UNIVERSITÀ DEGLI STUDI DI NAPOLI
"FEDERICO II"

FACOLTÀ DI INGEGNERIA

TESI DI LAUREA IN
TELECOMUNICAZIONI

TRASMISSIONE VIDEO SU RETE A COMMUTAZIONE DI PACCHETTO

RELATORE
ch.mo prof. Giovanni Poggi
CO-RELATORE
ing. Stefano Vignola

CANDIDATO
Antonio Caputo
matr. 039 / 820

ANNO ACCADEMICO 2000-2001

Ringraziamenti

Prima di iniziare è doveroso rivolgere un ringraziamento a chi mi ha permesso di fare quello che mi piace, quindi innanzitutto devo ringraziare il mio relatore Prof. Giovanni Poggi, che mi ha dato fiducia e si è mostrato sempre paziente e disponibile.

Un particolare ringraziamento va poi alla mia famiglia, che mi ha supportato (e sopportato) lungo tutto il cammino che mi ha portato a questo momento, e che mi ha permesso di completare la mia carriera di studi.

Un ringranziamento va anche al co-relatore ing. Stefano Vignola, che mi ha fornito un prezioso aiuto di natura tecnica (ma anche didattica) nei momenti di difficoltà.

Infine (ma non in ordine di importanza) voglio ringraziare tutti i miei amici, è anche grazie a loro che questi anni di studio sono scivolati via in modo sereno e divertente.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'ultimo decennio è stato caratterizzato dalla inarrestabile crescita della rete *Internet*, che ormai si appresta a diventare uno dei principali mezzi di comunicazione. Inizialmente erano disponibili poche semplici applicazioni, tipicamente basate sullo scambio di piccoli file di dati (e-mail, newsgroup, telnet, ftp), ma con il passare degli anni a questi si sono affiancati altri servizi che hanno radicalmente cambiato l'aspetto della rete. Primo fra tutti va sicuramente ricordato il *World Wide Web*, letteralmente ragnatela attorno al mondo, che permette di mettere a disposizione di tutti delle pagine contenenti informazioni di diversa natura (testo, immagini, suoni...). Il Web ha avuto certamente un ruolo determinante nella crescita di Internet, ed ha ancora oggi un ruolo predominante nello sviluppo vertiginoso della rete globale.

Negli ultimi anni, si è verificato un interesse sempre maggiore verso la trasmissione di flussi multimediali in tempo reale. Sin dalla sua nascita, Internet ha mostrato grosse potenzialità come mezzo di comunicazione alternativo. Se inizialmente, però, questo ruolo era sostanzialmente affidato alla sola posta elettronica, con l'allargarsi dell'utenza ed il progredire della tecnologia sono state presentate molte alternative. Prima si è assistito alla nascita delle cosiddette "chat", grazie alle quali più persone possono conversare in tempo reale, tramite semplici messaggi di testo, il passo successivo è stato segnato dai programmi di istant messaging, con i quali è possibile selezionare la persona o le persone con le quali si vuole conversare. La naturale evoluzione di questi programmi consiste nella possibilità di comunicare utilizzando anche audio e video, realizzando di fatto una sessione di videoconferenza casalinga, ed è quello a cui si sta assistendo in questi mesi.

Tuttavia, per come è strutturata, Internet non è certamente il mezzo migliore per tale tipo di applicazioni, poiché, come tutte le reti a commutazione di pacchetto, è molto più efficiente nella trasmissione dati che non per il traffico multimediale. Infatti i principali requisiti di una trasmissione multimediale in tempo reale sono:

- disponibilità di un'adeguata ampiezza di banda;
- disponibilità di una sufficiente potenza di calcolo;
- possibilità di garantire limitati ritardi massimi nella consegna dei pacchetti;
- variazione contenuta dei ritardi di consegna.

I primi due punti sono complicati ulteriormente dalla natura eterogenea dell'utenza (con canali che vanno da pochi kbit/s ad alcuni Mbit/s, e terminali potenti come i PC o semplici come i cellulari 3G). Per superare questi limiti è essenziale ricorrere a tecniche *scalabili* di codifica e decodifica video, tali cioè da permettere la ricezione della sequenza desiderata ad un livello di qualità selezionato dal generico utente e compatibile con le risorse garantite dalla rete e disponibili al terminale. Inoltre, particolare attenzione deve essere posta sulla complessità computazionale, specialmente per quanto riguarda gli algoritmi di codifica (tipicamente la decodifica non è molto impegnativa).

Gli aspetti legati ai ritardi di consegna, invece, sono certamente più critici. In Internet il trasporto dei pacchetti dalla sorgente alla destinazione è affidato ad uno dei due protocolli **TCP** (Transmission Control Protocol) e **UDP** (User Datagram Protocol), che si trovano sopra il protocollo di livello rete **IP** (Internet Protocol). IP è un protocollo best-effort, cioè fa del suo meglio per trasmettere i pacchetti ma non garantisce che questi arrivino a destinazione, nè tanto meno che ci arrivino nell'ordine in cui sono stati spediti. Queste funzionalità, se veramente necessarie, vanno introdotte dai protocolli di livello trasporto. TCP allora aggiunge, tra le altre cose, la numerazione dei pacchetti e la ritrasmissione nel caso di perdite, garantendo così la consegna di tutti i pacchetti e rispettando anche l'ordine originario, chiaramente a discapito della velocità di trasmissione. Tutto questo rende TCP un protocollo adeguato per la trasmissione di dati, quindi per applicazioni nelle quali anche il dover aspettare la ritrasmissione di un pacchetto non è problematico, ma praticamente inutilizzabile per i flussi multimediali, per i quali paradossalmente ricevere un pacchetto con un ritardo elevato è più dannoso che non riceverlo affatto. Per questo motivo tipicamente, le applicazioni multimediali utilizzano come protocollo di trasporto UDP. Quest'ultimo non offre le stesse garanzie di TCP, ma risulta sicuramente più adatto in quanto l'unico suo scopo è di consegnare nel modo più veloce possibile ogni singolo pacchetto, senza però curarsi di problemi di riordino o di ritrasmissione. Tuttavia ci sono altri fattori che limitano l'uso dello stesso UDP nelle trasmissioni real-time;

infatti mancano funzionalità come la numerazione dei pacchetti[1], il timestamping[2], l'identificazione del contenuto del pacchetto[3], e altre ancora.

Queste ed altre funzionalità vengono garantite dal protocollo **RTP** (Real Time Protocol). Inizialmente disegnato per essere indipendente dal protocollo sottostante, tipicamente RTP viene posto al di sopra di UDP, completandolo quindi di quelle caratteristiche che ne fanno un protocollo adatto alle trasmissioni real-time.

Tuttavia, per poter garantire gli ultimi due punti di cui sopra, nemmeno questo è sufficiente, e solo tecniche che permettano di garantire la *qualità del servizio (QoS)* possono soddisfare quelle richieste. Internet non è stata pensata per garantire la QoS, e per questo motivo si stanno studiando diverse soluzioni che permetteranno di estenderla da questo punto di vista. Le più promettenti sono probabilmente *Integrated Services* e *Differentiated Services*, entrambe le quali permettono di aggiungere nuovi servizi all'attuale modello di rete senza doverne stravolgere la struttura.

In questo lavoro di tesi si è affrontato il problema della pacchettizzazione e spedizione di un flusso video codificato. La trasmissione avviene utilizzando il protocollo RTP. L'algoritmo di codifica utilizzato (realizzato in un precedente lavoro di tesi dal collega Paolo Parlato) è caratterizzato da un'elevata scalabilità (in risoluzione spaziale, risoluzione temporale e qualità) che lo rende fruibile ad un'utenza eterogenea, i vari livelli di scalabilità sono combinati in un flusso codificato *embedded*. Sono state implementati due diversi prototipi che si distinguono essenzialmente per la diversa modalità di trasmissione utilizzata (unicast in un caso e multicast nell'altro), entrambi i prototipi sono composti da due unità software, un *server* che si occupa di codificare il flusso video e di spedirlo su rete, ed un *client* che riceve i pacchetti ne decodifica il payload e riproduce a schermo la sequenza video.

Nel prototipo *unicast* si sfruttano le qualità del codec permettendo di scegliere i 3 parametri di codifica al fine di definire come meglio si crede il trade-off qualità/tasso trasmissivo. Tale prototipo è stato sottoposto ad una serie di prove con le quali si è cercato di stabilire quale fosse il modo migliore per sfruttare le caratteristiche di scalabilità del codec.

Nel prototipo *multicast* invece utilizzando la tecnica **MMG** (Multiple Multicast Group) si ha la possibilità di realizzare una comunicazione nella quale ogni host ricevente decide a quale qualità (e quindi a quale tasso) ricevere, e questo in mo-

[1] E' vero che perdere un pacchetto non è così drammatico per una trasmissione real-time, ma per una corretta decodifica del flusso è comunque necessario sapere che tale pacchetto è stato perso.

[2] Etichette che permettono di ricostruire la giusta tempificazione del flusso ricevuto.

[3] Questa informazione permette ad esempio di trattare con priorità diverse tipi di dati diversi.

do indipendente da quanto scelto dagli altri host, e soprattutto in modo trasparente all'host che sta trasmettendo.

Per entrambi i prototipi sono state implementate tecniche per la gestione dei pacchetti persi (*error concealment*) e per la sincronizzazione tra client e server.

Lo sviluppo del software è stato svolto presso il Laboratorio per le comunicazioni multimediali del *CNIT* (Consorzio Nazionale Interuniversatario per le Telecomunicazioni) di Napoli. Il lavoro si colloca all'interno del progetto denominato *Labnet*, il cui obiettivo è lo studio e l'implementazione di diverse tecniche di apprendimento a distanza. In particolare, ad esempio, è in via di realizzazione un sistema web-based per la fruizione a distanza di esperienze di laboratorio di misure elettroniche.

Il lavoro è organizzato nel modo seguente:

Capitolo 1 Introduzione alle problematiche affrontate nell'ambito della compressione video, descrizione di alcune delle più diffuse tecniche di codifica e degli standard di codifica video H.261, MPEG-1, MPEG-2, MPEG-4. Nella parte finale viene descritto in dettaglio il codec utilizzato in questo lavoro;

Capitolo 2 Descrizione dei modelli di riferimento per le architetture di rete, con particolare enfasi sulla descrizione delle caratteristiche salienti dei protocolli che compongono la suite TCP-IP;

Capitolo 3 Descrizione dei protocolli utilizzati tipicamente nelle trasmissioni in tempo reale. Viene quindi introdotto l'ambiente Integrated Services con il protocollo RSVP, l'alternativa rappresentata da Differentiated Services, ed inoltre viene descritto approfonditamente il protocollo RTP;

Capitolo 4 Vengono introdotte le applicazioni sviluppate per questo lavoro di tesi. Vengono descritte le tecniche utilizzate per risolvere i problemi legati alla perdita dei pacchetti alla sincronizzazione, particolare risalto è dato alla tecnica MMG;

Conclusione Si riassumono gli obiettivi raggiunti e si analizzano i punti sui quali bisogna ancora lavorare cercando anche di proporre delle possibili strade alternative.

Capitolo 1

La codifica video

La codifica video permette di eliminare la ridondanza contenuta nel segnale video, riducendo, quindi, lo spazio richiesto per la memorizzazione o equivalentemente il bit-rate necessario per la trasmissione. In questo capitolo verranno introdotti i principi della codifica video, si discuterà inoltre su alcune tecniche di codifica molto diffuse. Verranno illustrati brevemente alcuni standard di codifica ed infine si introdurrà il codec utilizzato in questa tesi.

1.1 Cenni introduttivi

Un segnale video digitale consiste di una sequenza di frame video, ciascuna frame è fatta da un insieme di valori numerici che rappresentano i campioni dell'immmagine originaria.

Una sequenza video di buona qualità richiede uno spazio per la memorizzazione, o equivalentemente un bit-rate per la trasmissione, molto elevato. Si pensi ad esempio ad immagini con una risoluzione spaziale pari a 720×576 pixel e quantizzate ad 8 bit/pixel, nel caso di un frame-rate pari a 25 frame/s la trasmissione del segnale richierde un bit-rate r pari a:

$$r = 720 * 576 * 25 * 8 \simeq 80\,\mathrm{Mbit/s}$$

Un valore tanto elevato rende improponibile la trasmissione del segnale, ma anche la sua memorizzazione risulta difficoltosa, infatti un solo minuto della sequenza richiederebbe uno spazio pari a 600 Mbyte. Questi problemi rendono indispensabile il ricorso a tecniche di codifica video, queste ultime forniscono una rappresentazione efficiente del segnale permettendo un risparmio sulle risorse richieste.

1.1.1 Ridondanza statistica e psicofisica

Ogni segnale presenta un certo grado di ridondanza al suo interno, riducendo questa ridondanza si ottiene una rappresentazione più compatta.

La ridondanza è essenzialmente di due tipi:

ridondanza statistica: spesso il valore di un campione è molto simile a quello dei campioni che gli sono vicino, in altre parole ogni nuovo campione mostra una certa dipendenza da quelli vicini e quindi è in parte predicibile a partire da essi,

ridondanza psicofisica: tipicamente il segnale video contiene molte più informazioni di quelle che un osservatore può apprezzare, ad esempio il sistema visivo umano è poco sensibile alle elevate frequenze spaziali in modo particolare per quanto riguarda i colori. Quindi è possibile eliminare queste informazioni superflue mantenendo la stessa qualità soggettiva.

Nella classificazione fatta sopra non si distingue esplicitamente tra campioni appartenenti ad una stessa frame o a frame diverse, in effetti si usa distinguere fra *ridondanza intraframe*, presente tra i campioni vicini (spazialmente) di una stessa frame e *ridondanza interframe*, che riguarda i campioni vicini (temporalmente) di frame contigue.

1.1.2 La perdita di informazione

L'operazione di codifica può essere o meno invertibile, dove con invertibilità intendiamo la possibilità di ricostruire esattamente l'immagine di partenza. Quindi si può distinguere tra algoritmi di compressione **lossy** (con perdita di informazione e quindi non invertibili) e **lossless** (senza perdite, in questo caso è possibile ricostruire esattamente il segnale originale).

Un parametro che permette di confrontare i vari algoritmi di codifica è il *rapporto di compressione*, può essere definito come il rapporto fra il tasso del segnale originario e quello del segnale codificato. Quanto maggiore è questo valore tanto maggiore sarà la riduzione ad esempio del bit-rate del segnale codificato rispetto a quello del segnale originario. É chiaro però che per gli algoritmi lossy, che introducono anche una perdita di informazione, questo parametro da solo non basta a caratterizzare le prestazioni di codifica. In questi casi è necessario quantificare la perdita di qualità, questo tipicamente viene fatto attraverso delle misure oggettive

della distorsione come ad esempio l'*errore quadratico medio* (**MSE**, Mean Square Error), o il *rapporto segnale-rumore di picco* (**PSNR**, Peak Signal to Noise Ratio):

$$\mathrm{MSE} = \frac{1}{MN} \sum_{I=1}^{N} \sum_{J=1}^{M} \left[I(i,j) - \widehat{I}(i,j) \right]^2$$

$$\mathrm{PSNR} = 10 \log_{10} \frac{I_{\max}^2}{MSE}$$

dove I e $\widehat{I}$sono rispettivamente l'immagine originale e quella ricostruita, M ed N il numero di righe e di colonne che compongono l'immagine ed $I_{\max}$ è il massimo valore che può essere assunto dall'ingresso.

Per quanto sofisticate possano essere, non è detto che le misure oggettive della distorsione diano una buona indicazione della qualità dell'immagine ricostruita. Specialmente nel campo della codifica video, non è difficile trovare situazioni in cui immagini caratterizzate dalla stessa qualità oggettiva abbiano una qualità soggettiva molto diversa. In altre parole, non è detto che ad una maggiore qualità oggettiva debba necessariamente corrispondente una maggiore qualità soggettiva. Per questo motivo, spesso alle misure oggettive si affiancano quelle soggettive in modo da dare una valutazione più precisa delle prestazioni del codificatore.

1.1.3 Le risorse hardware

Abbiamo visto che nella valutazione di un algoritmo di codifica vanno presi in considerazione tanto il rapporto di compressione che le misure di qualità, tuttavia non bisogna mai dimenticarsi del contesto nel quale l'algoritmo si troverà ad operare, e quindi non si può trascurare un'altra caratteristica molto importante e cioè la richiesta di risorse hardware.

Tipicamente quanto più un algoritmo è efficiente (nel senso del trade-off compressione/qualità) tanto più è complesso, e quindi richiede maggiori risorse hardware. Tuttavia non è detto che ad un codificatore complesso debba necessariamente corrispondere un decodificatore altrettanto complesso, ed in questo senso è possibile distinguere tra:

algoritmi simmetrici, nei quali codificatore e decodificatore hanno pressapoco la stessa complessità computazionale. e quindi hanno bisogno di macchine di potenza paragonabile;

algoritmi asimmetrici, nei quali il codificatore ha una complessità computazionale maggiore rispetto a quella del decodificatore. Tipicamente in questo caso la codifica avviene su macchine di potenza nettamente superiore rispetto a quelle sulle quali avviene la decodifica.

In questa tesi è stato preso in considerazione un codec di tipo simmetrico, in quanto uno dei requisiti è la possibilità di effettuare tanto la codifica che la decodifica in tempo reale su computer di modeste prestazioni.

1.1.4 Le tecniche di compressione.

In genere gli algoritmi di codifica vengono costruiti mettendo insieme più strategie, al fine di soddisfare i requisiti del progetto cercando di riutilizzare quanto più possibile risultati che sono già consolidati, di seguito vengono illustrate alcune tra le più diffuse tecniche di compressione.

Intuitivamente possiamo immaginare una tecnica che ci permetta rappresentare un immagine usando solo un numero limitato di "tasselli" predefiniti, questo concetto è alla base della **quantizzazione vettoriale**, estensione della più nota quantizzazione scalare al caso multidimensionale. In pratica l'immagine viene suddivisa in blocchi di dimensioni $M \times N$ e ciascun blocco viene confrontato con un numero limitato di blocchi di riferimento (codebook). Tra questi si sceglie quello che meglio rappresenta il blocco originale e ne viene trasmesso l'indice che lo identifica all'interno del codebook, ovviamente si richiede che il codebook utilizzato sia disponibile anche per il decodificatore. Essendo il numero di blocchi di riferimento limitato, molto inferiore rispetto a quelle che sono tutte le possibili configurazioni di blocchi $M \times N$, trasmettere l'identificativo di un tale blocco è molto meno dispendioso che trasmettere l'intero blocco originale. La quantizzazione vettoriale è una tecnica molto generale con la quale si riesce a sfruttare bene la dipendenza statistica presente tra i pixel contigui, tuttavia la sua complessità computazionale cresce notevolmente al crescere delle dimensioni dei blocchi, questo impedisce di ottenere rapporti di compressione elevati. Per questo motivo spesso se ne usano versioni "vincolate" in cui, a patto di vincolare opportunamente la struttura del codice e la strategia di ricerca, si riesce ad ottenere una complessità computazionale ed una richiesta di memoria accettabili.

Nella **codifica con trasformata** l'ingresso è diviso in blocchi che vengono poi sottoposti ad una trasformazione lineare. Lo scopo della trasformazione è quello di rappresentare il blocco in un dominio trasformato nel quale i campioni di uscita

siano indipendenti (o quanto meno abbiano una bassa correlazione). Un'altra caratteristica richiesta è la compattazione dell'energia in pochi campioni di uscita: a valle della trasformazione la maggior parte dell'energia sarà contenuta in pochi campioni, rendendo di fatto trascurabili gli altri. Tornando nel dominio di partenza, l'errore commesso non trasmettendo le componenti a bassa energia si ridistribuisce su tutto il blocco divenendo meno visibile. É possibile dimostrare che la trasformata ottima secondo i due criteri di cui sopra è la trasformata di Karhunen-Loeve, essa tuttavia è di notevole complessità computazionale. Nella pratica vengono utilizzate trasformazioni sub-ottime che però hanno una complessità computazionale molto più bassa. La **DCT** (Discrete Cosine Transform) è una delle più utilizzate, anche perché data la particolare forma della funzione di correlazione delle immagini, le sue prestazioni si avvicinano molto a quelle della trasformata ottima ed inoltre esiste un algoritmo per il calcolo veloce simile alla FFT (Fast Fourier Transform), utilizzata per il calcolo della trasformata di Fourier. La codifica con trasformata è forse la tecnica più utilizzata nella codifica video, offre una elevata qualità abbinata ad un ottimo rapporto di compressione, tuttavia anche in questo caso la complessità computazionale è discretamente elevata.

La **codifica predittiva**, invece, sfrutta la ridondanza statistica esistente tra i campioni. Quindi invece di trasmettere i campioni viene trasmessa la differenza tra il campione attuale ed il campione predetto sulla base dei campioni precedenti. A causa della dipendenza statistica l'errore sarà molto contenuto e quindi è possibile rappresentarlo con un numero minore di bit rispetto al campione, ed in questo modo si riduce il bit-rate.

Attualmente molti sforzi di ricerca sono concentrati sulla **codifica per sintesi**, in questo caso si cerca di estrarre dall'immagine delle informazioni che permettano al decodificatore di *sintetizzare* l'immagine, ad esempio, come avviene nel caso della videotelefonia, le informazioni possono riguardare la posizione di alcuni punti caratteristici di un volto. In questo modo si riesce ad ottenere una compressione molto spinta, ma chiaramente tali tecniche richiedono un'elevata complessità di calcolo.

Per quanto riguarda la codifica interframe attualmente si possono individuare due tecniche principali, il **movement compensation** e il **conditional replenishment**. Nel primo caso si sfrutta il fatto che spesso nelle sequenze in movimento in pratica si hanno dei blocchi di immagine che compiono delle traslazioni, e per ognuno di essi invece che ritrasmettere tutto il blocco di dati, si può semplicemente trasmettere la traslazione corrispondente. Nel caso del conditional replenishment

invece si suddivide l'immagine in tanti blocchi, ogni blocco viene poi confrontato con quello che occupa la stessa posizione nella frame precedente (o comunque in un'altra frame utilizzata come riferimento) e il blocco viene trasmesso solo se risulta cambiato (tipicamente si confronta l'energia della differenza dei due blocchi con un opportuno valore di soglia). Dei due, quello che offre il maggiore rapporto di compressione è certamente il movement compensation, tuttavia richiede anche una complessità computazionale nettamente superiore.

Molto spesso, come dicevamo in precedenza, gli algoritmi di codifica vengono costruiti combinando alcune di queste (ed altre) tecniche, quindi ad esempio si può combinare la codifica con trasformata (per una singola frame) con il movement compensation (tra frame e frame). La scelta di quali utilizzare è essenzialmente dettata dal tipo di applicazione verso la quale ci si rivolge. Ad esempio se l'obiettivo è quello di contenere la complessità computazionale (ad esempio perché si vuol utilizzare il codec in tempo reale) si dovrà scartare il movement compensation e puntare sul conditional replenishment, se invece l'obiettivo primario è il rapporto di compressione (ad esempio per permettere l'archivazione di filmati) si possono utilizzare tecniche come la codifica con trasformata e il movement compensation.

1.2 H.261

Lo standard H.261 nasce nel 1990, appartiene alla famiglia di protocolli per la videoconferenza presentati dall'ITU[1]. In figura 1.1 sono mostrati gli schemi a blocchi del codificatore e del decodificatore.

Per quanto riguarda la codifica possiamo vedere un *blocco di codifica sorgente*, un *multiplexer*, che combina i vari flussi di dati (risultati della codifica, vettori di movimento, ecc.), un *buffer di trasmissione*, usato per la regolazione del bit-rate ed un *blocco di codifica di trasmissione*, che introduce ridondanza al fine di prevenire gli errori.

Il blocco di codifica sorgente è espanso in figura 1.2. L'immagine viene divisa in blocchi di 8×8 pixel e questi vengono sottoposti alla trasformata DCT, i coefficienti risultanti vengono quantizzati e poi vengono mappati su di un vettore seguendo la classica scansione a zig-zag[2] . Tale vettore è poi sottoposto ad una codifica **VLC** (Variable Length Coding) usando un ibrido della codifica di *Huffman* e della **RLE** (Run Length Encoding).

[1] H.320 per ISDN, H.323 per le LAN e H.324 per POTS.

[2] Che porta le componenti a minor *frequenza spaziale* in testa al vettore.

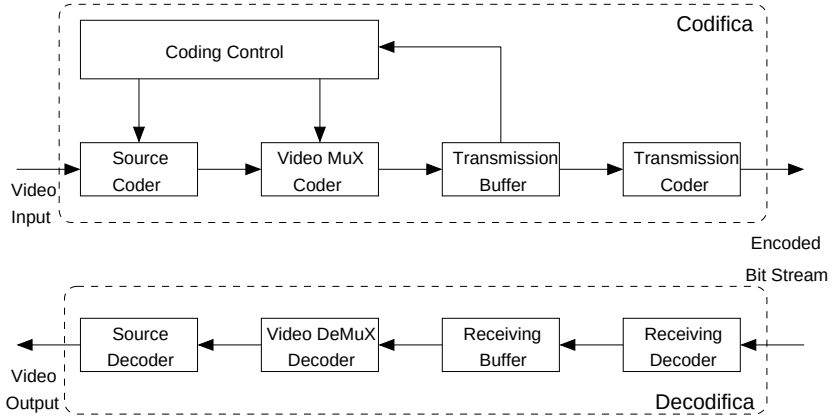


Figura 1.1: Schema a blocchi del codificatore e del decodificatore H.261.

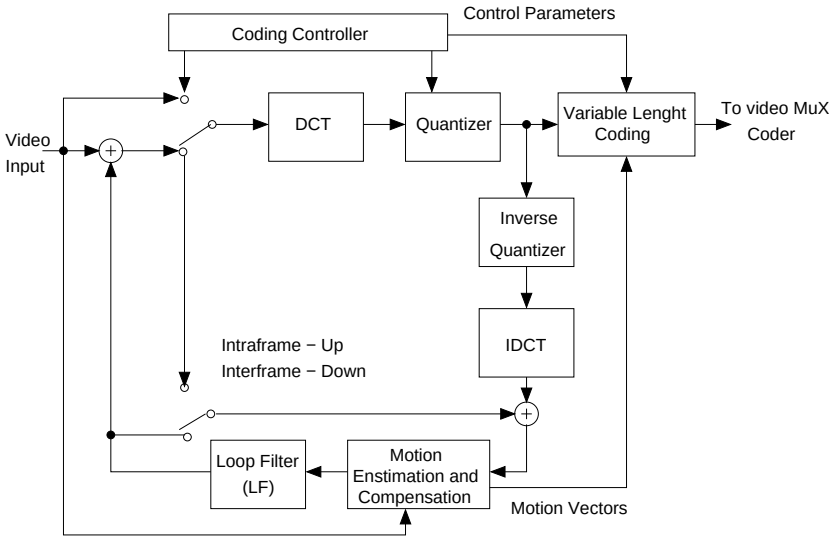


Figura 1.2: Schema a blocchi del codificatore di sorgente

Per quanto riguarda la codifica interframe, è prevista la codifica dell'errore di predizione e, in modo opzionale, si può introdurre anche la tecnica del movement compensation. E' interessante notare che H.261, dovendo essere usato per la videoconferenza, deve essere in grado di girare in tempo reale, ma in realtà le tecniche scelte non sono certe quelle più semplici dal punto di vista computazionale. Di fatto H.261 non è nato per essere eseguito su di un normale PC, ed esistono diversi set-top-box dedicati, da utilizzare esclusivamente per la videoconferenza.

1.3 MPEG

Il gruppo **MPEG** (Motion Picture Expert Group) è stato istituito nel 1988 all'interno del JTC1 (Joint ISO/IEC Technical Commitee) con lo scopo di affrontare il problema della codifica di immagini in movimento, con audio associato.Tra le caratteristiche più interessanti degli standard MPEG va sicuramente citata la possibilità di utilizzare formati d'immagine diversi, e la notevole flessibilità prevista nella composizione del bit-stream. Ad oggi sono stati prodotti tre standard MPEG-1, MPEG-2, MPEG-4.

1.3.1 MPEG-1

Nasce con lo scopo di permettere la codifica di immagini in movimento con audio associato con qualità VHS (360x288 pixel a $24 \div 30$ frame/s) mantenendo il il bit-rate entro il limite di *1.5 Mbit/s*, in modo ad esempio da permettere la memorizzazione di film su CD-ROM. Lo schema di codifica sostanzialmente è analogo a quello dello standard H.261 mostrato in figura 1.2 (il lavoro su MPEG è stato in parte basato sui risultati ottenuti con H.261)

Le frame vengono divise in 3 tipi diversi:

I-frame: Sono sottoposte a codifica intra, sono indispensabili al fine dell'implementazione di funzionalità come l'accesso casuale alla sequenza.

P-frame: Sottoposte a codifica inter, in particolare viene applicata la predizione del movimento utilizzando come frame di riferimento la frame precedente.

B-frame: Sottoposte a codifica inter, in questo caso però la predizione del movimento è di tipo bidirezionale, come frame di riferimento vengono utilizzate quella precedente e quella successiva.

Per le frame di tipo I la tecnica di codifica utilizzata è quella per trasformata, in particolare si ricorre alla DCT su blocchi di immagine 8×8, inoltre i coefficienti sono pesati in modo diverso tenendo conto delle caratteristiche della visione umana[3]. Per le frame di tipo P, invece, si utilizza il movement compensation, e i vettori di movimento di regioni adiacenti sono sottoposti a codifica differenziale visto che sono molto correlati. La tecnica utilizzata per le frame di tipo B è analoga alla precedente, con l'unica differenza che vengono usate due frame come riferimento. In una frame di tipo B, ad ogni blocco possono corrispondere due blocchi delle immagini di riferimento (uno da quella precedente ed uno da quella successiva), in questo caso in decodifica si può effettuare la media dei due blocchi.

La costruzione del bit-stream è lasciata libera, cioè è possibile scegliere in quale ordine si devono susseguire le frame di tipo I, P e B.

1.3.1.1 Differenze tra MPEG-1 e H.261

Come dicevamo prima, MPEG-1 ha in parte riutilizzato concetti già visti in H.261, anche se comunque il target dei due codec è molto diverso. Di seguito sono riportate le principali differenze tra i due codec.

1. MPEG-1 usa il movement compensation, anche bidirezionale, con un accuratezza di mezzo pixel, H.261 usa il solo movement compensation unidirezionale con accuratezza sempre pari ad un numero intero di pixel.

2. MPEG-1 supporta un range massimo per i vettori di movimento tra -512 e 512 mezzi pixel, contro i $\pm$ 15 pixel di H.261. Tuttavia questa differenza è giustificabile visto che H.261 è pensato per la videoconferenza, quindi per scene nelle quali il movimento è molto ridotto ed inoltre interessa sempre piccole zone.

3. MPEG-1 permette di definire matrici di pesatura per i coefficienti della DCT.

1.3.2 MPEG-2

MPEG-1, come abbiamo visto, è ottimizzato per applicazioni a 1.5 Mbit/s, con MPEG-2 invece viene offerta al possibilità di avere sequenze video a qualità ancora migliore e quindi con bit-rate necessariamente più alto, inoltre viene esteso il supporto per i formati di ingresso interlacciato.

[3] L'occhio umano è molto più sensibile alle basse frequenze spaziali che non alle alte, quindi è consigliabile rappresentare con maggiore accuratezza i primi coefficienti della DCT.

Un'altra grossa novità introdotta in MPEG-2 è la possibilità di ricorrere ad una codifica di tipo scalabile, anche chiamata codifica a livelli, perché è sempre possibile individuare un livello base (indispensabile) e più livelli di miglioramento (facoltativi). La codifica scalabile è importante, ad esempio, per la trasmissione su canali rumorosi, infatti è possibile proteggere i livelli più importanti (i livelli base) trasmettendoli su canali con prestazioni migliori. La codifica scalabile è molto utile anche su canali con bit-rate variabile, quando la banda è ridotta i layer meno importanti non vengono trasmessi. Un'altra utile applicazione della scalabilità consiste nella trasmissione progressiva, cioè l'utente può visionare un'anteprima del video, codificata con i soli livelli base al fine ad esempio di effettuare una scelta in un database, una volta che la decisione è stata presa potrà ricevere il video alla qualità migliore.

MPEG-2 supporta tre tipi diversi di scalabilità:

Scalabilità in SNR: Ci sono due possibili livelli di qualità; a livello base i coefficienti della DCT sono quantizzati in modo grossolano, mentre nel livello di enhancement viene codificata la differenza tra i coefficienti DCT originari e quelli quantizzati al livello base, chiaramente usando in questo caso passo di quantizzazione più fine.

Scalabilità in risoluzione: I due livelli sono ottenuti sfruttando uno schema a piramide laplaciana. L'immagine di ingresso viene sottocampionata e codificata, in questo modo viene prodotto il livello base, questo viene decodificato, sovracampionato e sottratto dall'immagine originale, l'immagine differenza viene quindi codificata in modo analogo a quanto fatto per il livello base.

Scalabilità in frame-rate: Il livello base è codificato al frame-rate minimo, e le sue frame decodificate sono usate come base per la predizione del movimento nelle frame che poi vanno a costruire i livelli di enhancement.

1.3.3 MPEG-4

Lo scopo di MPEG-4 è quello di fornire una serie di strumenti ed algoritmi per la codifica a basso tasso di sequenze audio/video e, a differenza degli altri standard della famiglia MPEG, è stato esplicitamente pensato per il funzionamento su rete. Tra i principali obiettivi possiamo individuare i seguenti:

- Elevata robustezza anche nell'ambito di reti molto inaffidabili.

- Estremo grado di interattività, per poter accedere in modo rapido al "contenuto" dei dati audio-visuali, in modo da poter visualizzare e presentare i dati con elevata flessibilità
- Codifica mista "pixel-based" e "sintetica".
- Possibilità di trasmettere i dati con un bit-rate tra i 4 e i 65 kbit/s per reti a bassa capacità, o fino a 4 Mbit/s per reti a banda larga.

L'idea innovativa alla base di MPEG-4 è quindi quello della codifica del contenuto piuttosto che dell'immagine. Una singola immagine viene quindi divisa in più "oggetti audio-visuali" (AVO, Audio Visual Object), come ad esempio lo sfondo, un particolare oggetto, un testo e così via.

Ciascun AVO può essere codificato/decodificato in modo indipendente dal resto, rendendo quindi possibile, ad esempio, la decodifica dei soli oggetti di interesse, questo chiaramente permette un elevato grado di interattività.

1.4 Il codec usato in questo lavoro

L'obiettivo perseguito nel progetto di questo codec [2] è stato quello di realizzare un algoritmo di codifica e decodifica video accessibile alla platea più ampia ed eterogenea possibile di utenti. Questo significa che l'algoritmo deve risentire il meno possibile delle eventuali limitazioni che caratterizzano il singolo utente, sia in termini di capacità del canale che di potenza di calcolo.

Deve essere permesso l'accesso tanto sui canali a banda larga quali ad esempio le reti ATM o le reti locali, quanto sulle reti a banda più limitata come ad esempio le reti geografiche. Questo è reso possibile dall'implementazione di tecniche di codifica scalabile; realizzando un bit-stream embedded, infatti ogni utente potrà scegliere i livelli da ricevere in accordo con la capacità della propria rete di accesso.

Si desidera inoltre rendere possibile in tempo reale tanto la decodifica che la codifica anche su terminali con limitata potenza di calcolo. L'algoritmo quindi è simmetrico, ed una particolare attenzione è stata posta nello scegliere tecniche di codifica che avessero un bassa complessità computazionale.

1.4.1 La scalabilità in risoluzione spaziale

Per realizzare una codifica scalabile in risoluzione spaziale è stata scelta la tecnica della *codifica piramidale laplaciana.*

Tramite un serie di filtraggi e decimazioni ci si riconduce ad una sequenza di immagini delle dimensioni di 180×144 pixel, questa viene codificata e trasmessa, e rappresenta quello che chiameremo **livello base**. Tale livello viene quindi decodificato (in modo da ottenere la stessa sequenza che si ha in ricezione) e sovracampionato ottenendo in questo modo una sequenza di immagini delle dimensioni di 360×288 pixel. Queste immagini vengono confrontate con quelle della sequenza originale e viene codificata e trasmessa la differenza tra le due sequenze, ottenendo in questo modo il **livello enhanced**. In questo modo, in ricezione sarà possibile scegliere ad esempio di ricevere il solo livello base, oppure si potrà ricevere anche il livello enhanced, nel qual caso la sequenza di livello base andrà decodificata sovracampionata e sommata alla sequenza di livello enhanced decodificata. In aggiunta è possibile ottenere in ricezione immagini delle dimensioni di 720×576 pixel, semplicemente sovracampionando e interpolando le immagini di livello enhanced. In figura 1.3 sono mostrati gli schemi di principio per la codifica e la decodifica.

Riassumendo, in ricezione sono disponibili le tre seguenti risoluzioni spaziali:

180 x 144 Si ottiene ricevendo il solo livello base.

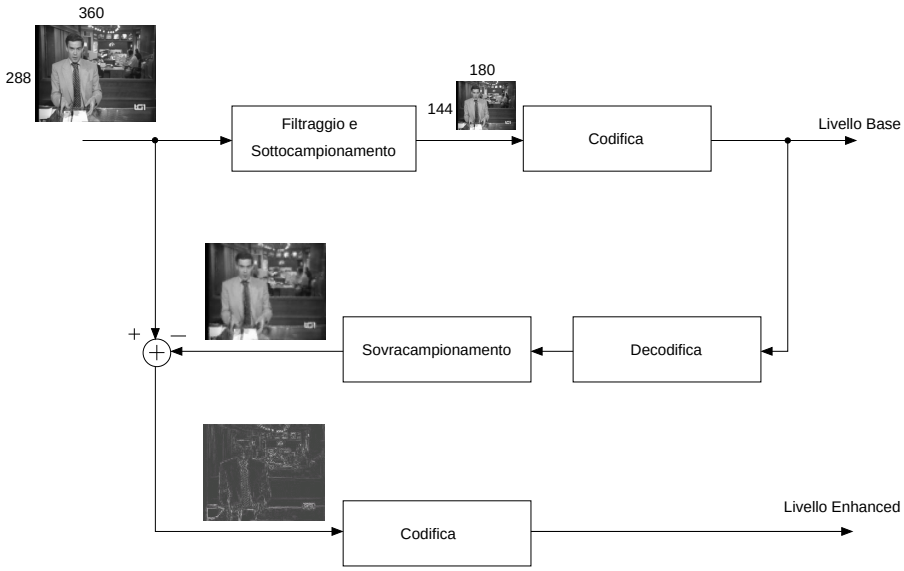
360 x 288 Si ottiene ricevendo i livelli base ed enhanced, le immagini di livello base decodificate e sovracampionate vanno sommate alla sequenza di livello enhanced decodificata.

720 x 576 Si ottiene semplicemente sovracampionando la sequenza decodificata di livello enhanced. In questo caso quindi non è richiesto nessun ulteriore impiego di banda rispetto al livello enhanced.

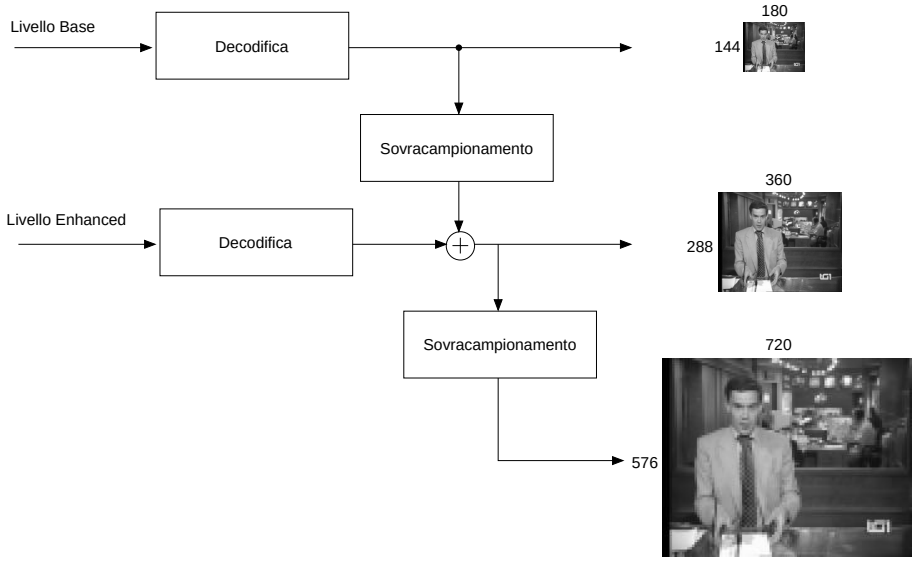
1.4.2 La scalabilità in risoluzione temporale

É chiaro che quante più frame al secondo trasmettiamo tanto più fluida sarà la sequenza, ma allo stesso tempo tanto maggiore sarà il tasso trasmissivo. Inoltre al crescere del frame rate devono crescere di pari passo le prestazioni del codificatore e del decodificatore, infatti se ad esempio il frame rate è di 10 frame/s avremo a disposizione per la codifica/decodifica della singola frame $1/10 = 0,1$ secondi mentre se il frame rate è di 25 frame/s avremo a disposizione $1/25 = 0,04$ secondi.

Per garantire la scalabilità temporale si ricorre ad una suddivisione delle frame in 3 livelli temporali. Nel primo livello viene trasmessa una frame ogni 4 quattro, quindi possiamo dire che vengono trasmesse tutte le frame con indice $4 * n$ con n



(a) Codifica.



(b) Decodifica

Figura 1.3: Scalabilità in risoluzione spaziale

intero. Nel secondo livello temporale invece viene ancora trasmessa una frame ogni quattro, ma in questo caso si trasmettono le frame con indice $4 * n + 2$, infine nel terzo livello temporale vengono trasmesse le frame con indice $4 * n + 1$ e $4 * n + 3$, quindi una ogni due. In figura 1.4 è mostrata la suddivisione in flussi temporali.

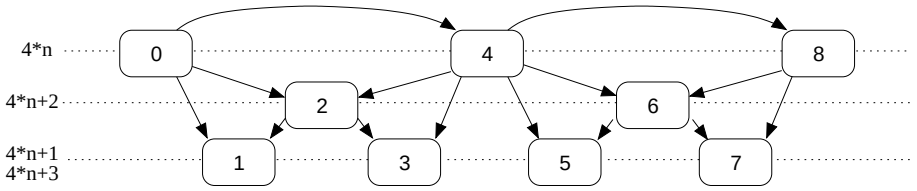


Figura 1.4: Scalabilità in risoluzione temporale.

Combinando questi livelli temporali è possibile ottenere 3 diversi frame rate.

- Se preleviamo il solo primo livello temporale, avremo un frame rate pari ad un quarto di quello della sequenza originale.
- Se preleviamo i primi due livelli temporali, di fatto vengono ricevute tutte le frame con indice pari, quindi il frame rate è pari alla metà di quello originale.
- Se preleviamo tutti e tre i livelli temporali, il frame rate è uguale a quello della sequenza originale.

Per la codifica interframe si è optato per il *conditional replenishment* (CR); ne viene usata la versione unidirezionale nel caso delle frame del flusso $4 * n$, mentre per le frame degli altri due flussi viene usato il CR bidirezionale. Le frecce in figura 1.4 indicano appunto le dipendenze per il CR. Ciò evidenzia anche che l'unico flusso indipendente (che cioè può essere decodificato senza ricevere i restanti flussi) è il primo, mentre il secondo flusso richiede il primo, e il terzo li richiede entrambi.

A causa di queste dipendenze, l'ordine con il quale vengono codificate le frame è diverso da quello con il quale vengono acquisite, infatti all'uscita del codificatore avremo una sequenza di questo tipo: 0 4 2 1 3 8 6 5 7 12... . Questo introduce dei piccoli ritardi sia in fase di codifica che di decodifica, in quanto è necessario riordinare le frame.

1.4.3 La scalabilità in precisione

Con la scalabilità in precisione (anche detta scalabilità in SNR) è possibile agire sul trade-off esistente tra qualità della ricostruzione e tasso trasmissivo. Per poterne

illustrare il funzionamento è necessario scendere più in dettaglio nella descrizione delle tecniche di codifica.

Come algoritmo di compressione si è optato per la *quantizzazione vettoriale tabellare con struttura gerarchica.* La quantizzazione vettoriale tabellare consente la codifica semplicemente accedendo ad una tabella, in questa tabella si entra con il blocco di pixel da quantizzare e si ricava l'indice della codeword che quantizza quel blocco. La creazione della tabella viene fatta off-line una volta per tutte, e permette ad esempio di utilizzare metriche anche sofisticate (tenendo conto ad esempio della percezione visiva) nella valutazione della distorsione. Tuttavia non è pensabile utilizzare un'unica tabella, perché questa dovrebbe avere dimensioni estremamente grandi, per questo motivo si organizza una gerarchia di tabelle più piccole.

Per ottenere la scalabilità in precisione si ricorre alla tecnica chiamata **TSVQ** (Tree Structured Vector Quantization), in pratica le parole codice sono organizzate in una struttura ad albero, è possibile quindi utilizzare un prefisso qualunque degli indici completi, riducendo in questo modo il tasso (ma anche la qualità ovviamente).

1.4.4 Il codec in dettaglio

Nei paragrafi precedenti sono state introdotte per sommi capi le tecniche utilizzate in questo codec, in questo paragrafo verrà invece mostrato più in dettaglio il suo funzionamento, analizzando gli schemi di codifica e decodifica intra/inter-frame.

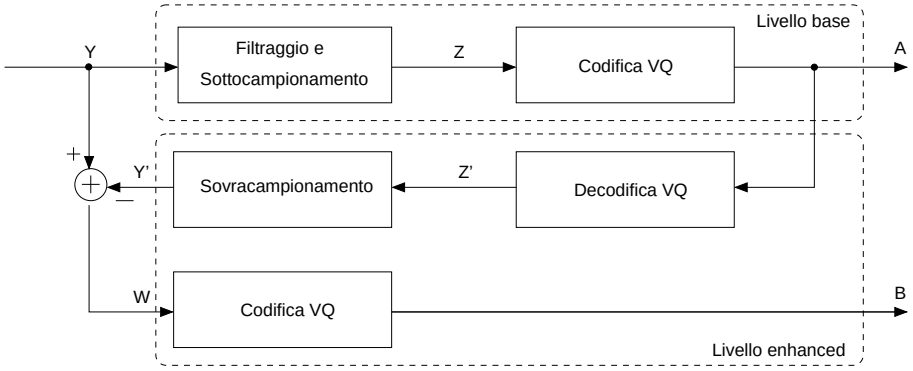
In tabella 1.1 viene illustrato il significato dei simboli usati negli schemi.

Invece in figura 1.5 vengono mostrati gli schemi del codificatore e del decodificatore intraframe.

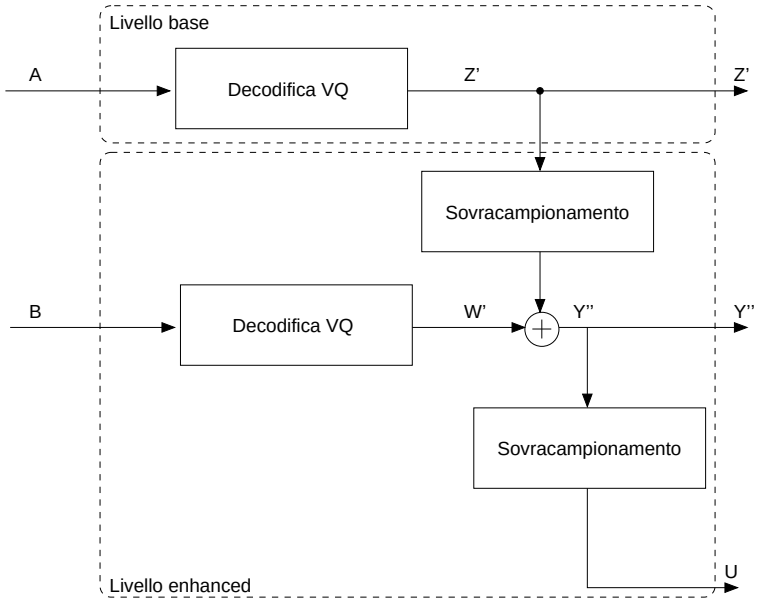
Come vediamo otteniamo due flussi codificati, quello dei vettori A corrisponde agli indici delle codeword che codificano le immagini di livello di base, mentre i vettori di tipo B contengono gli indici delle codeword che codificano il livello enhanced (differenza tra l'immagine di livello base sovracampionata e quella originale a 360×288 pixel).

La quantizzazione vettoriale opera su blocchi di 2×4 pixel (ogni pixel è rappresentato da un byte) e produce indici da 1 byte (codebook da 256 codeword), per cui per la codifica intra il rapporto di compressione è pari ad 8.

Nel caso della codifica interframe si usa la tecnica del conditional replenishment (CR), con una particolarità, la tecnica viene applicata agli indici (vettori tipo A) e non alle frame vere e proprie. Questo permette di ridurre notevolmente i calcoli in quanto tali vettori hanno dimensioni molto minori rispetto alle frame (bisogna divi-



(a) Codificatore



(b) Deocidificatore

Figura 1.5: Schemi per la codifica/decodifica intraframe.

Simbolo	Significato
Y	Immagine originaria a 360×288 pixel
Z	Immane filtrata e decimata, 180×144 pixel
A	Indici delle parole codice ottenute dalla VQ di Z
Z'	Versione ricostruita di Z a valle della VQ
Y'	Versione sovracampionata di Z
W	Errore di predizione da trasmettere per il livello enhanced
W'	Versione ricostruita di W a valle della VQ
Y"	Versione di Y ricostruita in ricezione
U	Versione sovracampionata di Y"
B	Indici delle parole codice ottenute dalla VQ di W
EA	Errore di predizione su A
TA	Soglia di decisione del CR su A
A"	Frame base di riferimento
B"	Frame enhanced di riferimento

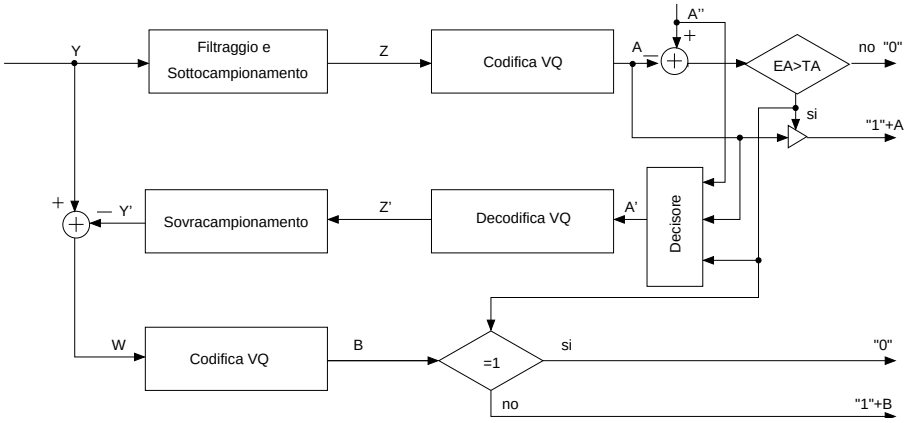
Tabella 1.1: I simboli usati negli schemi di codifica e decodifica.

dere le dimensioni delle frame per il rapporto di compressione), inoltre per ridurre l'overhead dovuto ai bit che segnalano l'esito del CR, questo è applicato a blocchi di dimensioni 3×3 indici, cioè in definitiva, su macroblocchi di 6×12 pixel. Tuttavia è bene precisare che è possibile effettuare il CR sugli indici invece che sui pixel perché i codici sono stati progettati in modo che ad indici vicini corrispondano blocchi molto simili (anche se questo è vero solo localmente), quindi confrontare pixel non è molto diverso che confrontare indici.

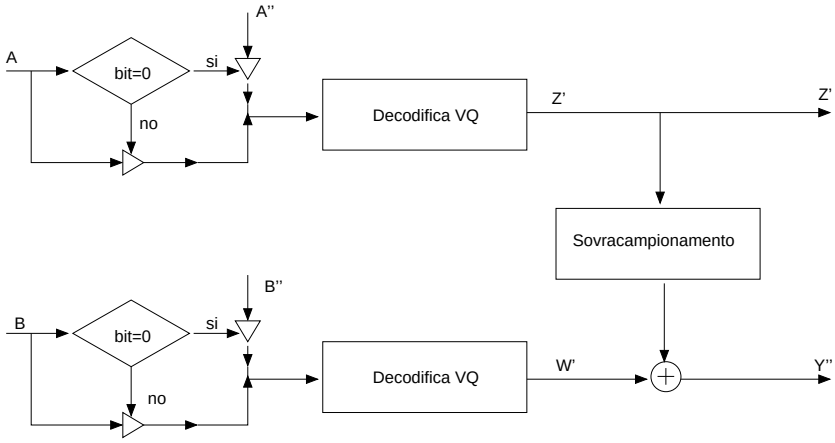
In figura 1.6 sono mostrati il codificatore ed il decodificatore interframe nel caso di conditional replenishment unidirezionale (il caso bidirezionale è del tutto analogo).

Come vediamo dallo schema del codificatore, i blocchi di indici vengono confrontati con quelli di riferimento, se l'energia della differenza supera una certa soglia il blocco viene trasmesso (insieme ad un bit di segnalazione pari ad 1), altrimenti viene trasmesso unicamente un bit di segnalazione pari a 0.

Per quanto riguarda il livello enhanced, solo i blocchi corrispondenti ai blocchi di indici per i quali il CR non ha avuto successo vengono trasmessi, in pratica si ipotizza di poter estendere anche a livello enhanced i risultati del CR effettuato a livello base. Questa scelta permette di avere una complessità computazionale molto ridotta, tuttavia probabilmente avere CR separati sui due livelli produrrebbe risultati di qualità migliore.



(a) Codificatore



(b) Decodificatore

Figura 1.6: Schemi per la codifica/decodifica interframe.

Capitolo 2

L'architettura di rete

Lo studio e la progettazione delle architetture di rete sono semplificati dall'approccio a livelli. In questo capitolo verrà introdotto il modello ISO-OSI, si passerà poi alla descrizione più dettagliata dei protocolli che compongono lo stack TCP/IP.

2.1 Il modello ISO-OSI

Un approccio molto utilizzato nel disegno e nello studio delle architetture di rete è quello della divisione in livelli. Ogni livello assolve solo ad alcuni compiti e fornisce servizi al livello soprastante il quale li utilizza per assolvere i propri compiti. In questo modo è possibile dividere i compiti tra i vari livelli semplificando il progetto. La figura 2.1 mostra un esempio di modello a livelli.

Viene mostrata la comunicazione tra due host, anche se in realtà la comunicazione avviene al livello più basso (collegamento fisico) il livello i-esimo di un host comunica con il livello i-esimo dell'altro host tramite il protocollo di livello i-esimo. Sono inoltre evidenziate le interfacce tra i vari livelli: è tramite queste ultime che ogni livello accede ai servizi offerti dal livello sottostante.

La **ISO** (International Standard Organization) ha prodotto nel 1983 uno standard per l'interconnessione di sistemi denominato **OSI** (Open System Interconnection). OSI è un modello a 7 livelli, non ha avuto molto successo da un punto di vista delle implementazioni ma è tuttora molto utilizzato per scopi didattici, la figura 2.2 mostra la sua architettura.

Si vedono 2 host che comunicano tra di loro a partire dal livello trasporto, anche se in realtà il collegamento fisico passa attraverso più host.

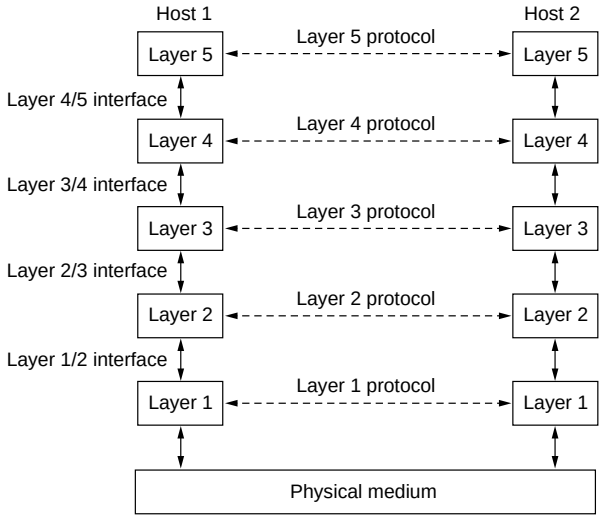


Figura 2.1: Il modello stratificato.

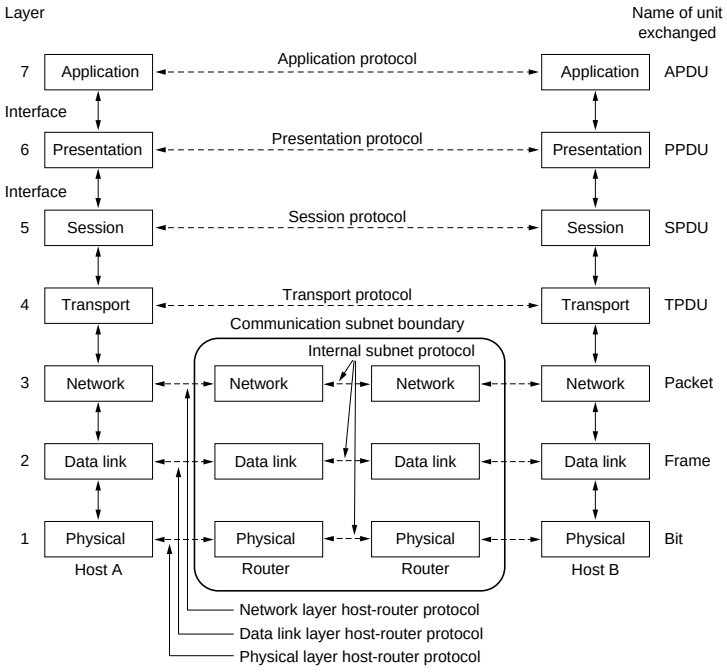


Figura 2.2: Il modello ISO-OSI.

Il livello **fisico** si occupa della trasmissione vera e propria dei dati, al livello superiore fornisce un astrazione di un canale virtuale sul quale viaggiano i bit.

Il livello **data link** offre un canale virtuale per pacchetti, i compiti svolti vanno dal controllo di errore e di flusso al framing.

Il livello **rete** offre un canale virtuale per pacchetti end-to-end, quindi risolve problemi come il routing, ed eventualmente si può occupare del controllo di congestione. Nei livelli sottostanti si ha la visibilità dei soli host collegati fisicamente, dal livello rete in poi si ha la visibilità di tutti gli host collegati alla rete.

Il livello di **trasporto** realizza un canale virtuale per messaggi end-to-end, si occupa quindi della divisione dei messaggi in pacchetti, inoltre se richiesto è al livello trasporto che viene garantita l'affidabilità del collegamento.

Il livello **sessione** implementa una sessione virtuale all'interno della quale trasmettere messaggi anche di tipo diverso in modo del tutto trasparente ai livelli superiori, a questo livello, se necessario, vengono risolti problemi come la sincronizzazione o la gestione dei token.

Il livello **presentazione** offre funzioni come la conversione tra i formati oppure come la compressione o la cifratura dei dati, a questo livello diviene determinante la semantica della informazioni trasmesse.

Il livello **applicazione** contiene una serie di applicazioni che sfruttano i meccanismi sottostanti, quindi si possono trovare applicazioni per il trasferimento di dati, di video, oppure applicazioni di terminale remoto, ecc.

2.2 Il modello TCP/IP

Nell'ambito delle architetture di rete si può tranquillamente dire che lo standard di fatto è la suite di protocolli TCP/IP[1]. Anche TCP/IP può essere studiato con un approccio a strati, e la figura 2.3 mostra un parallelo con il modello ISO-OSI.

Si nota che i due livelli più bassi della pila OSI corrispondono a quello che in TCP viene chiamato livello host-rete, in effetti questo strato sottostante è non meglio specificato, la suite TCP/IP copre la pila OSI a partire dal livello di rete. Si nota anche che in TCP/IP non sono previsti i livelli di presentazione e sessione, se si ritiene che alcune delle loro caratteristiche siano utili devono essere implementate dall'applicazione.

[1] Sarebbe più corretto dire TCP-UDP/IP.

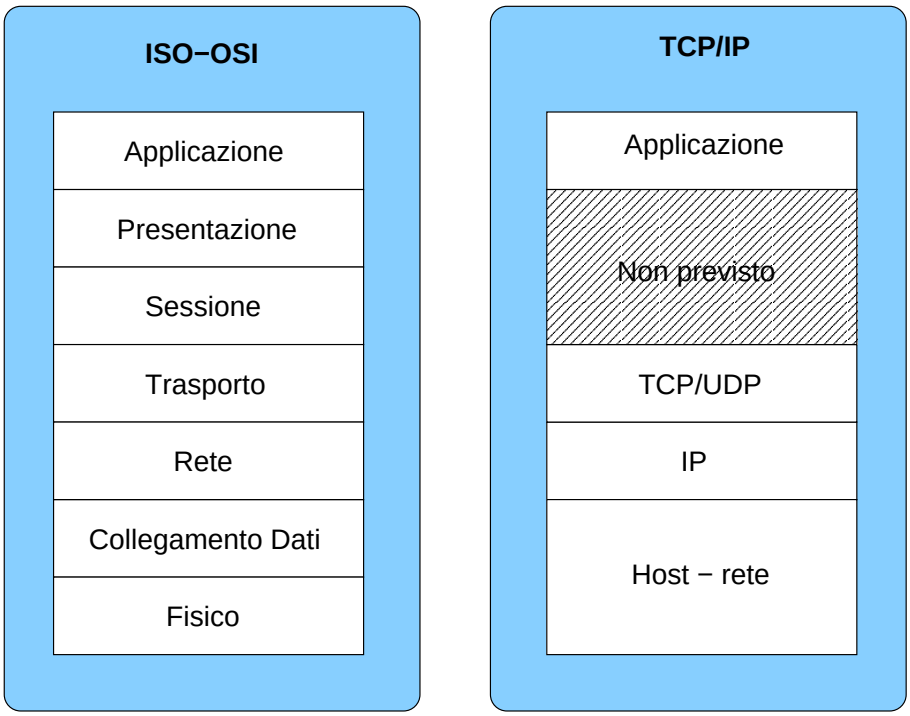


Figura 2.3: Lo stack ISO-OSI a confronto con quello TCP/IP.

Di seguito verranno presentate le caratteristiche salienti dei tre protocolli IP, TCP, UDP.

2.2.1 Internet Protocol (IP)

Secondo il modello ISO-OSI, IP [6] è un protocollo di livello 3, cioè di livello rete (vedi figura 2.3), è a livello IP che gli host hanno visibilità della rete e non più dei soli host direttamente collegati ad essi.

Per capire le funzionalità offerte da IP è sicuramente utile analizzarne la struttura dell'header riportato in figura 2.4.

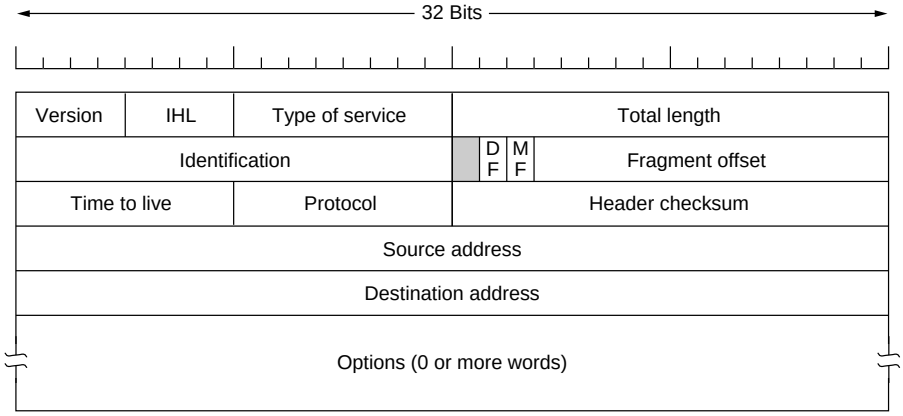


Figura 2.4: L'header di IP.

Ora analizzeremo i vari campi:

Version: 4bit

Descrive la versione del protocollo utilizzato.

IHL (IP Header Length): 4bit

Contiene la lunghezza dell'header (in parole di 32 bit), questa è un 'informazione necessaria perché con le opzioni l'header può essere anche più lungo dei canonici 20 byte.

Type of service: 8bit

Permette di indicare il tipo di servizio richiesto.

Total length: 16bit

Contiene la lunghezza totale del pacchetto (quindi header+payload)

Identification: 16bit

In caso di frammentazione tutti i frammenti conterranno lo stesso identificativo, quindi questo campo è necessario per permettere all'host di ricongiungere tutti i frammenti.

DF (Don't Fragment): 1bit

Flag che se settata ad 1 ordina ai router di non frammentare il pacchetto.

MF (More Fragment): 1bit

Flag che se settata ad 1 indica che questo datagramma è un frammento di un pacchetto più grande, e non è l'ultimo (che infatti avrà il flag posto a zero).

Fragment Offset: 13bit

Indica in che posizione del datagram va inserito questo frammento.

Time to live: 8bit

É il contatore utilizzato per determinare il tempo di vita dei pacchetti. Tipicamente viene decrementato ad ogni salto.

Protocol: 8bit

Indica quale è il protocollo di livello superiore utilizzato (e quindi permette di interpretare correttamente l'header che segue).

Header checksum: 16bit

Controllo di errore (solo sull'header).

Source address: 32bit

Indirizzo IP della sorgente.

Destination address: 32bit

Indirizzo IP della destinazione.

Options: 0 o più parole da 32bit

Estensioni dell'header che permettono di includere nuove funzionalità, oppure di provare nuove soluzioni.

É subito evidente che non è presente un campo con la numerazione dei pacchetti, infatti IP non ha bisogno di questo tipo di informazione in quanto non gestisce in alcun modo la perdita di pacchetti.

Inoltre è interessante notare la presenza del campo *Type of service*: secondo le idee dei progettisti, tale informazione avrebbe dovuto permettere di introdurre in

qualche modo il concetto di qualità del servizio sulla rete Internet. Infatti tale campo è composto a sua volta da più sottocampi che sono:

Precedence Indica di fatto una priorità, utilizza 3 bit e quindi si possono specificare 8 livelli diversi di priorità

D,T,R Questi 3 flag permettono all'host di specificare che cosa ritiene più importante tra ritardo, capacità di trasmissione e affidabilità.

Questi campi in teoria dovrebbero permettere ai router di effettuare delle scelte, ad esempio decidere quale è il migliore link in uscita per un determinato pacchetto, oppure quali pacchetti possono essere momentaneamente accodati e quali invece devono essere spediti subito. In realtà tali informazioni sono di fatto ignorate dai router probabilmente anche per motivi di semplicità.

Dall'analisi dei campi dell'header viene messo in evidenza inoltre che un pacchetto IP può subire una frammentazione, cioè può essere suddiviso in più pacchetti IP, questo tipicamente avviene perché le dimensioni del pacchetto eccedono la **MTU** (Maximum Transmission Unit), cioè la massima unità trasferibile. Questa è una cosa da tenere in considerazione, in quanto la perdita di anche un solo frammento comporta la perdita di tutto il pacchetto iniziale.

2.2.1.1 Multicast

IP sostanzialmente prevede tre distinte modalità di trasmissione che si distinguono per il tipo di indirizzo destinazione utilizzato, esse sono l'*unicast* il *broadcast* ed il *multicast*.

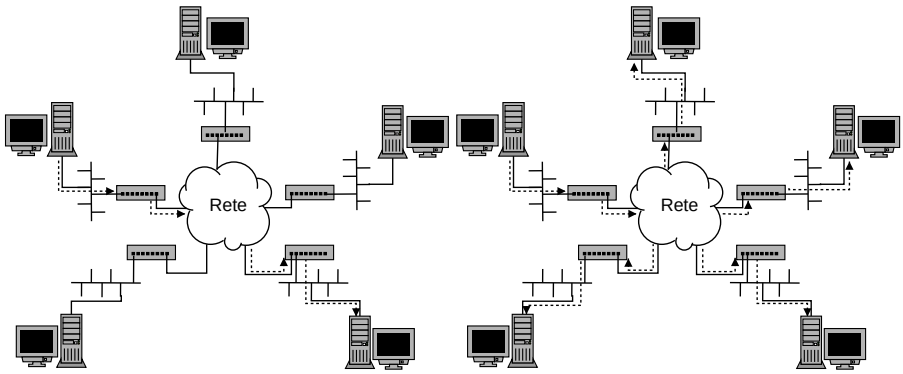
Il primo caso è quello più semplice, la trasmissione è del tipo uno a uno, l'indirizzo destinazione identifica in modo univoco un host, e i pacchetti vengono inoltrati verso quest'ultimo.

Nel caso del broadcast la trasmissione è del tipo uno a tutti (su una specifica rete), quindi tutti gli host riceveranno i pacchetti.

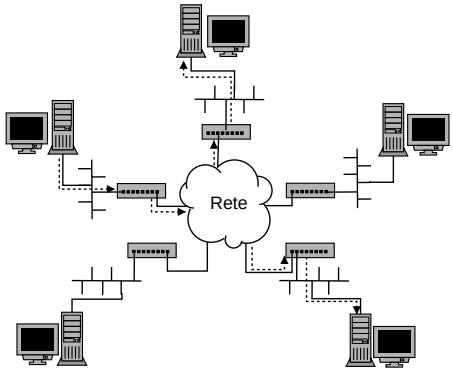
Infine c'è il multicast, in questo caso la trasmissione è del tipo uno a molti.

La figura 2.5 mostra quanto avviene, le linee punteggiate rappresentano il percorso dei dati.

Nel caso del multicast l'indirizzo IP utilizzato (appartenente agli indirizzi compresi tra `224.0.0.0` e `239.255.255.255`) identifica un cosiddetto gruppo multicast, tramite il protocollo **IGMP** (Internet Group Management Protocol) gli host possono comunicare al gateway della loro sottorete quali gruppi vogliono sottoscrivere



(a) Unicast (b) broadcast



(c) Multicast

Figura 2.5: Le tre modalità di trasmissione di IP

e quest'ultimo si occuperà di ricevere i pacchetti diretti a questi gruppi ed inoltrarli verso l'host. Il modo in cui il gateway riceve tali pacchetti dipende dal protocollo di routing multicast utilizzato.

Attualmente il multicast non è supportato su tutta la rete Internet in quanto solo pochi router implementano i protocolli di routing multicast, tuttavia è possibile sperimentarne l'utilizzo aderendo al circuito *Mbone.* In pratica diverse sottoreti multicast sono state collegate tramite dei *tunnel*, cioè dei semplici collegamenti unicast sui quali i pacchetti multicast viaggiano incapsulati in normali pacchetti IP, l'instradamento è gestito dai protocolli di routing multicast all'interno delle sottoreti mentre sui tunnel vengono usati i normali protocolli di instradamento unicast (figura 2.6).

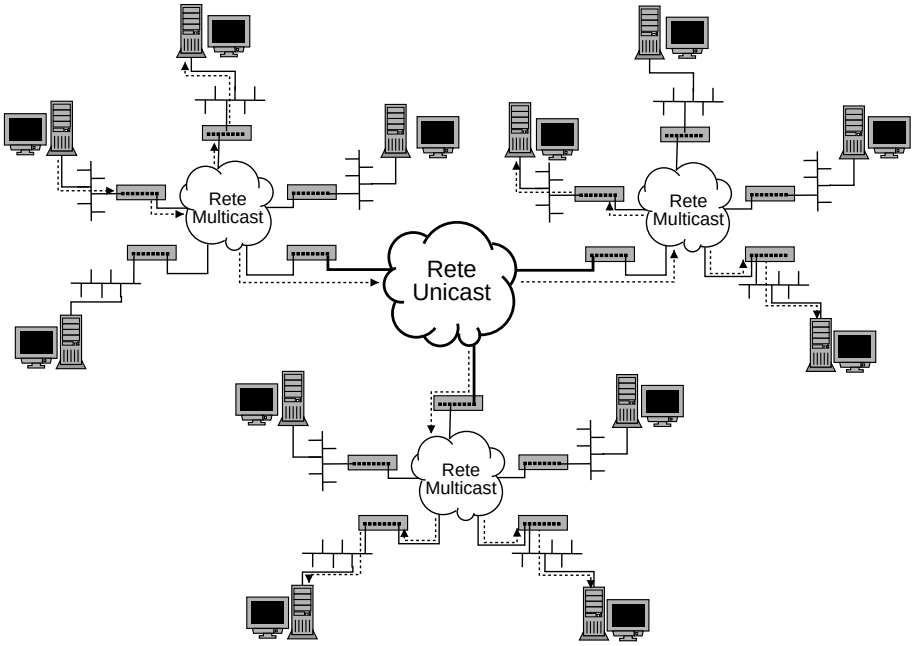


Figura 2.6: Mbone

2.2.1.2 IPv6

Quello descritto finora è il protocollo IP nella sua versione numero 4. Ormai questo protocollo ha i giorni contati, diversi problemi hanno reso indispensabile una nuova versione denominata *IPv6* (versione 6). Il problema principale è sicuramente legato agli indirizzi disponibili, ogni indirizzo IP è composto da 32 bit, il che significa che si hanno a disposizione 2^{32} possibili combinazioni, cioè poco più di 4 miliardi.

Tuttavia la forte crescita della richiesta degli indirizzi, unita allo spreco che si effettua assegnandoli per classi[2], ci sta portando velocemente verso l'esaurimento degli indirizzi disponibili. Nel 1990, intuendo la nascita di questi problemi, l' IETF iniziò a lavorare sulla nuova versione di IP, i principali requisiti su cui si è puntato sono riportati di seguito.

1. Supportare molti miliardi di host, anche nel caso di allocazione inefficiente dello spazio.
2. Ridurre la dimensione delle tabelle di routing.
3. Semplificare il protocollo in modo da permettere ai router di smistare più velocemente i pacchetti.
4. Fornire una maggiore sicurezza (mediante autenticazione e privacy).
5. Prestare più attenzione al tipo di servizio, in particolare per i dati real time.
6. Rendere facile l'evoluzione del protocollo.
7. Permettere la coesistenza con l'attuale versione di IP (IPv4).

Dopo aver analizzato diverse proposte fu scelta la soluzione indicata fino a quel momento come SIPP (Simple Internet Protocol Plus) e le fu dato il nome di IPv6. La nuova versione ha indirizzi molto più lunghi, 128 bit, ciò dovrebbe risolvere definitivamente il problema degli indirizzi, inoltre ha un preambolo più semplice (7 campi contro i 13 di IPv4), questa semplificazione è stata possibile anche grazie ad una migliore gestione delle opzioni, campi che prima erano obbligatori sono divenuti opzionali. IPv6, inoltre supporta direttamente la sicurezza, e maggiore attenzione è stata prestata alla diversificazione dei tipi di servizio.

Queste caratteristiche possono essere apprezzate meglio analizzando l'header (vedi figura 2.7).

Version: 4bit

Versione del protocollo

Priority: 4bit

Può essere utilizzato per distinguere, tramite priorità, i vari flussi di pacchetti, i valori da 0 a 7 sono riservati a trasmissioni che possono essere rallentate, i valori da 8 a 15 sono riservati al traffico in tempo reale.

[2]Le grosse aziende comprano intere classi di indirizzi, raramente però li utilizzano tutti.

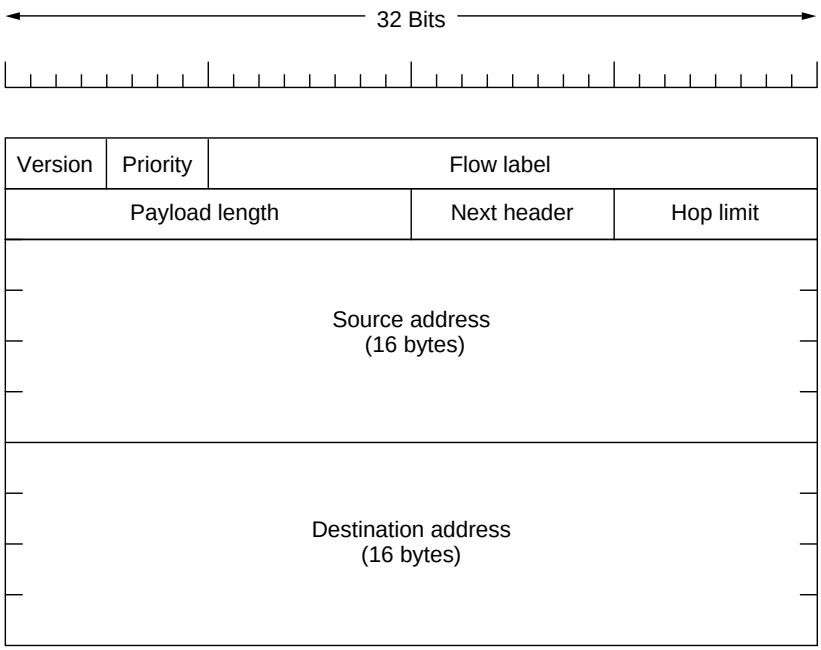


Figura 2.7: L'header di IPv6

Flow Label: 24bit

L'etichetta di flusso verrà utilizzata per permettere di creare una pseudoconnessione con particolari esigenze, come ad esempio il mantenere contenuti i ritardi o l'avere a disposizione una certa banda. I router guarderanno nelle loro tabelle interne per stabilire quale trattamento applicare. Le etichette di flusso sono un tentativo per avere contemporaneamente la flessibilità delle reti a datagrammi e le garanzie delle reti connection oriented.

Payload Length: 16bit

Indica quanti byte seguono l'header, quindi misura il solo payload.

Next Header: 8bit

Indica quale dei sei header di estensione segue quello standard, oppure nel caso che non ci siano altri header di estensione indica a quale gestore di protocollo di trasporto debba essere passato il pacchetto.

Hop Limit: 8bit

É analogo al campo TTL di IPv4, indica quanti salti può effettuare il pacchetto prima che venga considerato obsoleto.

Source Address: 128bit

Indirizzo della sorgente.

Destination Address: 128bit

Indirizzo della destinazione.

Si può notare che mancano del tutto i campi legati alla frammentazione, infatti in IPv6 si è cercato di evitare che i pacchetti possano essere frammentati. Quindi innanzitutto tutti gli host e i router conformi con IPv6 devono supportare pacchetti di almeno 576byte, questo già rende la frammentazione meno probabile, inoltre quando un router riceve un pacchetto troppo grande, invece di frammentarlo al volo risponde con un messaggio di errore, tutto questo nell'ottica che il fatto che l'host spedisca pacchetti della giusta dimensione è in ultima analisi più efficiente della frammentazione al volo. Rispetto ad IPv4 manca anche il campo checksum, in effetti le reti attuali sono molto più affidabili, ed inoltre il calcolo della checksum rallentava troppo le operazioni di routing.

2.2.2 User Datagram Protocol (UDP)

UDP [5] è un protocollo non orientato alla connessione che, come mostrato in figura 2.3 si pone a livello 4 (trasporto) della pila OSI.

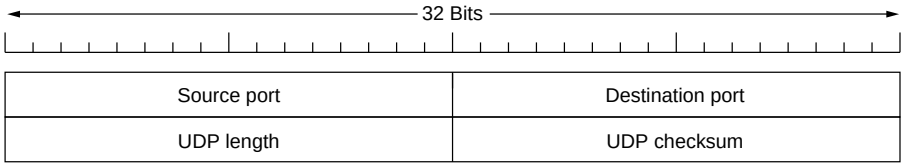


Figura 2.8: L'header di UDP.

In figura 2.8 è mostrato l'header di UDP. Segue la descrizione dei campi che lo compongono:

Source port: 16bit

Numero di porto della sorgente.

Destination port: 16bit

Numero di porto della destinazione.

UDP length: 16bit

Lunghezza di tutto il pacchetto UDP.

UDP checksum: 16bit

Controllo di errore sull'header.

Come vediamo UDP aggiunge veramente poco ad IP, in pratica la differenza fondamentale tra i due è l'introduzione dei numeri di porto, grazie ai quali diviene possibile effettuare una multiplazione di più flussi di traffico diretti verso uno stesso indirizzo IP. Quindi ad esempio più applicazioni possono funzionare contemporaneamente su uno stesso host, le informazioni verranno correttamente instradate verso la giusta applicazione grazie all'indicazione del numero di porto. Di fatto gli end-point a livello trasporto sono specificati dalle coppie [*Indirizzo IP*, *Numero di Porto*].

2.2.3 Transmission Control Protocol (TCP)

TCP [4] offre numerose funzionalità, ed è senza dubbio il protocollo di trasporto più utilizzato. Tra le caratteristiche salienti dobbiamo certamente ricordare che è un protocollo orientato alla connessione ed affidabile. In pratica gli applicativi vedono la connessione TCP come un circuito fisico dedicato ed hanno la certezza che tutto quello che viene spedito arriva all'altro capo della connessione, ed inoltre che anche l'ordine viene preservato. Anche TCP sfrutta il multiplexing delle connessioni grazie ai numeri di porto così come si è visto per UDP.

Una trasmissione inaffidabile e basata sui pacchetti (cioè quello che offre IP) viene trasformata in un flusso di byte con garanzie di affidabilità.

Inoltre vengono usati meccanismi di controllo di flusso e di congestione per evitare di congestionare l'host ricevente o la rete, queste funzionalità vengono garantite utilizzando meccanismi *sliding window* con gestione dell'*acknoledgement.* Tutto questo chiaramente ha un prezzo, l'uso di tecniche sliding window rende necessario un buffering dei dati ricevuti e quindi introduce dei ritardi nel passaggio dei dati ai livelli superiori.

L'analisi del header di TCP (figura) ci aiuta capire meglio il suo funzionamento.

Source port: 16bit

Numero di porto della sorgente.

Destination port: 16bit

Numero di porto della destinazione.

Sequence number: 32bit

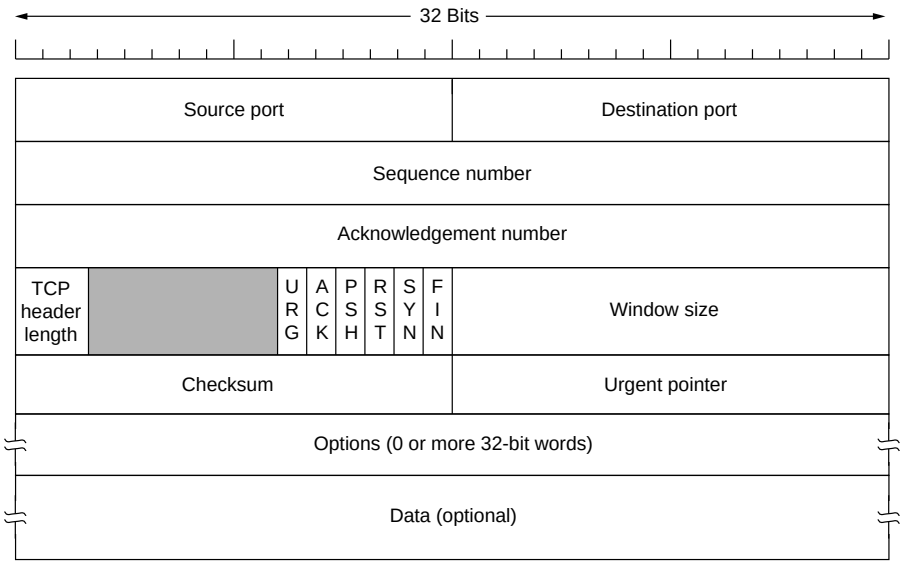


Figura 2.9: L'header di TCP

Identifica, nello stream di byte da trasmettere, la posizione del primo byte nel pacchetto.

Acknoledgement number: 32bit

Contiene il numero del byte immediatamente successivo all'ultimo correttamente ricevuto dalla destinazione.

TCP header length: 4bit

Indica la lunghezza dell'header, questa informazione è necessaria in quanto il campo *opzioni* è di lunghezza variabile.

Flag: 6bit

URG (Urgent) indica che il campo *urgent pointer* è valido, ACK (Acknoledgement) indica che il campo *acknoledgement number* è valido, PSH (Push) serve a sollecitare lo svuotamento del buffer di trasmissione, RST (Reset) resetta la connessione, SYN (Syncronization) sincronizzazione usata nella creazione e nella chiusura della chiusura, FIN (Final) il trasmettitore ha raggiunto la fine dei dati da trasmettere.

Window size: 16bit

Specifica la dimensione della finestra di ricezione.

Checksum: 16bit

Controllo di errore, a differenza di IP e UDP il controllo è esteso anche ai dati.

Urgent pointer: 16bit

In determinate situazioni TCP permette la trasmissione ad alta priorità di dati, in questo caso l'urgent pointer indica la posizione di inizio di questi dati all'interno del pacchetto.

Options: 0 o più parole da 32bit

Tra le principali opzioni abbiamo la dimensione massima del payload, oppure la possibilità di utilizzare il *selective repeat* come algoritmo sliding window.

Capitolo 3

La trasmissione in real-time

Le trasmissioni in real-time richiedono una serie di accorgimenti che non è possibile ottenere con i normali protocolli di TCP/IP. Vengono presentate allora alcune soluzioni; RTP per una gestione efficiente del traffico real-time, Differentiated Services, Integrated Services e RSVP per la qualità del servizio.

3.1 I problemi del real-time

Viene fatta qui una carrellata su quelli che sono i principali requisiti per realizzare una buona trasmissione real-time [7]. Come si vedrà, esistono due diversi tipi di requisiti, i primi più legati alla qualità del servizio mentre gli altri sono legati ad una gestione efficiente della trasmissione ed alla collezione delle informazioni provenienti dai vari host.

Nel primo caso la soluzione deve essere necessariamente affidata a tecniche che ci permettano di riservare la qualità del servizio, vengono quindi analizzate le due alternative più promettenti che sono *Integrated Services* e *Differentiated Services.*

Per la gestione della sessione invece è possibile utilizzare il protocollo *Real Time Protocol (RTP),* che completa UDP di quelle informazioni utili se si vuole trasmettere in tempo reale. Chiaramente le due soluzioni (qualità del servizio e RTP) non sono mutuamente esclusive ma bensì complementari, in quanto affrontano problemi diversi. C'è da dire però che al momento le tecniche per la prenotazione delle risorse di rete sono ancora in fase sperimentale, e quindi non disponibili sulla rete, anche perché richiedono modifiche alle infrastrutture della stessa. Per RTP, invece, non c'è alcun problema, essendo un protocollo di livello applicazione non necessita di

modifiche particolari ai router o agli altri elementi della rete, è l'applicazione che lo vuole utilizzare che si deve accollare l'onere di implementarlo.

3.1.1 La qualità del servizio

Una trasmissione in tempo reale deve essere caratterizzata da:

- Ritardi d'arrivo dei pacchetti contenuti.
- Jitter d'interarrivo ridotto.
- Adeguate risorse di rete.

Si pensi ad esempio ad una applicazione di videoconferenza nella quale due o più persone si scambiano audio (ed eventualmente anche video) realizzando di fatto una conversazione tramite la rete. Se i pacchetti arrivassero con ritardi elevati sarebbe limitata l'interattività della sessione[1], tutto questo renderebbe innaturale e poco pratica la conversazione.

Contenere il *jitter* (variazione del ritardo d'arrivo dei pacchetti) consente invece di ottenere una riproduzione fluida del flusso ricevuto. Per ammortizzare il jitter è certamente possibile implementare un sistema di buffering nella ricezione dei pacchetti, ma bufferizzare i pacchetti comporta l'introduzione di un ulteriore ritardo nella riproduzione.

Per risorse di rete intendiamo soprattutto l'ampiezza di banda. Infatti, se la capacità del collegamento è inferiore all'ampiezza di banda richiesta le code all'interno dei router tenderanno a crescere. In questo modo si avrà un aumento del ritardo d'arrivo dei pacchetti ed inoltre si potranno avere delle perdite.

Da quanto detto risulta subito chiaro tra i due protocolli di livello trasporto la scelta ricade inevitabilmente su UDP. TCP, infatti, al fine di garantire l'arrivo di tutti i pacchetti ed il loro riordino, introduce ritardi (anche molto variabili) che si vanno a sommare a quelli introdotti dalla rete, e si dimostra quindi del tutto inadatto alla trasmissione in tempo reale. Non bisogna però pensare che UDP sia la soluzione a tutti i problemi. Infatti, sebbene per la sua semplicità, esso non influenzi i ritardi dei pacchetti, non è comunque in grado di garantire sull'attuale rete Internet tutte le caratteristiche che sono state elencate in precedenza. Si richiedono, infatti, meccanismi che permettano di riservare e garantire la qualità del servizio. Esistono

[1] Si pensi ad esempio al fastidio del ritardo introdotto dall'uso dei satelliti nelle telefonate intercontinentali

diversi filoni di ricerca che puntano su soluzioni diverse, i più promettenti sembrano però essere il modello *Integrated Services* insieme al protocollo *RSVP*, ed il modello *Differentiated Services.* Tali modelli vengono descritti rispettivamente nei paragrafi 3.2 e 3.3.

3.1.2 Le informazioni ausiliarie

Per semplificare il compito delle applicazioni che producono/ricevono flusso real time è importante avere alcune informazioni ausiliarie come:

- Timestamping
- Numerazione dei pacchetti
- Identificativo del payload

Un *timestamp* è un'etichetta temporale che permette di risalire all'istante in cui il payload è stato acquisito e/o codificato. Conoscere i timestamp permette di ricostruire l'esatta tempificazione della sequenza ricevuta e quindi di riprodurla in modo più simile all'originale. Inoltre grazie ai timestamp si possono ricostruire informazioni quali il ritardo dei pacchetti oppure il jitter, e questo permette, ad esempio, di valutare la qualità della trasmissione.

In una trasmissione real-time come abbiamo visto perdere un pacchetto è preferibile rispetto a riceverlo con un forte ritardo, nonostante questo può essere molto utile numerare i pacchetti[2]. In questo modo, infatti, si riesce a stabilire se ci sono state perdite di pacchetti e si possono applicare tecniche che permettono di compensare in qualche modo questa perdita (tecniche di *error concealment*).

Utilizzare identificativi per i diversi tipi di payload (ad esempio per i diversi standard di codifica) consente di utilizzare applicazioni diverse per trattare uno stesso flusso, semplificando anche la realizzazione di applicazioni che gestiscano contemporaneamente più flussi diversi.

Nel paragrafo precedente si è concluso che per le trasmissioni real time è consigliabile affidarsi ad UDP, tuttavia esso non dispone delle caratteristiche che sono state elencate in questo paragrafo. Per questo motivo è stato pensato il protocollo *RTP* che viene presentato nel paragrafo 3.5.

[2]Tipicamente la numerazione dei pacchetti viene usata nei meccanismi che garantiscono l'affidabilità del collegamento. Quindi ad esempio sono indespensabili per gestire le ritrasmissioni dei pacchetti persi, oppure per riordinare i pacchetti ricevuti.

3.2 Integrated Services (IntServ)

Attualmente la rete Internet fornisce unicamente il servizio *best effort*, cioè tutte le entità che entrano in gioco nell'inoltro dei pacchetti cercano di fare del loro meglio ma di fatto non vengono date garanzie di nessun tipo. Con IntServ, [10], l'intenzione è di estendere (senza stravolgere) l'attuale modello della rete aggiungendo nuove classi di servizio a quella già esistente.

Nella rete Internet attuale tutti i pacchetti vengono trattati allo stesso modo e sono normalmente serviti da nodi a disciplina FIFO. In una rete che supporta IntServ, invece, ogni flusso[3] può ricevere una determinata qualità del servizio, che è stata negoziata all'inizio tra l'applicazione e la rete tramite il protocollo **RSVP** [11] (Resource reSerVation Protocol). Lo stesso RSVP viene anche utilizzato per settare la rete al fine di garantire che tutti i pacchetti di quel flusso beneficino della stessa qualità del servizio. Quindi prima di iniziare la trasmissione, l'applicazione, utilizzando RSVP, deve costruire i percorsi e prenotare le risorse.

3.2.1 Il controllo del traffico (Traffic Control)

Il meccanismo che si occupa di garantire la qualità del servizio necessaria si chiama *controllo del traffico*, in figura 3.1 è mostrato il modello di riferimento della sua implementazione nei router.

Sicuramente gli elementi principali di questo modello sono i seguenti:

1. *Packet Scheduler.*
   Lo schedulatore si occupa della spedizione dei pacchetti dei diversi stream usando un insieme di code e altri meccanismi basati ad esempio su timer. Deve essere implementato laddove i pacchetti sono accodati, cioè tipicamente nell'output driver del sistema operativo. Gli algoritmi di scheduling possono seguire un approccio a *priorità, round robin* oppure *Weighted Fair Queueing (WFQ).*
   Lo schedulatore inoltre decide quali pacchetti scartare in caso di sovraccarico del nodo ed implementa anche funzioni di stima del traffico. Analizza cioè il traffico e ne trae statistiche che ad esempio utilizza per il *policing*[4].

[3] Un flusso viene definito come una sequenza distinguibile di pacchetti che risultano dall'attività di un singolo utente e che richiedono la stessa qualità del servizio.

[4] Si controlla se l'host trasmittente rispetta le caratteristiche contrattate per il traffico, in caso che ciò non avvenisse vengono prese delle contromisure.

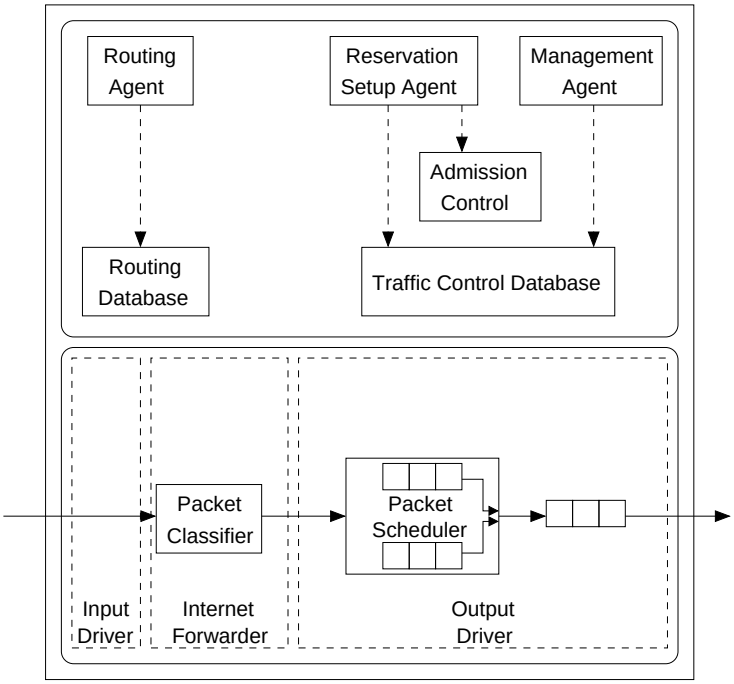


Figura 3.1: Controllo del traffico

2. *Packet Classifier.*
   Per implementare il controllo sul traffico (e anche ad esempio la tariffazione) ogni pacchetto arrivato viene mappato in una specifica classe, e ogni pacchetto di una stessa classe riceverà lo stesso trattamento dallo schedulatore.
   Questa mappatura è effettuata dal classificatore. Una classe può corrispondere tanto ad un unico flusso che ad una categoria di flussi dalle caratteristiche comuni. La scelta della classe appropriata è fatta localmente in ogni router e riflette l'importanza che il router dà al singolo flusso.
   La classificazione può essere fatta ad esempio sulla base degli indirizzi sorgente e destinazione, sul tipo di protocollo oppure sul numero di porto.

3. *Admission Control.*
   É l'algoritmo di decisione che il router utilizza per stabilire se può garantire o meno la qualità del servizio richiesta da un nuovo flusso senza influenzare le precedenti richieste (che sono state già accettate).
   La decisione è presa basandosi sui dati acquisiti dallo schedulatore oppure su stime della situazione peggiore con gli attuali flussi.

Osservando la figura 3.1 si nota che è divisa in due parti, nella parte superiore troviamo il *codice di background*, queste routine si occupano di creare le strutture dati che controllano il meccanismo di instradamento. Il routing agent implementa un particolare protocollo di routing e crea il database di routing. Il reservation setup agent implementa il protocollo utilizzato per riservare le risorse (tipicamente RSVP). Se l'admission control da la conferma ad una nuova richiesta, i cambiamenti appropriati sono apportati al database dello schedulatore e del classificatore al fine di garantire la qualità del servizio richiesta. Infine, ogni router supporta un agente per il network management, questo agente deve essere in grado di modificare il database dello schedulatore e del classificatore, ad esempio, per settare le politiche dell'admission control.

Nella parte inferiore troviamo il *percorso di inoltro*, che viene seguito da ogni pacchetto e deve essere molto ottimizzato affinché il router sia in grado di offrire buone prestazioni.

Nel modello iniziale di IntServ sono state pensate tre diverse classi di servizio corrispondenti a tre diversi tipi di applicazioni.

1. Applicazioni in tempo reale "hard" o intolleranti.
   Queste applicazioni hanno vincoli stringenti sui tempi di consegna dei pacchetti e sulla banda disponibile, ad esempio perché sono state progettate inizialmente per funzionare su reti a commutazione di circuito.

2. Applicazioni in tempo reale "soft" o tolleranti.
   Sono applicazioni che accettano un comportamento aleatorio della rete, e quindi possono tollerare se un pacchetto non arriva oppure arriva molto in ritardo. Tutte le applicazioni multimediali pensate per la rete attuale sono di questo tipo.

3. Applicazioni elastiche.
   Aspettano indefinitamente un pacchetto e quindi non sono compromesse da un ritardo (anche se questo non vuol dire che l'utente non ne sia infastidito). Rientrano in questa categoria le classiche applicazioni interattive come telnet, ftp, ecc

A questa distinzione corrisponde la distinzione nelle seguenti classi di servizio:

1. Servizio garantito

2. Servizio a carico controllato

3. Servizio best effort

L'ultima classe corrisponde al classico servizio offerto da Internet ed in esse rientrano oltre alle applicazioni elastiche anche tutte le altre applicazioni non interattive (come ad esempio la posta elettronica), le altre due classi di servizio vengono invece illustrate di seguito.

3.2.2 Il servizio garantito (Guaranteed Service)

Questa classe fornisce al flusso di dati generati dall'applicazione cliente un limite stretto sul ritardo di trasmissione end-to-end del generico pacchetto, inoltre viene garantita anche la banda che sarà fornita al flusso. Il servizio garantito quindi rappresenta sicuramente quello con le garanzie massime possibili, ed ha caratteristiche tali da permettere di emulare una comunicazione su circuito dedicato.

Le informazioni relative al traffico generato, le caratteristiche della rete, i parametri di prenotazione ecc. vengono scambiate tra i nodi tramite il protocollo RSVP.

3.2.3 Il servizio a carico controllato (Controlled Load)

Il servizio a carico controllato può essere considerato come una sorta di "controlled best-effort", nel senso che ha le stesse caratteristiche di un servizio best-effort (quindi nessuna garanzia) su di una rete poco carica, tuttavia si cerca di fare in modo che il servizio non cambi anche al crescere del carico della rete.

In altre parole l'utente che richiede un servizio a carico controllato ottiene un servizio best-effort che però risulta essere poco sensibile alle variazioni del carico della rete. É chiaro quindi che per riuscire a mantenere queste promesse è fondamentale il ruolo dell'admission control.

3.2.4 Resource reSerVation Protocol (RSVP)

Il protocollo RSVP è usato dagli host per richiedere una specifica qualità del servizio alla rete, ma è anche usato dai router per spedire informazioni sulla qualità del servizio lungo il percorso seguito dal flusso al fine di stabilire e mantenere la classe di servizio richiesta.

Le risorse sono riservate per i *flussi simplex*, cioè in una sola direzione, quindi sono ben distinti i ruoli di trasmettitore e ricevitore anche se nulla vieta che una stessa entità possa essere allo stesso tempo trasmettitore e ricevitore, in realtà RSVP è indipendente da IntServ, quindi può essere utilizzato anche in altri sistemi che garantiscono la qualità del servizio, la sua integrazione con IntServ è illustrata in [12].

Il protocollo gira al di sopra di IP, tuttavia non trasporta in alcun modo dati, ed è quindi scorretto considerarlo un protocollo di trasporto. Esso fa parte (insieme ad ICMP, IGMP ed ai protocolli di routing) di quei protocolli che possiamo immaginare confinati ancora a livello tre anche se ne sfruttano i servizi.

É interessante notare che RSVP è esplicitamente pensato per funzionare in ambiente multicast, quindi il caso unicast è visto come una semplice specializzazione. Questo permette di fare in modo che ogni ricevitore possa richiedere una qualità del servizio diversa da quanto fatto dagli altri.

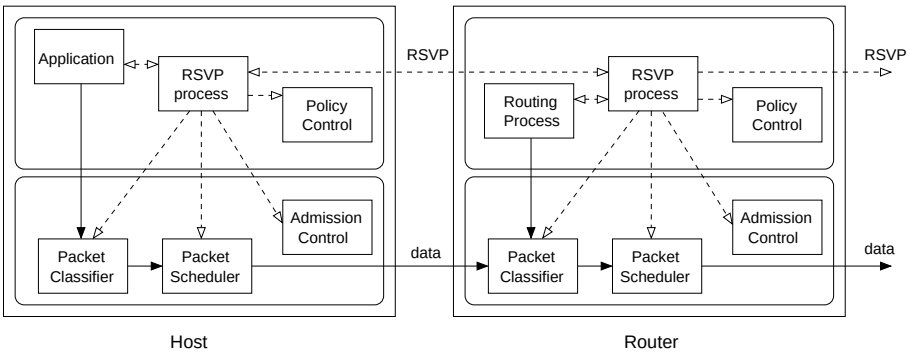


Figura 3.2: RSVP negli host e nei router.

In figura 3.2 è mostrato il modo in cui il processo RSVP interagisce negli host e nei router.

Nel processo di prenotazione il ruolo fondamentale è giocato dai messaggi *Path* e *Resv* (figura 3.3). I primi vanno dal trasmettitore ad ricevitore, trasportano informazioni sul flusso e sono usati per costruire il percorso che sarà seguito dal flusso. I messaggi *Resv* seguono il percorso contrario e sono quelli che di fatto compiono la prenotazione. Come si vede la prenotazione è di tipo *receiver initiated*, questo approccio è molto vantaggioso nel caso multicast, e permette una flessibilità maggiore rispetto al caso in cui la prenotazione viene effettuata dal trasmettitore.

Un messaggio *Path* contiene le seguenti informazioni:

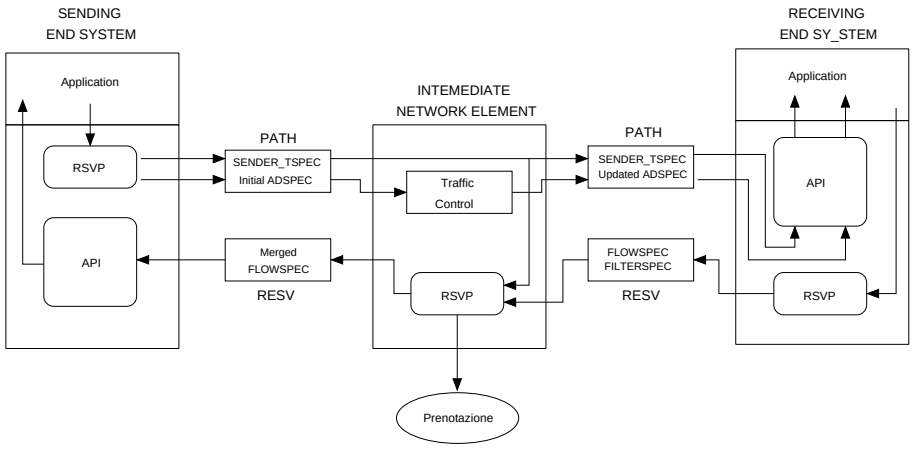


Figura 3.3: Il meccanismo di prenotazione.

- L'indirizzo dell'ultimo nodo RSVP lungo il percorso.
- L'indirizzo e il numero di porto della sorgente, oppure un'etichetta di flusso nel caso in cui il protocollo di livello rete la supporti (come ad esempio IPv6).
- Le caratteristiche del traffico che sarà generato dalla sorgente (*Tspec*).
- Una struttura opzionale chiamata *Adspec* che è aggiornata ad ogni nodo attraversato e serve per stabilire le caratteristiche del percorso dalla sorgente alla destinazione.

Quando un nodo riceve un pacchetto *Path* crea oppure aggiorna lo stato del percorso (path state) che viene poi utilizzato dai messaggi *Resv* per poter seguire il giusto percorso verso la sorgente. Ad ogni path state è associato un timer che ne indica la validità, una volta scaduto il timer (che può essere resettato se viene ricevuto un messaggio di tipo *Path)* il path state viene cancellato. Infatti RSVP è un protocollo *soft state* (detto anche connection less), e quindi i percorsi creati non vengono mantenuti in eterno, ma tendono a "scadere", però periodicamente vengono mandati i messaggi *Path* al fine di mantenere lo stato del percorso nei nodi. Questo comportamento permette di adattarsi dinamicamente ai cambiamenti che si possono avere sulla rete, come ad esempio un cambiamento di un percorso di routing oppure il guasto di un router e così via.

Un messaggio *Resv* contiene le seguenti informazioni:

- Lo stile di prenotazione, che permette di effettuare una prenotazione per una sorgente ben precisa (*Fixed Filter*), di condividerla tra più sorgenti selezionate

mediante wildcard (*Wildcard Filter*) oppure di condivere la prenotazione tra più sorgenti selezionate esplicitamente (*Shared Explicit*).

- La struttura *Filterspec,* che permette di selezionare il flusso (attraverso l'indirizzo della sorgente oppure tramite protocolli di livello superiore).
- La struttura *Flowspec,* che contiene una descrizione del traffico a cui va applicata la prenotazione (*Tspec*) e la classe di servizio richiesta (*Rspec*).

3.3 Differentiated Services (DiffServ)

Le principali critiche volte al modello Integrated Services hanno soprattutto evidenziato che è un modello poco scalabile su reti di grosse dimensioni, il livello di granularità delle informazioni[5] è probabilmente troppo elevato, inoltre è richiesto un forte intervento anche sull'architettura di base della rete, ogni router deve supportare il protocollo RSVP. Questi motivi hanno spinto a valutare modelli alternativi che fossero più facilmente scalabili.

Differentiated Services [13], prende in considerazione un numero ben definito di servizi, indipendentemente dalle applicazioni in gioco, si affida ad un insieme limitato ed efficiente di meccanismi di instradamento, evita di memorizzare informazioni di stato per ogni singolo flusso richiedente garanzie di qualità del servizio. Il funzionamento è molto semplice, il traffico in ingresso alla rete subisce un processo di "condizionamento", in corrispondenza delle cosiddette zone di frontiera, i flussi vengono in pratica assegnati a dei macro-flussi identificati tramite un semplice campo dell'header denominato DS byte. Nel caso di IPv4 [14], ad esempio il DS in realtà va a sovrascrivere l'inutilizzato campo TOS. Ad ogni DS è associata una diversa strategia di instradamento che determina il modo in cui i pacchetti vengono gestiti all'interno della rete (*Per-Hop Behaviour, PHB*). Si noti che non c'è nessun protocollo di segnalazione o informazione sullo stato, ogni router deve mantenere esclusivamente le informazioni per il management dei diversi PHB che sono in numero finito e indipendenti dal numero di microflussi aggregati, il numero di informazioni da mantenere sono in definitiva proporzionali al numero di classi di servizio e non al numero di flussi d'applicazione.

[5] I router devono mantenere il *path state* per ogni singolo flusso.

3.3.1 Elementi funzionali dell'architettura DiffServ

Anche per realizzare il servizio DiffServ occorrono delle forme di controllo di ammissione e politiche di amministrazione del traffico (policing).

Possiamo individuare i seguenti elementi funzionali.

1. *Packet Classifier*: si occupa di classificare i vari pacchetti in base alle informazioni contenute in una parte dell'header, si individuano quindi quella porzione del traffico che può ricevere un servizio differenziato assegnandogli un PHB. I classificatori possono funzionare sulla base dell'analisi del solo campo DS (*Behaviour Aggregate*), oppure sulla base di più campi dell'header (*Multi-Field*), ad esempio valutando congiuntamente il campo DS con gli indirizzi sorgente e destinazione.

2. *Traffic Conditioner*: è la parte vitale del sistema, il suo obiettivo è di applicare le specifiche di condizione sui pacchetti preventivamente classificati, il conditioner consiste di un o più di questi elementi:

   (a) *Meter*: misura il traffico dei pacchetti, suddividendoli in "in-profile" e "out-profile"

   (b) *Marker*: per i pacchetti in-profile seleziona, se necessario, un nuovo codice DS, o pacchetti out-profile, invece, vengono bollati mediante un apposito codice.

   (c) *Shaper*: si occupa dei pacchetti out-profile, scegliendo di ritardarne la trasmissione per farli rientrare in-profile.

   (d) *Dropper/Policier*: effettua il policing dei flussi, inoltre si occupa di rigettare i pacchetti out-profile in caso di congestione.

In figura 3.4 sono mostrati gli elementi funzionali introdotti ed il modo in cui interagiscono tra di loro. Si noti che le sofisticate operazioni di classificazione, marcatura, policing e shaping sono necessarie solo ai confini della rete, all'interno della rete i router hanno solo bisogno di conoscere il DS.

3.3.2 Le classi di servizio

Si può effettuare una prima distinzione in classi.

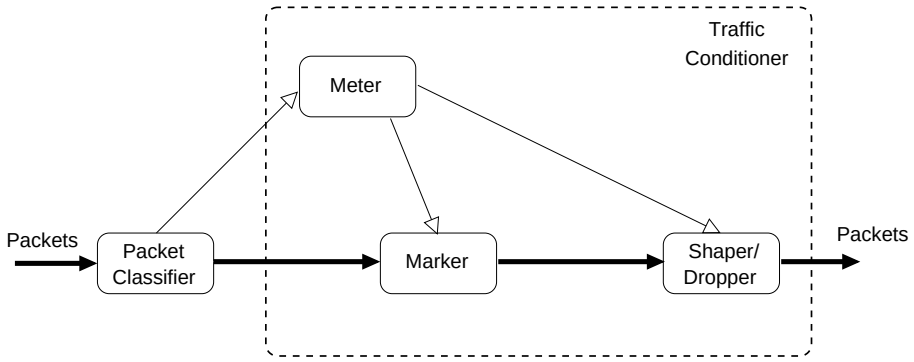


Figura 3.4: Elementi funzionali dell'architettura DiffServ

1. *Default service (DE)*: è la modalità di spedizione best effort tradizionale utilizzata in Internet. Tale servizio viene indicato resettando tutti i bit del campo DS.

2. *Premium service*: per applicazioni che richiedono un servizio caratterizzato da priorità più elevate.

Il premium service è composto a sua volta da due classi.

- *Expedited service*: per applicazioni che richiedono un servizio caratterizzato da un basso delay e jitter.

- *Assured service*: per applicazioni che richiedono una maggiore garanzia di raggiungibilità rispetto al servizio best-effort in caso di congestione. Se il traffico assured risulta in-profile ha una scarsa probabilità di essere ritardato. Questo servizio viene realizzato attraverso una accurata gestione dei buffer dei router. In pratica è analogo al servizio controlled load fornito da IntServ.

3.4 L'interoperabilità tra IntServ e DiffServ

Nei paragrafi precedenti sono state illustrate quelle che attualmente sono le due soluzioni maggiormente quotate per portare il concetto della qualità del servizio sulla rete Internet. Entrambe hanno i loro vantaggi e svantaggi, IntServ introduce il concetto fondamentale di QoS associata al singolo flusso, ma porta inevitabilmente a problemi di scalabilità, DiffServ elimina l'esigenza di mantenere informazioni di stato per i singoli flussi e non ha bisogno di alcun protocollo di segnalazione, tuttavia manca della capacità di "isolare" le singole richieste di QoS.

In realtà non è detto che le due soluzioni vadano applicate in modo esclusivo, si stanno facendo diversi tentativi [15] volti proprio a permettere l'interoperabilità delle due soluzioni. IntServ si mostra molto più utile ed efficiente per la rete d'accesso dell'utente mentre DiffServ è certamente più adatto per le reti di transito sulle quali passano grossi aggregati di traffico.

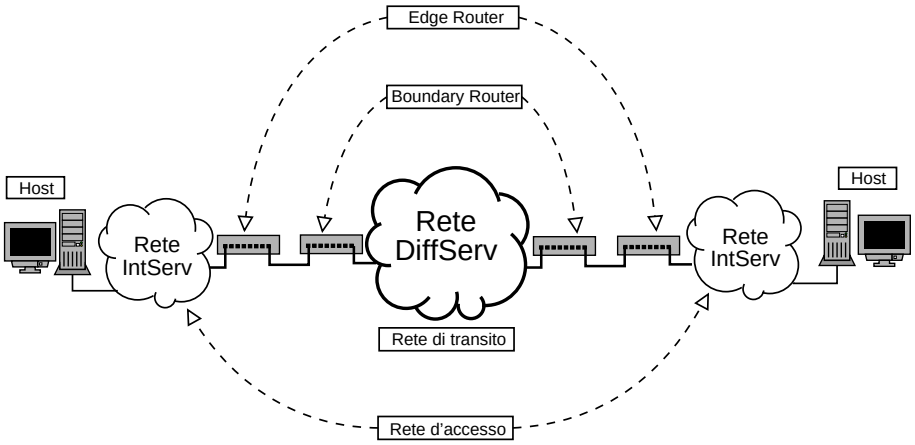


Figura 3.5: L'integrazione di IntServ e DiffServ

In figura 3.5 è appunto mostrata questa situazione. Secondo questo scenario, gli host sono collegati a reti di accesso che supportano IntServ e la segnalazione RSVP end-to-end, in realtà però il traffico prodotto arriva alla rete di transito dove è supportato DiffServ, e quindi i vari traffici vengono aggregati secondo i criteri adottati da quest'ultimo. In prossimità delle interfacce IntServ-DiffServ, ci sono gli *edge-router*, costituiti per metà da un router RSVP standard e per l'altra metà da un modulo di interfaccia capace di interagire con l'admission control della rete DiffServ. In prossimità delle interfacce DiffServ-IntServ, ci sono invece i *boundary router*, adibiti allo svolgimento delle normali funzioni di policing della rete DiffServ.

Una caratteristica importante è la possibilità di garantire la segnalazione RSVP end-to-end, in questo modo la rete DiffServ sottostante risulta del tutto trasparente agli host. In pratica l'host mittente genera il messaggio Path contenente la descrizione del traffico, all'interno della rete di accesso IntServ questo viene processato in maniera standard dai router RSVP. In prossimità del primo edge-router le informazioni di stato convogliate dal messaggio Path vengono memorizzate, ed il messaggio viene inoltrato verso la rete di transito che viene attraversata fino a raggiungere la rete IntServ all'altro capo, in quest'ultima il messaggio viene nuovamente processato dai router RSVP. A questo punto in prossimità dell'host ricevente viene generato

il messaggio Resv che deve effettuare la prenotazione delle risorse. Il meccanismo standard di prenotazione viene applicato all'interno della rete IntServ, arrivato all'edge-router il messaggio viene accettato o rifiutato a seconda della disponibilità di risorse sull'interfaccia a valle. In pratica viene sottoposto al controllo d'ammissione della rete DiffServ basandosi sul livello di servizio diffserv richiesto (ottenuto mediante un mapping delle classi IntServ), se viene accettato attraversa in modo trasparente la rete di transito fino a giungere all'altra sottorete IntServ. A questo punto il messaggio Resv giunge all'host mittente, il quale lo interpreta come una conferma della disponibilità di risorse e quindi può iniziare a trasmettere.

Con questa tecnica si riesce a garantire una qualità del servizio end-to-end anche su una rete eterogenea che utilizzi contemporaneamente le due diverse tecnologie.

3.5 Real Time Protocol (RTP)

RTP [16] è un protocollo di livello applicazione, ed è stato progettato per essere indipendente dal protocollo di trasporto, anche se tipicamente viene utilizzato al di sopra di UDP e lo completa di quelle funzionalità necessarie alle trasmissioni in real-time. É bene dire subito che RTP non garantisce in alcun modo la qualità del servizio, ne tanto meno riserva le risorse di rete, semplicemente fornisce alcune informazioni come numeri di sequenza e timestamp.

Viene affiancato da un altro protocollo, **RTCP** (Real Time Control Protocol) che permette un monitoraggio sulla trasmissione e fornisce anche un modo per distribuire informazioni sui partecipanti alla sessione.

RTP segue le linee guida fornite da Clark e Tennenhouse in [18], quindi è in un certo senso malleabile e può essere facilmente modificato/esteso al fine di supportare nuovi tipi di applicazione o architetture di rete. Quindi quando si decide di utilizzare RTP per una nuova applicazione è necessario produrre due documenti che completano la descrizione dell'uso di RTP:

- un profilo, che definisce i codici utilizzati come payload-type ed il loro mapping con il formato del payload. Nel profilo vanno inoltre specificate le estensioni/modifiche apportate al protocollo
- un documento in cui si indicano i formati usati per il payload, cioè viene spiegato in che modo il particolare payload è trasportato su RTP.

Un tipico scenario di utilizzo di RTP è la videoconferenza. Se sono usati sia l'audio

che il video questi tipicamente sono trasmessi in differenti sessioni RTP, una sessione RTP è identificata da una coppia di indirizzi di livello trasporto (indirizzo IP + numero di porto), nel caso di multicast l'indirizzo di destinazione è comune a tutti i partecipanti. Utilizzando due sessioni RTP è possibile fare in modo ad esempio che alcuni partecipanti ricevano sia l'audio che il video, mentre altri ricevano solo uno dei due.

RTP prevede anche l'utilizzo dei cosiddetti *Mixer* e *Traduttori.* I primi sono usati ad esempio per riunire i flussi provenienti da più sorgenti in un unico stream multimediale riducendone anche la larghezza di banda. In questo modo anche sorgenti collegate con link non velocissimi potranno partecipare alla sessione. I Traduttori invece possono essere utilizzati per aggirare dei problemi che si potrebbero avere quando ad esempio l'host si trova dietro un firewall.

In figura 3.6 è mostrato l'header di RTP.

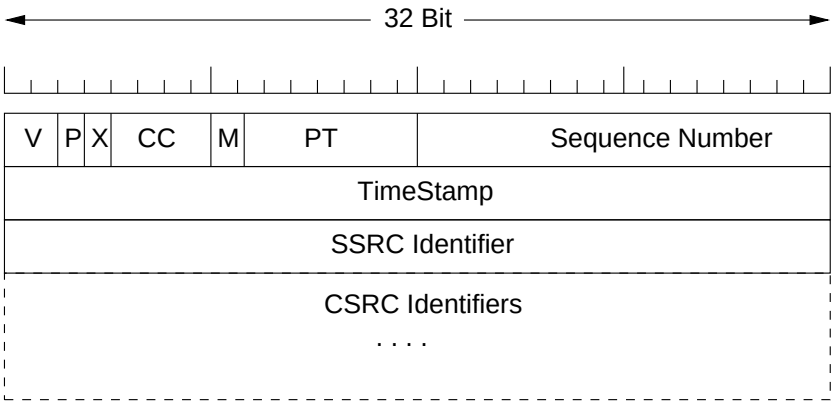


Figura 3.6: L'header di RTP

Version (V): 2bit

Indica la versione del protocollo, attualmente è 2.

Padding (P): 1bit

Se settato il pacchetto contiene uno o più ottetti di riempimento alla fine. L'ultimo ottetto conterrà il numero di ottetti di riempimento inseriti.

Extension(X): 1bit

Se settato l'header è seguito da un header extension (definito nel profilo).

CSRC count(CC): 4bit

Contiene il numero di Contributing Sources identifiers che seguono l'header base.

Marker(M): 1bit

L'interpretazione di questo bit è definita nel profilo. Si può anche decidere di non utilizzarlo, estendendo quindi il campo successivo ad 8 bit.

Payload type(PT): 7bit

Contiene l'identificativo del payload, i relativi codici vanno definiti nel profilo.

Sequence Number: 16bit

Numero di sequenza del pacchetto. Il valore iniziale è scelto a caso, e successivamente viene incrementato di 1 per ogni pacchetto

Timestamp: 32bit

Riflette l'istante di campionamento del primo ottetto del payload. Se i pacchetti sono generati periodicamente, viene considerato allora l'istante nominale di campionamento, e quindi il timestamp viene incrementato di 1 per ogni periodo di campionamento, il valore iniziale è comunque casuale. Più pacchetti possono avere lo stesso timestamp, come accade, ad esempio, se una frame video viene spedita utilizzando più pacchetti.

SSRC: 32bit

Il Syncronization Source identifier è un identificatore della sorgente scelto in modo random. Non ci possono essere più sorgenti con lo stesso SSRC.

CSRC: lista, da 0 a 15 elementi da 32bit l'uno

I Contributing Source identifier sono inseriti dai mixer, e sono i SSRC delle sorgenti che contribuiscono al flusso.

Le differenze più importanti rispetto ad UDP sono certamente introdotte dai campi *Payload Type, Timestamp* e *Sequence Number.* L'applicazione che riceve i dati grazie al PT è in grado di capire in che modo trattarli, il TS invece permette di riprodurre con l'esatta temporizzazione i dati ricevuti, infine il SN permette di stabilire se i pacchetti stanno arrivando nell'ordine opportuno e se ci sono state perdite. Si noti però che RTP non è un protocollo affidabile (come era prevedibile, essendo un protocollo pensato per il real-time), quindi non gestisce assolutamente la ritrasmissione dei pacchetti persi.

3.5.1 La personalizzazione

Come anticipato, RTP è un protocollo molto flessibile che può essere modificato per adattarlo meglio alle caratteristiche dell'applicazione che lo usa. Si viene così a creare un profilo che va presentato mediante un'opportuna documentazione.

Ci sono diversi punti in cui si può intervenire, i principali sono elencati di seguito:

1. Il significato dei campi *Payload type* e *Marker bit* è di fatto dipendente dall'applicazione. Quindi in un nuovo profilo si dovrà specificare il significato semantico del marker bit, e quali sono i payload type utilizzati.
   Tuttavia è anche possibile utilizzare gli 8 bit totali in un modo diverso, ad esempio assegnando più marker bit, oppure utilizzandoli come un unico campo.

2. É possibile definire delle estensioni per l'header. Quindi se il campo *X* è settato ad uno al normale header segue un header definito dall'applicazione il cui formato è mostrato in figura 3.7.

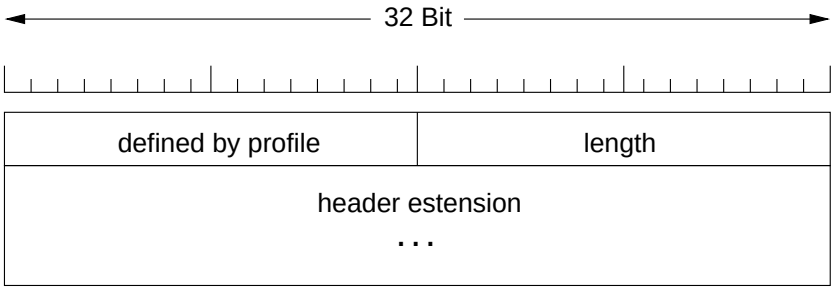


Figura 3.7: Header di estensione

3. Come vedremo in seguito lo standard definisce una serie di pacchetti RTCP, tuttavia è lasciata la possibilità di definirne uno nel profilo.

3.5.2 RTCP (Real Time Control Protocol)

RTCP è un protocollo affiancato ad RTP, sfrutta le capacità di multiplexing del protocollo sottostante (quindi ad esempio se il protocollo sottostante è UDP, una sessione RTCP usa numeri di porto differenti da quelli della corrispondente sessione RTP). RTCP ha essenzialmente 4 funzioni, descritte nel seguito.

1. Principalmente RTCP deve fornire un feedback sulla qualità della distribuzione dei dati, quindi permette di scambiare informazioni sul numero di pacchetti ricevuti o persi sul jitter e così via. Questo riscontro può essere utilizzato direttamente da quelle applicazioni che realizzano tecniche di codifica a tasso variabile. Queste informazioni sono trasportate nei pacchetti *RTCP sender report* e *receiver report.*

2. Ogni sorgente RTP, come abbiamo visto, è identificata da un *SSRC*, tuttavia questo identificativo cambia ogni volta che si fa partire l'applicazione, mentre può essere utile avere un identificativo che sia univoco e non cambi al cambiare della sessione. Per questo motivo si usa il *CNAME (Canonical Name)* che viene diffuso tramite RTCP.

3. Per ogni sessione, si assume che comunque il traffico RTCP debba mantenersi al disotto di un fissato limite, per questo motivo se il numero di partecipanti cresce ognuno dovrà ridurre il tasso di emissione di pacchetti di controllo.

4. Opzionalmente è possibile trasportare con RTCP alcune semplici informazioni sui partecipanti alla sessione, come ad esempio il nome, l'indirizzo e-mail, ecc.

Le prime tre funzioni sono obbligatorie nel caso di applicazioni multicast e sono invece raccomandate nel caso di applicazioni unicast.

Sono previsti diversi tipi di pacchetti.

Sender Report (SR): contiene le statistiche calcolate da una sorgente. É sempre composto da un primo blocco in cui ci sono le statistiche di questa sorgente (byte trasmessi, pacchetti trasmessi ed il timestamp del pacchetto). Possono seguire altri blocchi in cui si trovano le statistiche riguardanti altre sorgenti attive (ultimo numero di pacchetto ricevuto, jitter, timestamp dell'ultimo SR di questa sorgente).

Receiver Report (RR): contiene le statistiche calcolate da un ricevitore. Come l'SR anche il RR può essere composto da più sezioni. Nella prima sezione chiaramente comparirà solo il SSRC della sorgente, le altre sezioni sono analoghe a quelle viste per il SR.

Source Description Items (SDES): un pacchetto SDES può contenere più SDES Items, ognuno dei quali trasporta un'informazione diversa sulla sorgente. Le

possibilità sono le seguenti, CNAME, nome, indirizzo e-mail, numero di telefono, località geografica, applicazione utilizzata, note, estensione definita dall'applicazione.

BYE: indica la fine della partecipazione alla sessione. Opzionalmente può contenere anche il motivo dell'abbandono della sessione.

APP: pacchetto definito dall'applicazione.

Può capitare di dover spedire più pacchetti RTCP di diverso tipo, in questo caso è prevista la possibilità di raggrupparli tutti in un unico pacchetto di livello inferiore creando un pacchetto composito. Non ci sono regole precise sulla composizione di questi pacchetti ma solo alcune linee guida.

- I pacchetti SR e RR devono essere spediti il più spesso possibile (compatibilmente con i vincoli sulla banda) per massimizzare la risoluzione delle statistiche come, ad esempio, il jitter o il round trip time. Quindi ogni pacchetto composito dovrebbe contenere un pacchetto di report.
- I nuovi arrivati hanno la necessità di conoscere al più presto il CNAME degli altri partecipanti, quindi ogni pacchetto composito dovrebbe contenere un pacchetto SDES CNAME.

Capitolo 4

Il lavoro sviluppato

In questo capitolo viene presentato il lavoro svolto. Prima di tutto vengono ribaditi gli obiettivi, vengono quindi presentati i due prototipi implementati, le tecniche utilizzate nella loro realizzazione e le prove a cui sono stati sottoposti.

4.1 Obiettivi

Questo lavoro si pone come obiettivo principale la realizzazione di un sistema di comunicazione video accessibile alla platea più ampia ed eterogenea possibile di utenti. Il sistema deve quindi risentire il meno possibile delle eventuali limitazioni che caratterizzano i singoli utenti, sia dal punto di vista della capacità del canale d'accesso che da quello della potenza di calcolo disponibile.

Si desidera permettere l'accesso sui più svariati canali trasmissivi, da quelli a banda larga, caratteristici di reti ATM o delle reti locali, a quelli a banda più limitata, che si possono ad esempio incontrare nella trasmissione su reti geografiche (si pensi all'accesso via modem su doppino telefonico o al canale radio delle reti cellulari).

Decidendo poi di rendere disponibile il servizio anche per utenti che dispongono di terminali con limitata potenza di calcolo bisogna che la complessità computazionale sia molto ridotta, sia per quanto riguarda la codifica che la decodifica.

L'algoritmo di codifica utilizzato è fortemente scalabile e di bassa complessità, le caratteristiche salienti possono essere riassunte così:

- è possibile adattare il bit-rate della trasmissione alla capacità dei singoli collegamenti;

- è possibile ricevere il segnale video a diverse risoluzioni per adattarsi alle caratteristiche del monitor in ricezione (scalabilità spaziale);
- si può ricevere il segnale a diversi frame-rate (scalabilità temporale) per consentire la ricezione con terminali dalle prestazioni diverse;
- l'algoritmo di codifica è simmetrico, in modo da permettere la trasmissione/ricezione del segnale in modalità software anche con terminali di potenza ridotta.

Questi obiettivi sono stati perseguiti nel lavoro di progettazione del codec utilizzato in questa tesi che è stato presentato nel primo capitolo.

Innanzitutto l'algoritmo di codifica è stato modificato per consentire di variare i parametri che influenzano il tasso trasmissivo (e quindi la qualità), in modo da rispettare i vincoli imposti dal terminale trasmittente.

Tuttavia non è detto che questi vincoli siano propri anche del terminale ricevente. In un'ottica unicast questo però importa poco, l'host ricevente dovrà adattarsi ai parametri scelti dal terminale trasmittente (o comunque i due terminali potranno accordarsi sui parametri da utilizzare). Il discorso diviene più delicato in un contesto multicast. In questo caso ci saranno più host riceventi, ciascuno caratterizzato dai propri vincoli che difficilmente coincideranno con quelli degli altri host. Utilizzando un semplice sistema di codifica con tasso variabile il trasmettitore dovrebbe scegliere i parametri di codifica che producono il flusso che si adatta alle caratteristiche dell'host dalle prestazioni peggiori . In figura 4.1 è mostrata una situazione di questo tipo: accanto ai link in grassetto è indicata la loro capacità, le linee tratteggiate invece rappresentano il flusso prodotto dal codificatore ed il valore accanto a esse è l'ampiezza di banda occupata da questo flusso.

Come si vede tutti gli host sono costretti a ricevere un flusso a 128kbit/s anche se la loro rete di accesso permetterebbe di sostenere tassi superiori[1].

Una possibile soluzione a questa situazione è rappresentata dalla tecnica **MMG** (Multiple Multicast Group). Abbiamo anticipato che il codificatore è dotato di tre diversi tipi di scalabilità ed è in grado di combinarli in un flusso codificato embedded; l'idea allora consiste nel suddividere questo flusso in più sottoflussi, di cui uno è indispensabile mentre gli altri contengono le informazioni di *miglioramento.* Associando ciascuno di questi sottoflussi ad un diverso gruppo multicast si rendono

[1] Questa situazione è tipicamente indicata come *scenario del minimo comune denominatore.*

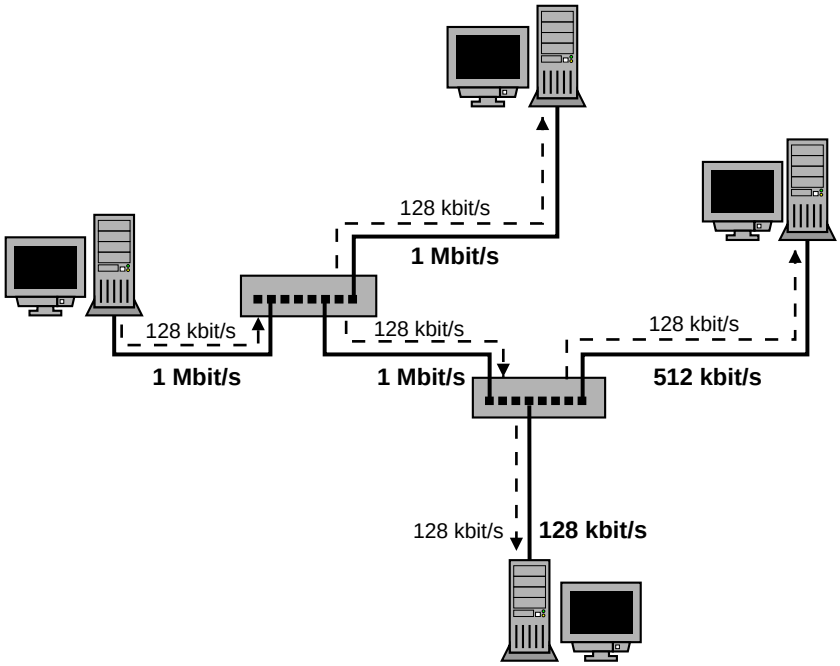


Figura 4.1: Distribuzione tra client eterogenei

possibili situazioni nelle quali ciascun host sceglie quali gruppi sottoscrivere (indipendentemente da quelli scelti dagli altri host) e quindi sceglie a quale tasso ricevere. Uno schema di questo tipo è anche indicato con il nome di *Receiver-driven Layered Multicast (RLM)* [20], in quanto la scelta è appunto effettuata dal ricevitore e non più dal trasmettitore che in questo caso trasmetterà sempre alla massima qualità possibile senza nemmeno sapere cosa hanno scelto i ricevitori.

In figura 4.2 è mostrata una situazione in cui si applica la tecnica MMG. Come si vede ogni host è libero di decidere la qualità del flusso ricevuto in modo indipendente da quanto hanno scelto gli altri. Sebbene la scelta sia lasciata ai ricevitori non è ancora ben chiaro in che modo quest'ultimi dovrebbero scegliere quali gruppi sottoscrivere.

Sono stati presentati molti lavori nei quali viene sfruttata la tecnica MMG e probabilmente il punto su cui si differenziano maggiormente è proprio quello dei criteri che spingono alla sottoscrizione dei vari gruppi. In [19], ad esempio, si propone che i ricevitori effettuino un controllo (non meglio precisato) per stabilire la banda disponibile sulla loro rete d'accesso; contemporaneamente la sorgente spedisce ai vari ricevitori delle informazioni sui vari gruppi, come il tasso prodotto ed il posizio-

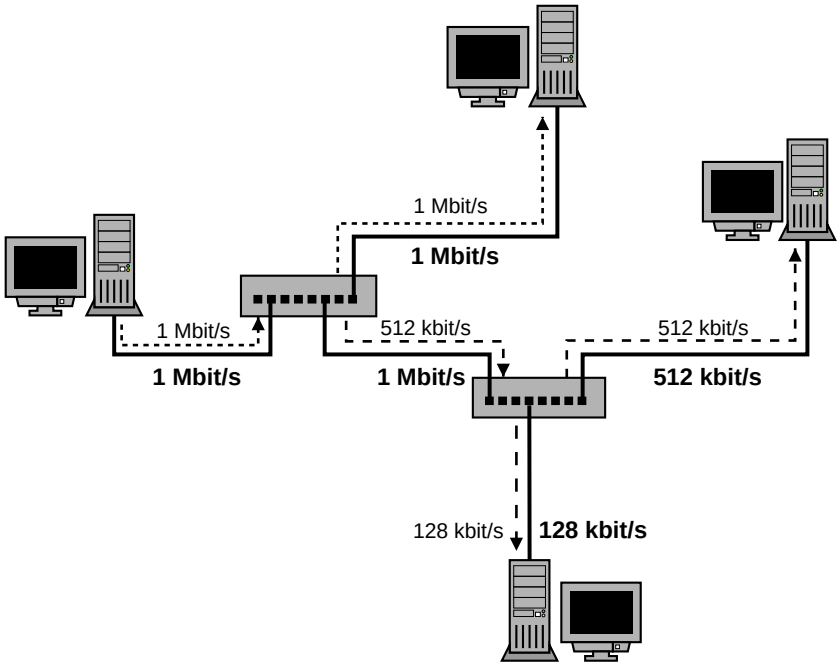


Figura 4.2: Distribuzione MMG

namento del particolare flusso nello stream embedded. Il ricevitore a questo punto ha tutte le informazioni sufficienti per scegliere i gruppi da sottoscrivere senza congestioncosiddettiare la sua rete di accesso. Il controllo sulla banda disponibile può essere fatto periodicamente in modo da adattarsi ai cambiamenti della rete.

Nell'articolo [24] invece si introducono i cosiddetti *Thin Streams*, ovvero i vari layer prodotti dall'algoritmo di codifica vengono a loro volta suddivisi in più stream dal tasso costante che poi vengono trasportati su più gruppi multicast, in questo modo si ha una separazione tra il livello di codifica e quello di rete. Calcolando la differenza tra il throuput effettivo e quello stimato, il ricevitore è in grado di stabilire se è possibile sottoscrivere altri Thin Streams oppure se è il caso di abbandonarne alcuni.

In [20] viene introdotta la tecnica RLM già anticipata, nella quale il ricevitore effettua i cosiddetti *join experiments*, in pratica, periodicamente tenta di sottoscrivere un ulteriore layer, e se riscontra che ci sono perdite lo abbandona, salvo poi ritentare successivamente, il tempo di attesa però cresce ogni volta che si hanno dei fallimenti, in questo modo si evita di sovraccaricare inutilmente la rete.

Infine in [25] viene proposta una tecnica basata sulle informazioni trasportate da

RSVP, quindi si combina la scelta dei gruppi da sottoscrivere con i meccanismi per la gestione della qualità del servizio.

4.2 Il prototipo unicast

Come primo passo si è deciso di realizzare un prototipo unicast che prevedesse la possibilità di sfruttare appieno la scalabilità offerta dal codec permettendo di variare il bit-rate, come protocollo di trasporto si è scelto di utilizzare RTP su UDP. Mentre UDP e TCP sono implementati direttamente nel sistema operativo e quindi le loro funzioni sono accessibili tramite primitive di sistema (le socket), RTP è un protocollo di livello applicazione e quindi deve essere implementato dall'applicazione che lo vuole utilizzare. Dopo una serie di ricerche in rete si è scelto di utilizzare la libreria *JRTPLIB*[2] per la sua semplicità d'utilizzo, la disponibilità su diverse piattaforme e la comprensibilità del codice di sorgente.

Sono state realizzate due applicazioni, un *server* ed un *client*, il primo si occupa di leggere le frame non codificate da disco, codificarle e trasmetterle sulla rete, il client invece riceve queste frame le decodifica e le visualizza in una finestra, riproducendo quindi la sequenza video. Per motivi di semplicità si è deciso di trasmettere ogni frame codificata in un unico pacchetto, mentre le frame di tipo $(k-1)*4+1$ e $(k-1)*4+3$ vengono spedite insieme in un unico pacchetto in quanto appartenenti allo stesso layer temporale[3].

4.2.1 L'interfaccia grafica

Per il client è stata realizzata l'interfaccia grafica mostrata in figura 4.3.

Sulla destra si possono vedere tre menu a scomparsa tramite i quali è possibile impostare i parametri di codifica. Si è deciso, infatti, di scegliere i parametri dal client (che in questo modo può decidere la qualità della sequenza che riceverà). In basso invece è possibile inserire l'indirizzo IP del server (per semplicità il numero di porto sul quale il server si mette in ascolto è fisso) nella classica notazione decimale puntata. Si può notare inoltre una casellina recante la scritta *Attiva Zoom*, selezio-

[2] Implementata da Jori Liesemberg (`jori@lumumba.luc.ac.be`) per il suo lavoro di tesi "Voice over IP in networked virtual environments" e disponibile liberamente all'indirizzo `http://lumumba.luc.ac.be/jori/jrtplib/jrtplib.html`

[3] Inoltre queste frame sono codificate con il conditional replenishment bidirezionale rispetto a dei riferimenti molto vicini temporalmente. Questo fa si che le loro dimensioni siano molto inferiori rispetto a quelle delle altre frame codificate.

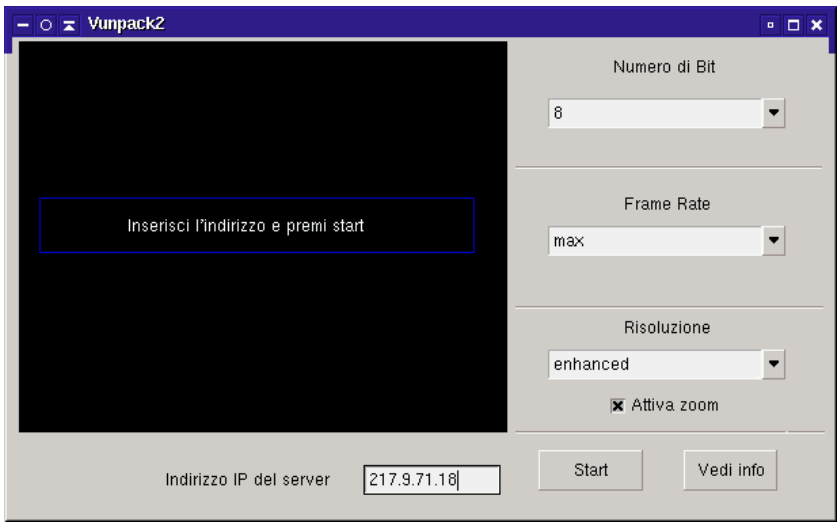


Figura 4.3: L'interfaccia del client

nandola le dimensioni dell'immagine visualizzata vengono raddoppiate, senza però richiedere ulteriori layer codificati, in altre parole attivando questa casella si raddoppiano semplicemente le dimensioni dell'immagine (effettuando anche una semplice interpolazione dei pixel) senza modificare il tasso. Nella tabella 4.1 sono mostrati i valori che i parametri di codifica possono assumere.

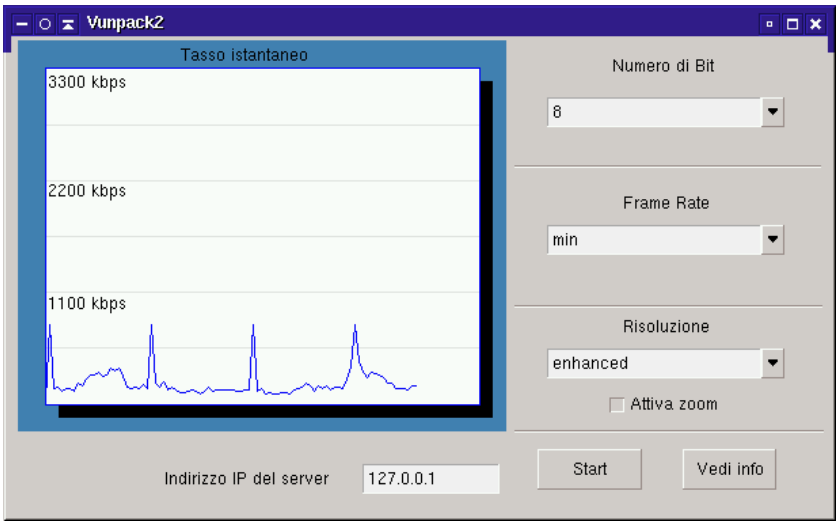
Parametro	Valori	Significato
Numero di bit	8	Vengono trasmessi gli indici completi
	7	Vengono trasmessi solo i primi 7 bit di ogni indice
	6	Vengono trasmessi solo i primi 6 bit di ogni indice
	...	...
Frame Rate	Max	Vengono trasmesse tutte le frame (25 frame/s)
	Med	Viene trasmessa una frame ogni 2
	Min	Viene trasmessa una frame ogni 4
Risoluzione	Base	La risoluzione spaziale è di 180×144 pixel
	Enhanced	La risoluzione spaziale è di 360×288 pixel

Tabella 4.1: I parametri di codifica

Una volta inseriti questi parametri cliccando sul pulsante *Start* si avvia la comunicazione tra client e server. In un primo momento viene stabilita una connessione TCP grazie la quale il client comunica al server i parametri di codifica, il server inizia a codificare la sequenza assecondando i parametri ricevuti e spedisce le frame codificate al client, questa volta però in pacchetti RTP. Il server quindi costruisce

i pacchetti RTP apponendo i giusti timestamp e numeri di sequenza. Il *marker bit* contenuto nell'header di RTP viene usato per segnare le frame con codifica intra.

Il client riceve questi pacchetti ne decodifica il payload e mostra le frame decodificate in una nuova finestra, mentre nella finestra principale viene diagrammato l'andamento del tasso istantaneo impegnato dalla trasmissione.



(a) La finestra principale

(b) La sequenza riprodotta

Figura 4.4: Il client in azione.

Il tasto *Vedi Info*, permette di accedere alle informazioni sull'altro host partecipante alla sessione, in particolare sono mostrate tutte le informazioni trasmesse tramite i pacchetti SDES del protocollo RTCP (vedi figura 4.5). Delle informazioni mostrate solo il CNAME è obbligatorio, quindi possono capitare situazioni nelle quali tutti i campi sono vuoti tranne quello del CNAME.

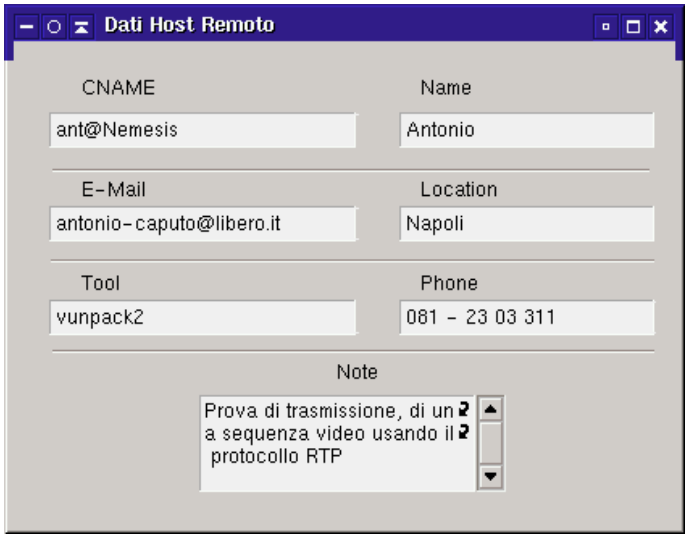


Figura 4.5: Informazioni SDES

Alla fine della trasmissione, nelle console in cui si sono stati lanciati gli eseguibili (il client ed il server) appaiono le informazioni riassuntive sulla sessione. I dati sono estratti dai report ottenuti grazie al protocollo RTCP, per cui mentre nel form di info venivano mostrati i dati SDES, in queste schermate vengono mostrati i dati dei sender report e dei receiver report. In particolare, essendo in questo caso ben distinti il ruolo di trasmettitore e ricevitore, si può vedere come dal lato server siano giunti sono receiver report mentre dal lato client solo sender report. Di seguito si mostra ad esempio un report ottenuto dal lato server.

```
****************************************
*     Informazioni sulla sorgente      *
- - - - - - - - - - - - - - - - - - - -
SSRC :1652890298
Ci sono pacchetti accodati :NO
TimeStamp unit :4e-05
- - - - - - - - - - - - - - - - - - - -
Non ci sono SenderReport
- - - - - - - - - - - - - - - - - - - -
Ci sono ReceiverReport
Fract. lost :0
Packet lost :0
Highest sn :41941
Jitter :12967 tsu
Last SR Timestamp :1515784184
SR Delay :63069
- - - - - - - - - - - - - - - - - - - -
CNAME :ant@Nemesis
- - - - - - - - - - - - - - - - - - - -
Info locali
Pacchetti ricevuti:0
Base seq. num. :0
Highest seq. num. :0
Jitter :0 tsu
- - - - - - - - - - - - - - - - - - - -
RoundTripTime :13606.4 ms
****************************************
```

Mentre dal lato client si è ottenuto il seguente report:

```
****************************************
*     Informazioni sulla sorgente      *
- - - - - - - - - - - - - - - - - - - -
SSRC :1788247824
Ci sono pacchetti accodati :NO
TimeStamp unit :4e-05
- - - - - - - - - - - - - - - - - - - -
Ci sono SenderReport
Last Timestamp :2860123955
Packet Count :541
Byte Count :921959
- - - - - - - - - - - - - - - - - - - -
Non ci sono ReceiverReport
- - - - - - - - - - - - - - - - - - - -
CNAME :ant@Nemesis
- - - - - - - - - - - - - - - - - - - -
Info locali
Pacchetti ricevuti:542
Base seq. num. :41399
Highest seq. num. :41941
Jitter :12967 tsu
****************************************
```

4.2.2 Le tecniche utilizzate

Di seguito vengono illustrate le tecniche che sono state impiegate nella progettazione del client e del server per la gestione della sincronizzazione, il recupero dalle perdite, ecc.

4.2.2.1 La codifica intraframe periodica

Il codec utilizzato prevede due modalità di codifica, la codifica *intraframe*, secondo la quale l'immagine viene codificata esclusivamente mediante la quantizzazione vettoriale, e la codifica *interframe*, che si ottiene applicando il conditional replenish-

ment. La prima frame deve necessariamente essere codificata in modo intra, visto che le successive frame codificate in modo inter la usano come riferimento.

Tuttavia non è sufficiente codificare una sola frame in modalità intra, infatti con l'andare del tempo la qualità della sequenza riprodotta tende a decadere a causa del fatto che, ad esempio, le zone in cui i cambiamenti dell'immagine sono ridotti (lo sfondo), in dipendenza dalla soglia del CR, potranno essere non aggiornate per lunghi periodi.

Per questo motivo si è deciso di produrre periodicamente (ogni 4 secondi, ma questo parametro può essere variato in modo semplice) una frame con codifica intra, che ripristini la qualità massima della riproduzione. In particolare la frame con codifica intra ha sempre un indice del tipo $(k-1)*4+4$ con $k = 25, 50, 75\ldots$. La scelta è ricaduta sulle frame del tipo $(k-1)*4+4$ perché sono le uniche ad essere sempre ricevute, indipendentemente dal livello di scalabilità a cui ci si pone.

A tal proposito in figura 4.6 viene mostrato un esempio dell'andamento del PSNR, dove si può notare come in corrispondenza della ricezione delle frame con codifica intra (sono quelle numerate 50, 100 e 150, in quanto in questo caso il frame rate era quello medio, ovvero la metà di quello massimo) si hanno dei picchi nel valore del PSNR.

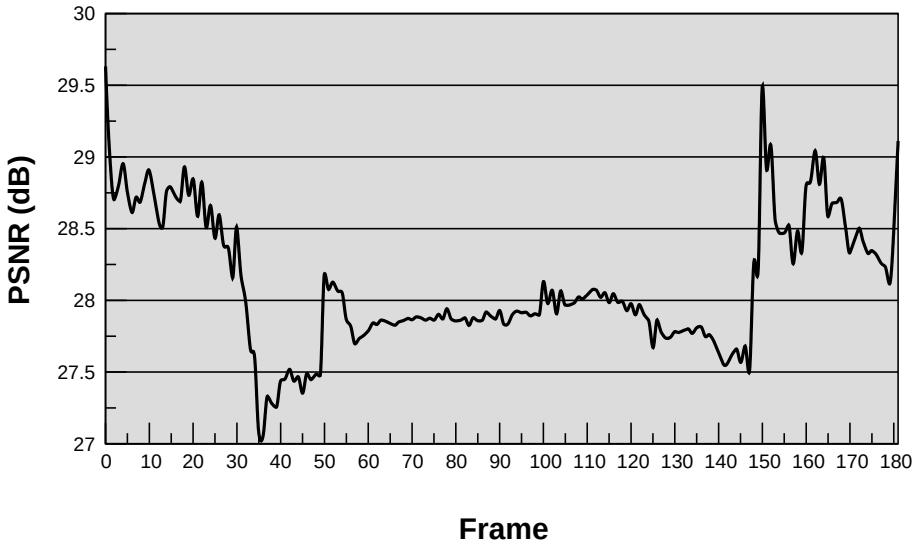


Figura 4.6: Andamento del PSNR.

Avere delle frame con codifica intra inoltre aiuta nella realizzazione di altri due compiti che sono l'*error concealment* e la *sincronizzazione,* che vengono spiegati di

seguito. É chiaro che tutto questo ha un costo, ovvero il tasso risulterà incrementato e soprattutto mostrerà una variabilità notevole (come si può vedere in figura 4.7, dove viene mostrato l'andamento del tasso istantaneo prodotto dal codec nel caso di frame rate minimo risoluzione enhanced e 8 bit per gli indici)

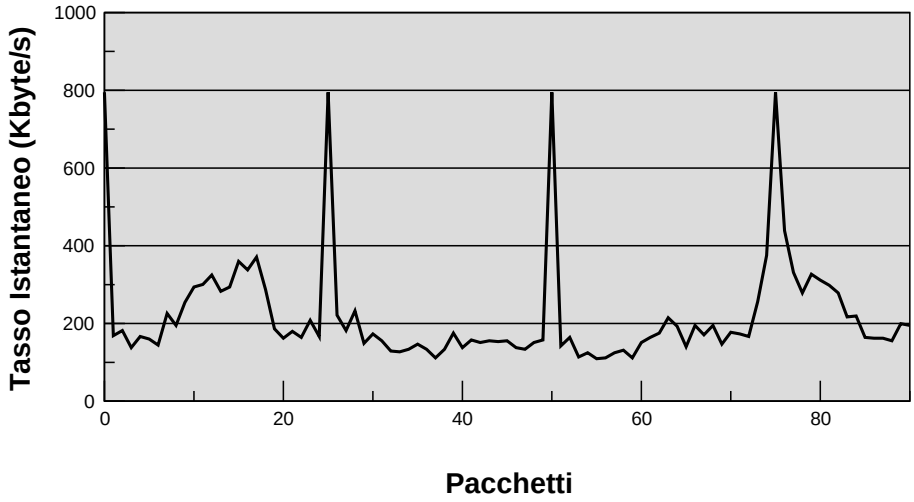


Figura 4.7: Andamento del tasso istantaneo.

4.2.2.2 Error concealment

Per quanto si possa regolare il tasso, inevitabilmente in una sessione si avrà la possibilità di perdere dei pacchetti, e il client deve essere quindi preparato per questa evenienza. Bisogna inoltre considerare che le frame sono codificate secondo un preciso schema sequenziale mostrato in figura 4.8, un'analogo schema ovviamente è deducibile anche per la decodifica.

Il problema sta nel fatto che non tutte le frame sono codificate allo stesso modo, ci sono le frame con codifica intra e quelle con codifica inter, inoltre queste ultime a loro volta si distinguono in frame con CR unidirezionale (le frame del tipo $(k-1)*4+4$ e con $k = 1, 2, 3\ldots$) e quelle con CR bidirezionale (quelle del tipo $(k-1)*4+1$, $(k-1)*4+2$ e $(k-1)*4+3$ con $k = 1, 2, 3\ldots$). Quindi ogni frame va indirizzata al giusto blocco di decodifica.

La tecnica implementata è molto semplice: se indichiamo con *numseq* il numero di sequenza del pacchetto appena ricevuto e con *oldnumseq* quello dell'ultimo pacchetto ricevuto in ordine, bisogna verificare se *numseq = oldnumseq + 1*, nel qual caso non ci sono state perdite e quindi il payload può essere correttamente decodi-

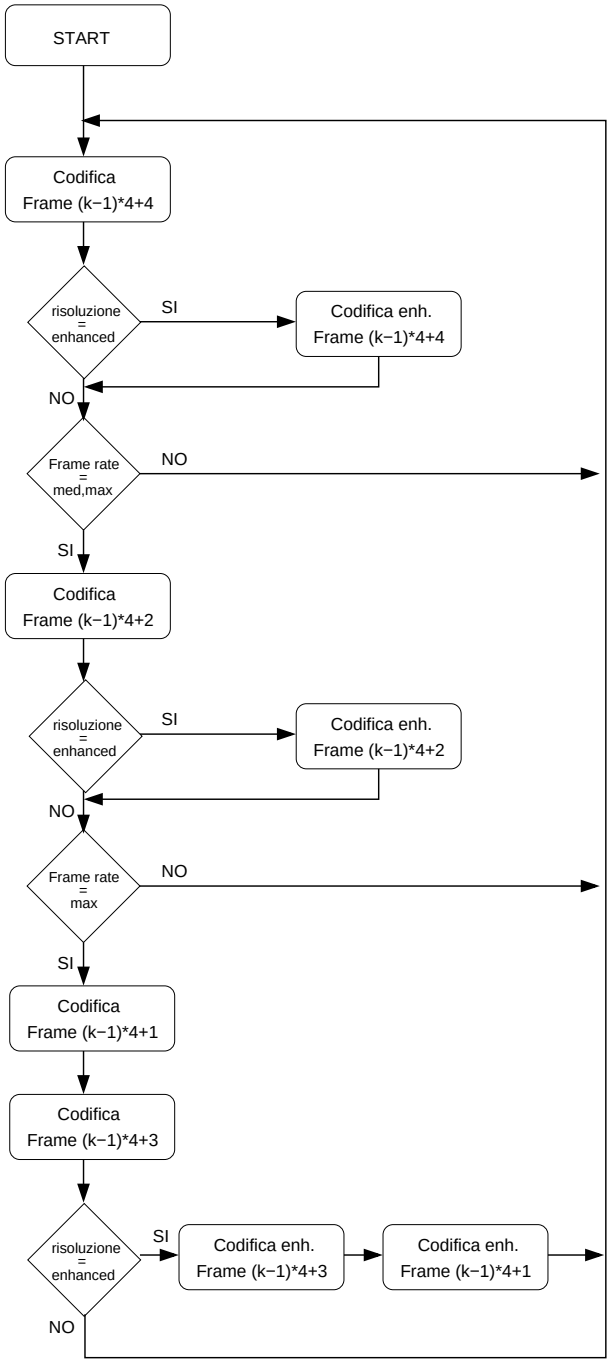


Figura 4.8: Lo schema di codifica

ficato. Se invece la condizione non è verificata vuol dire che c'è stata una perdita oppure è arrivata una frame fuori ordine, quindi il payload non viene decodificato ma semplicemente si utilizza come frame l'ultima decodificata con successo. A questo punto si avanza di un unità oldnumseq e ci si sposta sul successivo blocco di decodifica, se era arrivata una frame fuori ordine questa viene scartata e si riceve una frame nuova, si ripete il controllo e si continua in questo modo finché il controllo non risulta verificato, il che significa che la frame può essere correttamente decodificata.

Bisogna osservare che per la decodifica delle frame con CR si deve ricorrere alle frame precedenti (alcuni blocchi vengono sostituiti con quelli di queste frame di riferimento), questo implica che quando si verifica una perdita verranno usate come frame di riferimento delle frame sbagliate e quindi, anche avendo ricevuto correttamente un nuovo pacchetto, la frame decodificata sarà caratterizzata da una scarsa qualità. Tuttavia proprio grazie al funzionamento del conditional replenishment questo effetto tende ad attenuarsi se si ricevono nuove frame, infatti le zone in cui si è avuta attività sono presenti nella frame codificata e non richiedono il riferimento e quindi la loro decodifica è corretta, l'errore persiste in quelle zone in cui l'attività non è così forte da richiedere la trasmissione del blocchetto codificato.

Questo porta ad una sorta di *effetto memoria*, cioè anche se si continua a ricevere correttamente i pacchetti alcune zone resteranno "sporche". Però c'è anche da dire che non appena sarà ricevuta una frame con codifica intra questo errore sarà azzerato in quanto tutti i blocchetti verranno codificati. Da una parte quindi aumentare il periodo di trasmissione delle frame intra consente di contenere il tasso di trasmissione, dall'altra però ridurre tale periodo consente di avere una qualità media superiore.

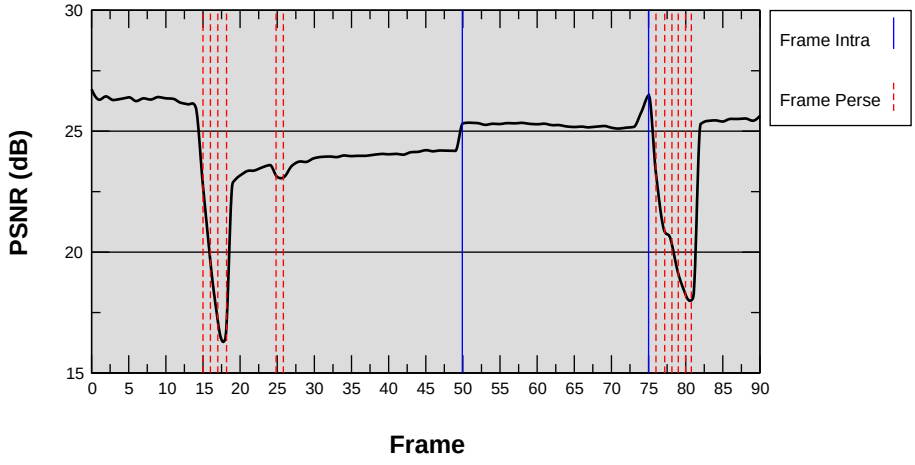


Figura 4.9: Andamento del PSNR nel caso di perdita di pacchetti.

La figura 4.9 mostra una situazione in cui si è verificata una perdita di pacchetti (questa evenienza è evidenziata dalle linee verticali tratteggiate), è evidente la caduta verticale del PSNR in questi casi, tuttavia le cose migliorano nettamente alla ricezione di frame con codifica intra (linee verticali continue). Tuttavia, si può notare che buona parte della qualità viene recuperata anche senza la ricezione di frame con codifica intra, questo è probabilmente imputabile al funzionamento stesso del conditional replenishment, come accennavamo sopra. Le zone in cui c'è una forte attività sono quelle sulle quali il CR fallisce e che quindi devono essere trasmesse, la loro ricezione permette di annullare i disturbi dovuti alle perdite, almeno nelle zone in movimento. Si può quindi affermare che già il solo conditional replenishment fornisce una certa robustezza nei confronti delle perdite, le zone in cui l'attività è bassa (laddove quindi il CR ha successo) resteranno però corrotte, il loro corretto ripristino è possibile solo ricevendo una frame con codifica intra, che quindi permette di ristabilire pienamente la qualità della riproduzione.

4.2.2.3 La sincronizzazione

Nel prototipo unicast la trasmissione inizia quando il server riceve un opportuno segnale di start da parte del client, quindi si potrebbe pensare che i due host siano già sincronizzati, in realtà ci sono almeno due situazioni in cui questo non è vero.

- Il primo pacchetto contenente la frame con codifica intra può andare perduto, in questo caso il client non può iniziare la decodifica dei successivi pacchetti in quanto necessita di un riferimento iniziale.
- Il server è di tipo concorrente, quindi è in grado di soddisfare le richieste di più host contemporaneamente, cosa accade allora se un host richiede la trasmissione quando la sequenza è stata già avviata? Anche questo host avrà bisogno di una frame di riferimento iniziale.

É chiaro che in entrambi i casi la soluzione più naturale consiste nell'attendere finché non viene ricevuta una frame con codifica intra. Quindi il client è stato progettato in modo tale che quando inizia la ricezione si mette in attesa di una frame intra, quindi riceve i pacchetti e va a controllare il marker bit, se il marker bit non è settato il pacchetto viene scartato mentre se risulta settato, il payload viene decodificato e si entra nel ciclo di decodifica.

Anche in questo caso quindi risulta determinante il fatto di spedire periodicamente una frame con codifica intra; risulta inoltre evidente che quanto minore è il

periodo di trasmissione di queste frame tanto meno tempo sarà necessario per la sincronizzazione, ma allo stesso tempo tanto più alto sarà il tasso di trasmissione.

4.2.3 Le prove

Come visto in precedenza, il prototipo unicast permette di scegliere praticamente qualsiasi combinazione possibile dei parametri di codifica, questo ci ha permesso di sottoporlo ad una serie di prove sul campo per cercare di capire i modi migliori per sfruttare l'algoritmo di codifica.

Il codec è dotato di tre diversi tipi di scalabilità. Questo lo rende, in teoria, adattabile a diversi tipi di linea, infatti, note le caratteristiche di banda del collegamento sarà possibile scegliere il valore dei parametri caratteristici al fine di ottenere il bit-rate più adeguato.

In una prima serie di prove è stato semplicemente valutato l'andamento del tasso istantaneo del PSNR e il valore del tasso medio al variare dei parametri di codifica. In questo caso le prove sono state effettuate trasmettendo su un collegamento con la capacità di 2 Mbit/s.

In una seconda serie di prove invece si è agito in modo differente, il sistema è stato testato su una linea della quale era possibile variare la capacità, e quindi per diversi valori fissati si è cercato di determinare quali valori dei parametri di codifica dessero il migliore compromesso in termini tasso e qualità, in questo caso è stato rilevato anche il numero di pacchetti persi.

La configurazione utilizzata è mostrata in figura 4.10.

Sul computer `altair.router-test.cnit.it` è stato installato il server, mentre su `corcaroli.labnet.cnit.it` il client. Il percorso che collega le due macchine passa attraverso gli switch LAN, i router e lo switch ATM. Come si vede dalla figura quest'ultimo è collegato ai router mediante delle linee seriali a 2Mbit/s. Il protocollo di comunicazione usato su questi collegamenti è in realtà Frame Relay, infatti lo switch ATM è in grado di gestire tanto Frame Relay che ATM (salvo poi effettuare internamente comunque una conversione verso ATM).

Tramite il software di controllo dello switch è possibile modificare le caratteristiche del collegamento, in particolare è possibile fissare il bit-rate[4]. Partendo quindi da un massimo di 2053 kbit/s (4844 celle/s) è possibile diminuire tale valore fino ad arrivare a 73 kbit/s (173 celle/s) con passi di 73 kbit/s (173 celle/s).

[4]in realtà quello che si fissa è il cell-rate in quanto, come dicevamo in precedenza, internamente lo switch lavora su celle ATM.

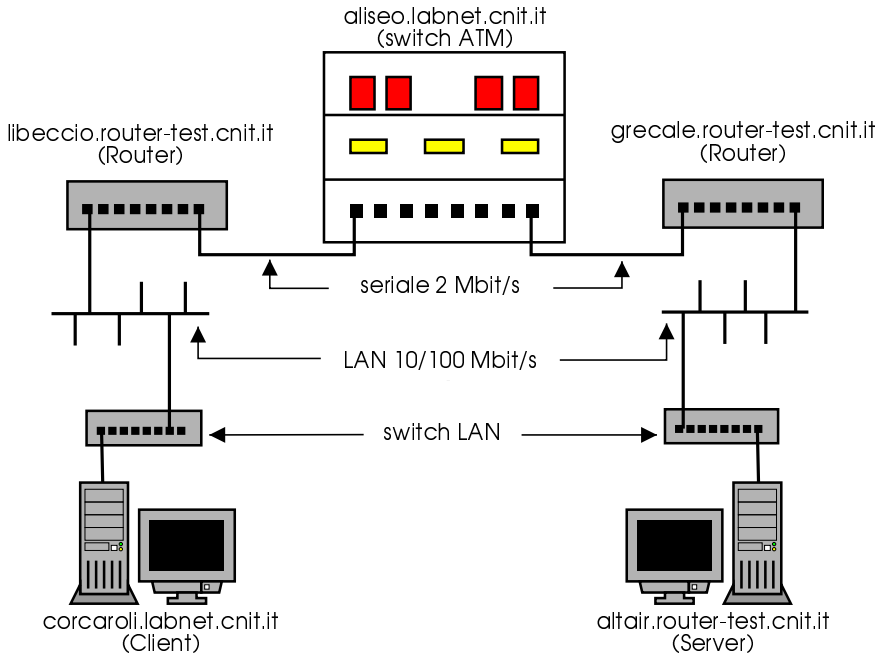


Figura 4.10: L'attrezzatura utilizzata per le prove

La riduzione della capacità del collegamento è resa possibile dal fatto che internamente lo switch ATM scarterà alcune celle (nel caso in cui il flusso di ingresso dovesse avere un bit-rate superiore a quello fissato) abbassando di fatto il bit-rate in uscita. E' chiaro che se il flusso ha un tasso medio al disotto del limite, ma presenta dei picchi nel bit-rate che superano quest'ultimo, subirà comunque delle perdite in quanto ovviamente il controllo viene fatto sul bit-rate istantaneo.

Agendo sui router è invece possibile cambiare la MTU (*Maximum Transmission Unit*), cioè la dimensione massima di un pacchetto IP che viaggia su quel link. Quindi se un pacchetto IP dovesse avere dimensione maggiore della MTU verrà frammentato in tanti pacchetti IP[5] ciascuno di dimensione al massimo pari alla MTU. Qualora anche uno solo di questi "frammenti" si dovesse perdere, tutto il pacchetto IP iniziale andrà perso (è noto che IP è un protocollo best-effort, non gestisce le ritrasmissioni). Questo fenomeno a parità di capacità del collegamento può influenzare il numero di pacchetti persi end-to-end, e quindi è da tenere in conto. Tuttavia per non introdurre troppe variabile si è preferito mantenere questo parametro fisso al valori di 1500 byte.

[5] A meno che il pacchetto non abbia il flag "Don't Fragment" settato.

4.2.3.1 Le prove a capacità fissa

In questo caso si è cercato di studiare il comportamento del codificatore al variare dei parametri di codifica. La capacità della linea è stata fissata al valore massimo possibile e cioè 2053 kbit/s ($\simeq$ 2 Mbit/s). Per ogni coppia di valori [frame rate , risoluzione] si è fatto variare il numero di bit usato per gli indici tra 4 e 8 bit. Utilizzare meno di 4 bit produce dei risultati molto scadenti da un punto di vista qualitativo e per questo tali combinazioni non sono state prese in considerazione.

Per ogni possibile combinazione sono stati calcolati il tasso istantaneo (medio, minimo, massimo e deviazione standard) il PSNR (medio, minimo, massimo e deviazione standard) e l'*overhead* (percentuale dello spazio del pacchetto occupato dagli header) medio dovuto alla pacchettizzazione. I primi due valori sono di chiara interpretazione, di seguito invece viene spiegato in che modo è stato calcolato l'overhead.

Le frame codificate, prima di essere spedite a destinazione, vengono inserite in un pacchetto RTP dal livello applicazione, al livello trasporto tale pacchetto viene incapsulato in uno UDP che a sua volta è incapsulato in un pacchetto IP al livello rete. Ogni livello quindi aggiunge un proprio header al pacchetto producendo quindi un incremento dell'overhead. L'header di RTP (nel caso in cui ci sia un'unica sorgente, cioè nel caso in cui il pacchetto non sia prodotto da un mixer) è costituito da 12 byte, quello di UDP da 8 byte, e quello di IP da 20 byte. Quindi saremmo portati a pensare che per ogni pacchetto c'è un overhead di 12+8+20 = 40 byte. In realtà non è così semplice, infatti bisogna tener conto anche del fatto che a livello IP può avvenire una frammentazione del pacchetto se quest'ultimo ha dimensioni maggiori della MTU.

In questo caso avremo il primo frammento che avrà un header RTP uno UDP ed uno IP, mentre i successivi frammenti avranno solo l'header IP[6], la figura 4.11 mostra un esempio. La somma dei payload degli N pacchetti restituisce il payload originario.

É chiaro quindi che solo il primo frammento avrà un overhead di 40 byte mentre gli altri avranno un overhead di 20 byte. Quindi per valutare l'overhead (in percentuale sul numero totale di byte ricevuti) medio sui pacchetti ricevuti si è usata la seguente espressione:

[6]Si noti che in questi discorsi si sta trascurando l'eventuale header introdotto a livello Data Link, a causa del fatto che non possiamo sapere a priori quale tipo di protocollo verrà utilizzato.

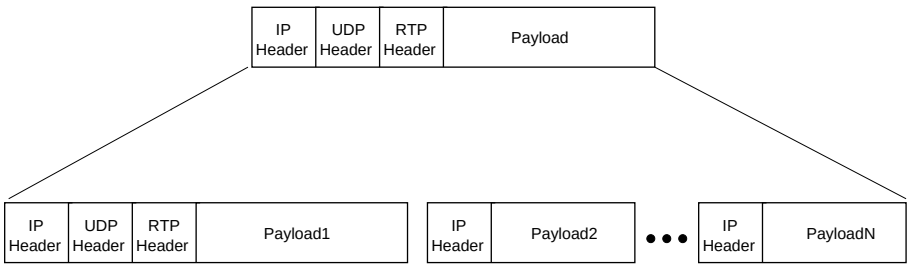


Figura 4.11: Esempio di frammentazione di un pacchetto.

$$\text{Overhead} = 100 * \frac{\left[\left\lfloor \frac{dim\,media}{MTU} \right\rfloor * 20 + 40\right] * num\,pacchetti}{somma\,paylod + \left[\left\lfloor \frac{dim\,media}{MTU} \right\rfloor * 20 + 40\right] * num\,pacchetti}$$

Tenere conto della frammentazione fa sì che l'andamento dell'overhead non sia necessariamente monotono con le dimensioni del pacchetto (e quindi ad esempio con il numero di bit) come ci si potrebbe aspettare.

Risoluzione livello *base* I dati relativi al PSNR, ottenuti a risoluzione base sono riportati nelle tabelle 4.2, 4.3, 4.4.

Bit	PSNR Medio (dB)	PSNR Min (dB)	PSNR Max (dB)	PSNR DevStd (dB)
8	24.964	24.312	25.924	0.574
7	24.187	23.590	25.168	0.566
6	23.651	23.074	24.572	0.543
5	23.063	22.526	23.980	0.549
4	21.735	21.238	21.601	0.562

Tabella 4.2: PSNR a livello base e frame rate massimo.

Bit	PSNR Medio (dB)	PSNR Min (dB)	PSNR Max (dB)	PSNR DevStd (dB)
8	24.969	24.312	25.920	0.577
7	24.192	23.590	25.156	0.572
6	23.655	23.074	24.572	0.545
5	23.065	22.526	23.975	0.551
4	21.738	21.238	21.601	0.564

Tabella 4.3: PSNR a livello base e frame rate medio.

Bit	PSNR Medio (dB)	PSNR Min (dB)	PSNR Max (dB)	PSNR DevStd (dB)
8	24.982	24.349	25.920	0.585
7	24.204	23.625	25.156	0.579
6	23.666	23.124	24.572	0.549
5	23.075	22.555	23.975	0.560
4	21.745	21.252	21.601	0.567

Tabella 4.4: PSNR a livello base e frame rate minimo.

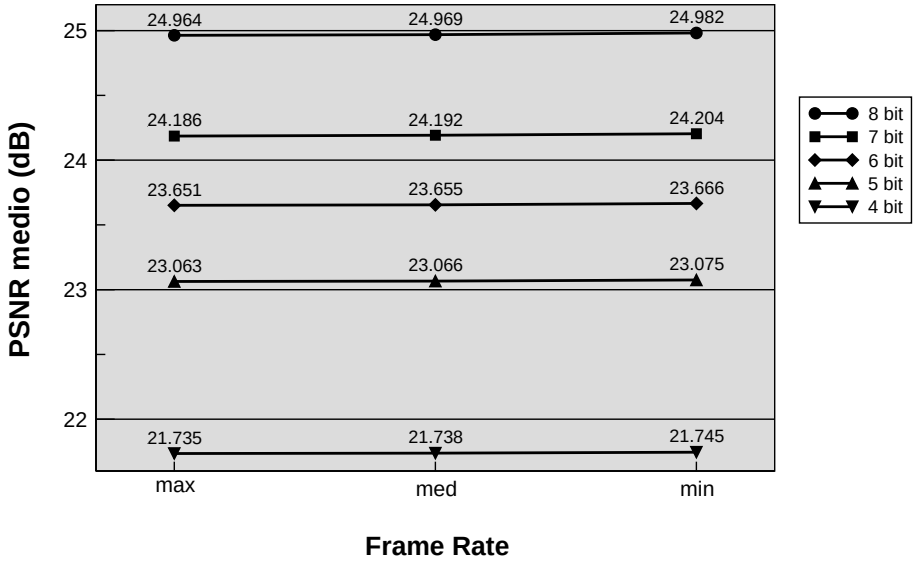


Figura 4.12: PSNR medio a livello base

In figura 4.12 sono riassunti i dati relativi all'andamento del PSNR medio nel caso di risoluzione base. Si può notare che il PSNR risulta ovviamente decrescente al diminuire del numero di bit utilizzati per gli indici, inoltre il passaggio da 5 bit a 4 bit porta un peggioramento più sensibile rispetto agli altri casi.

É anche interessante notare che il PSNR, anche se lievissimamente, risulta crescente al decrescere del frame rate, questo fenomeno è dovuto probabilmente al fatto che l'utilizzo del CR bidirezionale produce immagini ricostruite di qualità peggiore rispetto a quelle ricostruite con il CR unidirezionale (ovviamente però si ha un guadagno in termini di fattore di compressione). Infatti a frame rate *min* si utilizza un unico layer temporale, e le immagini sono tutte codificate con CR unidirezionale, il passaggio a frame rate *med* e *max* lo si ottiene aggiungendo gli altri due layer temporali, in questi layer si utilizza esclusivamente il CR bidirezionale. Oltre a questo va anche tenuto in conto che, a frame rate min, la percentuale di frame con codifica intra è maggiore.

Nelle tabelle 4.5, 4.6 e 4.7 sono riportati i dati relativi al valore del tasso rispettivamente nel caso di frame rate massimo, medio e minimo. Si può notare che il traffico prodotto ha una notevole variabilità, e quest'ultima risulta crescente con il tasso come si può capire dall'andamento della deviazione standard.

Bit	Tasso Medio (kbit/s)	Tasso Min (kbit/s)	Tasso Max (kbit/s)	Tasso DevStd (kbit/s)
8	117.219	35.352	640.625	84.978
7	104.627	32.617	561.523	73.319
6	92.009	29.785	482.422	63.649
5	79.397	27.051	403.320	52.986
4	66.770	24.316	342.219	42.312

Tabella 4.5: Tasso istantaneo a livello base e frame rate massimo.

Bit	Tasso Medio (kbit/s)	Tasso Min (kbit/s)	Tasso Max (kbit/s)	Tasso DevStd (kbit/s)
8	69.938	25.781	320.312	47.473
7	62.298	23.633	280.762	41.459
6	54.637	21.582	241.211	35.442
5	46.982	19.434	201.660	29.428
4	39.314	17.383	162.109	23.409

Tabella 4.6: Tasso istantaneo a livello base e frame rate medio.

Bit	Tasso Medio (kbit/s)	Tasso Min (kbit/s)	Tasso Max (kbit/s)	Tasso DevStd (kbit/s)
8	45.560	23.047	160.15	28.308
7	40.391	20.703	140.381	24.717
6	35.199	18.359	120.605	21.131
5	30.148	15.967	100.830	17.543
4	24.816	13.623	81.055	13.958

Tabella 4.7: Tasso istantaneo a livello base e frame rate minimo.

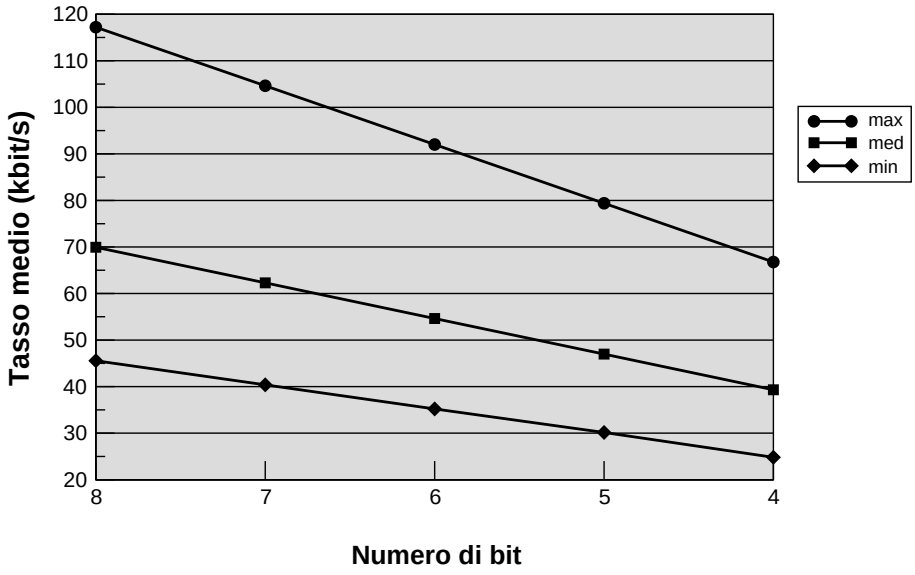


Figura 4.13: Tasso medio a livello base

I dati relativi al tasso medio per i tre valori di frame rate sono riassunti nella figura 4.13, questa mostra che ovviamente al diminuire del numero di bit il tasso medio decresce, tuttavia si può anche notare che la pendenza della curva è più marcata nel caso di frame rate massimo che non negli altri due casi. In altre parole, la variazione percentuale del tasso medio al variare del numero di bit è più piccola nel caso di frame rate medio o minimo, questo rende di fatto poco conveniente agire sul numero di bit in tale situazione

Per concludere l'analisi delle prestazioni a livello base riportiamo i dati relativi all'overhead nelle tabelle 4.8, 4.9 e 4.10. L'overhead risulta essere crescente al

Bit	Overhead (%)
8	5.540
7	6.199
6	7.036
5	8.135
4	9.642

Tabella 4.8: Overhead a livello base e frame rate massimo

Bit	Overhead (%)
8	5.585
7	6.270
6	7.249
5	8.314
4	9.936

Tabella 4.9: Overhead a livello base e frame rate medio

Bit	Overhead (%)
8	4.287
7	4.836
6	5.549
5	6.507
4	7.870

Tabella 4.10: Overhead a livello base e frame rate minimo

decrescere del numero di bit, così come era lecito aspettarsi. In particolare si può notare come il valore tenda a raddoppiare circa, passando da 8 bit a 4 bit. Il valore massimo, pari al 9.936 %, lo si ottiene nel caso di frame rate medio e indici da 4 bit,

tale valore è discretamente alto e utilizzare pacchetti di dimensioni ridotte potrebbe portare a valori di overhead troppo alti.

I dati dell'overhead sono riassunti in via grafica nella figura 4.14.

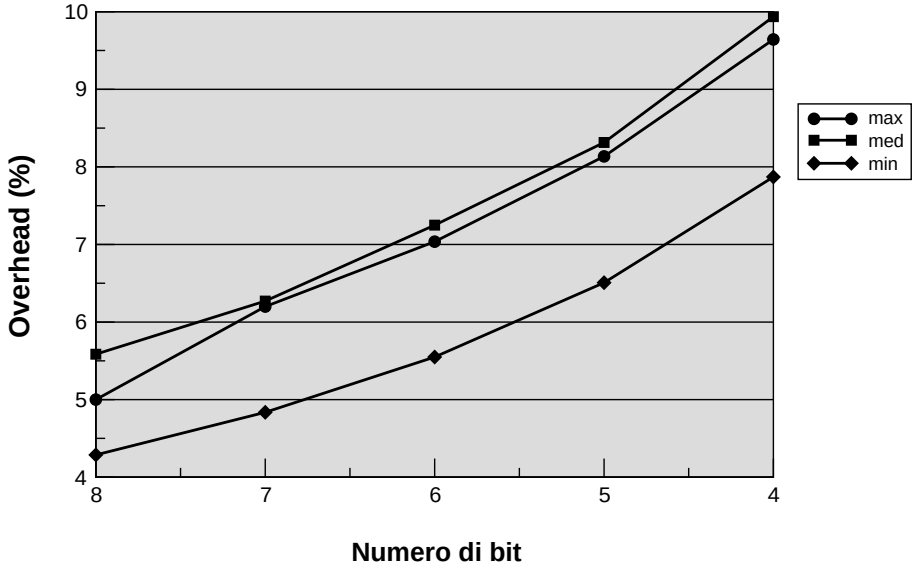


Figura 4.14: Overhead a livello base

Risoluzione livello *enhanced* Di seguito vengono analizzati i dati relativi al livello di risoluzione *enhanced.*

Le tabelle 4.11, 4.12 e 4.13 riportano i valori assunti dal PSNR per i tre valori possibili del frame rate.

Bit	PSNR Medio (dB)	PSNR Min (dB)	PSNR Max (dB)	PSNR DevStd (dB)
8	28.078	27.058	29.632	0.447
7	26.688	25.894	28.047	0.501
6	25.570	24.878	26.732	0.506
5	24.461	23.845	25.533	0.520
4	22.739	22.157	23.622	0.550

Tabella 4.11: PSNR a livello enhanced e frame rate massimo.

Sostanzialmente vengono confermate le osservazioni già fatte per il livello di risoluzione base. Chiaramente in questo caso i valori del PSNR sono mediamente più alti, tuttavia possiamo notare che mentre a risoluzione base, passando da 8 bit a 4 bit si aveva una riduzione del PSNR medio di circa 3 dB, in questo caso mediamente

Bit	PSNR Medio (dB)	PSNR Min (dB)	PSNR Max (dB)	PSNR DevStd (dB)
8	28.098	27.058	29.632	0.462
7	26.703	25.894	28.047	0.511
6	25.579	24.875	26.732	0.511
5	24.467	23.845	25.533	0.524
4	22.743	22.160	23.672	0.551

Tabella 4.12: PSNR a livello enhanced e frame rate medio.

Bit	PSNR Medio (dB)	PSNR Min (dB)	PSNR Max (dB)	PSNR DevStd (dB)
8	28.144	27.058	29.632	0.489
7	26.735	25.917	28.047	0.530
6	25.603	24.882	26.732	0.525
5	24.484	23.871	25.533	0.536
4	22.753	22.160	23.672	0.560

Tabella 4.13: PSNR a livello enhanced e frame rate minimo.

si assiste ad una riduzione di 6 dB. Quindi le immagini a livello enhanced risultano essere più sensibili alla variazione del numero di bit utilizzati per gli indici. La figura 4.15 riassume i dati relativi al PSNR medio. Anche le osservazioni fatte a proposito

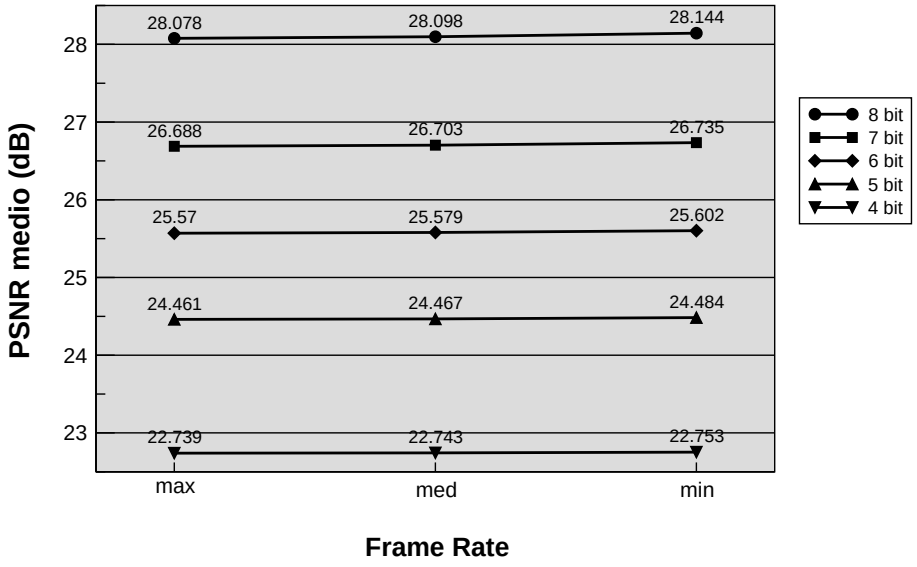


Figura 4.15: PSNR a livello enhanced.

del andamento del tasso sono sostanzialmente confermate per il livello di risoluzione enhanced come mostrano le tabelle 4.14, 4.15 e 4.16 e la figura 4.16. Proprio dalla

figura tuttavia si può dedurre che la pendenza delle curve risulta essere leggermente inferiore rispetto a quello osservato nel caso di livello di risoluzione base, cioè a risoluzione enhanced il tasso medio risulta essere meno influenzato dalle variazioni del numero di bit usati per gli indici.

Bit	Tasso Medio (kbit/s)	Tasso Min (kbit/s)	Tasso Max (kbit/s)	Tasso DevStd (kbit/s)
8	566.409	164.746	3179.690	422.879
7	503.365	151.074	2784.182	369.533
6	440.271	137.207	2388.671	316.165
5	377.208	123.535	1993.166	262.817
4	317.104	109.766	1597.66	209.451

Tabella 4.14: Tasso istantaneo a livello enhanced e frame rate massimo.

Bit	Tasso Medio (kbit/s)	Tasso Min (kbit/s)	Tasso Max (kbit/s)	Tasso DevStd (kbit/s)
8	337.904	116.895	1589.841	237.391
7	299.638	106.348	1392.098	207.313
6	261.333	95.996	1194.34	177.229
5	223.051	85.449	996.582	147.151
4	184.739	75.098	798.828	117.067

Tabella 4.15: Tasso istantaneo a livello enhanced e frame rate medio.

Bit	Tasso Medio (kbit/s)	Tasso Min (kbit/s)	Tasso Max (kbit/s)	Tasso DevStd (kbit/s)
8	221.943	109.375	794.922	141.538
7	196.029	97.607	696.045	123.597
6	170.074	85.791	597.168	105.663
5	144.146	73.975	498.291	87.725
4	118.185	62.158	399.414	69.793

Tabella 4.16: Tasso istantaneo a livello enhanced e frame rate minimo.

Per concludere vengono mostrati i dati relativi all'overhead (tabelle 4.17, 4.18 e 4.19). Dalla lettura dei dati si nota un differente andamento rispetto a quello osservato a livello base, questo comportamento è ancora più evidente osservando la figura 4.17.

Come si può vedere, l'overhead non è monotono con il numero di bit, tuttavia questa possibilità era stata anticipata quando è stata introdotta la modalità di

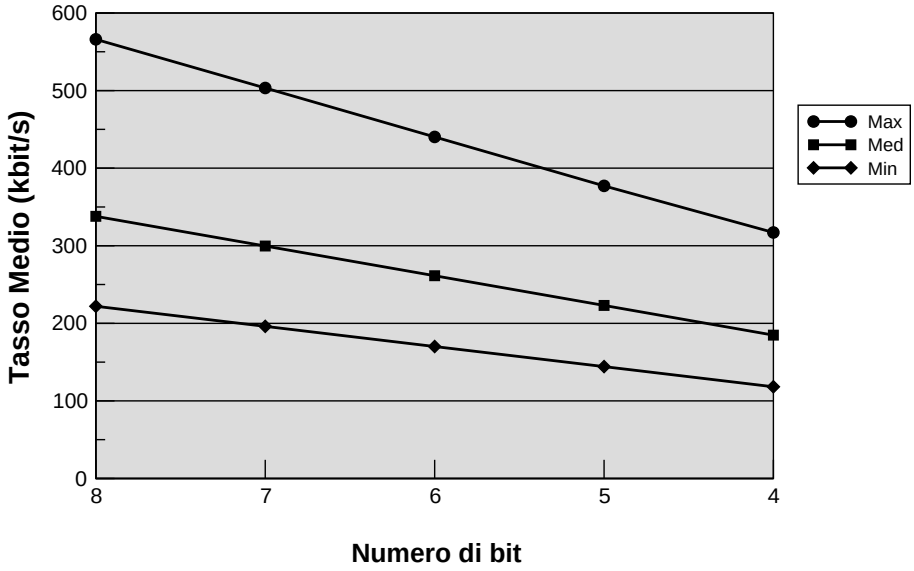


Figura 4.16: Tasso medio a livello enhanced

Bit	Overhead (%)
8	3.400
7	3.815
6	2.941
5	3.423
4	4.096

Tabella 4.17: Overhead a livello enhanced e frame rate massimo

Bit	Overhead (%)
8	3.428
7	2.607
6	3.043
5	3.503
4	4.229

Tabella 4.18: Overhead a livello enhanced e frame rate medio

Bit	Overhead (%)
8	2.617
7	2.960
6	3.406
5	2.710
4	3.305

Tabella 4.19: Overhead a livello enhanced e frame rate minimo

calcolo dell'overhead. Questo situazione si è manifestata solo a livello enhanced a causa delle maggiori dimensioni dei pacchetti, che quindi sono più soggetti alla frammentazione operata da IP.

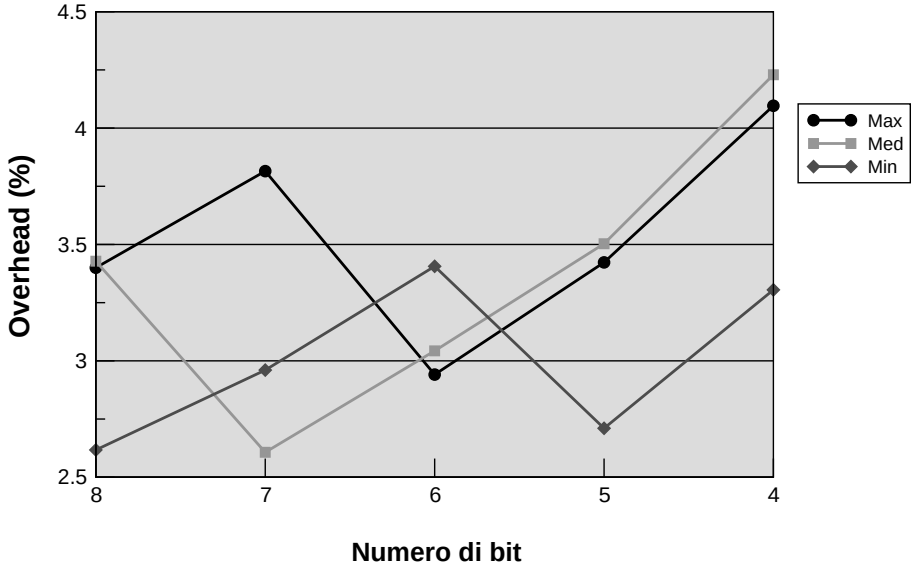


Figura 4.17: Overhead a livello enhanced

4.2.3.2 Le prove a capacità variabile

In questa seconda serie di prove si è cercato di stabilire quale fosse il modo migliore di agire sui parametri di codifica al fine di adattare il tasso alla capacità della linea.

Sono stati adottati cinque diversi valori di capacità corrispondenti ai valori disponibili su alcune reti di accesso comuni (tabella 4.20)

Capacità	Rete d'accesso
1 Mbit/s	XDSL
512 kbit/s	ADSL
256 kbit/s	ADSL
128 kbit/s	ISDN
64 kbit/s	ISDN

Tabella 4.20: I diversi valori utilizzati per la capacità della linea

Le prove hanno evidenziato che anche nelle situazioni in cui il tasso medio prodotto dal codec è molto inferiore alla capacità della linea si possono avere delle

perdite, soprattutto a causa dell'elevata *burstyness* causata dalle frame con codifica intra.

Inoltre si è potuto osservare che agire sul numero di bit è poco efficace in quanto la riduzione del tasso è tutto sommato contenuta. Molto più efficace risulta invece agire sul frame rate, considerando anche che il PSNR medio in questo modo risulta praticamente invariata, è ovvio che la scelta dipende anche dall'applicazione a cui ci rivolgiamo, nel caso di videoconferenza anche il frame rate minimo risulta più che sufficiente per avere una buona comprensibilità della sequenza, se invece si vuole riprodurre una sequenza più dinamica probabilmente avere un frame rate basso è troppo limitante.

Per ogni valore della capacità della linea si è cercato di determinare una o più combinazioni che ottimizzassero la trasmissione, ovvero che minimizzassero le perdite tenendo conto anche della qualità della sequenza. La qualità (spaziale) delle singole frame può essere valutata in modo oggettivo tramite il PSNR, ed il PSNR medio rappresenta quindi efficacemente la sola qualità spaziale, non tiene in conto in alcun modo della qualità temporale della sequenza, ovvero della diversa gradevolezza che si può riscontrare aumentando o riducendo il frame rate. Il modo migliore per valutare questo qualità è attraverso parametri soggettivi, e questo è quanto è stato fatto anche se nella tabella 4.21, dove vengono riassunti i risultati, viene riportato esclusivamente il PSNR medio. In ogni scelta si è cercato di dare un peso eguale tanto al valore del PSNR (quindi misura oggettiva) che alla gradevolezza riscontrata nella riproduzione (misura soggettiva).

Capacità	Combinazione scelta	Tasso Medio (kbit/s)	PSNR medio (dB)
1 Mbit/s	max, enhanced, 6bit	440.271	25.570
	med, enhanced, 8bit	337.904	28.098
512 kbit/s	med, enhanced, 6bit	261.333	25.579
	min, enhanced, 8bit	221.943	28.144
256 kbit/s	min, enhanced, 4bit	118,185	22.753
	max, base, 8bit	117.219	24.964
128 kbit/s	max, base, 4bit	66.702	21.735
	med, base, 8bit	69.9375	24.969
64 kbit/s	med, base, 4bit	39.314	21.738
	min, base, 8bit	45.5604	24.982

Tabella 4.21: Dati riassuntivi delle prove a capacità variabile

Si può vedere che le due scelte effettuate per la capacità di 256 kbit/s sono caratterizzate da una diversa risoluzione spaziale e quindi apparentemente non con-

frontabili. In realtà è stata già discussa la possibilità di attivare una modalità zoom, grazie alla quale un'immagine di livello base viene sovracampionata raggiungendo le dimensioni di un'immagine di livello enhanced. Visivamente si è riscontrato che la qualità della sequenza riprodotta a livello base ma con lo zoom era leggermente migliore di quella a livello enhanced, il tasso trasmissivo è praticamente lo stesso ma bisogna osservare che nella seconda combinazione il frame rate è quello massimo. L'immagine di livello base "zoomata" mostrava sicuramente un dettaglio inferiore a quella di livello enhanced, quest'ultima tuttavia risultava più "sporca" a causa del basso numero di bit utilizzati per gli indici, alla fine questo rende più gradevole alla vista l'immagine di livello base che non quella di livello enhanced.

Per confermare ulteriormente queste sensazioni si è misurato il PSNR della sequenza zoomata (in confronto all'immagine originale di 360×288 pixel) per poterlo confrontare con quello dell'immagine di livello enhanced. Per garantire le stesse condizioni questo calcolo è stato effettuato utilizzando il frame rate minimo (tabella 4.22).

Combinazione	PSNR medio (dB)	Tasso medio (kbit/s)
min, enhanced, 4bit	22.753	118.185
min, base(zoom), 8bit	23.038	45.560

Tabella 4.22: Il livello base con lo zoom

I dati mostrano che, anche se di poco, il PSNR medio è a favore dell'immagine di livello base, ma soprattutto è interessante notare il valore del tasso medio, la proporzione è quasi di 3 a 1. In figura 4.18 sono mostrate le due frame finali ottenute per le due diverse combinazioni.

4.3 Il prototipo multicast

Sfruttando l'esperienza acquisita nella programmazione del prototipo unicast è stato possibile costruire con il minimo sforzo anche un prototipo multicast. Le due applicazioni condividono molte soluzioni e si differenziano ovviamente soprattutto nella modalità di spedizione e ricezione dei pacchetti.

Sulla base di quanto riscontrato nelle prove a cui è stato sottoposto il prototipo unicast, e anche per motivi di semplicità, si è deciso (almeno per il momento) di non sfruttare la scalabilità in precisione, e quindi di supportare unicamente la scalabilità in risoluzione spaziale e temporale.

(a) Livello enhanced, 4 bit

(b) Livello base con zoom, 8 bit

Figura 4.18: Due frame codificate a basso bit-rate a confronto

4.3.1 Le tecniche utilizzate

Le tecniche già viste per il prototipo unicast vengono riproposte anche per il prototipo multicast, quindi anche in questo caso ritroviamo l'utilizzo periodico di una frame intra, l'error concealment e la sincronizzazione. Tuttavia in questo caso la tecnica più importante è senza dubbio la MMG; il modo in cui tale tecnica è stata implementata in questo prototipo è spiegato nel paragrafo seguente.

4.3.1.1 La tecnica Multiple Multicast Group (MMG)

Il flusso codificato viene diviso in sei diversi layer così come mostrato dalla figura 4.19.

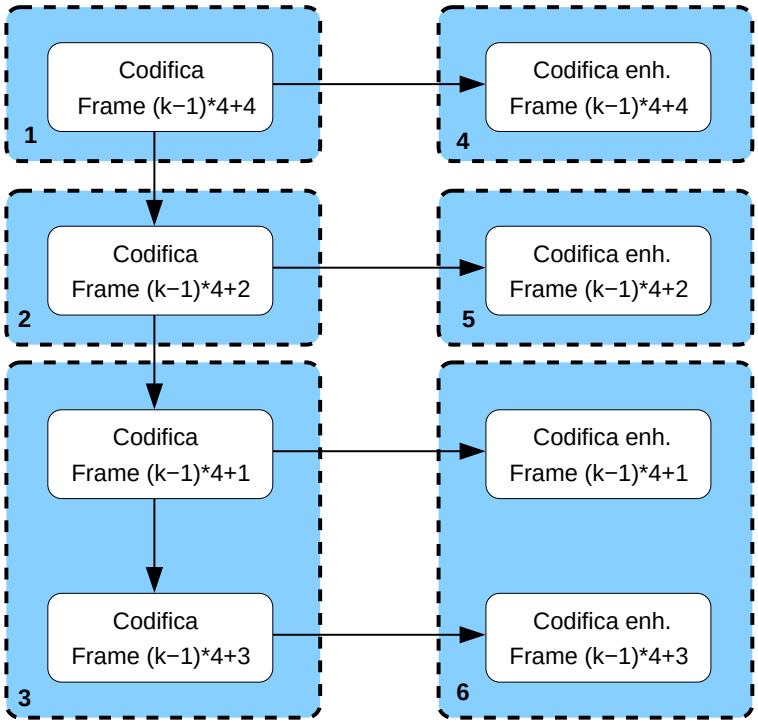


Figura 4.19: La suddivisione in layer

Le frecce servono per sottolineare in che modo si susseguono le varie operazioni di codifica. I blocchi scuri, che rappresentano i sei layer, racchiudono al loro interno i blocchi di codifica che producono il bit-stream dei vari layer. La situazione si può riassumere quindi in questo modo:

Layer 1: Frame $(k-1)*4+4$ di risoluzione base.

Layer 2: Frame $(k-1)*4+2$ di risoluzione base.

Layer 3: Frame $(k-1)*4+1$ e $(k-1)*4+3$ di livello base.

Layer 4: Frame $(k-1)*4+4$ di risoluzione enhanced.

Layer 5: Frame $(k-1)*4+2$ di risoluzione enhanced.

Layer 6: Frame $(k-1)*4+1$ e $(k-1)*4+3$ di livello enhanced.

Quindi, ad esempio, sottoscrivendo i layer 1 e 2 si ottiene un flusso a frame rate medio e risoluzione base, oppure sottoscrivendo i layer 1 e 4 si ottiene un flusso a frame rate minimo e a risoluzione enhanced. Tutte le possibili combinazioni sono riportate nella tabella 4.23. Volendo sfruttare la scalabilità in precisione si potreb-

Layer sottoscritti	Frame Rate	Risoluzione
1	Min	Base
1, 2	Med	Base
1, 2, 3	Max	Base
1, 4	Min	Enhanced
1, 2, 4, 5	Med	Enhanced
1, 2, 3, 4, 5, 6	Max	Enhanced

Tabella 4.23: Combinazioni possibili dei layer

bero affiancare altri sei layer corrispondenti a quelli indicati sopra ma nei quali, ad esempio, vengono utilizzati indici da 4 bit, e l'utente potrebbe scegliere se ricevere da uno o dall'altro gruppo. Questo comporta una crescita notevole nel numero di layer utilizzati (e quindi di gruppi multicast da creare) che in questo modo passa a dodici, e ciò ha contribuito a farci scartare il supporto per la scalabilità in precisione nel prototipo multicast.

Il server crea quindi sei diverse sessioni multicast verso sei diversi end-point (indirizzo multicast - numero di porto), a ciascuna di esse è associato un layer diverso. Solo sul primo di questi sei gruppi però viene attivata la trasmissione delle informazioni SDES opzionali, in quanto trasmetterle su tutti i gruppi risulterebbe ridondante. Dal client si possono scegliere ancora una volta i parametri come il frame rate e la risoluzione spaziale, ma in questo caso, a differenza di quanto avviene con il prototipo unicast, nessuna connessione TCP è stabilita con il server, ma le scelte vengono tradotte nella sottoscrizione dei gruppi multicast opportuni (secondo quanto indicato nella tabella 4.23).

Presso il laboratorio multimediale del *CNIT* è stato possibile effettuare delle prove molto interessanti sul funzionamento di questo prototipo. Infatti la rete locale di tale laboratorio è in realtà composta da più LAN connesse a livello 3 della pila OSI, cioè da veri e propri router che supportano il multicasting, il protocollo di routing multicast utilizzato è il **PIM** (Protocol Indipendent Multicast) nella modalità *sparse* [23]. Il nome deriva dal fatto che PIM non si appoggia a nessun protocollo di routing unicast per svolgere il suo compito (cosa che invece fanno altri protocolli di routing multicast). La gestione dell'inoltro dei pacchetti è in parte centralizzata, è previsto infatti un router particolare, detto *Rendezvous Point*, al quale arrivano i pacchetti multicast e le richieste di sottoscrizione dei gruppi, esso si occupa inizialmente di inoltrare i flussi dei gruppi multicast verso gli host che ne hanno fatto richiesta. Dopo il transitorio iniziale, quando la situazione si è stabilizzata, è possibile effettuare la commutazione verso il funzionamento non centralizzato, I vari router intermedi vengono istruiti sulle modalità di inoltro e quindi si passa da un approccio *center-based* ad un approccio *group-based.*

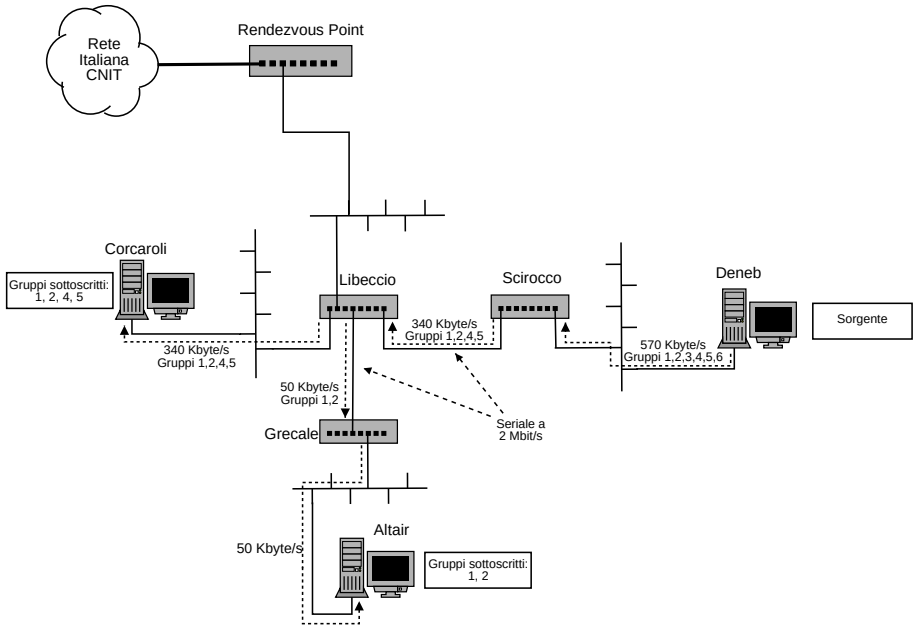


Figura 4.20: Il funzionamento del prototipo multicast

La figura 4.20 mostra la rete sulla quale si è testato il prototipo multicast. Sull'host DENEB è stato installato il server, mentre sui due host CORCAROLI e ALTAIR È stato installato il client. Su ALTAIR si è deciso di ricevere i primi due gruppi, mentre

su CORCAROLI si è deciso di ricevere i gruppi 1, 2, 4, 5. Nel transitorio iniziale si è constatato che il traffico relativo a tutti i gruppi raggiungeva il Rendezvous Point, finito il transitorio però la situazione si presentava come quella di figura 4.20. Le linee punteggiate rappresentano il traffico multicast che attraversa il collegamento, accanto ad esse ci sono i valori del tasso medio misurato. Si può vedere chiaramente che il traffico relativo ai gruppi che non sono stati sottoscritti da nessuno (gruppi 3 e 6) non lascia la LAN dove si trova la sorgente, ogni router inoltra su una sua interfaccia di uscita solo il traffico relativo a quei gruppi che sono stati sottoscritti dagli host raggiungibili a partire da essa.

4.3.1.2 La codifica intraframe periodica

Anche in questo caso periodicamente viene inoltrata una frame con codifica intra, così come per il prototipo unicast queste sono sempre del tipo $(k-1)*4+4$ con $k = 25, 50, 75\ldots$, ciò vuol dire che tali frame sono sempre inoltrate sul gruppo 1 (quelle enhanced sul gruppo 4). Si è scelto di operare in questo modo perché il gruppo 1 è l'unico che viene ricevuto in ogni caso (il gruppo 4 è l'unico che viene ricevuto in ogni caso, quando si è scelto la risoluzione enhanced) e quindi raggiunge sicuramente tutti i ricevitori.

4.3.1.3 La sincronizzazione

Sulla sincronizzazione è il caso di spendere qualche parola in più. Come abbiamo visto questo prototipo utilizza la tecnica *MMG*, questo vuol dire che i dati arrivano al ricevitore separati su più flussi, però su uno (due, se consideriamo anche le frame enhanced) di questi flussi si trovano le frame con codifica intra (che sono quelle che consentono la sincronizzazione). Una volta ricevuta quindi una frame con il marker bit settato (frame intra) non c'è modo di sapere se le frame che si stanno ricevendo sugli altri flussi sono quelle che effettivamente la dovrebbero seguire, infatti i numeri di sequenza non sono d'aiuto perché ogni flusso è trasportato su di una diversa sessione RTP ed ogni sessione viene inizializzata con un numero di sequenza scelto a caso.

Una possibile soluzione potrebbe essere quella di settare il marker bit delle prime frame, di tutti i layer, che seguono immediatamente quella con codifica intra, tuttavia questo approccio non è molto "pulito" in quanto violerebbe la semantica che si è affidata al marker bit (segnare le frame con codifica intra). Per questo motivo con il prototipo multicast al momento è permessa la sincronizzazione solo se si decidere di

ricevere unicamente i due (o solo il primo dei due) flussi sui quali sono trasportati le frame intra.

Questo problema è tipico delle applicazioni MMG che d'altra parte, presentano anche altre difficoltà. Ad esempio, ogni diversa sessione sarà caratterizzata da un diverso SSRC mentre tutte avranno lo stesso CNAME, sarà però compito del software usato in ricezione riconoscere che tutte quelle coppie SSRC/CNAME in realtà si riferiscono ad un'unica applicazione e gestirle di conseguenza. Questi motivi hanno spinto diversi ricercatori a richiedere che nella nuova versione del protocollo RTP ci fosse il supporto nativo per le applicazioni multisessione (e quindi in particolare per la tecnica MMG). In particolare la proposta ufficializzata M. Speer e S. McCanne in [21] è stata abbracciata dai progettisti di RTP come si può leggere nell'ultimo Internet Draft di RTP [22] in scadenza per Gennaio 2002.

4.3.1.4 Error concealment

La tecnica utilizzata è del tutto analoga a quella vista per il prototipo unicast. La differenza sta nel fatto che in questo caso ci sono un massimo di sei diverse sessioni multicast, il controllo sui numeri di sequenza, che nel prototipo unicast veniva fatto in globale, in questo caso viene effettuato su ogni singola sessione, indipendentemente dalle altre. C'è una precisa corrispondenza tra le sessioni e i diversi blocchi di codifica (la figura 4.19 mostra questa corrispondenza per quanto riguarda la codifica), quindi non c'è il pericolo di decodificare un pacchetto con il blocco di decodifica errato, in questo caso è semplicemente necessario stabilire se c'è stata una perdita o meno.

4.3.2 L'interfaccia grafica

L'interfaccia grafica del prototipo multicast deriva direttamente da quella del prototipo unicast.

Confrontando la figura 4.21, in cui è mostrata l'interfaccia grafica del client multicast, con la figura 4.3, in cui è mostrata l'interfaccia grafica del client unicast, si possono notare le differenze. Innanzitutto si nota la mancanza del menu di selezione del numero di bit, in quanto, come detto più volte, questa funzionalità non viene sfruttata dal prototipo multicast, inoltre si nota l'assenza della casellina in cui inserire l'indirizzo IP del server. Tale indirizzo nel prototipo unicast era utilizzato dal client per creare la connessione TCP con il server ma come abbiamo già detto,

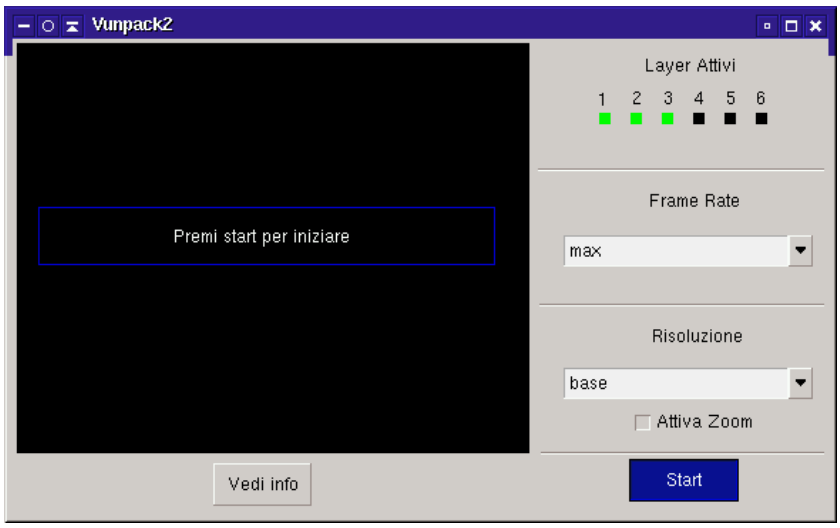


Figura 4.21: L'interfaccia grafica del prototipo multicast

nel prototipo multicast non viene creata nessuna connessione TCP, quindi quell'indirizzo è inutile. Alla pressione del tasto *Start*, il client effettua la sottoscrizione dei gruppi opportuni, scelti in base ai parametri (risoluzione e frame rate) impostati, e si mette in attesa della frame con codifica intra, i led in alto a destra si illuminano rispecchiando lo stato dei gruppi multicast (per gruppi attivi si intende quelli che sono stati sottoscritti).

Conclusioni e sviluppi futuri

In questo lavoro è stato presentato un sistema per la trasmissione, su reti a commutazione di pacchetto, di video in tempo reale destinato ad un utenza eterogenea, sia per quanto riguarda le caratteristiche della rete di accesso utilizzata, che per la potenza di calcolo disponibile. Il codec utilizzato è dotato di una forte scalabilità, è possibile infatti agire su tre distinti parametri che sono la qualità, il frame rate e la risoluzione spaziale. Esso combina in modo efficace la quantizzazione vettoriale (per la riduzione della ridondanza intraframe) ed il conditional replenishment (per la riduzione della ridondanza interframe) e nonostante questo è caratterizzato da una complessità computazionale contenuta, il che rende possibile effettuare tanto la codifica che la decodifica in tempo reale su processori di fascia media.

Il lavoro si è concentrato sullo strato di rete, mentre sono stati implementati due prototipi per la trasmissione dei dati codificati su rete, per il supporto della trasmissione in real-time si è fatto ricorso al protocollo RTP (che si affida ai servizi di UDP).

Il primo prototipo realizza una trasmissione di tipo unicast, il flusso codificato viene pacchettizzato ed inoltrato verso l'host destinazione, quest'ultimo decodifica le frame ricevute e le mostra a video. I vari tipi di scalabilità offerti dal codec possono essere combinati in diversi modi, questo ha permesso di studiare quali sono le combinazioni migliori al variare delle caratteristiche della rete di accesso. Si è evidenziato inoltre che, allo stato attuale delle cose, la scalabilità in precisione probabilmente non dà quei benefici che ci sarebbe potuti aspettare.

Il secondo prototipo implementato realizza una trasmissione di tipo multicast. In questo caso vengono sfruttate più a fondo le caratteristiche di scalabilità del codec tramite la tecnica MMG. I vari sottoflussi che compongono il flusso embedded vengono inoltrati su gruppi multicast diversi, in questo modo ogni host ricevente può stabilire, in modo indipendente da quanto fanno gli altri, quali livelli utilizzare e questo si traduce nella scelta dei gruppi da sottoscrivere.

Per entrambi i prototipi si è prevista la spedizione periodica di una frame con codifica intra al fine di contenere il decadimento della qualità della sequenza riprodotta, e soprattutto indispensabile per permettere la sincronizzazione degli host che dovessero inserirsi nella sessione quando questa è già iniziata. La sincronizzazione va perfezionata per il prototipo multicast, infatti in questo caso ci si è imbattutti nel problema di dover sincronizzarsi contemporaneamente su diversi flussi di pacchetti.

Inoltre è stata implementata anche una semplice tecnica di error concealment per reagire all'eventualità della perdita dei pacchetti.

Ci sono ancora diversi punti su cui lavorare per rendere funzionale l'applicazione realizzata.

Innanzitutto deve essere ancora implementato il codice necessario all'acquisizione delle frame dall'hardware video, allo stato attuale delle cose, le frame non codificate vengono lette da disco.

Inoltre attualmente esistono due entità separate, il server che si occupa della codifica e della trasmissione ed il client, che invece si occupa della ricezione, della decodifica e del rendering a schermo della sequenza video. Queste due unità vanno fuse in un'unica applicazione che sia in grado di svolgere entrambe i ruoli. Molta attenzione va posta alle prestazioni poichè, se è vero che sia il client che il server sono in grado di girare in tempo reale, non è detto che questo sia possibile quando entrambi vengono eseguiti in modo congiunto sulla stessa macchina. L'integrazione deve essere quindi seguita anche da un lavoro di ottimizzazione delle prestazioni.

Il traffico generato ha uno spiccato carattere impulsivo a causa della presenza delle frame con codifica intra. Sarebbe opportuno "smussare" il traffico per evitare di avere perdite concentrate proprio sulle frame intra. Ci sono due modi possibili di agire.

1. Si può intervenire a livello di rete. Le frame codificate vengono frammentate ed inserite in pacchetti di dimensioni più piccole e poi, tramite algoritmi tipo leaky bucket o token bucket, si effettua lo "shaping" del traffico;

2. Si può intervenire a livello di codifica. É necessario modificare lo schema di codifica per evitare la trasmissione delle frame intra. Si può ad esempio trasmettere periodicamente un blocchettino di immagine che sia codificato in modo intra, ogni volta si trasmettere un blocchettino differente, dopo un certo tempo tutta l'area dell'immagine sarà stata coperta.

La prima soluzione è certamente quella più veloce da implementare in quanto non

richiede modifiche sostanziali all'architettura sottostante, tuttavia non è molto adatta ad una trasmissione in tempo reale. Gli algoritmi di shaping introducono necessariamente dei ritardi tanto in trasmissione che in ricezione (dove bisognerà ricomporre i dati frammentati), senza contare poi che la perdita anche di un solo frammento comporterebbe la perdita di tutta la frame codificata. Inoltre questo modo di agire va contro il principio dell'*Application Level Framing (ALF)* espresso da Clark e Tennenhouse in [18]. Nello spirito di ALF, gli stream di dati prodotti da applicazioni real-time vanno pacchettizzati in aggregati, chiamati *Application Data Units (ADU)*, che sono significativi per l'applicazione e che quest'ultima è in grado di utilizzare indipendentemente dalla sorte degli altri pacchetti.

La modifica dell'algoritmo di codifica è certamente più complessa da effettuare, ma produce senza ombra di dubbio i risultati migliori, non introduce ulteriori ritardi ed il tasso risulta molto meno variabile. Attualmente esistono già diversi lavori in cui questa tecnica è usata con successo, probabilmente i risultati più interessanti sono mostrati dal codec PVH presentato da McCanne, Vetterli e Jacobson in [20].

Certamente questi problemi hanno evidenziato che probabilmente la soluzione migliore sarebbe stata quella di affrontare contemporaneamente tanto la progettazione dell'algoritmo di codifica che dello strato di rete.

Bibliografia

[1] A. Bovik (ed.), “Handbook of image and Video Processing”, Accademic Press 2000.

[2] P. Parlato, “Codifica video a bassa complessità per la trasmissione in tempo reale su reti eterogenee”, Tesi di Laurea, relatore prof. G. Poggi, Università degli Studi di Napoli “Federico II”, anno accademico 2000/2001.

[3] A. S. Tanenbaum, “Reti di computer”, Prentice Hall International 1996.

[4] J. Postel (ed.), “Trasmission Control Protocol - DARPA Internet Program Protocol Specification”, RFC 793, USC/Information Sciences Institute, Settembre 1981.

[5] J. Postel (ed.), “User Datagram Protocol - DARPA Internet Program Protocol Specification”, RFC 768, USC/Information Sciences Institute, Agosto 1980.

[6] J. Postel (ed.), “Internet Protocol - DARPA Internet Program Protocol Specification”, RFC 791, USC/Information Sciences Institute, Settembre 1981.

[7] C. Aras, J. Kurose, D. Reeves, H. Schulzrinne, “Real-Time Communication in Packet-Switched Networks”, Proceedings of the IEEE, Vol. 82, No. 1, Gennaio 1994.

[8] M. Vitez, “Trasmissione di musica a degrado controllato su Internet”, Tesi di Laurea, relatore prof. F. Babich, Università degli Studi di Trieste, anno accademico 1997/1998.

[9] J. Liesenborgs, “Voice iver IP in networked virtual environments”, Tesi di Laurea, relatore prof. W. Lamotte, Universiteit Maastricht, anno accademico 1999/2000.

[10] B. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, Luglio 1994.

[11] R.Baden, L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification”, RFC 2205, Settembre 1997.

[12] J. Wroclawski, “The use of RSVP on IETF Integrated Services”, RFC 2210, Settembre 1997.

[13] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services”, RFC 2475, Dicembre 1998.

[14] K. Nichols, S. Blake, F. Baker, D. Black, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, RFC 2474, Dicembre 1998.

[15] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, “ A Framework for Integrated Services Operation over Diffserv Networks”, RFC 2998, Novembre 2000.

[16] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Trasport Protocol for Real-Time Applications”, RFC 1889, Gennaio 1996.

[17] H. Schulzrinne, “RTP Profile for Audio and Video Conferences with Minimal Control”, RFC 1890, Gennaio 1996.

[18] D. Clark, D. L. Tennenhouse, “Architectural considerations for a new generation of protocols” in SIGCOMM Symposium on Communications Architectures and Protocols, (Philadelphia, Pennsylvania), pp. 200-208, IEEE, Sett. 1990. Computer Communications Review, Vol. 20 No. 4, Sett. 1990.

[19] N. Chaddha, A. Gupta, “A Frame-work for Live Multicast of Video Streams over the Internet”, International Conference on Image Processing, pp. 1-4, 1996.

[20] S. McCanne, M. Vetterli, V. Jacobson, “Low-Complexity Video Coding for Receiver-Driven Layered Multicast”, IEEE Journal on selected areas in communications, Vol.15 No. 6 pp. 983-1001, Agosto 1997.

[21] M. Speer, S. McCanne, “RTP Usage with Layered Multimedia Streams”, Internet Draft, Dicembre 1996.

[22] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Trasport Protocol for Real-Time Applications”, Internet Draft, Luglio 2001.

[23] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, “Protocol Indipendent Multicast-Sparse Mode (PIM-SM): Protocol Specification”, RFC 2362, Giugno 1998.

[24] L. Wu, R. Sharma, B. Smith, “Thin Streams: An Architecture for Multicasting Layered Video”, 1997.

[25] D. Hoffman, M. Speer, “Hierarchical Video Distribution over Internet-Style Networks”, 1996.

Indice analitico