UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

MULTIMEDIA 2021/2022

Francesco Crisci 1219620

Contents

Chapter 1

Introduzione

Per multimedia si intendono quelle applicazioni che vengono utilizzate quotidianamente in molte situazioni diverse.
La codifica di sorgente serve a comprimere il segnale per trasmettere dei pacchetti più piccoli e facili da trasportare. Per la trasmissione dei file da un dipositivo ad un altro si usano le onde radio, ovvero dal dispositivo alla base station si usano tecniche di modulazione e attualmente si trasmettono tramite LTE(attualmente sta venendo rimpiazzato dal 5G).
La modulazione tramuta i bit in segnali analogici che possono essere trasmessi attraverso diversi mezzi, nel caso odierno parliamo delle fibre ottiche.
Un protocollo gestisce la sincronizzazione temporale, la correzione di errori attraverso l'internet.
M2M vuol dire machine-to-machine, ovvero macchine che comunicano tra di loro senza l'intervento di un utente umano. TIpo una cella fotovoltaica che raccoglie i dati e sucessivamente li trasmette a un altro computer per l'analisi.
La trasmissione da un nodo a un altro è funzione della distanza. Ci si accorge che se si vogliono soddisfare certe condizioni, i due nodi devono essere, necessariamente, avvicinati tra loro. Tutto questo a trovato soluzione con l'uso dell'edge computing, ovvero passare una copia di un dato da un nodo a un altro per poi non dover ridurre fisicamente la distanza tra i due nodi.
Le nuove applicazioni come Virtual Reality, Augmented Reality e MR richiedono bassa latenza e banda alta. Applicazioni

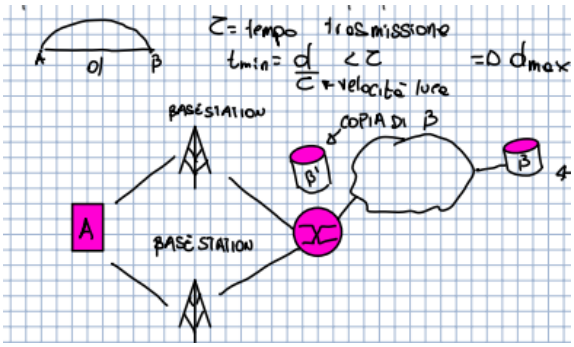

Figure 1.1: Schema grafico dell'edge computing

come guida autonoma richiedono latenza molto bassa, affidabilità molto alta e una possibile banda medio-alta.
Il traffico multimediale sicuramente è dominante sia ogi che nei prossimi periodi, sopratutto per le applicazioni vicino a video e audio, ma anche per virtual reality e augmented reality.
Una maggiore banda, in realtà bit-rate, non sarà abbastanza. Necessiteremo della differenziazione dei servizi. Aumentando il bit-rate a ultranza non è la soluzione ottimale da seguire. La soluzione otimale è la differenziazione dei servizi.

Un pò di sigle

- LPWAN = Low Power Wide Area Network
- NB-IoT = Narrow Band Internet of Things, è un'aggiunta allo standard LTE che permette di servire nodi che abbiano poca energia e che vogliono poca latenza.
- SDN = Software Defined Network, permettono di gestire la rete in base al tipo di traffico, a differenza di internet che non ne tiene conto. Permette di usufruire di più servizi contemporaneamente senza che la banda usata per un servizio vada a saturare la banda degli altri. Se con internet io facessi uno stream mentre sto scaricando un file, la qualità del primo ne risentirebbe molto. Con l'SDN si previene questo tipo di problema. Così viene introdotto il network slicing.

1.1 Classificazione dei servizi multimediali

Dato un insieme di entità, ci sono dei metodi di classificazione di esse. Nei casi di servizi multimediali si possono fare diverse classificazioni. Ma cosa si intende per applicazione multimediale? Deriva da medium che in plurale è media, ovvero una rappresentazione, memorizzazione o trasporto dell'informazione.
Multimedia è un flusso di informazione rappresentato, memorizzato o trasportato da un media differente. Esempi di multimedia sono i seguenti:

- Testo;
- Dati;
- Suono;
- Immagini;
- Video;

Un multimedia è quindi una rappresentazione di media misti come segnali digitali.

1.1.1 Suddivisione tra Analogico e Discreto

La prima suddivisione che si può effettuare è tra sorgenti analogiche e sorgenti discrete:

- Sorgenti analogiche: insieme di tutti i fenomeni che sono analogici in natura come immagini, video e audio. Hanno bisogno di digitalizzazione, ovvero operazioni come campionamento, quantizzazione e codifica.
- Sorgenti discrete: insieme di tutti quegli elementi nella quale si possono identificare le unità informative che la compongono. Un esempio sono i grafici creati al computer, in quanto quei grafici sono costituiti da u insieme discreto di elementi. Altri esempi sono il testo, il computer, l'audio e i "dati". Essi sono digitali in natura. Operazioni come quantizzazione e campionamento potrebbero non essere necessarie per questo tipologia di segnali.

1.1.2 Suddivisione in segnale Statico e Tempo variante

I segnali possono essere divisi in segnali con sorgente statica, ovvero che rimane costante nel tempo, o tempo variante.

- Sorgenti statiche: sono tempo invarianti, così come testo, immagini (non le gif), grafica e "data". Sono elastiche, ovvero possono tollerare ritardi, anche grandi, senza risentirne in termini di qualità. Ad esempio il testo e le immagini di pagine html non devono necessariamente comparire contemporaneamente. Sono generalmente molto sensibili alla perdita di pacchetti.
- Sorgenti tempo varianti: sono sorgenti dinamiche come audio, video, streaming di testo. Richiedono sincronizzazione tra i differenti tipi di media. Ad esempio i flussi audio e video di un broadcast televisivo devono essere sincronizzati per il ricevitore. Sono sensibili ai ritardi dell'E2E(end-to-end) ed al jitter(variazione del tempo di consegna dei pacchetti). Sono poco sensibili alla perdita di pacchetti, l'esatto opposto delle sorgenti statiche.

1.2 Streaming

Le applicazioni di streaming possono essere iniziate prima che la trasmissione di essa possa essere terminata. Richiede un servitore, ovvero un provider, che mandi i dati con un passo fermo e un ricevitore che memorizzi temporaneamente dei dati in un buffer facendo poi caching sui dati in eccesso. ESempi di applicazioni streaming sono le news, l'intrattenimento, l'educazione remota, le teleconferenze, internet radio, broadcast televisivi, telecontrollo (tipo per i bracci meccanici) e gaming.
Una rappresentazione dello straming è visibile in figura 1.2. Da notare che la curva rossa ha la stessa pendenza che aveva il flusso originario, ovvero la curva nera, e la stessa pendenza del flusso riprodotto, ovvero la curva arancione. Solitamente per contenuti di tipo store, la pendenza del grafico rosso è diverso dagli altri due, in quanto dipende dal tempo di trasmissione.
Ma come posso migliorare? Adotto la codifica di sorgente alla rete. Meglio ancor se trasmetto dei meta dati, ovvero informazioni su un video, ad esempio, in modo che anche se la mia rete fa schifo, io posso dare qualche informazione all'utente. Ovvero, scorriamo una versione campionata e a risoluzione molto più bassa di quella originale e pianificata.

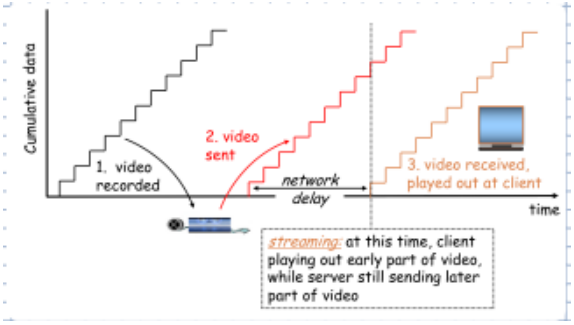

Figure 1.2: Grafico raffigurante l'attività di streaming

1.2.1 Playout Buffering

Permette di rendere più liscio, to smooth, variazioni sporadiche dallo streaming entrante. I dati sono passati al livello applicativo ad un rate costante.
Come possibile vedere in figura 1.3, abbiamo una rappresentazione qualitativa di come il playout buffering funzioni. Nella

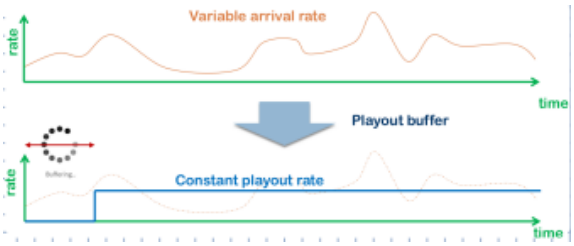

Figure 1.3: Grafico raffigurante l'attività di Playout

prima parte si memorizza una quantità di dati trasmessa in modo da dare tempo a altri di arrivare e successivamente fare caching. Così facendo, io posso iniziare a vedere lo streaming in maniera molto fluida. Ci possono essere casi un cui si presentano delle problematice, che vedremo nei seguenti paragrafi.

1.2.2 Underrun

In una situazione di underrun, come mostrato in figura 1.4, i dati vengono consumati più velocemente di quanto arrivino al buffer. Dopo un periodo, il buffer si scuota e l'utente avverte un'interruzione. Ciò accade principalmente dopo una

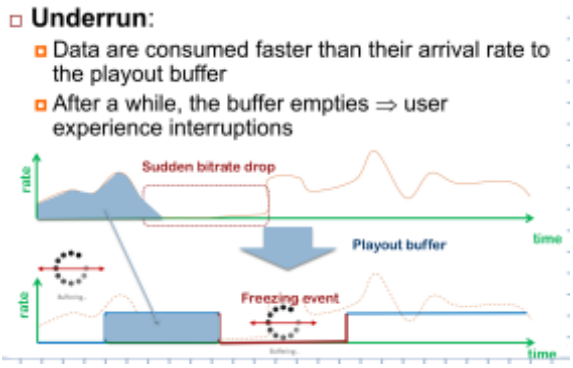

Figure 1.4: Grafico raffigurante l'attività di Underrun

caduta di bit-rate accidentale. Il grafico sovrastante rappresenta la sorgente di streaming mentre il grafico sottostante rappresenta l'utente che riceve il flusso di dati.
Ci si adatta dinamicamente per la qualità che viene trasmessa all'utente. Solitamente, le prime richieste vegono servite con qualità bassa per poi aumentarla nel tempo. Ci possono essere momenti in cui la qualità può sia diminuire che cambiare. Per questo motivo, la qualità viene adattata dinamicamente anche in base alla quantità di dati che il buffer può memorizzare e alla velocità di trasmissione.

1.2.3 Overrrun

Il fenomeno opposto all'underrun è l'overrun, ovvero quando i dati arrivano più velocemente di quanto il buffer riesca a svuotarsi. Solitamente è molto raro e non accade. In caso contrario, il client deve interrompere la richiesta al server per non far riempire il buffer. Dopo un periodo il buffer va in overflow e l'untent riscontra dei glitch. Di solito accade a causa di programmazione scarsa o sbagliata.

1.3 Interactivity

L'interattività è una funzionalità di tipo VCR: il client può mettere pausa, tornare indietro, andare avanti più veloce e molto altro. Ci sono dei delay iniziali che sono solitamente più grandi dei tempi di funzionamento di un comando.
Esempi di realtime interaction sono la telefonia IP, le video conferenze e i mondi interattivi distribuiti. Ha dei requisiti E2E: per l'audio se il tempo di interazione è minore di 150 ms il risultato è molto buono, mentre se risulta maggiore di 400ms è molto scarso.
Include livello applicativo (packetization) e delay della rete (network delay). I delay più ampi si possono notare e si ha interattività impari. Sommatoriamente, questà è la suddivisione delle applicazioni streaming: Le applicazioni di streaming

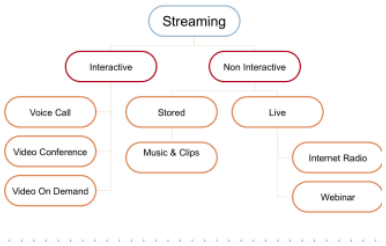

Figure 1.5: Streaming

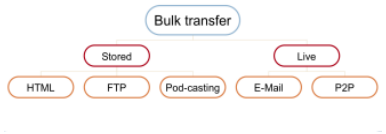

Figure 1.6: Bulk

possono essere messe di fianco alle applicazioni di bulk, ovvero applicazione che usano memorizzazione a pacchetti.

Chapter 2

Digitalizzazione dei segnali

2.1 Introduzione

In questo capitolo parliamo della digitalizzazione dei segnali.
I dati possono essere sia analogici, quindi continui, sia discreti. I sistemi di comunicazione più moderni usano tescniche di trasmissione digitale. Le sorgenti sono facilmente digitalizzabili associandogli una semplice rappresentazione binaria. Questa operazione è sempre invertibile.
Nel caso di una conversione da analogico a digitale si usa obbligatoriamente un analog-to-digital conversion (ADC).

2.1.1 Analog-to-doigital conversion

L'ADC richiede tre passi: Un segnale deve essere analogico in tutte le sue dimensioni, sia nel dominio che nel codominio.

- Discretizzazione del dominio: Consideriamo segnali che si evolvono nel tempo ma questa tecnica si applica per ogni tipo di dominio continuo. Il teorema di campionamento di Shannon determina il rate di campionamento minimo in base alla dinamica generale del segnale e dalla caratteristica del ricevitore.
- Discretizzazione del codominio: bisogna discrettizzare l'asse dei valori. Questa operazione prende il nome di quantizzazione. Consiste nel dividere il codominio in sotto intervalli e identificare un punto che rappresenta ognuno di questi sotto intervalli. Lo spazio continuo dei campioni viene mappato in un set finito di valori quantizzati. Questa operazione è irreversibile in quanto può perdere informazione non recuperabile in codifica.
- Codifica del segnale: viene associato a ogni elemento del segnale una rappresentazione binaria.

Il campionamento è teoricamente invertibile nel momento in cui si può campionare con una frequenza maggiore del doppio della semibanda positiva del segnale.

2.1.2 Spettro del segnale

Un segnale periodico piò essere rappresentato come somma di sei e coseni la cui frequenza sia multipla della frequenza del segnale originale. Se un segnale è periodico 2 secondi, ha una frequenza di 0,5 Hz. Un segnale periodico che ha periodo di 2 secondi può essere rappresentato come somma di seni e coseni che hanno frequenze multiple di 0,5 Hz(Figura 2.1):
Il segnale x(t) si può rappresentare andando a specificsare i coefficienti di seni e coseni con frequenze multiple di 1/Tp.

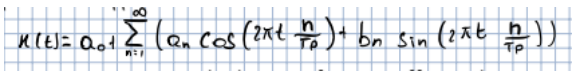

Figure 2.1: Segnale espresso in seni e coseni

I coefficienti si possono rappresentare in forma esponenziale. Questa è la rappresentazione in forma esponenziale. La trasformata di Fourier ci permette di esprimere il segnale x(t) come segue(Figura 2.2 e 2.3):
Lo spettro del segnale è, dunque, l'insieme delle componenti in frequenza del segnale stesso. Il mio segnale si scompone in una serie di armoniche che hanno una distribuzione in frequenza, la cui ampiezza va diminuendo man mano che si va a frequenze più alte. Lo spettro è l'insieme delle componenti in frequenza che hanno potenza non trascurabile.
Se la banda che rappresentiamo è più piccola di quella effettiva del segnale, in frequenza si creerà della distorsione e in alcune parti del segnale ci saranno delle onde che nel segnale originario non erano presenti.

$$x(t) = \sum_{k=0}^{\infty} c_k e^{j2\pi t \frac{k}{T_p}}, \quad (3)$$

dove $c_0 = a_0$ e $c_k = \frac{a_k - jb_k}{2}$, $c_{-k} = \frac{a_k + jb_k}{2} = c_k^*$, sicché[2]

$$c_k = \frac{2}{T_p} \int_{-\frac{T_p}{2}}^{\frac{T_p}{2}} x(u) \frac{\cos\left(2\pi u \frac{n}{T_p}\right) - j \sin\left(2\pi u \frac{n}{T_p}\right)}{2} \mathrm{d}u = \frac{1}{T_p} \int_{-\frac{T_p}{2}}^{\frac{T_p}{2}} x(u) e^{-j2\pi u \frac{k}{T_p}} \mathrm{d}u\,. \quad (4)$$

Figure 2.2: Segnale secondo Fourier

$$\left\{\ldots, \left(c_{-2}, -\frac{2}{T_p}\right), \left(c_{-1}, -\frac{1}{T_p}\right), (c_0, 0), \left(c_1, \frac{1}{T_p}\right), \left(c_2, \frac{2}{T_p}\right), \ldots\right\}$$

Figure 2.3: Dominio

La banda è la frequenza della componente oltre la quale la potenza delle armoniche può essere trascurata. Ovvero le armoniche che non mi trasmettono informazione.
Nella figura 2.4 è possibile vedere come risulta un segnale dopo che è stata eseguita la trasformata di Fourier. Lo spettro è

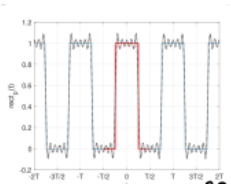

Figure 2.4: Segnale dopo Fourier

l'insieme dei coefficienti complessi rappresentati con modulo e fase. L'insieme di questi grafici (figura 2.5) ci da lo spettro.

L'operazione di campionamento risulta invertibile quando la frequenza di campionamento supera la banda del segnale. Questo accade perchè quando noi prendiamo un segnale continuo, con un certo spettro, e lo campioniamo, il segnale che otteniamo ha uno spettro diverso da quello originario perchè abiamo cambiato il segnale originario. Lo spettro del segnale campionato si ottiene come ripetizione periodica dello spettro del segnale di origine.
L'operazione opposta al campionamento prende il nome di interpolazione. Passo da un segnale discreto a uno continuo. L'alliasing è l'effetto di distorsione dovuno al sottocampionamento, consiste nel fatto che lo spettro del segnale si sporca nella parte esterna dello spettro. Più l'alliasing è severo, più vengono corrotte le parte interne del segnale. Se il segnale ha uno spettro che cade molto rapidamente nelle code, e l'alliasing risulta essere presente in quell'area, la distorsione è presente ma non si avverte molto.
La semibanda positiva del segnale è l'insieme delle frequenze dopo le quali le potenze dei segnali possono essere trascurate. Nei segnali reali, però, la banda non è mai illimitata. Le componenti con potenza maggiore di zero sono infinite. Introduciamo sempre distorsione anche se non vogliamo. Per limitarla si usa il prefiltraggio del segnale, usato sia per le immagini che per l'audio.
Supponiamo di partire da un segnale, e che il nostro sistema non ci permette di campionare a frequenze volute e che quindi viene introdotta distorsione. Vado a modificare il segnale prima di campionarlo, in modo da andare a campionare un segnale che abbia una banda compatibile alla mia frequenza di campionamento. Elimino le componenti con frequenze al di fuori dell'intervallo della mia frequenza di campionamento. Faccio una convoluzione con un filtro che ha risposta impulsiva adeguata, il cui specchio è una finestra, in modo da ottenere un segnale che ha uno spettro con code che tendono a zero molto velocemente.
Campionando questo nuovo segnale l'alliasing non viene introdotto, ma la distorsione è stata introdotta nella fase di prefiltraggio. Il prefiltro mi fa predistorcere il segnale per non introdurre l'alliasing. Nella codifica audio si fa prefiltraggio che tenga conto delle frequenze meno visibili.
Quindi l'ADC è irreversibile? Il cmapionamento potenzialmente lo è, la quantizzazione come già detto non lo è, mentre la codifica può essere lossless e quindi reversibile o lossy e quindi non reversibile. La codifica può essere quindi potenzialmente invertibile.
Vediamo ora un quadro d'insieme(Figura 2.5): Il segnale viene convertito da analogico a digitale e viene successivamente codificato.Dopo la codifica viene trasferito sulla rete. Dal lato del ricevitore, il decodificatore tramuta il segnale in valori quantizzati e con un convertitore digital-to-analog ottengo una copia del segnale di partenza. Il segnale copiato è simile al segnale originario in quanto è stata introdotta distorsione. In uno scenario ideale, il segnale che viene ricevuto è uguale a quello trasmesso, ma sappiamo che nel mondo reale ciò non accade mai. Nella rete ci possono essere delle perdite di pacchetti, e quindi il segnale che vado a decodificare può risultare già modificato rispetto a quello originale.
Il campionatore il cui effetto sullo spettro del segnale risulta essere ripetizione periodica del segnale originale.

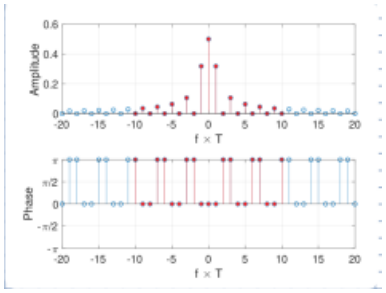

Figure 2.5: Spettro

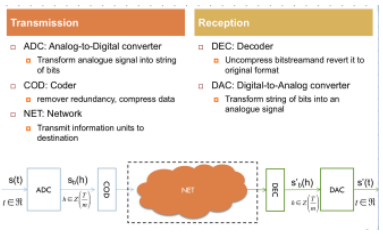

Figure 2.6: Trasmissione e ricezione di un segnale

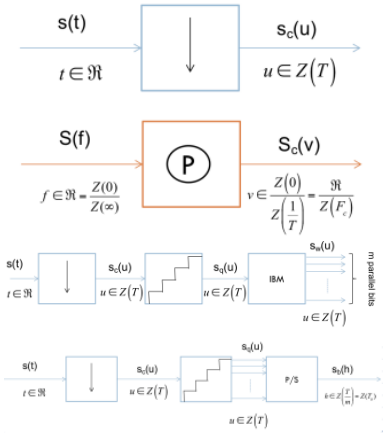

Figure 2.7: Campionamento

Nelle seguenti figure 2.7 è possibile vedere uno schema della conversione ADC: Il periodo è indicato con Fc, che risulta essere uguale alla frequenza di campionamento. Successivamente c'è la quantizzazione dei valori campionati: per ogni campione viene generata una parola di mbit, che risulta essere log_2 del numero dei livelli di quantizzazione alla quale può aggiungere qualcosa per la correzione di errori. Si può anche sequenzializzare la sequenza di bit per ottenero uno stream di bit. Produco così una sequenza si bit. Ogni bit viene generato ogni tot secondi e il periodo di generazione risulta essere il rapporto (tasso di campionamento/numero bit per campione)xT/m. Il periodo dipende quindi sia dal periodo di campionamento che dal numero di bit da generare. Supponendo di campionare con una frequenza doppia la banda del segnale e quantizzare su 2^m livelli, il bitrate risulta essere 2Bm o $Fclog_2m$. La sequenzializzazione mi darà un segnale binario e discreto.
L'operazione inversa viene eseguita dal ricevitore: leghiamo la sequenza in blocchi di m bit. Dalla parola originaria ri-otteniamo il simbolo a cui era associata la parola. Ora con un'operazione simile all'interpolazione nel tempo possiamo ottenere una copia del segnale originario. Si usano forme d'onda continue associate ai simboli usati nell'interpolazione. Si usa la funzione sinc(x) = sinx/x ed ha la caratteristica di avere la forma di seno smorzato nel tempo e simmetrico.
Ad ogni impulso, io moltiplico per il sinc e sommo i vari sinc assieme. La caratteristica dei sinc è che si annullano nei punti di intersezione. Nei punti corrispondenti ai campioni ho esattamente i campioni e negli altri punti ho qualcosa che mi approssima progressivamente il segnale. Il problema del sinc è che ha una durata infinita ed è causale, in teoria.
Il segnale ricostruito non è uguale all'originale, viene così quantizzato l'errore. Per farlo, si usa il mean square erro, la media dello scarto quadratico tra il segnale originale e la copia(Figura 2.8): Quando arrivo a costruire il source bitstream, esso deve essere maneggiato dai livelli di rete. I dati che lo costituiscono vengono divisi in sequenze di bit, messi in un

$$MSE = \lim_{T\to\infty}\left\{\frac{1}{T}\int_{-T/2}^{+T/2}\left(s(t)-s'(t)\right)^2 dt\right\}$$

Figure 2.8: Mean Square Error

certo pacchetto a cui si aggiungono anche i dati di intestazione(header), utilizzati per trasportare informazioni di controllo utili alla rete per riuscire a recapitare informazione e destinazione.
Nella figura 2.9 è possibile vedere come i pacchetti vengono trasmessi in rete e come essi si comportano. Vediamo ora, in

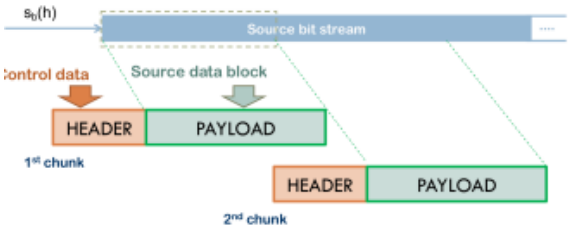

Figure 2.9: Pacchetto nella rete

figura 13, lo schema completo di quello che accade tra rete, trasmettitore e ricevitore: I protocolli utilizzano informazioni

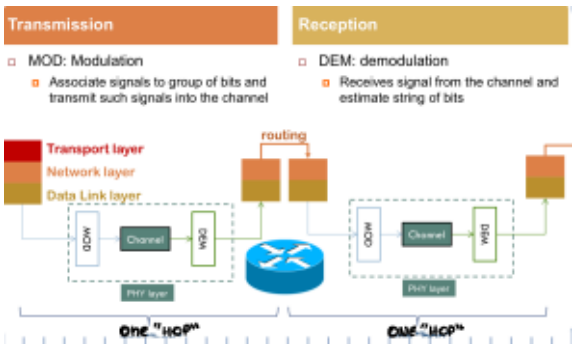

Figure 2.10: Schema della rete

di controllo contenuti negli header e parity bit, o più in generale codici molto potenti, per la correzione di errori.
Il segnale modulato serve solo per il livello fisico. I livelli più alti lavorano sul pacchetto, e quindi su stringhe di bit. Queste connessioni rappresentate sono connessioni punto-punto e non ci sono incertezze sul ricevitore. Nella depachetizzazione, quando il destinatario riceve il pacchetto, vengono estratti i bit all'interno del payload. I bit che costituiscono il payload vengono passati all'applicazione nell'ordine con la quale sono stati generati, ricostruendo così lo stream bit originario, a patto che però non ci siano stati errori.

2.1.3 domande d'esame

- Abbiamo un segnale con una banda da 10KHz e viene campionato a 30KHz. Viene quantizzato su 256 livelli. Per ridurre la distorsione è più conveniente raddoppiare il numero di livelli di quantizzazione o portare la frequenza di campionamento a 50KHz?
  Risposta: Aumentare i livelli, in quanto il campionamento è già maggiore del doppio della banda, per cui aumentarlo non apporterebbe migliorie.

- Abbiamo un segnale con una banda da 10KHz, può assumere valori da s0 a s1000. Viene rappresentato con 10 bit e viene campionato con una frequenza di 30KHz. Cosa possiamo fare per ridurre la distorsione?
  Risposta: Non possiamo fare nulla meglio di così, in quanto il segnale così può essere invertito, se facciamo le cose bene, in quanto il segnale aveva ampiezze discrete e quindi la quantizzazione può essere omessa.

- Ho un segnale che assume valori da s0 a s1100. Ha una banda di 10KHz. Definire il valore minimo dela frequenza di campionamento e il numero di bit per la codifica per evitare la distorsione.

Risposta: Useremo 11 bit = $upper(log_2(1100))$. Almeno 20 Khz per la frequenza di campionamento.

2.2 Codifica del testo

La codifica serve ad associare ad ogni si,bolo della sorgente una sequenza binaria univocamente interpretabile dal ricevitore come sequenza di quel simbolo.
Il testo è un esempio discreto della quale può essere effettuata la codifica. Il testo è formato da insiemi di caratteri definiti come repertoires. In ogni repertoires, ogni carattere è associato a un numero specifico chiamato codepoint. Ogni codepoint è memorizzato in un codice in byte.
Diversi repertoires possono essere associati a diversi codepoint. Per visualizzare correttamente il testo, il repertoires usato per la codifica deve essere conosciuto.
Nella codifica ascii originaria venivano definiti codepoint per lettere maiuscole e caratteri speciali, poi sono state aggiunte le lettere minuscole. Sono 127 simboli standard dell'ascii e vengono usati 7 bit per questo. L'ultimo bit, il 128esimo, è il bit di parità e assume valore 0 o 1 a seconda se si vogliano un numero di bit con valore 1 pari o dispari. I caratteri di controllo risultano essere 33.
Si è poi passato all'extended ascii repertoire, da 128 caratteri a 223, includendo così anche caratteri come lettere accentate. Questo standard era utile solo agli USA e l'europa occidentale. Sono nate così nuove versioni ascii per paesi diversi per soddisfare le esigenze di paesi tipo cina e india.
Nasce così l'unicode, uno standard che permette la codifica consistente dei diversi caratteri dei singoli repertoires.
UCS, l'universal character set è originariamente pensato come un insieme di caratteri mappabili con due byte, standard USC-2 = UTF-16. Viene successivamente aumentato a una codifica a 4 byt, USC-4 = UTF-32. Ogni carattere è identificato in maniera non ambigua da un nome e da un numero intero definito come codepoint. Il primo blocco da 64mila caratteri, dovuti a 16 but, è chiamato Basic Multilingual Plane.
I caratteri ascii possono essere rappresentati con 8 bit. Altri caratteri come lambda, o lettere di altre lingue, richiedono più byte. Come fa il decodificatore a capire e a interpretarli se non sono a prefisso? Introduciamo una transcodifica.
La UTF-8 permette di tradurre in maniera univoca da 1 a 6 ottette. Se il bit più significativo del primo byte è 0, allora il decodificatore interpreterà i 7 bit successivi come standard ascii. Altrimenti i numeri di 1 consecutivi sono il numero di ottette. Se ho 110, ho due ottette(considerando i 5 bit rimanenti del primo ottetto e i 6 bit rimanenti del secondo ottetto in quanto inizierà come 10 dato i bit di controllo), se il prefisso è 1110, ho 3 ottette e così via.
Ottette consecutive iniziano con 10 che sono bit di controllo e sono i primi 2 bit delle ottete. L'UDC si ottiene concctenando i bit rimanenti di ogni ottetto. L'10 a ogni ottetto si aggiunge per avere ridondanza r aumentando la robustezza del codice.

2.3 Codifica audio

La voce ha una densità di potenza spettrale nell'intervallo 200, 3400Hz. La larghezza di banda si approssima a 4KHz. Il rate di campionamento deve essere maggiore o uguale a 8KHz. La quantizzazione deve avvenire su 256 livelli, ovvero si usano 8 bit, o 128 se ti trovi in Asia o America. Il bitstream risultante è di 64kbit/s e presenta molta ridondanza.
Velocità in m/s moltiplicati per il tempo in cui li raccolgo mi da la dimensione.
La banda della musica è compresa tra i 15 e i 20000Hz. Campionando a 44.1KHz e quantizzando con 16 bit per campione, il potere di rumore di quantizzazione è -100dB. Ha un efficienza di codifica dell'87%. Il bitrate si calcola come 44.1x16x2(Dato che sono 2 i canali dello stereo)/0.87 = 1.622 Mbit/s.
La dimensione di un compact disc è di 730 MB. Il tempo totale di durata della soundtrack + 730x8/(1,62x0,87) che mi da il bitrate dell'audio. Da come risultato circa 70 minuti.
Per latenza si intende il ritardo della codifica che si aggiunge all'intero ritardo M2E. Il ritardo di decodifica può essere algoritmico o computazionale.
Per resilienza si intende connessione mobile che soggre di grandi problemi di latenza(?).
Statisticamente il 50% delle conversazioni è silenzio. Con le tecniche di soppressione si può salvare molta banda, se fatta bene ovviamente, perchè altrimenti può tagliare la fine delle parole. C'è una degradazione della qualità della conversazione se ciò acade. Sono stati introdotti successivamente codificatori molto avanzati:

- Waveform coders: cercano di ripetere il segnale negligendo la sua natura. Ha generalmente complessità bassa e qualità alta, ma richiede molto bitrate.
- Voice coders: non tentano di mantenere la forma del segnale, cercano invece di modellare come la sorgente del suono è prodotto.
- hybrid coders.

2.4 Meccanismi di codifica

La codifica a livello di simbolo di carattere serve a rappresentarlo in forma digitale, serve a dare un significato logico al flusso di byte, per questo motivo è necessario che i file che mandiamo abbiano una tipologia.
Il livello 4 offre 2 tipi di servizio: Il tcp che realizza una relazione logica a byte stream e l'udp che utilizza un modello basato sui messaggi.
Con il tcp, i byte generati vengono trasmessi e ricevuti nello stesso ordine al ricevitore, ma la pacchettizzazione non funziona molto bene. Posso trasmettere 10 byte, successivamente un blocco da 12, poi uno da 6 e così via. Per risolvere il problema, si è pensato di inviare dei byte specifici per la sincronizzazione, in modo da capire come interpretare il flusso di byte.
Si è introdotto un meccanismo che faceva uso di caratteri speciali che non risultavano stampabili, usati come caratteri di controllo per comunicazioni orientate a carattere di tipo sincrono. Il carattere SYN, tipo 00010110, carattere di sincronizzazione idle, svegliava il ricevitore e gli permetteva di acquisire lo stream dei byte.
Il ricevitore potrebbe perdere dei bit in fase di sincronizzazione, allora il trasmettitore invia SYN più volte, in modo che se dei bit si perdono, il ricevitore riesce a capire come è formata la sequenza di sincronismo.
Il ricevitore entra in funzione di HUNT e cerca di capire qual è la sequenza di sincronizzazione. Quando riesce a interpretare e entra in fase di sincronismo, il ricevitore inizierà a interpretare i bit. Il trasmettitore deve indicare al ricevitore che sta inviando il messaggio effettivo. Ci sono due caratteri di controllo: il DLE (00010000), ovvero il data link escape, e il STX(00000001) start of text. Questo canale rende la comunicazione più robusta. Quando il messaggio è terminato, vengono usati due caratteri: il DLE e il ETX(00000011), ovvero l'end of text.
Questo meccanismo funziona bene se il messaggio contiene solo caratteri stampabili. Non posso trasmettere caratteri terminatori all'interno del messaggio. Una possibile soluzione sarebbe inserire la lunghezza del messaggio all'invio, ma non si fa cosi. Cosa fare?
Inseriamo ulteriori caratteri di controllo, ma come si fa a non confonderli con altri caratteri terminatori? Ogni volta che c'è un carattere di controllo, i ltrasmettitore ne aggiunge un altro. Cosi facendo, si spezzano il DLE trasmesso con DLE e ETX. Il ricevitore rimuoverà il DLE aggiuntivo. Questa non è una procedura ambigua in assenza di errori. Vediamo ora un piccolo esercizio:
La sequenza mandata è: SYN SYN SYN DLE STX DLE DLE DLE DLE ETX SYN DLE STX SYN SYN SYN DLE ETX
I messaggi che vengono ricevuti sono due:
DLE DLE
SYN SYN
Una seconda sequenza è la seguente: SYN DLE DLE ETX DLE STX DLE DLE STX DLE DLE DLE ETX DLE ETX
Questa seconda sequenza presenta un errore, in quanto il primo messaggio è errato. Viene trasmesso un DLE che viene seguito da una codifica sbagliata perchè contiene un ETX, seguito da un DLE e STX, quindi il messaggio non è terminato. Vengono dunque scartati tutti i caratteri fino all'inizio di un nuovo messaggio.

2.5 Digitalizzazione di immagini e video

Come si acquisisce una foto digitale?
L'immagine del piano reale viene convertita in una matrice finita di pixel digitali da una matrice di diodi fotosensibili. Questa matrice di diodi prende il nome di filtro a mosaico. I fotodiodi sono pensati per reagire a una certa gamma di colore, si usa un filtro a mosaico e digitalmente si ricreano le componenti di colore attraverso l'analisi delle componenti adiacenti.
Questa operazione a livello di digitalizzazione consiste nel campionare lo spazio, c'è una digitalizzazione del dominio. Dato che l'immagine è composta da pixel, minore saranno i pixel impiegati per l'immagine, minore sarà il dettaglio.
Si utilizza una codifica per rosso, verde e blu con 8 bit ciascuno, quindi 256 livelli disponibili per colore. In caso del bianco e nero, vengono usati 8 bit per la scala dei grigi.
Abbiamo una profondità di colore di 24 bit e 16 milioni di colori. La rappresentazione RGB viene usata nei display digitali, ma ci sono vie alternative per rappresentare i colori:

- CMY, ciano, magenta e giallo. Risulta essere una rappresentazione sottrattiva, il bianco è il colore base e la loro sovrapossizione risulta nel nero. Usata solitamente nelle stampanti.
- CMYK, ciano, magenta, giallo e nero.
- YCbCr, dominanza, combinanza blu e combinanza rossa. Risulta essere analoga alla rappresentazione RGB, di fatto si può passare da una rappresentazione all'altra in quanto sono identiche a livello informativo. Risulta più facile da comprimere in quanto la maggior parte dell'informazione si trova all'interno della parte Y e non nelle altre due. Questo ci permette di comprimere molto di più le parti Cb e Cr. Cambiando il dominio, solitamente non si perde il codominio.

Una immagine in formato raw occupa uno spazio che è dato dal numero di pixel moltiplicato per la profondità di colore. Un video è una sequenza di immagini che vengono riprodotte con una cadenza predeterminata, detta frame rate. Il frame rate tipico è di 24 o 30 frame al secondo, ma può cambiare in base a vari fattori. La dimensione di un video non compresso è dato dalla dimensione dell'immagine moltiplicata per il frame rate. Un film della durata di 90 minuti, senza compressione e a colori, occuperebbe 2700Gbit, circa 1 Tb di memoria.

2.6 Codifica di sorgente

La codifica di sorgente serve a eliminare la ridondanza del senale e a eliminare parte dell'informazione, rendendo più facile la comunicazione del segnale. Serve a ridurre la dimensione dei dati che devono essere trasmessi. Aggiunge ridondanza al segnale, rendendolo più robusto e permettendo il riconoscimento di errori e la loro correzione.
La codifica può essere lossless, ovvero che non introduce perdita di informazione durante la codifica e riduce la ridondanza dei dati. La decodifica ripristina i dati originali.
La codifica può essere lossy, ovvero introduce perdita di informazione durante la codifica. La decodifica produce un segnale simile all'originale ma con possibili errori o distorsioni.

2.6.1 Compressione lossless

Abbiamo due modalità per rappresentare lunghezze binarie, ovvero codeword:

- lunghezza fissa, si fa uso dell'ascii, molto facili da interpretare ma non efficienti.
- lunghezza variabile, utf-8, più efficiente ma richiede molto più lavoro in fase di decodifica.

Il codice morse è un esempio di codice a lunghezza variabile, si usano sequenze più corte per i caratteri di uso più comune. Non è un codice a prefisso, per cui ho bisogno di pause tra la trasmissione delle lettere. Per capire quanto buono è un codice si usa la lunghezza di codice, ovvero $L = \sum_{s_i in S} L_i p_i [bit/s]$
La miglior codifica lossless per una sorgente di simboli indipendenti ha una lunghezza media compresa tra H e H+1, dove H è l'entropia del codice definita come segue: $\sum_{s_i in S} log_2(\frac{1}{p_i})$
La lunghezza del codice ottima è ottimizzata quando tutti i simboli sono equiprobabili. Se l'albero che mi rappresenta la parola di codice è sbilanciato, i simboli saranno sicuramente non equiprobabili. La lunghezza è la media della profondità delle foglie dell'albero. Con un albero bilanciato la media diminuisce. Huffman ha ottenuto un modo per avere un codice a prefisso ottimo.
La codifica di Huffman risulta essere una codifica di tipo entropica ed è in grado di produrre un codice a lunghezza minima, ovvero entro un bit dall'entropia della sorgente. Si può ottenere un codice del genere se la sorgente è ideale, ovvero se la distribuzione dei simboli è nota a priori e i simboli sono indipendenti tra loro.
Alcuni esempi sono:

- Testo, dove i simboli non sono indipendenti. La probabilità di trovare una u è più alta quando la lettera che la precede è una q.
- Audio, un brano consiste in silenzi e segmenti di voce. Il silenzio si può modellare con una gaussiana, nel caso della voce abbiamo una correlazione e una distribuzione dei diversi elementi che dipendono dalla sillaba.
- Immagini, hanno una correlazione spaziale.

Come si possono usare codifiche dj sprgenti eggicienti quando le due condizioni non sono soddisfatte?
Trasformiamo la sorgente finchè non troviamo una soluzione più vicina possibile alla teoria di conversione entropica (è possibile vedere uno schema nella figura 2.11).

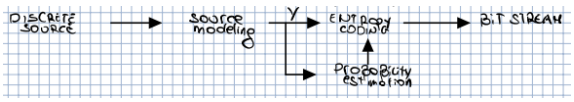

Figure 2.11: Schema della lavorazione per una codifica entropica

Sbiancamento

Il processo si dice bianco quando tutti i simboli sono indipendenti. Sbiancamento indica la procedura per rendere un processo bianco. Questa operazione si divide in:

- Pattern matching: identificare e rappresentare i pattern ripetuti all'interno del segnale. Riconosco i pattern e mando gli indici dei pattern. Si posono avere pattern come mostrati in figura 2.12, ovver delta,2delta,3delta e 4delta fanno parte del pattern 1.

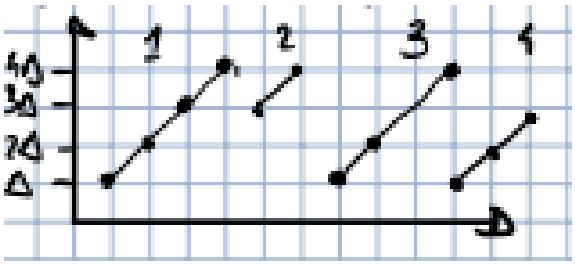

Figure 2.12: Pattern

- Codifica del codice: si usa un dizionario e si trasmettono solo le parole presenti al'interno di esso.
- Predizione: immaginiamo di avere un segnale variabile nel tempo e si cerca di predirre il simbolo successivo. Calcolando la differenza tra la predizione e il simbolo ricevuto si ottengono dei simboli indipendenti. Come possibile vedere in figura 2.13, solitamente si usa una predizione a linea retta. Vengono usate per la predizione di segnali

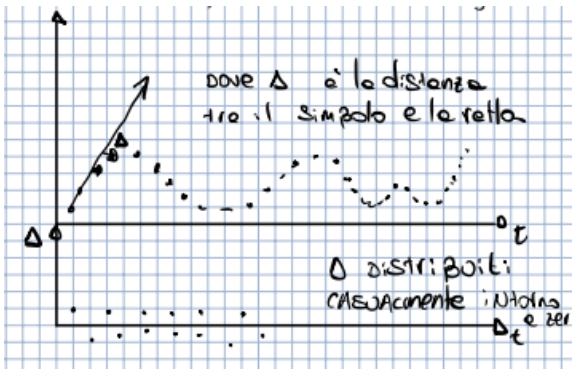

Figure 2.13: Predizione a linea retta

audio e tipicamente introduce una distorsione.

Un esempio di sbiancamento attraverso trasformazione si usa per il formato mp3, ovvero si osserva il segnale in un altro dominio, ovvero non un dominio temporale ma un dominio di frequenze, ottenuto tramite le trasformate di Fourier.

Run-Length Coding

Un'altra tecnica che si usa spesso per rompere la correlazione di simboli è la Run-Lenght Coding (RLC), e viene impiegata in molti campi. Una volta che si ha una serie di bit, questa tecnica permette di rompere le correlazioni. Si possono identificare sequenze di simboli idetici e cercare la ripetizione di quel simbolo della sequenza. Dato che le sorgenti sono binarie, basta ripetere quel simbolo per le volte che serve, senza sapere quale valore sia associato a ogni simbolo. Si ripeteranno tanti uni, seguiti da tot vole zeri, seguiti al loro volta da tot uni o viceversa. Vediamone ora un esempio:
Una sorgente emette sempre simboli con un pattern predeterminato, ovvero S-E-N-W. Può emettere un numero casuale di simboli uguali prima di passare al prossimo. Si ottiene una sequenza del tipo:
SSSS EE NNN WWWW S EEEEE W, che viene tradotta in 4 2 3 4 1 5 0 1
Una volta che la sequenza di bit è trasformata in una serie di numeri, essi possono essere codificati attraverso la codifica di Huffman vista prima.
Anche con questa modalità c'è una sorta di correlazione, ovvero un numero alto sara sempre seguito da un umero basso e così via. Utilizzo così una codifica di Huffman, dove separo i vari simboli trovati dopo lo sbilanciamento. Si possono usare varianti nella quale si usano gli 0 per la sequenza e gli 1 per spezzare la sequenza.

0000010011000 si tradurrebbe in 5,2,0,3.
Queste tecniche si usano per sequenze di valori che accettano anche valori più grandi di 1. Si usa il formato (skip, value), ovvero si utilizza oltre alla grandezza di simboli identici anche il simbolo che spezza la sequenza:
6 7 0 0 3 0 2 2 0 0 .... 0, ottenendo (0,6)(7,0)(2,3)(1,2)(0,2)(0,0).
La coppia speciale 0,0 indica che tutti gli elementi della sequenza rimanenti sono 0.
La parte value può essere codificata attraverso la metodologia sss, quindi la sequenza diventerebbe (0,011/110)(0,011/111),(2,010/11) e così via.

Codifica (sss,value)

I tre bit sss vengono usati per codificare il valore di value. Ad esempio:
01011001110110100 può essere scritto come (010,11),(001,1),(101,10100) che ha come risultato 3,1,20. Risulta essere una codifica costosa, ma permette di risparmiare sulle sequenze più corte. Il valore sss mi dice quanti bit impiegare per rappresentare i valori, mentre value è il valore effettivo che voglio rappresentare. Ho, però, un limite fisso sul codificare da 0 a 127 valori, ovvero quando sss vale 111. Come ovvio a questo problema? Riservo una terna di valori che mi indica che la codifica va ampliata, ad esempio un sss che vale 000 non va a codificarmi alcun bit, per cui la uso come sequenza di controllo, e quindi i prossimi sss,value mi dicono gli altri bit da aggiungere per ottenere il valore che io ho codificato. Questo trucco si può usare iterativamente fino a che non viene raggiunto il valore desiderato.

2.7 Compressione di immagini

In questa sezione viene discussa la codifica jpeg. Viene usata principalmente in formato lossy, più l'immagine è compressa più farà schifo una volta interpretata dal ricevitore. Quando un'immagine è ricostruita prende il nome di artefatto. Quando l'immagine viene decompressa a schermo occupa lo stesso spazio dell'immagine che non è stata compressa, ovvero dell'immagine raw. Come parametri per la qualità si usano le seguenti formule:

- $PSNR = 10log_{10}(\frac{255^2}{E[e^2]})$ dove e è l'errore.
- $SSIM(x,y) = \frac{(2u_xu_y+c_1)(2\sigma_{xy}+c_2)}{(u_x^2+u_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$, richiede più computazione ma è più vicina alla percezione umana

La compressione jpeg avviene attraverso 5 passi fondamentali:

- conversione YUV, da RGB a YCbCr.
- partizione in blocchi.
- DCT, descrete cosin transform. Si cambia il dominio del segnale.
- quantizzazione, serve a ridurre il range di valori che possono assumere i bit.
- codifica entropica.

Si effettua una conversione da RGB a YCbCr, si porta meno informazione sul piano di luminanza. Si può fare un sotto-campionamento, ovvero un'operazione di CIF per ridurre l'informazione.
Si lavora dividendo l'immagine su blocchetti di 8x8 pixel, in quanto le immagini in uno stesso blocco tendono a essere omogenee. Sottraiamo ai valori del nostro blocco il valore 128, in modo che il blocco che ottengo sia bilanciato intorno allo zero. Il blocco viene trasformato attraverso la DCT, un'operazione invertibile attraverso l'utilizzo di questa formula: $F[i,j] = \frac{1}{4}C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7}P[x,y]cos\frac{(2x+1)i\pi}{16})sin\frac{(2y+1)j\pi}{16})$, dove la funzione C(h) è definita come segue:
$\begin{cases} 1 \quad se \quad h = 1,...,7 \\ \frac{1}{\sqrt{2}} \quad se \quad h = 0 \end{cases}$
La DCT contiene i valori più significativi in alto a sinistra. Rappresenta la media dei pixel della matrice 8x8 della quale si è calcolata la DCT. Gli elementi all'interno della matrice tendono allo zero se ci si sposta verso l'angolo in basso a destra.
Dopo la DCT, si effettua la quantizzazione sul blocco 8x8 usando una matrice nota sia al trasmettitore che al ricevitore. Dovrebbe prendere il nome di compressione per essere più tecnici. Durante questa fase si introduce distorsione.
Come si aumenta la compressione?
Si aumentano i coefficienti della matrice di quantizzazione, in quanto più zeri si hanno nella matrice risultante, meno informazione sarà quella ricevuta.
L'ultimo passo è la codifica entropica che risulta essere un misto di varie operazioni. Inizialmente si usa il vectorin, che viene riportato come segue.

Vectoring

Prende i 64 valori della matrice e li inserisce all'interno di un vettore. Si utilizza un pattern a zig zag, si parte da in alto a sinistra e si prosegue fino ad arrivare alla fine all'angolo in basso a destra. Il primo elemento è il componente DC, ovvero l'elemento più grande. Si prendono i i coefficiendi DC di tutti i vettori, si inseriscono in un vettore e si esegue una codifica differenzale. I delta di due cofficienti DC vicini è solitamente piccolo. Ottenuta la sequenza differenziale, si può codificare i diversi elementi con una modalità sss,value, dove i valori negativi si rappresentano con il complemento a 1.
Ora si passa a codificare i valori AC con un RLE(che equivale all'RLC). In fine si prendono i campi sss della codifica del DC e so effettua un'altra codifica differenziale. Il passo finale è la codifica di Huffman di quanto trovato prima.

2.8 Compressione video

Una sequenza di immagini in sequenza a 24 o 30 frame/s compone un video. C'è oltre alla compressione delle immagini una correlazione temporale: probabilmente il frame successivo sarà molto simile al frame precedente.
Il frame rate deve risultare il minimo possibile per avere un'immagine fluida.
Le informazioni che vengono passate sono solitamente solo quelle dei pixel che variano, evitando così di trasmettere i pixel che rimangono costanti. Più l'immagine risulta essere nitida più la complessità della codifica risulta essere alta.
Quando si parla di intraframe ci si riferisce all'interno del frame e si sfrutta una ridondanza spaziale. Quando si parla di interframe ci si riferisce allo spazio tra due frame e si sfrutta una ridondanza temporale.

2.8.1 Group of Picture-GOP

Data una sequenza di frame, li si divide in gruppi di frame di dimensione prestabilita (è uno dei parametri di codifica) ognuno dei quali prende il nome di GOP, group of Pictures.

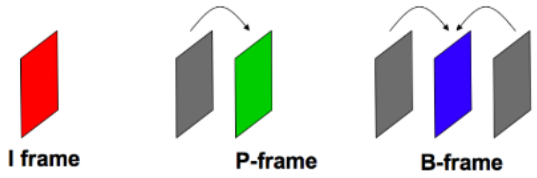

Figure 2.14: Esempio di frames

2.8.2 Frame

I Frame

Un GOP inizia sempre con un frame di tipo I, attraverso l'intraframe coding, e viene codificato come se fosse un'immagine e funge da riferimento per gli altri frame. Questo risulta essere la codifica più pesante possibile perchè non usa la ridondanza temporale. Simile alla codifica jpeg, soltanto che si lavora su una matrice 16x16 invece che su una matrice 8x8, resa più veloce grazie al parallelismo. Questo porta a problemi con gli artefatti, però viene sminuito dal fatto che un frame rimane meno tempo sullo schermo.

P Frame

Sono codificati prendendo come riferimento i frame precedenti. Viene utilizzata una codifica predittiva.

B Frame

Simili ai P Frame ma tengono in considerazione sia il frame precedente che quello successivo. In base a quello più vicino al B frame in considerazione viene decisa la codifica da utilizzare.

2.8.3 Motion estimation

P Frame

Idealmente sarebbe ottimo inviare solo i pixel che cambiano o descrivere il movimento dell'immagine, ma nella realtà risulta essere molto complicato.

Si divide l'immagine a griglie e si cerca di capire come quel quadratino è spostato nel frame successivo. Per cercarlo, si considerano i pixel intorno al pixel bersaglio nel frame successivo, e controllo se c'è qualcosa che gli somigli molto. Codifico un vettore di spostamento. I blocchi saranno simili ma non identici: identificato il blocco più simile a quello di partenza, calcolo la differenza. Spedisco al ricevitore il motion vector e l'errore di predizione.

B Frame

Identico a prima, ma si guarda sia nel frame prima che in quello dopo. Nella figura 2.15 è possibile vedere un esempio di GOP: Visto che i P e i B frame sono più leggeri, sarebbe comodo fare GOP molto lunghi, ma ciò creerebbe problemi in

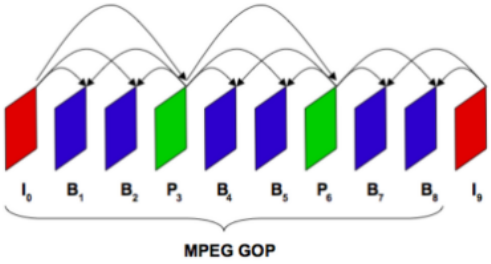

Figure 2.15: Esempio di GOP

quanto certamente si avrà all'interno dei cambi di scena, che renderebbero d'obbligo una ricodifica di tutto.
Avere GOP troppo lunghi crea problemi con gli artefatti che si prolungano per tutto il GOP.

Chapter 3

Rete

3.1 Performance di rete

KPI = strumenti per misurare alcune metriche del flusso di rete. Keep Performance Indicator.

- Bandwidth W [Hz]: insieme di frequenze in un certo mach trasmissivo che subiscono un'attenuazione limitata durante la propagazione. Di fatto, le frequenze che in un certo caso possono essere trasmesse senza avere un'attenuaziuone eccessiva
- Bitrate: quantità di bit che posso trasmettere su un cavo in un secondo. Ovvero la massima velocità raggiungibile.
- Teorema della capacità di shannon: lega banda e bitrate. In certe condizioni il bitrate massimo che posso ottenere da una certa banda è limitato da $W * log_2(1 + SMR)$, dove SMR è il signal to mask ratio.
- Throughput: La velocità di trasferimento dal punto di vista della comunicazione. Simile al bitrate ma a livello di applicazione.
- Overhead: Il costo di trasportare dati in aggiunta rispetto a quelli codificati, com egli header dei livelli ISO-OSI. Abbiamo:
  - SDU: Service Data Unit, ovvero il payload o carico pagante.
  - PDU: Protocol Data Unit

  L'SDU di un livello non è altro che la PDU del livello sovrastande, Vedi figura 3.1 Overhead altro non significa che

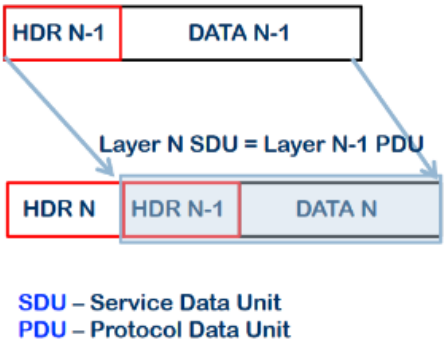

Figure 3.1: Overhead

  aggiungere controlli rispetto alle informazioni trasmesse.

3.2 Bluetooth

Il bluetooth ha bisogno di avere una prima parte predeterminata (header) in modo da semplificare la codifica della prima parte, che indica come codificare la seconda. Vedi figura 3.2. Il trailer serve a trovare eventuali errori. Per trasmettere l'header si usa il bitrate più basso, quello più sicuro, mentre il resto può avere un rate diverso.

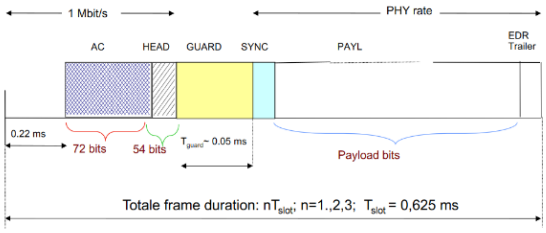

Figure 3.2: Bluetooth

3.2.1 Calcolare il Throughput

Il thourhput è un rate, definito come delivered data/time, dipende dunque dall'intervallo temporale. Definendo b(t) come il totale di bit consegnati fino al tempo t, possiamo definire le seguenti metriche di throughput:

- Throughput instantaneo, ovvero la derivata su t di b(t): s(t) = b'(t)
- Average Throughput in un intervallo di tempo T: s(t;T) = $\frac{(b(t)-b(t-T)}{T}$
- Long-term, o steady-state, o asymptotic throughput: S = $\lim_{T\to\infty} s(t;T)$

Richiede come bound fondamentale che S sia minore o uguale a R0, dove R0 indica il bitrate.

3.2.2 Overhead

In uno stack di protocollo di rete, ogni layer offre un servizio ai livelli sovrastanti. L'SDU rappresenta il frame di dati che posono essere processati da uno specifico servizio. Per servire l'SDU, il servizio necessita di aggiungere alcune segnalazioni e informazioni di controllo definite come heade. Il frame ottenuto considerando Header e SDU prende il nome di PDU. L'overhead è il costo di trasportare dati di controllo o segnale aggiuntivi al payload. Il throughput che l'applicazione può ricevere deve essere obbligatoriamente interrotto dal costo di overhead. La formula risulta essere:
$S_n^g = S_n X \frac{|SDU_n|}{|PDU_n|}$

3.3 Delay

Il ritardo può essere causato da 4 fattori:

- Bodal Processing: controllare errori nei bit e determinare l'output link. Può essere migliorato attraverso l'uso di meccanismi di routing.
- Queueing: il tempo di attesa in un link di output per la trasmissione. Dipende a livello di congestione del router.Risulta essere il tempo che un messaggio aspetta in coda per essere processato. Dipende fondamentalmente dalla struttura dei collegamenti e anche dal traffico di pacchetti.
- Trasmission delay: Dovuto al tempo per mandate i bit attraverso il link. Risulta essere il rapporto tra la lunghezza di banda R e la lunghezza del pacchetto L:
  $\frac{L}{R}$.
  Può essere ridotto aumentando il bitrate.
- Propagation delay: Data d come lunghezza fisica del link e s come tempo di propagazione medio, circa $2 * 10^8$, il propagation delay è definito come:
  $\frac{d}{s}$.
  Non dipende dal bit rate ma dal tempo che impiega il mezzo fisico a attraversare la distanza. Gioca un ruolo nell'intasamento dei nodi: se è troppo lungo, i nodi si intasano (tempo di vulnerabilità = il tempo tra due trasmissioni necessario perchè non si sovrappongano).

Il delay nodale viene definito come: $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$
Il queueing delay può essere calcolato come segue: $S = \frac{LxA}{R}$
dove R è la bandwidth del link, L la lunghezza del pacchetto e A la media dei pacchetti che arrivano. Se S tende a 0, il tempo medio di queueing delay è piccolo, se risulta 1 vuol dire che il ritardo è diventato grande. Se è più grande di 1, arriva più lavoro di quello che può essere processato e la rete si congestiona.

3.3.1 Jitter

Il jitter è una variante del delay molto presente durante le attività di streaming. Jitter larghi implica lunghi ritardi di playout e grandi playout buffer. Il contenuto viene bufferizzato per un pò di tempo prima che venga utilizzato. Jitter larghi comportano queue più lunghe. Riportiamo la Pollaczeck Khintchine formula per M/G/1 queueing systems:
$m_q = \frac{\sigma_v^2}{2(1-m_v)} - \frac{m_v}{2}$
Dove m_q è la lunghezza media della queue, σ_v^2 indica il jitter e m_v è la media degli arrivi in un tempo medio di servizio.

3.4 Metrica di Pipe Capaity- BDP

La BDP: bandwidth delay product[bits][pcks], indica il numero di bit che possono essere contemporaneamente in "volo" all'interno di una connessione. O il numero totale di bit che possono essere consegnati al ricevitore in un tempo uguale al ritardo end-to-end (d_{e2e}). Pensando al flusso di dati generato dalla sorgente come un fluido, e alla connessione come un tubo digitale, il BDP sarebbe la quantità di liquido che riempie quel tubo.
Più largo è il BDP, più largo sarà il numero di bit/pacchetti che il mittende deve trasmettere mentre sta aspettando di essere riconosciuto dal ricevitore, in modo da usare completamente la capacità del link.

3.5 Metriche di affidabilità

Sono legate al tasso di errori per pacchetto. I pacchetti con errori sono uno dei problemi maggiori perchè i nodi sono fatti in modo che se arriva un pacchetto con errore lo richiedono, quindi si perde del tempo. Cosa accade se c'è troppo ritardo? Ho due scuole di pensiero:

- Drop tail policy: scarto tutto quello che mi arriva dopo che ho riempito il buffer
- Random early dropping: si buttano pacchetti a caso già in coda quando il buffer è a 2/3 di capienza.

Packet Loss Rate - PLOSS

frazione complessiva dei pacchetti trasmessi che non sono arrivati a destinazione a causa di una qualsiasi motivazione.

Packet delivery ratio

Conosciuta anche come Packet delivery Rate. La PDR = 1 - PLOSS, ovvero è la probabilità complementare alla PLOSS. Frazione complessiva di pacchetti che soino stati consegnati con successo a destinazione. Risulta essere il rateo tra il numero di pacchetti consegnati con successo sul numero di pacchetti trasmessi.
Il PDR può essere differente dai protocolli dei vari livelli, e dipende dal meccanismo di recupero di ogni layer.

Il vero internet

Funziona col best effort service, ovvero fa il meglio che può, senza dare garanzie. Il livello di garanzia è implementato nelle applicazioni multimediali.

3.6 Quality of Service e Quality of Experience

Il primo tentativo di classificare la qualità è stata la QoS. Ci si concentra solo sul servizio in senso stretto.

3.6.1 QoE

Si vuole muovere l'attenzione sulla persona. Il primo modo è ovviamente quello di chedere un parere agli utenti.

Esperimenti soggettivi

Chiedo alle persone di fare esperimenti e di dare una valutazione. Richiede molto tempo per preparazione e esecuzione.

Analisi dei dati

Parte molto importante, include lo screening degli utenti e similari.

Mean opinion score

Indica il livello di soddisfazione per un determinato servizio. Tipicamente con una scala numerica, oppure un like o cose così. Il test deve però essere riproducibile, in modo da poter replicare l'esperimento.

3.6.2 Esperimenti oggettivi

Prova a predirre quello che le persone direbbero senza dover impiegare persone vere.

Full reference

Sono quelle che funzionano meglio. Usano l'immagine di partenza e l'immagine modificata e cercano di vedere la differenza tra i due. Serve obbligatoriamente l'originale, cosa non sempre possibile. Le due metriche più usate sono PSNR e SSIM, per entrambi, valori più alti significano qualità più alta.

- PSNR: si basa sulla differenza pixel a pixel tra l'immagine originale e quella modificata. Però non funziona perchè valutando le cose pixel a pixel gli sfugge la visione completa, che può essere maggiore rispetto alle singole differenze. Di fatto, valuta la differenza e non la qualità in se.
- SSIM: Valuta tante cose: contrasto, geometria e similari. Presenta un valore da 0 a 1. Riesce a fornire valori più credibili.

Reduced Reference

Ho meno informazioni dell'immagine originale. Estraggo dall'immagine di partenza delle caratteristiche e idem da quella che devo confrontare. Vengono confrontate solo le informazioni estratto. Il problema ovviamente è come scegliere quali feature estrarre.

No reference

Diventa un pò un casino perchè sensza l'originale è praticamente impossibile.
Brisque e NIQE: analizzano per ragioni l'immagine che deve essere valutata e un punteggio più basso indica un'immagine migliore.

3.6.3 QoE

Si sposta l'attenzione sull'utente finale e sulla sua esperienza. Su questo aspetto impatta qualsiasi cosa: psicologia, mezzi, contesto sociale e similari.

3.7 Framework DASH

DASH, o Dinamic Adaptive Streaming Over HTTP, è un framework di streaming video (ISO/IEC 23009-1).
Dash è un insieme di regole che abilitano la trasmissione di flussi multimediali in maniera efficiente in internet. Definisce come rappresentare le possibilità di codifica video. Non definisce come vengono codificati, quali tecniche usare, come viene fruito il video. Così facendo posso sviluppare applicativi molto diversi tra loro ma compliant col DASH.
Uno dei principi è quello di poter usare le tecnologie già presenti e usare streaming video basato sul protocollo HTTP.
Sono reti overlay: sovrapposte alla rete fisica. Non sviluppa protocolli di basso livello a HTTP.
TCP-IP: non ha protocolli di controllo e quindi non introduce latenza, perchè si dava precedenza alla latenza piuttosto che all'affidabilità.
DASH si appoggia al TCP, due livelli più in alto rispetto al protocollo usato prima, ovvero l'UDT. I vantaggi che presenta sono i seguenti:

- Robustezza.
- Non c'è fireqall di sicurezza che filtra HTTP.
- HTTP non è filtrato dai middleboxes (RAGIONE PRINCIPALE).
- Qualità video che dipende dalla rete e dal device.
- Cambio server senza soluzione di continuità. Il server può cambiare man mano che ci si muove.
- Sposto l'intelligenza dalla rete al client, comporta scalabilità e utilizzo di reti non dedicate.

- Flessibilità nella tipologia di fornitura video.

Come funziona?
C'è uno streaming video, che può essere implementato in diversi livelli di qualità: basso, medio o alto. Il video è spezzato in segmenti, o chunk, di qualche centinaio di ms, fino ad un tempo massimo di 1s. Considerando ogni pezzo come unità a asè stante, valuto i tempi di download di ogni chunk e in base a quello decido quale livello di qualità chiedere per il blocco successivo.

3.7.1 MPD- Media presentation description

Un file che contiene tutte le informazioni sul video, o sul blocco video, che si sta aprendo: le qualità disponibili, gli indirizzi HTTP e altro. Nel MPD troviamo anche il bitrate della sorgente i nbase alla qualità che si andrà a chiedere. In figura 3.3 è riportato uno schema.

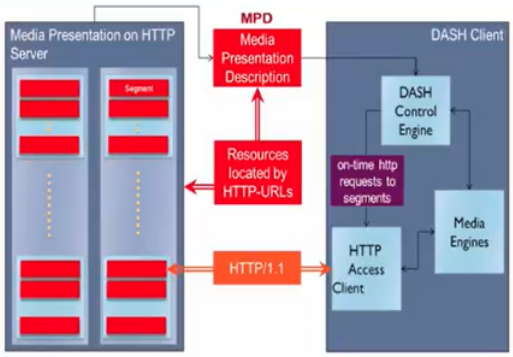

Figure 3.3: Schema MPD

Chapter 4

Esercizi svolti