include retransmission o traffico di controllo. Di conseguenza Ω misura il risparmio di egress infrastrutturale, non necessariamente la sola frazione di byte che arriva all'utente direttamente da un peer: l'iniezione originaria e i relay restano correttamente a carico dell'infrastruttura.

Nel live, usando i due lower bound precedenti,

$$\Omega \leq \min \left\{ 1 - \frac{1}{N'} \frac{U_P}{Nr} \right\}.$$

Dimostrazione. Da $R_I \geq r$ segue

$$\Omega = 1 - \frac{R_I}{Nr} \leq 1 - \frac{r}{Nr} = 1 - \frac{1}{N}.$$

Da $R_I \geq Nr - U_P$ segue invece

$$\Omega \leq 1 - \frac{Nr - U_P}{Nr} = \frac{U_P}{Nr}.$$

Poiché entrambi i limiti valgono contemporaneamente, Ω non può superare il loro minimo. Questo conclude la dimostrazione.

Per video on-demand la prima limitazione può scomparire: se almeno una copia completa dei chunk richiesti è già disponibile nello swarm, l'origin non deve necessariamente immettere continuamente un nuovo flusso. Restano però i vincoli di disponibilità e capacità.

Esempio numerico

Con $N = 1000$ utenti, $r = 4$ Mbit/s, upload medio $u = 2$ Mbit/s ed efficienza $\alpha = 0.6$,

$$U_P = 0.6 \cdot 1000 \cdot 2 = 1200 \text{ Mbit/s}.$$

La domanda totale è 4000 Mbit/s e quindi

$$R_I \geq \max\{4, 4000 - 1200\} = 2800 \text{ Mbit/s}.$$

L'offload non può superare

$$\Omega \leq \frac{1200}{4000} = 0.30.$$

Dimostrazione. I valori si ottengono per sostituzione diretta nelle relazioni già dimostrate: $U_P = \alpha Nu$, $D = Nr$ e $R_I \geq \max\{r, D - U_P\}$. Il rapporto U_P/D fornisce il limite del 30% all'offload. Questo conclude la verifica numerica.

Se invece $u = 8$ Mbit/s con la stessa efficienza, $U_P = 4800$ Mbit/s: il bound aggregato diventa $R_I \geq 4$ Mbit/s e l'offload teorico può avvicinarsi a $1 - 1/N = 99.9\%$. Questo è un limite ideale, non una previsione operativa: richiede che upload, chunk e connessioni siano utilizzabili esattamente dove servono.

5.6.6 Disseminazione e latenza

La capacità totale non dice quanto velocemente una nuova informazione si diffonda. Consideriamo un modello store-and-forward molto semplice. In questa sottosezione N indica il numero totale di nodi che devono possedere il chunk, contando anche il detentore iniziale. All'inizio un solo nodo possiede il chunk e, durante un intervallo di trasferimento τ , ogni nodo che lo possiede può consegnare una copia completa ad al massimo un nuovo peer.

Dopo k intervalli, il numero di nodi che possono possedere il chunk, inclusa la sorgente iniziale, è al massimo

$$M_k \leq 2^k.$$

Dimostrazione. Per induzione. A $k = 0$ esiste una sola copia e $M_0 = 1 = 2^0$. Supponiamo $M_k \leq 2^k$. Nell'intervallo successivo ciascuno dei M_k nodi può creare al massimo una nuova copia; quindi $M_{k+1} \leq M_k + M_k = 2M_k \leq 2^{k+1}$. Questo conclude la dimostrazione.

Per raggiungere N peer sono quindi necessari almeno

$$k \geq \lceil \log_2 N \rceil$$

intervalli e

$$T_{dist} \geq \tau \lceil \log_2 N \rceil.$$

Dimostrazione. Perché tutti gli N peer possiedano il chunk deve valere $2^k \geq N$. Applicando il logaritmo in base 2, $k \geq \log_2 N$; essendo k intero, il minimo possibile è il soffitto. Moltiplicando il numero di intervalli per la loro durata τ si ottiene il bound temporale. Questo conclude la dimostrazione.

Se il chunk ha dimensione C bit e ciascun upload avviene a rate u , nel modello store-and-forward

$$\tau = \frac{C}{u}$$

e quindi

$$T_{dist} \geq \frac{C}{u} \lceil \log_2 N \rceil.$$

Dimostrazione. Trasferire C bit a rate costante u richiede C/u secondi. La sostituzione nel bound precedente produce la relazione. Questo conclude la dimostrazione.

Il risultato non è un limite universale di ogni P2P. Sub-chunk, pipelining, network coding, upload multipli e ricezione da più sorgenti violano intenzionalmente le ipotesi del modello. La relazione mostra però perché dimensione del chunk e profondità dell'overlay influenzino la latenza live anche quando la banda aggregata è abbondante.

5.6.7 Vincolo di fan-out nel live

Sia T_c la durata media rappresentata da un chunk di un flusso a bitrate r . La dimensione media del chunk è

$$C = rT_c.$$

Dimostrazione. La bitrate è quantità di bit divisa per tempo. In un intervallo T_c un flusso a rate costante r produce rT_c bit. Questo conclude la dimostrazione.

Se un peer con upload aggregato u deve inoltrare una copia completa del chunk a k peer entro lo stesso intervallo T_c , una condizione necessaria è

$$\frac{kC}{u} \leq T_c,$$

ossia

$$u \geq kr.$$

Dimostrazione. Le k copie contengono complessivamente kC bit. Un upload aggregato u richiede almeno kC/u secondi per trasmetterli, indipendentemente da come la capacità venga suddivisa fra trasferimenti paralleli. Per completare entro T_c deve valere $kC/u \leq T_c$. Sostituendo $C = rT_c$ e dividendo per $T_c > 0$ si ottiene $u \geq kr$. Questo conclude la dimostrazione.

Il bound mostra perché un albero con fan-out elevato richieda nodi con upload elevato. La mesh distribuisce il fan-out su più peer e può pipelinizzare sotto-chunk, ma non elimina la conservazione della capacità totale.

5.6.8 Churn e disponibilità dei chunk

I peer entrano ed escono senza preavviso. La perdita di un nodo è poco rilevante se i chunk che possedeva hanno molte repliche; può essere critica se quel nodo era l'unica sorgente di una parte rara.

In un modello idealizzato, supponiamo che un chunk abbia m repliche su peer differenti e che ciascun peer sia online con probabilità q , in modo indipendente dagli altri. La probabilità che almeno una copia sia disponibile è

$$A_m = 1 - (1 - q)^m.$$

Dimostrazione. Una singola replica è indisponibile con probabilità $1 - q$. Per indipendenza, tutte le m repliche sono contemporaneamente offline con probabilità $(1 - q)^m$. L'evento "almeno una disponibile" è il complementare, quindi ha probabilità $1 - (1 - q)^m$. Questo conclude la dimostrazione.

Per esempio, con $q = 0.8$ e $m = 5$,

$$A_5 = 1 - 0.2^5 = 0.99968.$$

Il numero è volutamente ideale: nella realtà disponibilità e churn sono spesso correlati per area geografica, access network, ora del giorno e comportamento degli utenti. La replica di chunk rari resta comunque il meccanismo fondamentale per ridurre la dipendenza dal singolo peer.

5.6.9 Incentivi e fairness

Nei sistemi di file sharing un peer potrebbe consumare senza mai contribuire. Meccanismi ispirati al *tit-for-tat*, choking e optimistic unchoke premiano chi fornisce upload utile e consentono periodicamente di esplorare nuove relazioni. Il principio ha avuto un ruolo importante nell'ecosistema BitTorrent.

Nel multimedia la stessa politica non può essere applicata meccanicamente. Un terminale mobile può avere downlink elevato e upload limitato; un utente appena entrato non possiede ancora chunk da offrire; un live stream ha deadline che rendono inutile attendere una relazione "equa" se il contenuto sta per scadere. I sistemi ibridi separano quindi spesso incentivi, controllo di servizio e garanzia minima fornita dalla CDN.

5.6.10 NAT, browser e relay

La capacità di upload di un peer è utile soltanto se altri peer possono raggiungerlo. NAT e firewall possono impedire connessioni dirette o rendere alcune coppie molto più difficili di altre.

Nel browser, WebRTC DataChannel fornisce un meccanismo pratico per costruire collegamenti peer-to-peer: ICE raccoglie candidate, STUN aiuta a scoprire percorsi riflessi e TURN offre il relay quando la connessione diretta fallisce. Se il traffico passa attraverso TURN, però, la banda non è più realmente "gratuita" per il servizio: viene consumata capacità infrastrutturale e deve essere contabilizzata in R_I .

WebTransport risolve un problema differente. Offre stream e datagram fra browser e endpoint HTTP/3/QUIC, ma non sostituisce automaticamente ICE come meccanismo generale di NAT traversal peer-to-peer. Può essere utile per collegare client a relay o nodi di servizio e quindi entrare in un'architettura ibrida, senza rendere per questo il collegamento P2P.

5.6.11 Integrità, sicurezza e contenuti ostili

Un peer non controllato non deve poter iniettare arbitrariamente dati nel contenuto. Chunk identificati da hash, manifest autenticati e strutture ad albero di hash permettono di verifi-

care che i dati ricevuti corrispondano a quelli pubblicati. Senza verifica, un peer malevolo può eseguire un *pollution attack*, distribuendo blocchi corrotti che si propagano nello swarm.

Cifratura del trasporto, controllo di accesso e protezione del contenuto restano problemi distinti. Un overlay può essere tecnicamente efficiente ma incompatibile con una particolare politica DRM o con requisiti di privacy che impediscano ai peer di conoscere indirizzi o interessi reciproci.

5.6.12 P2P per on-demand e live

Nel video on-demand il contenuto esiste già integralmente. I peer possono prelevare chunk futuri, scambiare parti rare e conservare dati anche oltre il playhead. Un buffer relativamente profondo permette di recuperare da peer lenti o di usare la CDN soltanto quando la deadline si avvicina.

Nel live esiste invece una finestra mobile. Un chunk troppo vecchio non serve più; uno troppo nuovo può non essere ancora disponibile in nessun peer. Le politiche di scheduling devono mantenere corta la catena di replica, privilegiare la deadline e assicurare che almeno alcuni nodi stabili ricevano rapidamente ogni nuovo chunk.

Architettura	Carico turale	infrastrut- turale	Robustezza al churn	Caratteristiche principali
Client-server	cresce circa con N		alta lato server	controllo semplice e prestazioni prevedibili; scalabilità limitata dall'egress del servizio
CDN	distribuito edge	sugli	molto alta	capacità controllata, cache e prossimità geografica; costo infrastrutturale
P2P ad albero	molto basso nel caso ideale		bassa-media	poco signaling, ma un peer interno può interrompere un intero sottoalbero
P2P mesh/swarm	basso se l'upload utile è elevato		alta	più percorsi e buona elasticità; scheduling e prestazioni dipendono dallo swarm
CDN+P2P	la CDN copre il deficit		molto alta	offload opportunistico con fallback, startup e chunk rari garantiti dall'infrastruttura

Tabella 5.1. Confronto qualitativo delle architetture di distribuzione. Il carico P2P reale dipende da upload, churn, NAT, disponibilità dei chunk e deadline.

5.6.13 Architettura ibrida CDN+P2P

L'ibrido è spesso la soluzione più controllabile. Il player può procedere secondo una logica del tipo:

1. ottiene manifest e bootstrap dall'infrastruttura;
2. scopre peer compatibili e misura la loro capacità;
3. richiede ai peer i chunk con sufficiente slack temporale;
4. usa edge/origin per startup, chunk rari, timeout e deficit di capacità;
5. interrompe un trasferimento P2P se la deadline rende più sicuro il fallback infrastrutturale.

In questo modello il P2P non è una alternativa alla CDN ma un livello di *offload opportunistico*. La qualità minima resta governabile dal servizio, mentre la disponibilità di upload ai

bordi riduce il traffico costoso nei periodi in cui lo swarm è sano.

Le prestazioni vanno quindi misurate almeno mediante offload ratio, startup delay, re-buffering, distanza dal live edge, percentuale di chunk consegnati da peer prima della deadline, traffico TURN/relay e stabilità al churn. Un alto offload ottenuto aumentando troppo il buffer può essere ottimo economicamente ma pessimo per un servizio live: come in tutto il multimedia, una sola metrica non basta.

5.7 HTTP/2, QUIC e HTTP/3

HTTP/2 multiplexa più stream logici su una singola connessione TCP. Il vantaggio è evitare molte connessioni parallele; il limite è che il recupero delle perdite TCP opera sul byte stream globale. Una perdita può quindi arrestare temporaneamente l'avanzamento di più stream HTTP/2 anche se i loro frame logici sono indipendenti.

HTTP/3 usa QUIC. QUIC mantiene congestion control a livello di connessione ma affidabilità e ordinamento a livello di stream. La perdita che blocca uno stream non impedisce al trasporto di consegnare dati già disponibili su stream indipendenti.

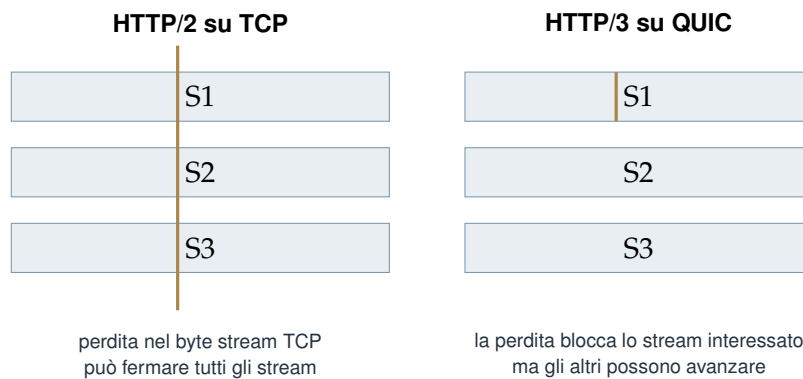


Figura 5.29. Differenza concettuale del head-of-line blocking di trasporto fra HTTP/2/TCP e HTTP/3/QUIC.

QUIC integra TLS nel handshake, supporta connection migration e può riprendere più rapidamente comunicazioni già note. Queste proprietà sono utili nei terminali mobili, ma HTTP/3 non garantisce di per sé una latenza bassa: segmentazione, buffer, encoder e ABR restano determinanti.

5.8 WebTransport

WebTransport fornisce a un'applicazione Web stream unidirezionali, stream bidirezionali e datagram, con sicurezza e multiplexing coerenti con il modello Web. È una primitiva di trasporto, non un media stack: non definisce codec, sincronizzazione audio/video, segnalazione di conferenza o attraversamento NAT come fa WebRTC.

Nel 2026 il binding WebTransport over HTTP/3 è ancora un Internet-Draft IETF in Working Group Last Call, destinato a Proposed Standard. È quindi corretto descriverne architettura e uso, ma non presentarlo come RFC già consolidato.

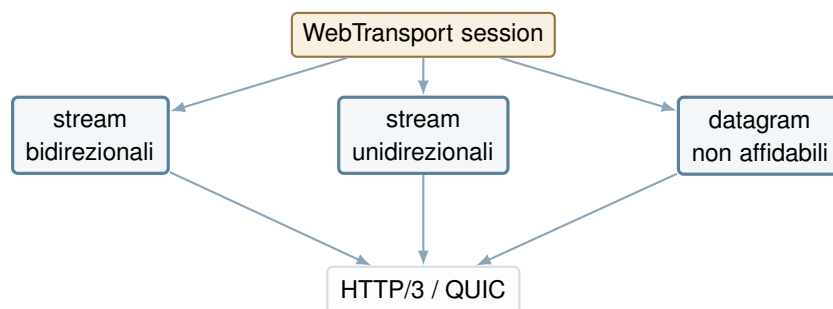


Figura 5.30. WebTransport espone semantiche affidabili e datagram nella stessa sessione, lasciando all'applicazione la semantica del media.

5.9 Media over QUIC

Media over QUIC Transport (MOQT) rappresenta nel 2026 una direzione di sviluppo particolarmente importante. L'obiettivo è fornire un modello publish/subscribe per distribuzione media a bassa latenza, capace di funzionare punto-punto oppure attraverso relay e di sfruttare stream, datagram, priorità e forme di partial reliability del trasporto QUIC o WebTransport.

Al 1 ottobre 2026 MOQT è ancora un Internet-Draft del gruppo IETF MoQ, non un RFC definitivo. La distinzione è essenziale: è corretto studiarne il modello architetturale, ma i dettagli di wire format possono ancora cambiare.

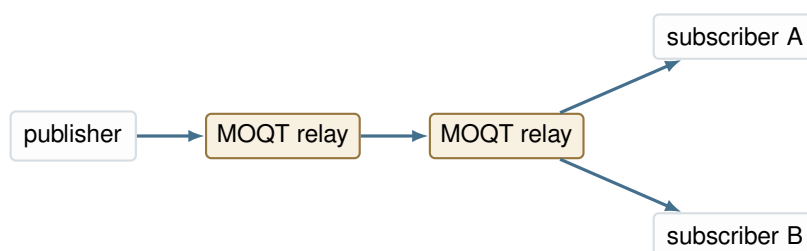


Figura 5.31. Architettura publish/subscribe di Media over QUIC Transport: relay intermedi possono distribuire oggetti media a più subscriber senza trasformarsi necessariamente in endpoint del contenuto.

MOQT non va confuso con "RTP su QUIC". Il suo oggetto logico è la pubblicazione e sottoscrizione di media object organizzati per priorità e dipendenze, adatti a distribuzione scalabile attraverso relay. Nel 2026 esistono inoltre lavori IETF su formati di streaming MOQT e su mapping compatibili con CMAF, anch'essi ancora in forma di Internet-Draft.

5.10 Contribution professionale: SRT e RIST

HLS/DASH ottimizzano soprattutto la distribuzione verso grandi popolazioni di player. Un altro problema è trasportare in tempo reale un feed professionale fra encoder, regia, cloud e distributore su una rete IP non gestita. Qui sono diffusi protocolli di contribution come Secure Reliable Transport (SRT) e Reliable Internet Stream Transport (RIST).

SRT è un protocollo open source orientato al trasporto sicuro e a bassa latenza su reti imprevedibili. Usa recupero selettivo delle perdite, buffer di latenza e cifratura integrata nelle implementazioni conformi ai relativi profili.

RIST è sviluppato in ambito Video Services Forum con forte attenzione all'interoperabilità multi-vendor. Le Technical Recommendations coprono Simple, Main e Advanced Profile e, nel 2026, anche estensioni per scenari ibridi satellite/IP e OTA/IP. La famiglia usa meccanis-

mi di recupero delle perdite e può trasportare flussi professionali su reti non gestite.

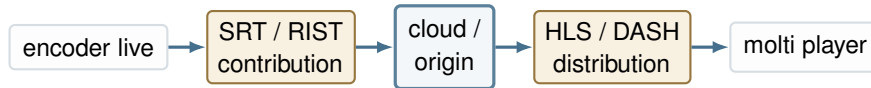


Figura 5.32. Contribution e distribution risolvono problemi differenti: il primo trasporta feed realtime fra nodi professionali, il secondo scala la consegna verso molti utenti finali.

5.11 Dalla QoS di rete all'adattamento applicativo

L'evoluzione descritta nel capitolo mostra tre strategie complementari. La prima cerca di rendere la rete più prevedibile con code, DiffServ, AQM e, in domini controllati, riservazione. La seconda modifica il trasporto: congestion control, QUIC, recovery selettivo e feedback rendono la comunicazione più robusta. La terza adatta direttamente il media: ABR, scalabilità, FEC e buffer reagiscono al servizio realmente ottenuto.

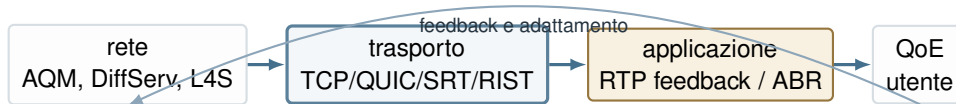


Figura 5.33. La qualità end-to-end nasce dalla cooperazione di rete, trasporto e applicazione, non da un singolo protocollo.

Nessun meccanismo risolve simultaneamente banda, latenza, perdita, scalabilità e qualità percettiva. Un sistema moderno progetta insieme codec, pacchettizzazione, trasporto, buffer, feedback, CDN e comportamento del player.

Sistemi

Un sistema multimediale completo non coincide con un singolo codec o protocollo. Integra almeno rappresentazione della sorgente, compressione, contenitore, sincronizzazione, segnalazione, trasporto, sicurezza e politiche di distribuzione. Le tecnologie cambiano nel tempo, ma queste funzioni restano riconoscibili: proprio per questo le architetture storiche di conferenza sono ancora utili per comprendere WebRTC, streaming e piattaforme moderne.

Il capitolo segue due fili paralleli. Il primo ricostruisce l'evoluzione dei sistemi di conferenza, dalle reti a circuito a H.323, SIP e WebRTC. Il secondo studia come audio e video vengono organizzati in contenitori, partendo da RIFF/AVI e arrivando a ISO-BMFF/MP4, CMAF, Matroska/WebM e alle API Web contemporanee.

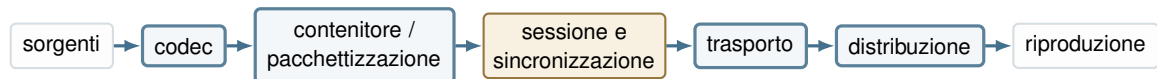


Figura 6.1. Funzioni fondamentali di un sistema multimediale. Gli standard storici e moderni differiscono soprattutto nel modo in cui distribuiscono queste funzioni.

6.1 Sistemi di conferenza su reti a circuito e a servizio garantito

Le raccomandazioni H.32x nacquero quando reti diverse offrivano caratteristiche di trasporto molto differenti. Invece di assumere un substrato IP comune, ciascuna famiglia combinava codec, multiplexing, controllo e segnalazione adatti alla rete sottostante. La Tabella 6.1 riassume le architetture storiche principali. La colonna relativa alle reti a larga banda basate su B-ISDN/ATM è mantenuta esclusivamente come riferimento storico: ATM non è parte dell'architettura multimediale moderna descritta nel resto del libro.

	H.320	H.324	H.321	H.310	H.322
Rete	ISDN	PSTN modem	B-ISDN/ATM storico	B-ISDN/ATM storico	LAN con QoS
Audio	G.711, G.728	G.722, G.723.1, G.729	G.711, G.728	G.722, G.711, G.728, MPEG	G.711, G.728, G.722
Video	H.261	H.261/H.263	H.261	H.261/MPEG-2	H.261
Dati	T.120	T.120	T.120	T.120	T.120
Mux	H.221	H.223	H.221	H.221/H.222	H.221
Controllo	H.242	H.245	H.242	H.245	H.242
Chiamata	Q.931	V.25ter / mo-dem	Q.931	Q.931	Q.931

Tabella 6.1. Architetture storiche della famiglia H.32x. La tabella serve a confrontare le funzioni, non a suggerire l'impiego corrente di queste tecnologie.

6.1.1 H.320

H.320 definisce terminali e procedure per comunicazioni audiovisive su ISDN a banda stretta. Il rate disponibile è costruito aggregando canali B da 64 kbit/s; la raccomandazione deve quindi coordinare codec, multiplexing e segnalazione entro una capacità nota e relativamente stabile.

L'architettura separa chiaramente le funzioni. H.261 codifica il video CIF/QCIF; G.711, G.722 o G.728 rappresentano differenti opzioni audio; T.120 gestisce collaborazione e dati; H.221 multiplexa i flussi; H.242 negozia capacità e modalità operative; Q.931 stabilisce e rilascia la chiamata ISDN. Questa decomposizione anticipa la separazione moderna fra codec, trasporto dei media e segnalazione.

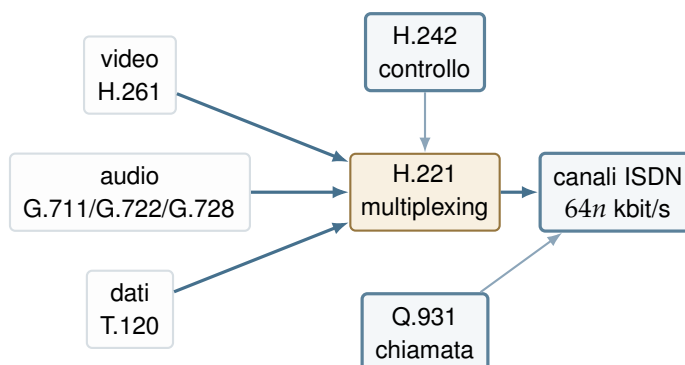


Figura 6.2. Architettura funzionale H.320. Media, dati, controllo e segnalazione sono standardizzati separatamente e poi integrati sul collegamento ISDN.

T.120

La famiglia T.120 definisce protocolli per collaborazione dati in una conferenza multipunto. L'obiettivo è rendere interoperabili applicazioni e terminali differenti fornendo controllo della conferenza, trasporto affidabile dei dati e servizi condivisi indipendenti, per quanto possibile, dalla rete sottostante. L'architettura è suddivisa in più raccomandazioni, ciascuna dedicata a una funzione specifica:

- T.121 definisce un Generic Application Template (GAT) che specifica un insieme di linee guida da utilizzare nello sviluppo dei protocolli applicativi e nella gestione del controllo degli applicativi. In particolare in T.121 viene specificato come registrare un utente nella conferenza, come consentire politiche locali e remote di gestione, come negoziare le caratteristiche applicative con gli altri utenti.
- T.122 specifica come interoperare in una struttura multipunto, in modo da consentire a uno o più partecipanti l'invio di dati a tutti gli afferenti alla conferenza.
- T.123 si occupa del trasporto dei dati sulla rete tramite un'azione di sequenzializzazione, numerazione e controllo di flusso. Il T.123 è anche responsabile delle azioni di connessione, disconnessione, invio e ricezione dei dati definendo una serie di profili e tecniche di correzione degli errori.
- T.124 fornisce un Generic Conference Control (GCC) per inizializzare ed amministrare la conferenza multipunto. T.124 raccoglie le informazioni di controllo della conferenza, ammette utenti e dati e mantiene lo stato generale della sessione, compresi l'elenco dei partecipanti e le applicazioni disponibili per ciascuno di essi.
- T.125 specifica come devono essere trasmessi i dati all'interno della conferenza, tramite canali applicativi privati o broadcast, garantendo una comunicazione accurata ed efficiente.
- T.126 descrive come devono essere scambiati i dati per una lavagna condivisa (whiteboard), sia in formato compresso che in formato non compresso, specificando il meccanismo di aggiornamento delle informazioni sulle lavagne degli utenti.
- T.127 specifica lo scambio di file nella conferenza, consentendo di trasmettere uno o più file, in formato nativo o compresso, a uno o più partecipanti.

- T.128 specifica in che modo i partecipanti alla conferenza possono condividere programmi in modo da effettuare un lavoro collaborativo.

L'architettura T.120 prevede la presenza di un sistema responsabile della gestione di tutti gli eventi, garantendo che i comandi siano correttamente interpretati ed eseguiti nell'ordine corretto. Le topologie di rete possibili sono 3: a stella, in cascata ad albero e con rilancio da una stazione a un'altra.

H.221

Il sistema di multiplexing è gestito dal protocollo H.221 che prevede una tecnica di tipo Time Division Multiplexing (TDM) per impacchettare e mantenere sincronizzati i flussi audio, video e dati. H.221 è in grado di gestire canali da 64 kbit/s fino a 1920 kbit/s.

H.242

Il sistema di controllo è basato su H.242 e viene utilizzato per negoziare le capacità dei terminali, i codec e i relativi parametri, le modalità audio e video e i meccanismi per aprire e chiudere i canali di comunicazione. Inoltre consente di gestire la comunicazione e i vari eventi che possono accadere durante la comunicazione stessa.

Q.931

L'instaurazione della chiamata usa la segnalazione Q.931 e coinvolge tre entità: chiamante, rete e chiamato. I messaggi del protocollo gestiscono setup, avanzamento della chiamata, connessione e rilascio delle risorse al termine della comunicazione.

I messaggi che caratterizzano Q.931 sono i seguenti: alerting, per indicare che è stata originata una chiamata; call proceeding, cioè chiamata in corso; connect, cioè la procedura di chiamata è andata a buon fine e la chiamata è stata accettata; connect acknowledge, per indicare l'avvenuta connessione; setup, per inviare il messaggio di invito alla comunicazione; setup acknowledge, per indicare che il setup è stato ricevuto; suspend, per sospendere la comunicazione in corso e utilizzare il canale di comunicazione per effettuare un'altra comunicazione; suspend acknowledge, per indicare la corretta ricezione della sospensione; suspend reject, che indica la non accettazione della sospensione della comunicazione; resume, per riprendere una comunicazione sospesa; resume acknowledge, per confermare la riattivazione di una comunicazione sospesa; resume reject, che indica che la comunicazione sospesa non può essere ripresa; disconnect, per chiudere la comunicazione; release, che indica che il canale sta per essere rilasciato; release complete, che indica il completamento della fase di rilascio della linea; status enquiry, per verificare lo stato della rete.

Un esempio di una sessione di comunicazione gestita da Q.931, con U che identifica l'utente e N che identifica la rete, è il seguente: setup UN, call proceeding NU, alerting NU, connect NU, scambio dei dati, disconnect UN, release NU, release complete UN.

6.1.2 H.324

H.324 nasce per comunicazioni multimediali a bassa bitrate sulla rete telefonica generale commutata. Il profilo storico usa modem V.34, con rate di segnalazione fino a 33.6 kbit/s. Revisioni successive della raccomandazione hanno esteso il quadro anche a interfacce ISDN e reti mobili, ma il caso PSTN/V.34 resta quello che spiega le scelte iniziali di forte compressione e multiplexing a overhead contenuto.

Il supporto di ciascun tipo di media è opzionale, ma quando audio, video o dati sono presenti devono esistere modi comuni che permettano interoperabilità fra terminali. Le scelte tipiche comprendono G.723.1 per la voce, H.263 per il video e T.120 per i dati; versioni e profili possono supportare ulteriori codec. H.223 multiplexa i canali logici e H.245 negozia ca-

pacità e controllo. La chiamata sul modem utilizza le procedure della famiglia V.25ter/V.250 insieme al controllo del modem, non una segnalazione ISDN Q.931 end-to-end.

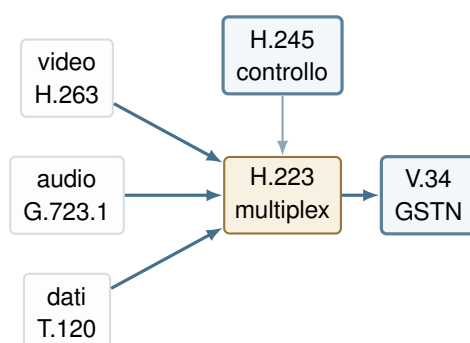


Figura 6.3. Architettura H.324 nel caso storico GSTN/V.34. La bassa capacità del modem rende centrale la condivisione dinamica fra media e controllo.

H.223

Il multiplexing è gestito da H.223. La bitrate del canale è divisa in un numero certo di canali logici, virtualmente separati uno dall'altro. Il flusso è organizzato in unità all'interno delle quali i diversi canali logici, associati ai media e al controllo, vengono multiplexati secondo le regole H.223. La ripartizione della capacità può quindi variare nel tempo in funzione dei canali attivi e dei rispettivi requisiti.

H.223 può associare ai canali logici differenti livelli di protezione e adattamento all'errore. La reazione a una corruzione dipende dal media e dal profilo: i dati affidabili possono richiedere meccanismi di recupero, mentre audio e video realtime possono preferire concealment o continuazione del flusso per non violare la deadline. Protezione e ritrasmissione non sono quindi proprietà uniformi di ogni canale H.223.

H.245

Il controllo è gestito dal protocollo H.245. I controlli di H.245 sono trasportati sui canali di controllo H.245 che, a differenza dei canali legati ai media, sono sempre disponibili ed aperti. I messaggi sono relativi alle caratteristiche e capacità dei terminali e alla gestione dei canali dei media in termini di apertura, chiusura e negoziazione.

Le funzioni di cui si occupa H.245 sono: stabilire i requisiti della comunicazione tra sorgente e destinazione; gestire le differenti funzioni di master e slave nella comunicazione in modo da evitare conflitti operativi; gestione delle capacità dei terminali, indicando che tipo di media possono essere processati; gestione dei canali logici; gestioni dei comandi di cambiamento delle caratteristiche della conferenza in corso; controllo di flusso mediante vincoli di bitrate su ogni singolo canale logico; gestione di segnalazione verso punti multipli; gestione dell'instaurazione della conferenza, mediante password e moderatori; gestione della chiusura della conferenza.

I messaggi principali di H.245 sono: master-slave determination, per determinare chi è master e chi è slave; terminal capability set, con informazioni relative alle capacità multimediali dei terminali, open logical channel, per aprire un canale di comunicazione logico; close logical channel, per chiudere un canale di comunicazione logico; request mode, per richiedere al terminale un particolare modo di funzionamento; send terminal capability set, per forzare un terminale corrispondente ad indicare le sue capacità multimediali; end session command, per terminare la sessione.

6.1.3 H.321

H.321 è una raccomandazione storica che adatta l'architettura H.320 a servizi audiovisivi su B-ISDN a larga banda. I codec audio previsti comprendono G.711, G.722 e G.728; il codec video di riferimento è H.261. La parte dati è gestita da T.120, il multiplexing da H.221 e il controllo da H.242, mentre l'instaurazione della chiamata usa Q.931.

6.1.4 H.310

H.310 è una raccomandazione storica per sistemi audiovisivi broadband su B-ISDN. Rispetto a H.321 ammette un insieme più ampio di codec e meccanismi di multiplexing, inclusi strumenti MPEG. Fra le opzioni audio compaiono G.711, G.722, G.728 e MPEG-1 Audio; per il video sono previsti H.261 e strumenti MPEG-2 Video. I dati sono gestiti da T.120. Il multiplexing può avvenire con H.221 oppure con H.222. Il controllo è affidato a H.245 mentre l'instaurazione della chiamata a Q.931. H.222 appartiene alla famiglia MPEG-2 Systems e svolge funzioni di multiplexing adatte a transport/program stream; non è semplicemente una nuova versione di H.221.

6.1.5 H.322

H.322 copre sistemi videotelefonici su LAN configurate e gestite per fornire una qualità di servizio garantita equivalente a quella assunta da H.320. L'ipotesi è quindi molto più forte del normale best-effort: la rete sottostante deve rendere non necessari ulteriori meccanismi di protezione o recupero oltre a quelli già richiesti ai terminali H.320.

L'architettura conserva perciò molti elementi della famiglia H.320: audio G.711, G.722 e G.728, video H.261, collaborazione T.120, multiplexing H.221 e controllo H.242. Storicamente questa soluzione rappresenta il tentativo di portare il modello quasi-circuitale su una LAN con QoS garantita. H.323 affronta invece LAN e reti IP che non forniscono necessariamente tale prestazione deterministica.

6.2 Conferenze a commutazione di pacchetto

I sistemi a commutazione di pacchetto hanno assunto un ruolo fondamentale grazie al successo e alla diffusione di Internet, basata sul protocollo IP. IP viene utilizzato sia in ambito locale che geografico, senza alcuna distinzione di funzionamento. Un punto chiave che differenzia fortemente la commutazione di circuito da quella di pacchetto è la tariffazione, che risulta tipicamente legata al tempo di connessione nella commutazione di circuito mentre è funzione del numero dei byte o pacchetti scambiati nel caso della commutazione di pacchetto. Inoltre, è da sottolineare che un modello economico vincente e diffuso per la commutazione di pacchetto è una tariffazione flat, indipendente dal traffico effettivo. Questo modello ha il vantaggio di offrire all'utente connettività in attesa che venga riempita di servizi, per cui stimola l'utente ad utilizzare la rete anche per servizi che convenzionalmente sono stati forniti da altre tipologie di reti.

6.2.1 H.323

H.323 definisce un sistema multimediale per reti a pacchetto che non forniscono necessariamente la qualità deterministica assunta da H.322. La famiglia combina segnalazione di chiamata, registrazione e admission control, negoziazione delle capacità e media RTP/RTCP.

Quattro componenti logici ricorrono nell'architettura:

- **terminal:** endpoint audio/video/dati. L'audio è la funzione fondamentale; video e collaborazione sono opzionali;
- **gateway:** traduce segnalazione e, quando necessario, media fra H.323 e reti o sistemi

differenti;

- **gatekeeper:** gestisce una zona H.323 e può fornire address translation, admission control, bandwidth control, instradamento della segnalazione, accounting e policy;
- **MCU:** supporta conferenze multiparty mediante funzioni di controllo e, a seconda dell'architettura, mixing o switching dei media.

Un alias H.323 può essere mappato verso un indirizzo di trasporto o verso numerazione E.164. E.164 limita a 15 cifre il numero internazionale completo, includendo country code e numero nazionale: non si sommano quindi 15 cifre nazionali alle 1–3 cifre del country code.

I media audio/video usano RTP e RTCP. H.245 negozia capacità, relazione master/slave e logical channel. H.225 comprende due funzioni distinte: RAS fra endpoint e gatekeeper e segnalazione di chiamata basata su messaggi della famiglia Q.931.

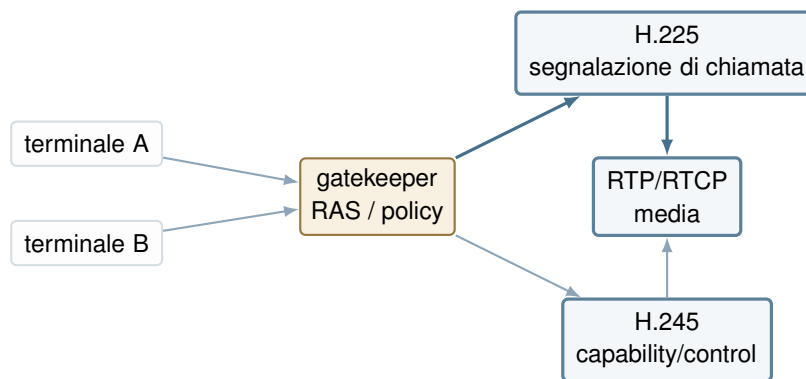


Figura 6.4. Separazione funzionale H.323 fra registrazione/ammissione, segnalazione H.225, controllo H.245 e trasporto RTP/RTCP.

RAS e segnalazione di chiamata H.225

Il canale RAS gestisce discovery del gatekeeper, registrazione, ammissione, variazione di banda, disengage e stato. Un endpoint può conoscere staticamente il proprio gatekeeper oppure scoprirlo mediante messaggi di discovery. Registrarsi nella zona e ottenere autorizzazione per una specifica chiamata sono eventi distinti.

Per una chiamata in uscita, un terminale registrato può inviare Admission Request (ARQ). Il gatekeeper risponde con Admission Confirm (ACF) oppure Admission Reject (ARJ). L'ACF può anche indicare se il call la segnalazione deve fluire direttamente fra endpoint o essere instradata attraverso il gatekeeper.

La segnalazione di chiamata H.225 usa messaggi come Setup, Call Proceeding, Alerting, Connect e Release Complete. Il gatekeeper può rimanere fuori dalla segnalazione dopo l'ammissione oppure restare sul percorso per policy, controllo o accounting.

H.245

H.245 controlla la sessione multimediale una volta creato il contesto di chiamata. Le funzioni principali sono:

- TerminalCapabilitySet per scambiare codec, formati e capacità;
- MasterSlaveDetermination per risolvere ruoli in procedure che richiedono una decisione univoca;
- OpenLogicalChannel e CloseLogicalChannel per creare e rilasciare canali media con i relativi parametri di trasporto;
- RequestMode e comandi di flow control per modificare il modo operativo durante la

sessione;

- EndSessionCommand per terminare il controllo H.245.

Versioni successive della famiglia H.323 hanno introdotto procedure di *fast start* e tunneling di H.245 nella segnalazione H.225 per ridurre round trip e tempo di setup. Il principio è anticipare o incorporare parte della negoziazione media invece di attendere l'apertura completa di un canale H.245 separato.

Esempio di call flow

Consideriamo due endpoint T1 e T2 già registrati al gatekeeper GK e una modalità in cui il segnalazione di chiamata può essere diretta.

1. T1 invia ARQ a GK per chiedere l'ammissione della chiamata;
2. GK risponde ACF e fornisce i dati necessari al routing;
3. T1 invia H.225 Setup a T2;
4. T2 può inviare Call Proceeding e Alerting;
5. T2 richiede a sua volta l'ammissione al proprio gatekeeper e riceve ACF;
6. T2 invia Connect a T1;
7. T1 e T2 scambiano TerminalCapabilitySet e completano le procedure H.245 necessarie;
8. vengono aperti i logical channel per audio/video e comunicati gli indirizzi RTP/RTCP;
9. il media fluisce via RTP e il feedback via RTCP;
10. al termine i logical channel vengono chiusi, H.245 termina la sessione e H.225 usa Release Complete;
11. gli endpoint notificano il disengage al gatekeeper mediante RAS.

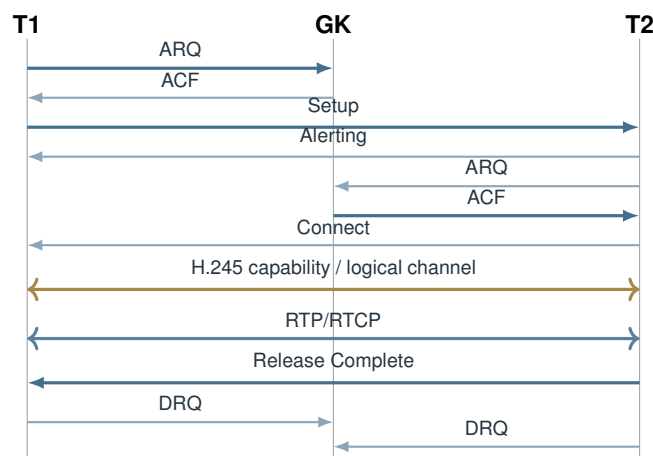


Figura 6.5. Call flow H.323 semplificato con endpoint già registrati. RAS, segnalazione di chiamata, H.245 e media sono fasi e canali logicamente distinti.

H.323 e VoIP

H.323 è stato uno dei primi ecosistemi completi per Voice over IP e videoconferenza interoperabile. Ha avuto particolare rilevanza in reti enterprise, gateway verso PSTN e sistemi di videoconferenza dedicati. La sua architettura è più ampia di una semplice chiamata telefonica: include registrazione, admission control, capability negotiation e conferenze multiparty.

Il terminale può essere un telefono IP, un apparato di sala, un gateway o un softphone. La mobilità logica deriva dal fatto che identità e alias non devono coincidere permanentemente

con un'interfaccia fisica; un endpoint può registrarsi da un nuovo indirizzo purché politiche e segnalazione lo consentano.

SIP ha successivamente acquisito un ruolo dominante in molte architetture VoIP perché adotta un modello testuale più vicino ai protocolli Internet e separa in modo netto segnalazione e descrizione SDP. H.323 resta però didatticamente utile perché mostra in forma esplicita funzioni che nei sistemi moderni esistono ancora: discovery, registrazione, admission, capability exchange, media control e multipoint conferencing.

6.2.2 SIP e SDP

Session Initiation Protocol (SIP) è il protocollo IETF testuale usato per creare, modificare e terminare sessioni. Non trasporta il media e non prescrive un codec: trasporta segnalazione e, normalmente, descrizioni SDP che permettono agli endpoint di concordare come comunicare.

Questa separazione è fondamentale. SIP risponde a domande come “chi sto chiamando?”, “dove si trova?”, “ha accettato?”; SDP descrive media e parametri; RTP/RTCP o altri trasporti portano i campioni. Un proxy SIP può quindi partecipare a tutta la segnalazione senza trovarsi sul percorso RTP.

User Agent, proxy, registrar e redirect

Un endpoint SIP è uno User Agent. Per una specifica transazione assume il ruolo di User Agent Client (UAC) quando genera una richiesta e User Agent Server (UAS) quando la riceve. I ruoli sono per transazione, non per l'intera sessione: durante un dialogo lo stesso endpoint può generare nuove richieste e diventare UAC per quelle transazioni.

Le funzioni infrastrutturali più comuni sono:

- il **registrar**, che associa temporaneamente un Address of Record a uno o più Contact;
- il **proxy**, che instrada richieste e risposte e può applicare autenticazione, policy, forking e record-route;
- il **redirect server**, che non inoltra la richiesta verso il destinatario ma restituisce al chiamante uno o più contatti da provare direttamente.

Un SIP URI può avere forma sip:utente@dominio. Identità telefoniche E.164 possono essere integrate mediante gateway e procedure di mapping, ma un SIP URI non è semplicemente un numero telefonico travestito: identifica una risorsa nel modello di routing SIP.

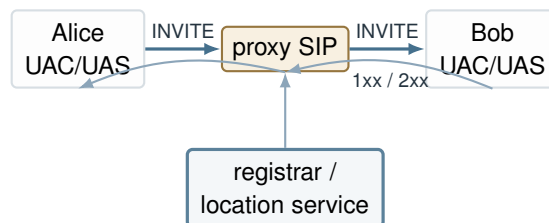


Figura 6.6. Proxy SIP: la segnalazione attraversa l'infrastruttura, mentre il percorso del media può essere differente.

Transazioni, dialoghi e sessioni

Una *transaction* SIP associa una richiesta alle relative risposte. Un *dialog* è una relazione persistente fra due User Agent identificata da Call-ID e tag locali/remoti e mantiene il contesto di richieste successive. La *media session* descritta da SDP è ancora un oggetto differente. Separare i tre concetti evita di confondere stato SIP e stato RTP.

INVITE crea o modifica una sessione; ACK conferma la risposta finale 2xx all'INVITE oppure completa la transaction INVITE nei casi di risposta non-2xx; BYE termina un dialogo stabilito; CANCEL annulla una richiesta pendente senza terminare un dialogo già accettato; OPTIONS interroga capacità; REGISTER aggiorna la location; SUBSCRIBE e NOTIFY formano il framework di eventi quando supportato.

Le risposte sono raggruppate per classe:

Classe	Significato	Esempi
1xx	provisional	100 Trying, 180 Ringing, 183 Session Progress
2xx	success	200 OK
3xx	redirection	301 Moved Permanently, 302 Moved Temporarily
4xx	client/request failure	400 Bad Request, 401 Unauthorized, 404 Not Found, 486 Busy Here, 488 Not Acceptable Here
5xx	server failure	500 Server Internal Error, 503 Service Unavailable
6xx	global failure	600 Busy Everywhere, 603 Decline

Tabella 6.2. Classi principali delle risposte SIP. Le singole response code appartengono alla specifica del protocollo e alle sue estensioni.

Call flow INVITE

Un flusso semplificato attraverso un proxy stateful è:

1. Alice invia INVITE al proxy;
2. il proxy localizza Bob e inoltra l'INVITE;
3. Bob può inviare risposte provvisorie, per esempio 180 Ringing;
4. Bob accetta con 200 OK, normalmente contenente la risposta SDP se l'offerta era nell'INVITE;
5. Alice invia ACK;
6. il media fluisce secondo i parametri negoziati;
7. uno dei due endpoint invia BYE; l'altro conferma con 200 OK.

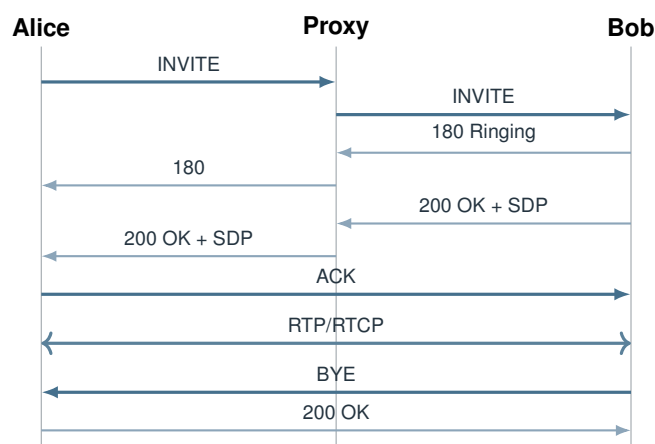


Figura 6.7. Call flow SIP semplificato. Il proxy partecipa alla segnalazione; il media RTP può fluire direttamente fra gli endpoint.

Con un redirect server il primo INVITE riceve una risposta 3xx con un nuovo Contact. Il chiamante può quindi emettere un nuovo INVITE verso quell'indirizzo. Il redirect non rimane necessariamente sul percorso delle successive transazioni.

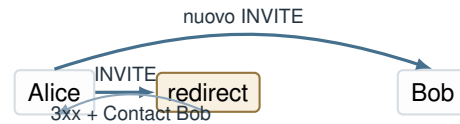


Figura 6.8. Redirect SIP: il server restituisce una nuova destinazione, ma non deve inoltrare il dialogo successivo.

Esempio di messaggio

Un INVITE minimale può assumere la forma seguente; gli identificatori e gli indirizzi sono puramente illustrativi.

```

INVITE sip:bob@example.net SIP/2.0
Via: SIP/2.0/UDP client.example.org;branch=z9hG4bK-123
From: Alice <sip:alice@example.org>;tag=a73kszlfl
To: Bob <sip:bob@example.net>
Call-ID: 93j2@example.org
CSeq: 1 INVITE
Contact: <sip:alice@client.example.org>
Content-Type: application/sdp
Content-Length: ...
  
```

```

v=0
o=alice 2890844526 2890844526 IN IP4 192.0.2.10
s=-
c=IN IP4 192.0.2.10
t=0 0
m=audio 49170 RTP/AVP 111
a=rtpmap:111 opus/48000/2
  
```

Il body SDP non modifica la semantica SIP: SIP trasporta la descrizione, mentre il modello offer/answer stabilisce come due descrizioni vengono confrontate per trovare parametri compatibili.

SDP e offer/answer

Session Description Protocol (SDP) è un formato testuale per descrivere sessioni multimediali. Non è un protocollo di trasporto e non crea da solo una chiamata. Le linee principali includono:

- v= versione SDP;
- o= identificazione dell'origine e versione della descrizione;
- s= nome della sessione;
- t= intervallo temporale;
- c= informazioni di connessione;
- m= media description, porta, protocollo e formati;
- a= attributi, inclusi mapping RTP e parametri del codec.

Nel modello offer/answer un endpoint propone un insieme di media e formati; l'altro risponde selezionando una configurazione compatibile o rifiutando singole media description. Non è corretto descrivere la negoziazione come semplice "scelta del codec minimo comune": possono intervenire direzioni send/recv, payload type dinamici, parametri fmtp, ICE, sicurezza e più stream simultanei.

SIP/SDP restano importanti anche quando il media usa codec e reti molto più moderni

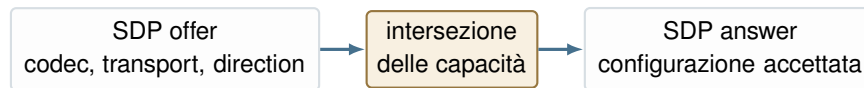


Figura 6.9. Offer/answer SDP: la risposta seleziona parametri compatibili per ciascun media stream senza trasformare SDP in un protocollo di trasporto.

rispetto a quelli disponibili alla nascita dei protocolli: la segnalazione e la descrizione della sessione evolvono in modo in parte indipendente dalla codifica audio/video.

6.3 WebRTC

WebRTC porta comunicazioni audio, video e dati realtime nel modello di sicurezza e interoperabilità del Web. Non è un singolo protocollo: è un'architettura che combina acquisizione dei media, negoziazione, attraversamento NAT, trasporto RTP/RTCP protetto, controllo di congestione e data channel.

Una distinzione fondamentale riguarda la segnalazione applicativa. WebRTC non impone come due browser debbano scambiarsi SDP e candidate ICE: un'applicazione può usare WebSocket, HTTP, SIP o un protocollo proprio. Una volta negoziata la sessione, il percorso dei media è indipendente dal server di segnalazione, salvo che ICE debba ricorrere a TURN o che l'architettura multiparty usi una SFU/MCU.

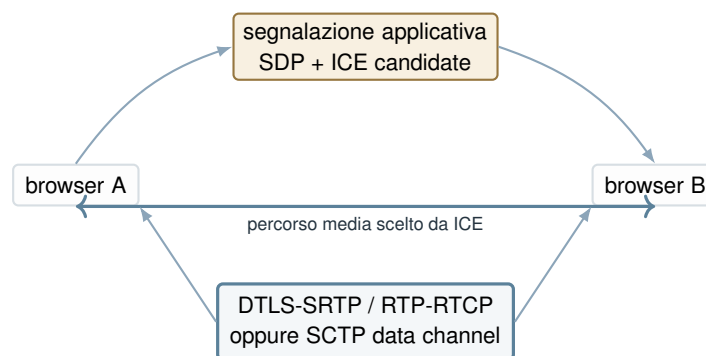


Figura 6.10. Separazione WebRTC fra segnalazione applicativa e percorso dei media. Il server che scambia SDP non deve necessariamente trasportare RTP.

6.3.1 ICE, STUN e TURN

Un endpoint dietro NAT conosce i propri indirizzi locali ma non può assumere che siano raggiungibili dal peer. Interactive Connectivity Establishment (ICE) raccoglie più *candidate* e prova coppie di percorsi mediante connectivity checks.

Le tre classi concettuali più importanti sono:

- **host candidate:** indirizzo direttamente disponibile sull'host;
- **server-reflexive candidate:** indirizzo/porta osservati tramite STUN dall'esterno del NAT;
- **relayed candidate:** indirizzo allocato su un server TURN, attraverso il quale viene inoltrato il traffico.

STUN aiuta a scoprire come una comunicazione appare all'esterno ma non è un relay. TURN fornisce invece un relay esplicito e consuma banda server-side. ICE preferisce percorsi diretti quando funzionano e usa il relay quando le policy di NAT/firewall rendono impossibile la connettività diretta.

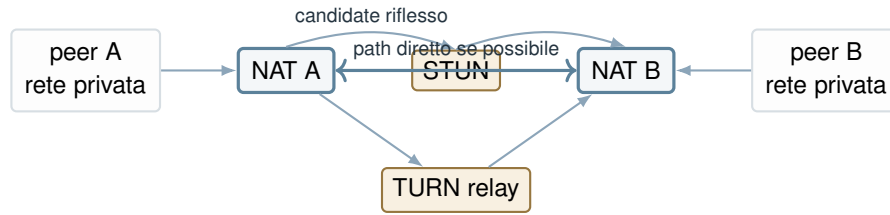


Figura 6.11. ICE prova percorsi host/server-reflexive e può ripiegare su TURN quando il path diretto non è utilizzabile.

Con *trickle ICE* i candidate possono essere inviati al peer man mano che vengono scoperti, senza attendere la fine dell'intera fase di gathering. Questo riduce il tempo di setup senza cambiare il principio di selezione delle coppie candidate.

6.3.2 Sicurezza dei media

La cifratura dei media è parte integrante del modello WebRTC. RTP e RTCP vengono protetti da SRTP/SRTCP; il materiale crittografico viene negoziato tramite DTLS-SRTP. Le tre funzioni restano distinte: RTP definisce sequenza e timing, SRTP protegge il flusso, DTLS autentica la sessione crittografica e deriva le chiavi necessarie.

Il fingerprint del certificato DTLS viene normalmente trasportato nella descrizione SDP. In questo modo la segnalazione collega l'identità della sessione negoziata al canale DTLS che sarà effettivamente stabilito sul percorso ICE.

6.3.3 Negoziazione, RTP/RTCP e adattamento

Le API WebRTC usano il modello offer/answer per negoziare codec, extension header, direzioni delle tracce e parametri di trasporto. ICE candidate e descrizioni SDP possono essere aggiornati mentre la sessione evolve; aggiungere o rimuovere tracce non equivale necessariamente a creare una nuova connessione di rete.

Il media usa RTP/RTCP protetto. Feedback RTCP e RTP header extension possono fornire informazioni su perdita, keyframe necessari, timing di trasporto e condizioni di rete. L'algoritmo di congestion control del sender usa queste osservazioni per adattare la bitrate; non è RTP da solo a imporre un particolare algoritmo di controllo.

In caso di perdita, un ricevitore può chiedere la ritrasmissione di pacchetti recenti quando la deadline lo consente oppure richiedere un nuovo frame intra se il riferimento video è diventato inutilizzabile. La scelta fra NACK, FEC, keyframe e concealment è quindi guidata dal tempo residuo e dalla struttura del codec.

6.3.4 DataChannel

WebRTC non trasporta soltanto media. RTCDataChannel usa SCTP sopra DTLS e il percorso ICE per offrire più stream logici all'applicazione. Può fornire consegna ordinata e affidabile, oppure modalità unordered e partial-reliability quando l'applicazione preferisce scartare dati vecchi piuttosto che attendere indefinitamente la loro ritrasmissione.

6.3.5 Conferenze multiparty: mesh, SFU e MCU

Con due partecipanti il peer-to-peer è naturale; con molti partecipanti il fan-out diventa il problema dominante. Supponiamo N partecipanti e un flusso video di bitrate r per partecipante. In una mesh completa ogni peer invia una copia agli altri $N - 1$ peer:

$$R_{mesh,peer} = (N - 1)r.$$

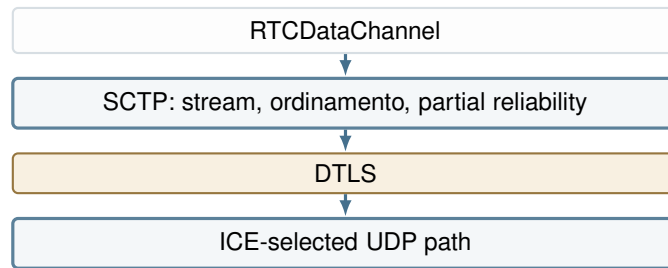


Figura 6.12. Stack concettuale del WebRTC DataChannel. La semantica dei dati applicativi è separata da RTP/SRTP.

Il numero complessivo di flussi diretti è

$$M_{mesh} = N(N - 1).$$

Dimostrazione. Ogni partecipante ha $N - 1$ destinatari e invia a ciascuno un flusso di bitrate r , quindi l'upload è $(N - 1)r$. Ripetendo il conteggio per i N mittenti si ottengono $N(N - 1)$ flussi diretti. Questo conclude la dimostrazione.

Con una Selective Forwarding Unit ogni client invia, nel modello più semplice, una sola copia del flusso alla SFU:

$$R_{SFU,peer} = r.$$

Dimostrazione. L'unico destinatario upstream del peer è la SFU; viene trasmessa una sola copia di bitrate r . Il fan-out non scompare: se la SFU inoltra ogni sorgente a tutti gli altri, l'infrastruttura produce ancora $N(N - 1)$ flussi downstream. Questo conclude la dimostrazione.

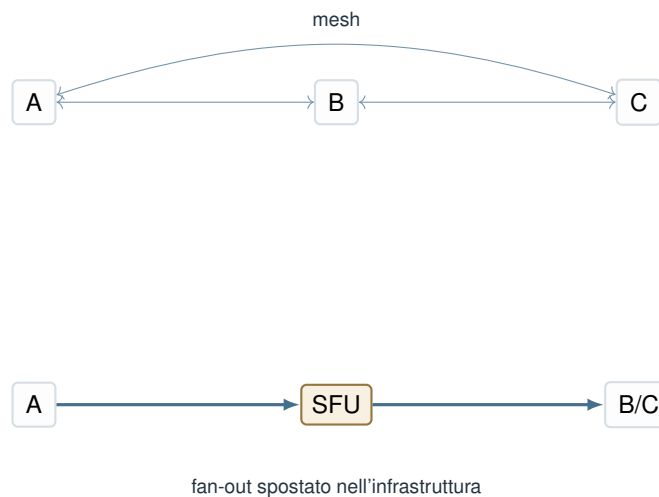


Figura 6.13. Mesh e SFU: la SFU riduce l'upload dei client ma assume il costo di inoltra verso i destinatari.

La SFU può sfruttare *simulcast*, nel quale il sender produce più representations indipendenti, oppure Scalable Video Coding (SVC), nel quale layer temporali/spaziali/qualitativi dipendono da un base layer. Il server seleziona ciò che ogni destinatario può usare in funzione di banda, layout e capacità di decodifica.

Una MCU esegue un lavoro differente: decodifica i flussi, li compone o transcodifica e produce un nuovo stream. Può alleggerire i terminali ma introduce elaborazione, latenza e perdita generazionale sul server.

6.3.6 WebRTC e WebTransport

WebRTC e WebTransport sono complementari. WebRTC fornisce una soluzione integrata per media realtime: ICE/TURN, sicurezza, RTP/RTCP, feedback, jitter buffering e API di traccia. WebTransport espone invece stream e datagram generici sopra HTTP/3/QUIC e lascia all'applicazione la semantica del protocollo.

Una videoconferenza interoperabile trae quindi vantaggio dallo stack WebRTC esistente. Un'applicazione sperimentale che vuole definire pacchettizzazione, timing e affidabilità proprie può usare WebTransport, magari insieme a WebCodecs; deve però progettare esplicitamente ciò che WebRTC già fornisce.

6.4 Contenitori multimediali

Un codec e un contenitore risolvono problemi differenti. Il codec rappresenta in forma compressa campioni audio o video; il contenitore organizza uno o più stream, ne descrive formato, timeline, indici, metadati e relazioni. Lo stesso codec può quindi essere trasportato in contenitori differenti, e lo stesso contenitore può ospitare codec diversi.

Questa separazione è fondamentale per leggere l'evoluzione da AVI a MP4, CMAF e WebM. I principi ricorrenti sono:

- identificare ogni stream e il codec associato;
- rappresentare il tempo in una scala numerica esplicita;
- associare campioni e timestamp;
- permettere seek e accesso casuale mediante indici;
- separare payload, metadati e struttura;
- supportare file completo oppure consegna progressiva/frammentata.

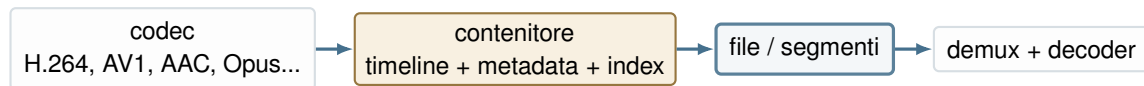


Figura 6.14. Codec e contenitore sono livelli distinti: il contenitore non sostituisce la codifica dei campioni e il codec non descrive da solo la struttura del file.

6.4.1 RIFF e AVI

Audio Video Interleave (AVI) nasce nell'ecosistema RIFF, Resource Interchange File Format. RIFF organizza il file come una gerarchia di chunk identificati da FourCC, codici di quattro byte leggibili come RIFF, LIST, avi h, movi e così via.

Un chunk RIFF elementare contiene:

- FourCC di 4 byte;
- campo size di 4 byte;
- payload di lunghezza indicata;
- eventuale byte di padding per mantenere l'allineamento a parola.

Se il payload contiene N byte, lo spazio occupato dal chunk, trascurando strutture esterne, è

$$S_{\text{chunk}} = 8 + N + (N \bmod 2).$$

Dimostrazione. L'header usa 8 byte. Se N è pari non serve padding e $N \bmod 2 = 0$; se N è dispari viene aggiunto un byte e $N \bmod 2 = 1$. Sommando i tre contributi si ottiene la relazione. Questo conclude la dimostrazione.

Struttura generale AVI

Un file AVI tradizionale ha forma concettuale

```

RIFF 'AVI '
  LIST 'hdrl'
    'avih'          main AVI header
  LIST 'strl'       stream 0
    'strh'          stream header
    'strf'          stream format
    ['strd'] ['strn']
  LIST 'strl'       stream 1
    ...
  LIST 'movi'
    '00dc' / '00db' video chunks
    '01wb'          audio chunks
    ...
  ['idx1']          legacy index

```

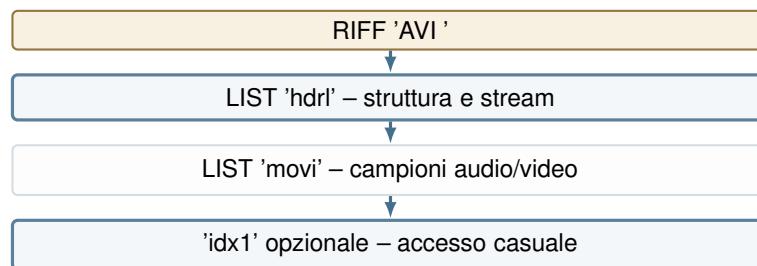


Figura 6.15. Gerarchia essenziale di AVI tradizionale nel modello RIFF.

L'header principale `avih` descrive proprietà globali come microsecondi per frame, stima del massimo byterate, numero totale di frame, numero di stream, buffer suggerito e dimensioni dell'immagine. Una rappresentazione C storicamente usata è:

```

typedef struct {
    DWORD dwMicroSecPerFrame;
    DWORD dwMaxBytesPerSec;
    DWORD dwPaddingGranularity;
    DWORD dwFlags;
    DWORD dwTotalFrames;
    DWORD dwInitialFrames;
    DWORD dwStreams;
    DWORD dwSuggestedBufferSize;
    DWORD dwWidth;
    DWORD dwHeight;
    DWORD dwReserved[4];
} AVIMAINHEADER;

```

I flag possono indicare presenza dell'indice, interleaving, necessità di usare l'indice e origine come file di cattura. Questi campi appartengono alla struttura del contenitore: non definiscono il codec video.

Stream list: `strh` e `strf`

Ogni stream possiede una lista `strl`. Il chunk `strh` contiene un `AVISTREAMHEADER`; il chunk `strf` descrive il formato specifico. Per video `strf` può contenere una `BITMAPINFOHEADER`; per PCM audio una struttura della famiglia `WAVEFORMATEX`.

I FourCC di tipo stream più comuni includono `vids` per video, `auds` per audio e `txts` per testo. Il campo handler può identificare il codec, mentre `dwScale` e `dwRate` definiscono la base temporale dello stream.

La frequenza nominale è

$$f_{stream} = \frac{dwRate}{dwScale}.$$

Dimostrazione. Per definizione $dwScale$ rappresenta quante unità della scala temporale corrispondono a un campione e $dwRate$ quante unità di quella scala trascorrono in un secondo. Il numero di campioni al secondo è quindi il rapporto fra le due quantità. Questo conclude la dimostrazione.

Se uno stream contiene N unità temporali consecutive, la durata ideale è

$$T = N \frac{dwScale}{dwRate}.$$

Dimostrazione. Ogni unità dura $dwScale/dwRate$ secondi, reciproco della frequenza nominale. Moltiplicando per N si ottiene la durata. Questo conclude la dimostrazione.

Nell'esempio classico DV NTSC,

$$dwRate = 2997, \quad dwScale = 100,$$

quindi

$$f = \frac{2997}{100} = 29.97 \text{ frame/s.}$$

Verifica. Applicando la definizione $f_{stream} = dwRate/dwScale$ ai valori 2997 e 100 si ottiene direttamente 29.97 frame/s. Questo conclude la verifica.

Audio PCM nell'header AVI

Per audio PCM con frequenza di campionamento f_s , C canali e b bit per campione, il byterate è

$$R_B = f_s C \frac{b}{8}.$$

Dimostrazione. Ogni secondo contiene f_s campioni per canale, quindi $f_s C$ campioni complessivi. Ogni campione usa $b/8$ byte. Il prodotto fornisce i byte al secondo. Questo conclude la dimostrazione.

La dimensione di un frame audio interleaved è

$$nBlockAlign = C \frac{b}{8}.$$

Per $f_s = 32000$ Hz, $C = 2$ e $b = 16$,

$$nBlockAlign = 4 \text{ byte}, \quad R_B = 32000 \cdot 2 \cdot 2 = 128000 \text{ byte/s.}$$

Verifica. Con due canali a 16 bit ogni frame PCM usa $2 \cdot 16/8 = 4$ byte. A 32000 frame/s il byterate è $32000 \cdot 4 = 128000$ byte/s. Questo conclude la verifica.

Questi valori coincidono con i campi tipici di un WAVEFORMATEX PCM.

movi e identificazione dei chunk

La lista `movi` contiene il payload dei singoli stream. Il FourCC di un data chunk usa normalmente due cifre per il numero di stream e due caratteri per il tipo:

Suffisso	significato tipico
db	video non compresso
dc	video compresso
pc	palette change
wb	audio

Per esempio 00dc può identificare frame video compressi dello stream 0 e 01wb dati audio dello stream 1. AVI permette inoltre di raggruppare chunk in liste `rec` per favorire un accesso a blocchi interleaved.

L'interleaving non modifica il tempo logico dello stream: cambia la collocazione fisica dei chunk per evitare che la lettura sequenziale richieda salti molto grandi fra audio e video.

Indice idx1

Il chunk legacy `idx1` contiene, per ciascun data chunk, FourCC, flag, offset e dimensione. I flag possono segnalare keyframe, liste `rec` e chunk che non avanzano il tempo di presentazione.

L'indice separa quindi due problemi:

- la lista `movi` conserva il payload nell'ordine fisico del file;
- `idx1` permette al player di localizzare rapidamente un campione senza scandire tutto il contenuto precedente.

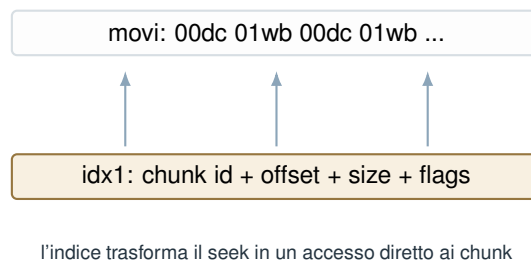


Figura 6.16. Ruolo dell'indice AVI tradizionale rispetto alla lista dati.

AVI DV Type 1 e Type 2

Nella terminologia DV-in-AVI, il Type 1 conserva audio e video nel medesimo stream DV interleaved, identificato storicamente mediante `iavs`. L'audio è già contenuto nel payload DV e uno splitter lo estrae quando necessario.

Il Type 2 espone invece anche uno stream audio AVI separato, tipicamente `auds`, oltre allo stream video `vids`. Il file può quindi risultare più semplice per applicazioni che si aspettano stream AVI indipendenti, al prezzo di una duplicazione o riorganizzazione dei dati audio rispetto alla rappresentazione DV originaria.

La distinzione Type 1/Type 2 riguarda il modo in cui DV viene mappato nel contenitore AVI; non è una distinzione generale fra "AVI con uno stream" e "AVI con due stream" applicabile a ogni codec.

6.4.2 OpenDML e AVI 2.0

OpenDML estende AVI per file più grandi, indicizzazione più robusta e workflow professionali. Il limite pratico del RIFF AVI storico deriva dai campi di dimensione a 32 bit e dalla compatibilità con software e filesystem dell'epoca. OpenDML permette di concatenare più segmenti RIFF: il primo resta identificato come AVI, i successivi come AVIX.

A 80 Mbit/s, un file da 1 GB decimale contiene $8 \cdot 10^9$ bit e dura

$$T = \frac{8 \cdot 10^9}{80 \cdot 10^6} = 100 \text{ s.}$$

Dimostrazione. La durata è dimensione in bit divisa per bitrate. Sostituendo i valori, $8 \cdot 10^9 / (80 \cdot 10^6) = 100$. Questo conclude la dimostrazione. Se si intende 1 GiB, cioè 2^{30} byte, la durata diventa circa 107.4 s.

OpenDML introduce un modello gerarchico di indici. Un superindex nello stream header punta a standard index distribuiti nel file; ciascun standard index punta poi ai chunk effettivi. Il principio è un *indice di indici*.

```

AVI 2.0 / OpenDML
RIFF 'AVI '
LIST 'hdrl'
LIST 'strl'
    strh
    strf
    indx      -> superindex
LIST 'movi'
    media chunks
    ix00      -> standard index stream 0
    ix01      -> standard index stream 1
RIFF 'AVIX'
LIST 'movi'
    ...

```

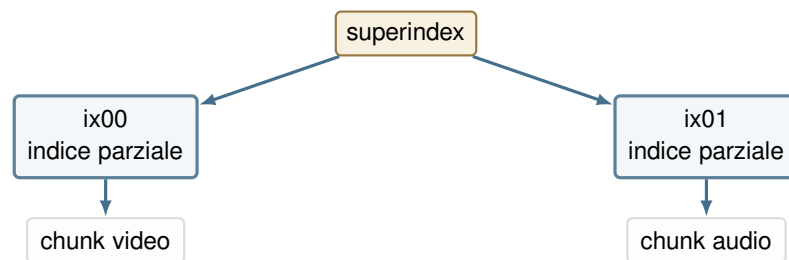


Figura 6.17. OpenDML usa una gerarchia di indici per evitare un unico indice monolitico e permettere file AVI estesi.

La lezione architetturale di AVI resta attuale: un contenitore deve mantenere coerenti payload, tempo e indice. I formati moderni rendono questi concetti più espliciti e più adatti a streaming e frammentazione.

6.4.3 MPEG-2 Transport Stream

MPEG-2 Transport Stream (MPEG-TS) nasce per broadcast e trasporto su canali nei quali sono importanti sincronizzazione rapida, multiplexing di più programmi e resilienza a errori locali. È ancora rilevante in televisione digitale, contribution/distribution professionale e in alcuni workflow HLS e IP.

Il flusso è costruito con pacchetti di dimensione fissa pari a 188 byte. L'header base occupa 4 byte; se non è presente un adaptation field e l'intero resto del pacchetto è payload, il massimo payload è quindi 184 byte e l'efficienza rispetto al solo pacchetto TS è

$$\eta_{TS} = \frac{184}{188} \simeq 0.9787.$$

Dimostrazione. Il pacchetto contiene 188 byte totali e 4 byte di header base. Restano $188 - 4 = 184$ byte disponibili. Il rapporto fra payload e totale è $184/188$, pari a circa 97.87%. Questo conclude la dimostrazione. Un adaptation field o altre informazioni riducono ulteriormente il payload effettivo.

Ogni pacchetto contiene un Packet Identifier (PID) di 13 bit che identifica il flusso logico a cui appartiene il payload. La Program Association Table (PAT), trasmessa sul PID 0, associa i programmi ai PID delle rispettive Program Map Table (PMT). La PMT elenca poi i PID dei componenti elementari, per esempio video, audio e dati associati a quel programma.

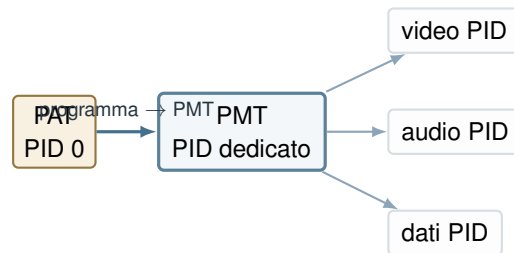


Figura 6.18. Segnalazione essenziale MPEG-TS: PAT individua la PMT del programma e la PMT individua i PID dei componenti elementari.

Il clock di sistema viene ricostruito tramite Program Clock Reference (PCR), trasportato nell'adaptation field dei pacchetti appartenenti al programma. Il PCR usa una base a 90 kHz e una estensione che porta la risoluzione nominale del riferimento a 27 MHz. Serve a disciplinare il clock del decoder e non coincide con il timestamp di presentazione del singolo frame.

I Packetized Elementary Stream (PES) possono invece trasportare Presentation Time Stamp (PTS) e, quando necessario, Decoding Time Stamp (DTS). Entrambi usano una scala temporale di 90 kHz. PTS indica quando l'unità deve essere presentata; DTS quando deve essere consegnata al decoder. La distinzione è necessaria quando l'ordine di codifica non coincide con l'ordine di presentazione, come accade con immagini B.

Con timestamp a 33 bit e clock a 90 kHz, l'intervallo prima del wrap è

$$T_{wrap} = \frac{2^{33}}{90000} \simeq 95443.7 \text{ s} \simeq 26.5 \text{ h.}$$

Dimostrazione. Un contatore a 33 bit possiede 2^{33} stati. Se avanza a 90000 tick al secondo, il tempo necessario a percorrerli tutti è $2^{33}/90000$ secondi, cioè circa 95443.7 s. Dividendo per 3600 si ottengono circa 26.5 ore. Questo conclude la dimostrazione.

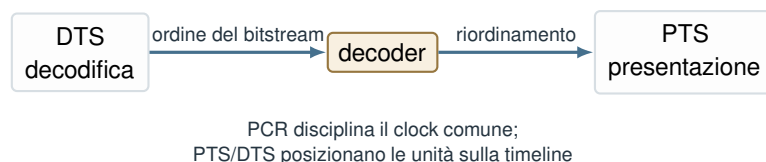


Figura 6.19. Ruoli distinti di PCR, DTS e PTS nel trasporto MPEG.

A differenza di MP4, che è naturalmente orientato a file e sample table, MPEG-TS privilegia una sequenza di pacchetti piccoli e autonomamente riconoscibili. La scelta spiega la sua lunga vita nei sistemi broadcast e nei trasporti continui, mentre fMP4/CMAF è diventato dominante in molti nuovi workflow OTT.

6.4.4 ISO Base Media File Format e MP4

ISO Base Media File Format (ISO-BMFF) usa una struttura gerarchica di *box*. Ogni box dichiara dimensione e tipo; box contenitore possono includerne altri. MP4 è una famiglia di file format costruita su questa base e specializzata mediante registrazioni e specifiche MPEG.

Tre box sono particolarmente utili per comprendere un file MP4 non frammentato:

- `ftyp`: brand e compatibilità dichiarata;
- `moov`: struttura logica, track, timescale, sample table e metadati necessari alla riproduzione;
- `mdat`: payload dei campioni compressi.

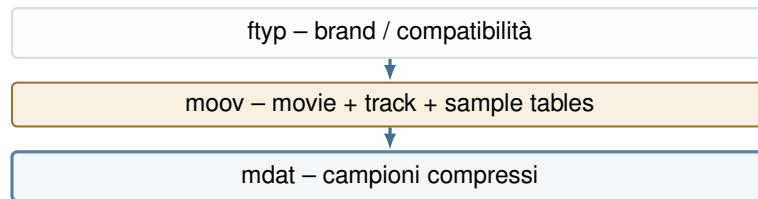


Figura 6.20. Struttura minima concettuale di un file ISO-BMFF/MP4 non frammentato.

Ogni track possiede una propria timeline. Se la timescale è q tick al secondo e un sample o intervallo dura d tick, la durata fisica è

$$T = \frac{d}{q}.$$

Dimostrazione. Un secondo corrisponde per definizione a q tick. Un tick dura quindi $1/q$ secondi; d tick durano d/q secondi. Questo conclude la dimostrazione.

Per esempio, con timescale $q = 90000$ e durata campione $d = 3003$,

$$T = \frac{3003}{90000} \simeq 0.0333667 \text{ s},$$

che corrisponde a circa 29.97 campioni/s quando ogni campione ha la stessa durata.

Verifica. La durata segue direttamente da $T = d/q$. Il rate equivalente è $1/T = 90000/3003 \simeq 29.97$ campioni/s. Questo conclude la verifica.

Le sample table del `moov` mappano numero di sample, durata, posizione nel file, dimensione e proprietà di accesso casuale. In questo modo l'ordine fisico dei byte in `mdat` resta separato dalla timeline logica.

6.4.5 Fragmented MP4

Un file MP4 tradizionale può richiedere che informazioni complete sulle sample table siano note prima di finalizzare `moov`. Nel fragmented MP4 la presentazione viene suddivisa in frammenti. Un `moov` iniziale descrive la struttura generale; coppie `moof`/`mdat` successive descrivono e trasportano i campioni di intervalli progressivi.

```

ftyp
moov
moof -> metadata fragment 1
mdat -> samples fragment 1
moof -> metadata fragment 2
mdat -> samples fragment 2
...
  
```



Figura 6.21. Fragmented MP4: ogni coppia moof/mdat può essere prodotta e consumata progressivamente.

La frammentazione rende naturale la registrazione live, l'invio progressivo e la segmentazione usata da HLS/DASH. Un frammento non è necessariamente equivalente a un segmento HTTP: il packaging può raggruppare più frammenti o rendere disponibili porzioni più piccole in funzione del profilo.

6.4.6 CMAF

Common Media Application Format (CMAF) definisce un insieme comune di regole per media basati su ISO-BMFF frammentato, con l'obiettivo di ridurre duplicazioni fra ecosistemi di distribuzione adattativa. Lo stesso insieme di chunk codificati può essere referenziato da manifest HLS o DASH quando profili, codec e modalità operative sono compatibili.

CMAF distingue concettualmente:

- *track*: sequenza temporale di campioni di un singolo media;
- *fragment*: unità ISO-BMFF autonoma dal punto di vista della struttura moof/mdat;
- *chunk*: porzione consecutiva della track resa disponibile progressivamente;
- *segment*: oggetto o intervallo indirizzabile nel sistema di distribuzione.

Questa granularità è alla base dello streaming a bassa latenza: un client può iniziare a consumare chunk prima che l'intero segmento sia disponibile.

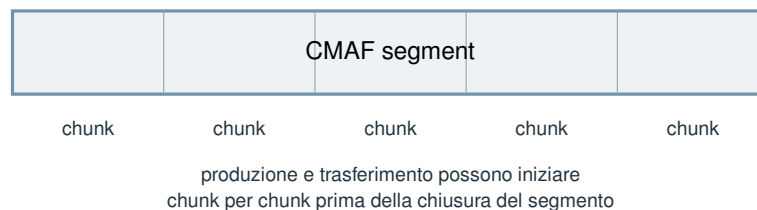


Figura 6.22. Relazione concettuale fra segmento e chunk CMAF nella consegna progressiva.

6.4.7 Matroska e WebM

Matroska usa Extensible Binary Meta Language (EBML), una struttura binaria gerarchica basata su elementi identificati e lunghezze variabili. Un Segment contiene metadata, Track, Cluster, Cue, Attachment e altri elementi. I Cluster raggruppano Block e SimpleBlock secondo la timeline; i Cue forniscono punti di accesso casuale.

WebM definisce un sottoinsieme e un profilo orientato al Web, associato storicamente a codec video VP8/VP9 e audio Vorbis/Opus e, nelle specifiche moderne, ad AV1 dove supportato. Il punto importante è che Matroska/WebM e MP4 sono contenitori differenti: un browser può supportare lo stesso codec in uno dei due e non necessariamente nell'altro.

6.4.8 Confronto dei contenitori

Non esiste un contenitore "migliore" in assoluto. La scelta dipende dai codec ammessi, dalla necessità di progressive download o live fragmentation, dall'ecosistema di player, dalla gestione di metadati e DRM e dai requisiti di interoperabilità.

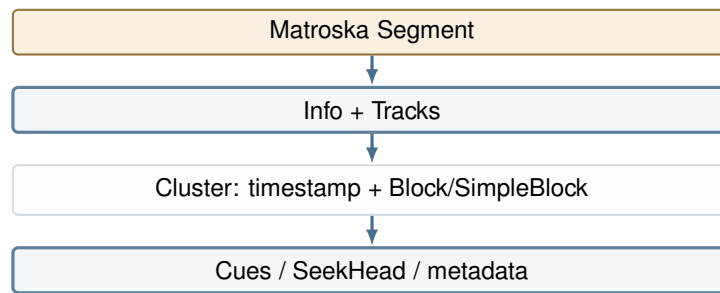


Figura 6.23. Struttura concettuale Matroska: EBML permette una gerarchia estensibile di elementi e cluster temporali.

Famiglia	Struttura	Accesso/streaming	Ruolo tipico
AVI/RIFF	chunk FourCC	idx1/OpenDML; streaming limitato	cattura e interoperabilità legacy
MP4/ISO- BMFF	box gerarchici	sample table o frammenti	file, broadcast, OTT, mobile
fMP4/CMAF	moof/mdat + pro- fili comuni	progressivo e bassa latenza	HLS/DASH e distribuzione adattativa
Matroska WebM	elementi EBML + Cluster	Cue + cluster pro- gressivi	file aperti e Web

Tabella 6.3. Confronto concettuale fra le principali famiglie di contenitori trattate nel capitolo.

6.5 Pipeline multimediale nel Web

Il browser moderno offre più livelli di API multimediali. Un elemento audio o video nasconde demultiplexing, decodifica, sincronizzazione e rendering. MSE permette all'applicazione di alimentare dinamicamente la timeline con segmenti. WebCodecs espone encoder e decoder a un livello più basso. Web Audio permette di costruire grafi di elaborazione audio. WebRTC e WebTransport forniscono invece semantiche di comunicazione e trasporto.

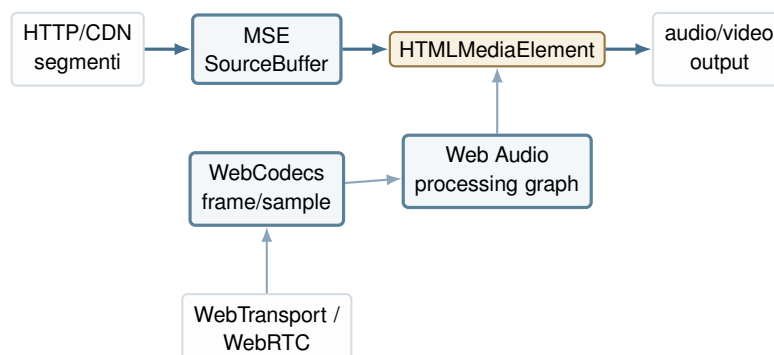


Figura 6.24. Livelli della pipeline media Web. Le API risolvono problemi differenti e possono essere combinate, ma non sono sostituibili una con l'altra.

6.5.1 Media Source Extensions

Media Source Extensions (MSE) estende `HTMLMediaElement` permettendo a JavaScript di costruire progressivamente il media da byte stream supportati dal browser. Nel 2026 MSE versione 2 è ancora sviluppata dal W3C Media Working Group; una Working Draft è stata pubblicata il 7 agosto 2026.

L'applicazione crea uno o più `SourceBuffer`, scarica segmenti compatibili e li aggiunge

alla timeline. Un `SourceBuffer` conserva stato di parsing, intervalli temporali disponibili e regole per append/remove. MSE non sceglie automaticamente la rappresentazione e non implementa HLS o DASH: un player ABR usa manifest, rete e stato del buffer per decidere cosa scaricare e usa MSE come interfaccia verso la pipeline del browser.

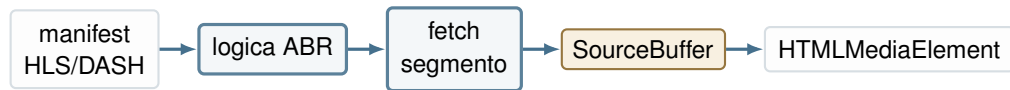


Figura 6.25. MSE non contiene l'algoritmo ABR: riceve i segmenti scelti dall'applicazione e li integra nella timeline del player.

La continuità temporale richiede che timestamp, edit, discontinuity e append window siano gestiti coerentemente. Nel live streaming la rimozione dei segmenti ormai troppo vecchi limita la memoria e mantiene una finestra mobile attorno al live edge.

6.5.2 Timed text e accessibilità

Un sistema multimediale completo deve trattare testo sincronizzato. Sottotitoli, closed captions, descrizioni, karaoke, cue applicativi e metadata temporizzati condividono la timeline del media.

WebVTT è particolarmente diffuso nell'ecosistema Web. TTML e i profili IMSC sono comuni in broadcasting, OTT e interoperabilità professionale. Il formato testuale deve specificare almeno intervallo temporale e contenuto; profili più ricchi aggiungono posizionamento, stile, regioni, writing mode e metadata.

La traccia testuale deve restare coerente durante seek, variazioni della velocità di riproduzione, cambio di rappresentazione e discontinuità. Accessibilità e localizzazione fanno quindi parte dell'architettura del sistema e non di una fase editoriale separata.

6.5.3 Web Audio

Web Audio API costruisce un grafo di nodi che elaborano stream audio. Sorgenti, filtri, gain, panner, analyser, mixer e destinazioni vengono collegati in una pipeline schedulata rispetto a un clock audio.



Figura 6.26. Esempio di grafo Web Audio. La composizione a nodi separa sorgenti, processing e rendering finale.

Web Audio non sostituisce il codec o il contenitore. Opera su audio già reso disponibile alla pipeline del browser oppure generato sinteticamente e consente elaborazione, mixing e analisi a livello di campioni/blocchi.

6.6 Protezione del trasporto e protezione del contenuto

La cifratura del trasporto e la protezione del contenuto risolvono problemi differenti. TLS, QUIC, DTLS e SRTP proteggono dati mentre viaggiano fra endpoint. Dopo che un endpoint autorizzato ha ricevuto e decifrato il traffico, la protezione del trasporto ha completato il proprio compito.

Per alcuni servizi è necessario controllare anche accesso, chiavi e politiche d'uso del contenuto memorizzato o distribuito. Common Encryption (CENC) definisce schemi comuni di cifratura per media ISO-BMFF in modo che gli stessi segmenti possano essere usati con sistemi di gestione licenze differenti.

Encrypted Media Extensions (EME) estende `HTMLMediaElement` con API che collegano il player a un Content Decryption Module (CDM). Nel 2026 la seconda generazione della specifica continua sul Recommendation track del W3C Media Working Group; una Working Draft è stata pubblicata il 7 luglio 2026.

EME non definisce un DRM universale, non prescrive un sistema di licenze e non cifra direttamente i campioni. Standardizza la negoziazione delle *key system capability*, la creazione di sessioni e lo scambio dei messaggi necessari affinché CDM e servizio di licenza possano ottenere le chiavi appropriate.

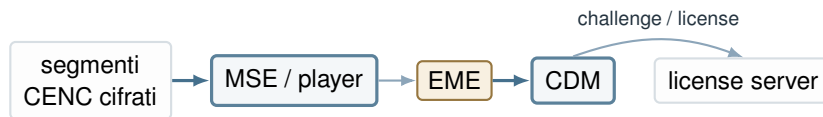


Figura 6.27. Separazione fra media cifrato, API EME, CDM e servizio di licenza. HTTPS protegge il trasporto ma non sostituisce questa catena di autorizzazione.

6.7 WebCodecs e pipeline applicative

WebCodecs espone a JavaScript primitive per codificare e decodificare audio, video e immagini usando implementazioni di codec disponibili nel browser. La Working Draft del 21 settembre 2026 chiarisce un principio importante: WebCodecs non specifica né richiede un codec particolare; un browser può supportarne un sottoinsieme o, in teoria, nessuno.

Le primitive principali operano su oggetti come `EncodedVideoChunk`, `EncodedAudioChunk`, `VideoFrame` e `AudioData`. L'applicazione deve gestire backpressure, timestamp, configurazione del decoder/encoder e vita degli oggetti.

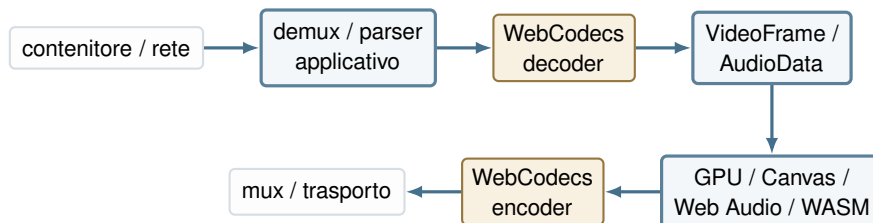


Figura 6.28. Pipeline applicativa con WebCodecs. Demultiplexing, sincronizzazione, mux e trasporto restano responsabilità separate.

La differenza rispetto a WebRTC è sostanziale. WebRTC fornisce sessione, NAT traversal, sicurezza, RTP/RTCP, congestion control e jitter management; WebCodecs espone soltanto primitive media. La differenza rispetto a MSE è altrettanto importante: MSE consegna byte stream a un player integrato, mentre WebCodecs restituisce frame e campioni direttamente all'applicazione.

Combinare WebCodecs e WebTransport permette di costruire pipeline interattive con pacchettizzazione e semantica applicativa personalizzate. Il vantaggio è il controllo; il costo è che sincronizzazione, buffering, containerizzazione, segnalazione, congestion control e resilienza devono essere progettati esplicitamente o forniti da altri componenti.

Nel settembre 2026 il W3C mantiene anche registrazioni specifiche per codec come AAC, MP3, Opus e AV1. Queste registrazioni descrivono stringhe di codec, configurazioni e formato dei chunk; non obbligano le implementazioni a supportare tali codec.

6.8 Architettura multimediale moderna

I capitoli del libro possono essere collegati in un'unica catena. Sensori e sorgenti producono testo, immagini, audio e video; i codec riducono la ridondanza; i contenitori associano campioni, timeline e metadati; RTP, QUIC, HTTP o protocolli di contribution trasferiscono i dati; WebRTC, HLS/DASH o applicazioni Web personalizzate organizzano sessione e distribuzione; CDN, SFU, TURN, relay ed edge spostano traffico e calcolo nei punti più opportuni.

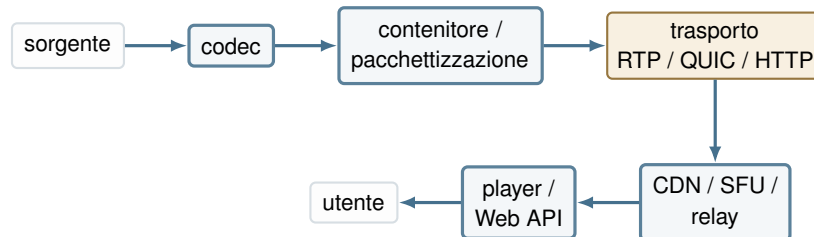


Figura 6.29. Architettura end-to-end contemporanea: ogni livello risolve un problema specifico e può retroagire sugli altri attraverso feedback e adattamento.

Non esiste una corrispondenza uno-a-uno fra “sistema multimediale” e un singolo standard. Un’applicazione è una composizione di funzioni interoperabili; la progettazione consiste nel scegliere per ciascuna il compromesso appropriato fra qualità, bitrate, latenza, complessità, affidabilità, sicurezza e scalabilità.

Le tecnologie storiche mantenute nel libro rendono leggibile questa evoluzione. H.221 e H.223 risolvono un problema di multiplexing che oggi ricompare nei container e nei trasporti multiplexati; H.242, H.245, SIP/SDP e offer/answer mostrano l’evoluzione della negoziazione; MCU e SFU mostrano due modi diversi di spostare il costo della conferenza; AVI, MP4 e CMAF mostrano l’evoluzione della relazione fra payload, tempo e indice.

Cambiano implementazioni e standard, ma i problemi fondamentali restano gli stessi: rappresentare, comprimere, sincronizzare, trasportare, adattare, proteggere e riprodurre informazione multimediale.

Indice analitico

- AAC, 95
- ABR, 141
- AC-1, 98
- AC-2, 98
- AC-3, 99
- Acceleratore grafico, 25
- Adaptive Bitrate Streaming, 141
- ADPCM, 83
- Aliasing, 28
- Ambisonics, 99
- APCM, 83
- AQM, 129
- ARQ, 45, 124
- ASCII, 13
- Audio, 3, 27, 81
 - compressione lossless, 86
- Audio multicanale
 - 5.1, 95
- Audio object-based, 99
- Audio spaziale, 99
- AV1, 78, 116
- AV2, 117
- AVC, 114
- AVI, 166
- AVIF, 78
- Banda, 4
- Bark, 89
- Best effort, 133
- BitTorrent, 148
- Box, 172
- BT.2020, 43
- BT.601, 41
- BT.709, 43
- Buffering, 122
- CABAC, 114
- Cache, 142
- Campionamento, 28
- Canale, 121
- CAVLC, 114
- CD, 35
- CDN, 4, 8, 142
- CELP, 85
- Chunk scheduling, 143
- Churn, 148
- CIF, 43
- Classificazione pacchetti, 130
- CMAF, 141, 173
- CMY, 17
- CMYK, 17
- Code con priorità, 133
- CoDel, 132
- Codice Morse, 48
- Codifica aritmetica, 58
- Codifica learning-based, 79
- Collaborazione, 10
- Colorimetria, 19
- Common Encryption, 175
- Commutazione circuito, 153
- Commutazione pacchetto, 157
- Companding, 33
- Compensazione del moto, 106
- Compressione, 3
- Compressione immagini, 64
- Conferenze, 153, 157
- Congestione, 129
- Contenitore multimediale, 166
- Content addressing, 148
- Conversione A/D, 28
- Correzione errori, 124
- CRT, 23
- CTU, 115
- DCT, 70
- DEFLATE, 68
- Delta modulation, 83
- DHT, 143
- DiffServ, 7, 136
- Disuguaglianza di Kraft, 50
- Divergenza relativa, 45
- DMD, 25
- Dolby, 97
- Dolby Atmos, 99
- DPCM, 81
- DPI, 21
- DRM, 175
- DSCP, 136
- DTLS-SRTP, 164
- DTS

- timestamp, 170
- DualQ, 137
- E-learning, 10
- EBML, 173
- ECN, 131
- Edge computing, 8
- EME, 175
- Entropia, 45
- EVS, 97
- Fair Queue, 130
- Fattore di forma, 22
- FEC, 7, 45, 124
- FGS, 118
- FIFO, 130
- FLAC, 86
- Fletcher, 88
- Fosfori, 23
- FourCC, 166
- FQ-CoDel, 132
- Fragmented MP4, 172
- Frame B, 101
- Frame I, 101
- Frame P, 101
- Frame rate, 36
- G.711, 157
- G.722, 157
- G.723.1, 86
- G.728, 86, 157
- G.729, 86
- Gaming, 10
- GAT, 154
- Gatekeeper, 157
- Gateway H.323, 157
- GCC, 154
- GIF, 66
- GPS, 130
- GPU, 25
- Grafica vettoriale, 27
- H.221, 155
- H.222, 157
- H.223, 156
- H.225, 158
- H.242, 155
- H.245, 156, 158
- H.261, 107, 157
- H.263, 109
- H.264, 114
- H.265, 115
- H.266, 117
- H.310, 157
- H.320, 153
- H.321, 157
- H.322, 157
- H.323, 157
- H.324, 155
- HDR, 19
- HDTV, 43
- HEIC, 78
- HEIF, 78
- HEVC, 78, 115
- HLG, 19
- HLS, 141
- HSV, 19
- HTJ2K, 75
- HTTP/2, 150
- HTTP/3, 150
- Huffman dinamico, 55
- Huffman statico, 49
- ICE, 163
- Immagini, 3, 15, 64
- IMSC, 175
- Informazione mutua, 45
- Interlacciamento, 23, 38
- Interleaving, 128
- IntServ, 136
- ISO Base Media File Format, 172
- ISO-8859, 14
- Jitter, 6, 121
- JPEG, 69
- JPEG 2000, 75
- JPEG AI, 79
- JPEG XL, 78
- JPEG XS, 77
- L4S, 137
- LC3, 97
- LCD, 24
- Leaky Bucket, 135
- Lempel-Ziv, 60
- Lossless, 48
- Lossy, 48
- Low-Latency DASH, 141
- Low-Latency HLS, 141
- LPC, 84
- LZ77, 60, 61

- LZ78, 60, 62
- LZSS, 60, 61
- LZW, 60, 63
- MAD, 103
- Matroska, 173
- MCU, 69, 157, 164
- Media over QUIC, 151
- Media Source Extensions, 174
- Metamerismo, 15
- MicroLED, 24
- MIDI, 36
- MJPEG, 101
- Modello psicoacustico, 87, 88
- Monitor, 22
- MOQT, 151
- MOS, 65
- Motion blur, 36
- MP1, 93
- MP2, 93
- MP3, 93
- MP4, 172
- MPEG-1, 87, 111, 157
- MPEG-2, 94, 111, 157
- MPEG-2 Transport Stream, 170
- MPEG-4 Visual, 112
- MPEG-DASH, 141
- MPEG-H 3D Audio, 99
- MPEG-TS, 170
- MS-SSIM, 65, 119
- MSE
 - Mean Squared Error, 65
 - Media Source Extensions, 174
- n-way XOR, 126
- NMN, 90
- NMT, 90
- NTSC, 41
- NYT, 56
- Offer / Answer, 162
- OLED, 24
- OpenDML, 169
- Opus, 96
- OSI, 1
- Overhead, 1, 139
- Overlay P2P, 8, 142
- P2P, 8, 142
 - disponibilità, 148
 - disseminazione, 146
 - integrità, 148
 - NAT traversal, 148
 - offload, 144
 - scalabilità, 144
- PAL, 40
- Palette, 66
- PAT, 170
- PBM, 26
- PCM, 33
- PCR, 170
- Peer-to-peer, 8, 142
- Perdita di pacchetti, 7, 121
- PGM, 26
- PHB, 136
- Piani colore, 15
- PID, 170
- PIE, 132
- Pixel, 15
- Playout buffer, 122
- PMT, 170
- PNG, 68
- Politiche di drop, 131
- PPI, 21
- PPM, 26
- PQ, 19
- Predizione lineare, 81
- Progressive scan, 38
- Protezione differenziata, 129
- Proxy SIP, 160
- PSNR, 65
- PTS, 170
- Pulldown, 37
- Q.931, 155
- QCIF, 43
- QoE, 10
- QoS, 7, 133
- Qualità dell'esperienza, 9
- Quantizzazione, 30, 71
- QUIC, 150
- Rarest first, 143
- RAS, 158
- Rate control, 108
- Rate-distortion, 66
- Realtà aumentata, 10
- Realtà virtuale, 10
- RED, 131
- Reed-Solomon, 35
- Registrar SIP, 160

- Reti multimediali, 4
- RGB, 16
- Ridondanza sorgenti, 3
- Ridondanza temporale, 101
- RIFF, 166
- Risoluzione, 21
- RIST, 151
- Ritardo, 6, 121
- RLE, 49
- RSVP, 136
- RTCP, 137
- RTP, 137
- RVLC, 113
- SAO, 115
- Scalabilità, 118
- SCTP, 164
- SDP, 160, 162
- SECAM, 41
- Seed, 143
- Segnale composito, 39
- SFU, 164
- SIF, 43
- Simulcast, 164
- SIP, 160
- Sistemi multimediali, 9
- SMR, 90
- Softphone, 159
- Sorgenti dinamiche, 2, 3
- Sorgenti multimediali, 2
- Sorgenti statiche, 2, 3
- Sottocampionamento, 41
- Sottocampionamento cromatico
 - 4:1:0, 43
 - 4:1:1, 42
 - 4:2:0, 42
 - 4:2:2, 42
 - 4:4:4, 42
- SPL, 88
- sRGB, 19
- SRT, 151
- SRTP, 164
- SSIM, 65, 119
- Stima del moto, 103
- Streaming, 11
- Streaming HTTP, 140
- STUN, 163
- SVC, 164
- SVG, 27
- Swarm, 8, 142
- T.120, 154
- T.121, 154
- T.122, 154
- T.123, 154
- T.124, 154
- T.125, 154
- T.126, 154
- T.127, 154
- T.128, 155
- TDM, 155
- Teleconferenze, 11
- Televisione analogica, 38
- Televisione digitale, 11
- Teorema del campionamento, 28
- Testo, 3, 13, 48
- TFT, 24
- Tit-for-tat, 148
- TNM, 90
- Token Bucket, 135
- Tracker P2P, 143
- Traffic shaping, 135
- TTML, 174, 175
- TURN, 163
- TV a colori, 40
- UCS, 14
- UHD, 43
- Unicode, 14
- User Agent, 160
- UTF-8, 14
- V.34, 155
- Video, 3, 36, 101
- Video Object Plane, 112
- Videoconferenza, 11
- VMAF, 119
- VoIP, 159
- VP9, 116
- VSB, 39
- VVC, 117
- Wavelet, 75
- Web Audio, 174
- Web Audio API, 175
- WebCodecs, 176
- WebM, 116, 173
- WebP, 77
- WebRTC, 163
- WebRTC DataChannel, 148, 164
- WebTransport, 150

WebVTT, 174, 175

WFQ, 131

Wide gamut, 19

XBM, 26

XPM, 26

YCbCr, 18

YPbPr, 18

YUV, 18

Zig-zag, 73